ВІСНИК
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
“Київський політехнічний інститут”

Інформатика, управління та обчислювальна техніка

55

2012

Міністерство освіти і науки України
Національний технічний університет України «КПІ»

ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТА УРКАЇНИ «КПІ»

Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 55

Київ “ВЕК+” 2012

УДК 004

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2012. – № 55. – 212 с.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 10 від 16.04.2012

Описано результати дослідження і створення компонентів обчислювальних й інформаційних систем і комплексів, пристроїв автоматики та передавання даних, систем автоматизації програмування, контролю й діагностики, штучного інтелекту тощо.

Для аспірантів, студентів, фахівців з обчислювальної техніки, систем керування, автоматизації програмування, штучного інтелекту та інших інформаційно-обчислювальних систем.

ISSN 0135-1729

Свідоцтво про державну реєстрацію № 16949-5719 Р від 17.06.2010

Збірник наукових праць українською, англійською та російською мовами

Web-ресурс – http://it-visnyk.kpi.ua

Підп. до друку 16.04.2012. Формат 60×84 1/16. Гарнітура Times. Папір офсетний № 1.
Наклад 150 прим.
Надруковано в ЗАТ “ВІПОЛ”, 03151 м.Київ, вул. Волинська, 60.

ЗМІСТ

SUMMARY

УДК 519.854.2

ПАВЛОВ А.А.,
МИСЮРА Е.Б.,
ЩЕРБАТЕНКО О.В.,
МИХАЙЛОВ В.В.,
МЕЛЬНИКОВ О.В.

ФОРМАЛЬНОЕ ОПИСАНИЕ ТРЕХУРОВНЕВОЙ МОДЕЛИ ОПЕРАТИВНОГО ПЛАНИРОВАНИЯ СИСТЕМ С СЕТЕВЫМ ПРЕДСТАВЛЕНИЕМ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ. ПОСТАНОВКА НОВЫХ ЗАДАЧ ИССЛЕДОВАНИЯ

Приводится формальное описание трехуровневой модели оперативного планирования в классическом виде, что позволяет сформулировать постановку новых задач построения результирующего расписания, имеющих эффективное точное решение.

A formal description in the traditional sense of the three-level model of operational planning is given, which allows to formulate the new problems of constructing the resulting schedule with effective exact solution.

Введение

В [1] формальное описание трехуровневой модели оперативного планирования систем с сетевым представлением технологических процессов методически реализуется в органическом единстве с описанием построения результирующего оперативного плана, что максимально просто позволяет обосновать посылку о том, что результирующий эвристический алгоритм построения расписания реализует решение в области глобального экстремума по любому из предложенных критериев.

В статье приводится формальное описание трехуровневой модели оперативного планирования в классическом виде, что позволяет сформулировать постановку новых задач построения результирующего расписания, имеющих эффективное точное решение.

Агрегированная модель представлена тремя уровнями. Первый уровень предназначен для решения на основе исходной информации следующей задачи.

Постановка задачи первого уровня модели (исходная постановка задачи)

Задано множество n комплексов взаимосвязанных работ $J = \{J_1, J_2, \ldots, J_n\}$ (комплекс работ $J_i,\ i = \overline{1, n}\ \overline{1, n}$, в дальнейшем называется заданием). На каждом подмножестве J_i частичный порядок задан ориентированным ациклическим графом. Частичная упорядоченность очевидным образом определяется технологией выполнения комплекса работ. Каждая следующая работа может начаться только по завершению предыдущих работ. Вершины графа отвечают работам, связи указывают на отношения предшествования. Конечные вершины отвечают завершению выполнения заданий. Для каждой вершины j графа известна l_j – детерминированная продолжительность выполнения (интегрированный показатель, отображающий выделенные ресурсы – материальные, человеческие, производственные; длительность выполнения каждого задания определяется его критическим путем); для каждой работы $j \in I$ (I – множество конечных вершин, идентифицирующихся с множеством заданий) задан вес ω_j; для отдельных заданий задан директивный срок окончания d_i. Величина веса определяется потенциальной сложностью, важностью и неоднозначностью (для работ, связанных с необходимостью получения нового научного решения) выполнения тех работ, без которых в целом задание не может быть выполнено. Для выполнения работ применяется множество ограниченных ресурсов.

Необходимо построить распределение (согласованный план) выполнения работ по ресурсам с учетом критериев оптимальности, указанных ниже, и их комбинаций.

Главная цель планирования в условиях рынка – максимизация прибыли предприятия. Поэтому максимизация прибыли является общим критерием оптимальности для всех уровней модели планирования. Прибыль рассчитывается как планируемый доход от реализации всех изделий (выполнения всех заданий) минус затраты *3* (все издержки), очевидным образом рассчитываемые по оптимальному расписанию в блоке принятия решений. В процедуру приня-

тия решений и максимизацию прибыли оптимизация по *З* не входит.

В модели рассматриваются следующие критерии оптимальности.

Задача 1. Максимизация суммарной прибыли предприятия в случае отсутствия директивных сроков.

В обеспечении прибыльности предприятия важное значение играет фактор времени. В выигрыше будет тот, кто обеспечивает максимально быстрое выполнение заказов и сокращение времени выхода на рынок новых товаров. При отсутствии директивных сроков прибыль от реализации *i*-го изделия (выполнения *i*-го задания) является функцией времени и равна $P_i(t) = \omega_i(T)\cdot(T - C_i)$, где $\omega_i(T)$ – весовой коэффициент изделия (задания) *i*, определенный экспериментальным путем; T – плановый период; $C_i \le T$ – момент окончания выполнения изделия (задания) *i*, соответствующий моменту окончания выполнения его конечной вершины. Критерий максимизации суммарной прибыли предприятия в этом случае определяется выражением

$$F_1 = \sum_{i=1}^{n} P_i(t) = \sum_{i=1}^{n} \omega_i(T)\cdot(T - C_i) + P - З = \\ = T\cdot\sum_{i=1}^{n}\omega_i(T) - \sum_{i=1}^{n}\omega_i(T)\cdot C_i + P - З \to \max, \quad (1)$$

где P – гарантированный минимальный доход от продажи (выполнения) всех n изделий (заданий). Таким образом, максимизируемая функция имеет вид

$$\max\left\{\sum_{i=1}^{n}\omega_i(T)\cdot(T - C_i)\right\} + P - З,$$

отсюда критерий оптимальности:

$$\min\sum_{i=1}^{n}\omega_i(T)\cdot C_i \ (\textit{критерий 1}).$$

Итак, критерий F_1 эквивалентен критерию минимизации суммарного взвешенного момента окончания выполнения заданий (МВМ) при заданном отношении порядка на множестве работ каждого задания.

Задача 2. Максимизация суммарной прибыли предприятия при условии: для всех заданий $i \in I$ введены директивные сроки d_i, которые не могут быть нарушены (планирование «точно в срок»):

$$\max\left\{\sum_{i=1}^{n}\omega_i U_i\right\} - З, \quad \text{где } U_i = \begin{cases}1, C_i = d_i\\ 0, C_i \ne d_i\end{cases},$$

ω_i – прибыль от выполнения *i*-го задания, если оно выполнено точно в срок. Критерий оптимальности:

$$\max\left\{\sum_{i=1}^{n}\omega_i U_i\right\} \ (\textit{критерий 2});$$

Задача 3. Максимизация суммарной прибыли предприятия при условии: для некоторых заданий $i \in \overline{1,k}$ заданы директивные сроки, которые не могут быть нарушены, для остальных заданий $d_i = 0$:

$$\max\left\{\sum_{i=1}^{k}\omega_i U_i + \sum_{i=k+1}^{n}\omega_i(T)\cdot(T - C_i)\right\} + P - З,$$

где $U_i = \begin{cases}1, C_i = d_i\\ 0, C_i \ne d_i\end{cases}$, P – гарантированный минимальный доход от продажи (выполнения) изделий (заданий) $i = \overline{k+1,n}$; ω_i – прибыль от выполнения *i*-го задания, если оно выполнено точно в срок; $\omega_i(T)$ – весовой коэффициент задания *i* (имеет тот же смысл, что и в задаче 1). Критерий оптимизации:

$$\max\left\{\sum_{i=1}^{k}\omega_i U_i + \sum_{i=k+1}^{n}\omega_i(T)\cdot(T - C_i)\right\} \ (\textit{критерий 3});$$

Задача 4. Максимизация суммарной прибыли предприятия при условии: для всех заданий $i \in I$ введены директивные сроки d_i, необходимо минимизировать суммарное взвешенное запаздывание выполнения заданий относительно директивных сроков:

$$\max\left\{P - \sum_{i=1}^{n}\omega_i\max(0, C_i - d_i)\right\} - З,$$

где P – гарантированный минимальный доход от продажи (выполнения) всех n изделий (заданий), если все они выполнены без запаздывания; ω_i – штраф за запаздывание окончания выполнения *i*-го задания относительно директивного срока на единицу времени. Критерий оптимизации:

$$\min\left\{\sum_{i=1}^{n}\omega_i\max(0, C_i - d_i)\right\} \ (\textit{критерий 4});$$

Величина $\omega_i\max(0, C_i - d_i)$ – уменьшение дохода P в случае выполнения задания *i* с запаздыванием $C_i - d_i$. Решение по выполнению или отказу от выполнения таких заданий принимается в блоке принятия решений.

Задача 5. Постановка задачи соответствует задаче 4 с дополнительным условием: для некоторых заданий $i \in \overline{1,k}$ директивные сроки не могут быть нарушены:

$$\max\left\{\sum_{i=1}^{k}\omega_i' U_i - \sum_{i=k+1}^{n}\omega_i'' \max(0, C_i - d_i)\right\} + P - 3,$$

где $U_i = \begin{cases}1, C_i = d_i\\ 0, C_i \neq d_i\end{cases}$, ω_i' – прибыль от выполнения i-го задания, если оно выполнено точно в срок; ω_i'' – штраф за запаздывание окончания выполнения i-го задания относительно директивного срока на единицу времени, P – гарантированный минимальный доход от выполнения заданий $i = \overline{k+1, n}$, если они выполнены в срок. Критерий оптимальности:

$$\max\left\{\sum_{i=1}^{k}\omega_i' U_i - \sum_{i=k+1}^{n}\omega_i'' \max(0, C_i - d_i)\right\} (\textit{критерий 5}).$$

Величина $\omega_i'' \max(0, C_i - d_i)$ – уменьшение дохода P в случае выполнения задания i с запаздыванием $C_i - d_i$. Решение по выполнению или отказу от выполнения таких заданий принимается в блоке принятия решений.

Задача 6. Для всех заданий $i \in I$ введены директивные сроки d_i. Для каждого задания указана величина ω_i – абсолютная прибыль от выполнения задания, не зависящая от момента окончания выполнения задания в том случае, если задание выполняется без запаздывания относительно директивного срока, иначе прибыль предприятия по этому заданию равна нулю. Задача – максимизировать суммарную прибыль предприятия:

$$\max\left\{\sum_{i=1}^{n}\omega_i U_i\right\} - 3, \quad \text{где } U_i = \begin{cases}1, C_i \le d_i\\ 0, C_i > d_i\end{cases},$$

где ω_i – прибыль от выполнения i-го задания, если оно выполнено без запаздывания относительно директивного срока; 3 – риск уменьшения прибыли из-за срыва выполнения задания в срок. Критерий оптимальности:

$$\max\left\{\sum_{i=1}^{n}\omega_i U_i\right\} \quad (\textit{критерий 6}).$$

Задача 7. Для всех изделий заданы директивные сроки d_i. Необходимо минимизировать суммарный штраф предприятия как за опережение, так и за запаздывание относительно директивных сроков:

$$\max\left\{P - \sum_{i=1}^{n}\omega_i |C_i - d_i|\right\} - 3,$$

где P – гарантированный минимальный доход от продажи (выполнения) всех n изделий (заданий), если все они выполнены без опережения и запаздывания; ω_i – штраф за отклонение момента окончания выполнения i-го задания от директивного срока на единицу времени. Критерий оптимизации:

$$\min\sum_{i=1}^{n}\ \omega_i |C_i - d_i|. \quad (\textit{критерий 7})$$

Величина $\omega_i |C_i - d_i|$ – уменьшение дохода P в случае выполнения задания i с запаздыванием $C_i - d_i$. Решение по выполнению или отказу от выполнения этих заданий принимается в блоке принятия решений.

Ограничения:
- простои ресурсов при выполнении работ допускаются;
- прерывания работ при выполнении запрещены.

Задачи 1–7 относятся к труднорешаемым, и для их решения не существует эффективных методов решения, как точных, так и приближенных. Предлагаемая трехэтапная модель планирования позволила создать эффективные эвристические алгоритмы, реализующие решение в области глобального оптимума.

Второй уровень модели является агрегированным представлением первого уровня. Агрегирование с целью уменьшения размерности исходной задачи осуществляется с помощью построения агрегированных работ и мультиресурсов. В результате выполнения агрегации свойства модели сохраняются.

Совокупность ресурсов и исполнителей разделяется на отдельные, достаточно автономные модули – *мультиресурсы* (мультиресурс – устойчивая группа совместно работающих ресурсов – например, бригада, группа однотипного оборудования, однопрофильное подразделение). Мультиресурс характеризуется суммарным количеством рабочих мест, общей длительностью переналадки оборудования. Мультиресурсы могут находиться как в одной, так и в разных организациях. В общем случае, если это обусловлено производственной необходимостью и позволит более эффективно выполнить заданный объем работ, то в состав мультиресурса может быть включено разнотипное оборудование.

Агрегированная работа – совокупность работ, выполняемых в одном мультиресурсе в рамках одного захода в мультиресурс по одному заданию.

Длительность выполнения агрегированной работы в мультиресурсе однозначно определяется ее критическим путем в данном мультиресурсе, который строится с помощью стандарт-

ных процедур. Например, для случая, когда в мультиресурсе с однотипным оборудованием выполняется некоторое количество в общем случае связных работ, составляющих агрегированную работу, формальная модель критического пути определяется последовательностью множеств работ p типов. Множество работ r-го типа выполняется после множества работ (r–1)-го типа, $r = \overline{2, p}$. Множество работ r-го типа состоит из n_r независимых работ. Тогда l_{jk} – длительность j-й агрегированной работы в данном k-м мультиресурсе – рассчитывается по формуле

$$l_{jk} = \sum_{r=1}^{p} \frac{\widehat{l}_r n_r}{m_k}, \qquad (2)$$

где $\widehat{l}_r$ – длительность выполнения работы r-го типа, m_k – количество однотипных рабочих мест k-го мультиресурса. Если существует несколько однотипных мультиресурсов $k = \overline{s,t}$, отличающихся количеством рабочих мест, в каждом из которых может выполняться агрегированная работа, то длительность выполнения для каждого из мультиресурсов определяется по формуле (2).

Некоторые агрегированные работы, принадлежащие разным заданиям и выполняемые в одном мультиресурсе, объединяются в общие агрегированные работы (семейства) так, чтобы не требовалась наладка для работы, если она принадлежит тому же семейству, что и агрегированная работа, выполненная перед ней. На графе связности это отображено общими вершинами. При объединении агрегированных работ в «общие вершины» исключается время переналадки для объединяемых работ, которое в отдельных случаях может существенно превышать длительность выполнения работ. Если объединение в «общие вершины» не реализуется, то при выполнении каждой агрегированной работы необходимо учитывать время переналадки, что существенно увеличит время прохождения заданий в системе. Длительность выполнения «общей вершины» равна сумме длительностей выполнения входящих в ее состав агрегированных работ.

«Общие вершины», включающие большое количество агрегированных работ, с одной стороны, имеют преимущество эффективного использования мультиресурса, так как число наладок существенно уменьшается. С другой стороны, выполнение такой «общей вершины» может задержать выполнение другой агрегированной работы, не принадлежащей «общей вершине». Поэтому разработаны эвристические правила построения «общих вершин», учитывающие критерии оптимальности, приоритеты заданий, которым принадлежат объединяемые агрегированные работы, директивные сроки и прогнозируемое время поступления агрегированной работы на выполнение в данный мультиресурс [1].

Постановка задачи второго (агрегированного) уровня модели

Задано множество n комплексов взаимосвязанных агрегированных работ $J = \{J_1, J_2, \ldots, J_n\}$ (комплекс агрегированных работ J_i, $i = \overline{1, n}$ $\overline{1, n}$, в дальнейшем называется заданием). На каждом подмножестве J_i частичный порядок задан ориентированным ациклическим графом. Каждая следующая агрегированная работа может начаться только по завершению предыдущих агрегированных работ. Вершины графа отвечают агрегированным работам, связи указывают на отношения предшествования. Конечные вершины отвечают завершению выполнения заданий. Для выполнения работ применяется множество мультиресурсов. Для каждой вершины j графа задана l_{jk} – длительность j-й агрегированной работы в k-м мультиресурсе, определяемая по формуле (2); для каждой агрегированной работы $j \in I$ (I –множество конечных вершин, идентифицирующихся с множеством заданий) задан вес ω_j; для отдельных заданий задан директивный срок окончания d_i. Некоторые агрегированные работы, принадлежащие разным заданиям и выполняемые в одном мультиресурсе, объединены в общие агрегированные работы (семейства), длительность выполнения которых равна сумме длительностей выполнения входящих в ее состав агрегированных работ.

Необходимо построить согласованный план выполнения комплексов агрегированных работ мультиресурсами с учетом критериев оптимальности 1–7.

Ограничения:

– простои мультиресурсов при выполнении работ допускаются;
– прерывания агрегированных работ при выполнении запрещены;
– агрегированная работа не передается в другие мультиресурсы до ее полного завершения;
– длительность выполнения агрегированной работы в мультиресурсе определяется ее критическим путем в данном мультиресурсе;
– для семейств агрегированных работ наладка мультиресурса включается только один раз перед выполнением семейства;
– прерывание выполнения семейства агрегированных работ запрещено.

Несмотря на меньшую размерность задачи второго уровня, она все же остается труднорешаемой. Поэтому требуется представить предприятие (систему планирования) в виде одного прибора с целью применения точного эффективного алгоритма решения задачи МВМ, позволяющего решать задачи большой размерности.

Третий уровень модели отвечает уровню, при котором предприятие (система планирования) представляется в виде одного прибора. Для этого длительности агрегированных работ приводятся к длительности их выполнения на одном приборе. Пусть l_{jk} – длительность j-й агрегированной работы при выполнении в k-м мультиресурсе. Тогда длительность выполнения j-й агрегированной работы на одном приборе при условии, что она может выполняться на одном из однотипных мультиресурсов $k = \overline{s,t}$, определяется по формуле Конвея-Максвелла:

$$l_j^1 = \frac{1}{\sum_{k=s}^{t} \frac{1}{l_{jk}}}. \qquad (3)$$

Сведение (3) с формулой (2) дает длительность агрегированной работы

$$l_j^1 = \sum_{r=1}^{p} \frac{\hat{l}_r n_r}{m_{st}}, \qquad (4)$$

где $\hat{l}_r$ – длительность выполнения работы r-го типа, $m_{st} = \sum_{k=s}^{t} m_k$ – суммарное количество однотипных рабочих мест мультиресурсов $\overline{s,t}$.

Т. к. одним из самых важных критериев планирования является длительность прохождения заданий в системе, то введем следующие дополнительные ограничения:

– длительность выполнения каждого задания определяется его критическим путем;
– общие агрегированные работы разных заданий лежат на их критических путях и выполняются в одном мультиресурсе.

Поэтому для каждого задания находится критический путь, определяющий минимальную длительность его выполнения. Некоторые агрегированные работы, принадлежащие разным критическим путям и выполняемые в одном мультиресурсе, объединяются в «общие вершины» по описанным выше правилам. На основе критических путей заданий и набора «общих вершин» строится ориентированный ациклический граф, называемый далее графом на критических путях.

Постановка задачи третьего уровня модели (уровень одного прибора)

Задано множество n критических путей для комплексов взаимосвязанных агрегированных работ $J = \{J_1, J_2, \ldots, J_n\}$ (комплекс работ J_i, $i = \overline{1,n}$, в дальнейшем называется заданием) и набор «общих вершин» на критических путях, объединяющий их в ориентированный ациклический граф на критических путях. Вершины графа отвечают агрегированным работам, связи указывают на отношения предшествования. Конечные вершины отвечают завершению выполнения заданий. Для каждой вершины j графа задана l_j^1 – длительность выполнения на одном приборе, определяемая формулой (3); для каждой агрегированной работы $j \in I$ (I – множество конечных вершин, идентифицирующихся с множеством заданий) задан вес ω_j; для отдельных заданий задан директивный срок окончания d_i. Предполагается, что все агрегированные работы выполняются последовательно на одном приборе.

Необходимо построить последовательность выполнения агрегированных работ одним прибором, оптимальную по критериям оптимальности 1–7.

Ограничения:

– простои прибора при выполнении агрегированных работ запрещены;
– прерывания агрегированных работ при выполнении запрещены.

Как показано выше, критерий максимизации прибыли сводится к критерию МВМ. Для решения задачи МВМ разработан эффективный точный ПДС-алгоритм [1], позволяющий решать задачи большой размерности. При назначении на выполнение заданий одним прибором определяется приоритет задания (отношение ожидаемой прибыли каждого задания к длине его критического пути), что позволило для каждого из критериев 2–7 построить соответствующую аппроксимирующую задачу МВМ [1] при условии, что веса всех вершин, кроме конечных, равны нулю. Задача решается на графе на критических путях. В результате решения аппроксимирующей задачи МВМ точным алгоритмом [1] получаем приоритетно-упорядоченную последовательность, определяющую очередность назначения заданий на выполнение: чем больше приоритет подпоследовательности, тем раньше в заданный срок агрегированные работы этой подпоследовательности назначаются на выполнение.

Полученная на третьем уровне приоритетно-упорядоченная последовательность агрегированных работ служит дополнительной информацией, позволяющей значительно повысить эффективность полученных решений на втором и первом уровнях. На третьем уровне мы получаем точное решение, и свойства этого решения максимально допустимо сохраняются на следующих этапах. Действительно, согласованное планирование 2-го этапа и по нему получение эвристического расписания 3-го этапа является лишь детализацией обобщенного решения в конкретном расписании модели 3-го этапа с максимально возможным сохранением свойств расписания третьего уровня, которому на агрегированном уровне соответствует глобальный оптимум нашего функционала.

Если в качестве критерия эффективности системы планирования используется критерий 4, то на третьем этапе планирования возникает следующая многоэтапная задача календарного планирования. После реализации блока принятия решений (рис. 8.2 [1]) в процессе согласования получаем результирующий портфель заказов с согласованными новыми директивными сроками, которые не могут быть нарушены. Следовательно, на третьем этапе построения результирующего плана выполнения работ получаем многоэтапную (сетевую) задачу календарного планирования, решение которой не связано с планами, полученными на 1 и 2 этапах. Они используются в блоке принятия решений для получения результирующего портфеля заказов и окончательно согласованных директивных сроков. Критериями оптимальности таким образом сформулированной многоэтапной задачи календарного планирования являются:

1) при условии выполнения директивных сроков и ограничения на работу оборудования без прерываний при выполнении множества независимых работ минимизировать суммарное время опережения директивных сроков. Требование на непрерывность работы оборудования в процессе реализации технологического процесса в большинстве случаев является естественным ограничением.

2) при условии выполнения директивных сроков минимизировать суммарное время опережения директивных сроков. В этом случае предполагается, что оборудование при выполнении множества независимых работ может работать с прерыванием.

Полученные результаты [1, 2, 3] в области построения ПДС-алгоритмов для труднорешаемых задач комбинаторной оптимизации позволяют построить точный эффективный алгоритм построения расписания на третьем этапе по критерию 1.

Для точного решения задачи по критерию 2 необходимо построить ПДС-алгоритмы для соответствующих задач календарного планирования с учетом возможности прерывания работы оборудования в процессе выполнения множества работ.

Выводы

Приведено формальное описание в классическом виде трехуровневой модели оперативного планирования систем с сетевым представлением технологических процессов. Показано, что результирующий эвристический алгоритм построения расписания реализует решение в области глобального экстремума по любому из предложенных критериев. Сформулированы на примере критерия 4 постановки новых задач построения результирующего расписания, имеющих эффективное точное решение на основе результатов, полученных в области построения ПДС-алгоритмов для труднорешаемых задач комбинаторной оптимизации [1].

Список литературы

1. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.
2. Конструктивные полиномиальные алгоритмы решения индивидуальных задач из класса NP. / А.А.Павлов, А.Б.Литвин, Е.Б.Мисюра, Л.А.Павлова, В.И.Родионов, под редакцией А.А.Павлова.– К.: Техника, 1993.– 126 с.
3. ПДС-алгоритми для важкорозв'язуємих комбінаторних задач. Теорія і методологія розробки / О.А.Павлов, Л.О.Павлова. – Ужгород: Поличка «Карпатського краю» №15, – 1998.– 320 с.

УДК 004.4’23:004.453

ЛУЦКИЙ Г.М.,
СТИРЕНКО С.Г.,
ЗИНЕНКО А.И.,
ГРИБЕНКО Д.В.

ПРЕДСТАВЛЕНИЕ ЗАДАЧ В СИСТЕМАХ РАСПАРАЛЛЕЛИВАНИЯ С ИЗМЕНЯЕМОЙ ЗЕРНИСТОСТЬЮ

В этой статье рассматриваются подходы к представлению вычислительных задач в различных современных системах параллельного программирования, и предлагается новый подход, который позволят описывать задачу в терминах абстрактных операций без явного использования примитивов параллельного программирования. Предложенный метод позволяет построить систему времени выполнения, обеспечивающую параллельное выполнение программы пользователя с возможностью изменения зернистости в зависимости от типа используемого аппаратного обеспечения. Система может найти широкое применение в распределённых и грид вычислениях.

This article studies the approaches to description of computational tasks in different modern parallel programming systems and proposes a new approach that allows to define a task in terms of abstract operations without explicit usage of parallel programming primitives. The proposed technique allows to build a runtime system providing parallel execution of the user code that could change granularity on-the-fly with regards to the hardware being used. Such a system can be used in distributed and grid computing.

Введение

Информатика и средства вычислительной техники являются одной из наиболее быстро развивающихся областей современной науки. С момента начала активного её развития сохраняется положительная динамика, что явно следует из соблюдения таких экспоненциальных зависимостей, как например закон Мура [1] и его следствия. Это также подчёркивается большим количеством научных и государственных организаций, декларирующих развитие вычислительной техники одним из приоритетных направлений. [2, 3] Постоянный рост требований к вычислительным ресурсам для решения наиболее важных научных задач современности привёл к созданию ряда технологий, позволяющих предоставлять ресурс для их решения, в частности кластерных вычислений [4] и международной грид-инфраструктуры. [5]

Традиционным подходом к повышению производительности вычислительных комплексов являлось усовершенствование аппаратной компоненты путём увеличения тактовых частот, степени интеграции элементов, и другими подобными методами. Однако в последнее десятилетие рост производительности за счёт их применения практически прекратился [6], в связи с чем широкое использование приобрели параллельные и распределённые вычисления [7]. Несмотря на потенциально большие приросты производительности в теории, на практике применение параллельных или распределённых вычислительных систем не всегда достаточно эффективно. Показано [8], что теоретически линейный рост скорости вычисления при горизонтальном масштабировании системы не только подвержен деградации из-за возникновения накладных расходов на выполнение служебных функций системы, но и также ограничен сверху численной характеристикой применяемого в программной компоненте алгоритма в следствие закона Амдала. Из вышесказанного следует, что при решении сложных научных задач на современных компьютерных системах на скорость получения результатов влияет не только аппаратное, но и программное обеспечение, в том числе прикладное. [9]

Развитие параллельных и распределённых вычислений в аппаратной части привело также к появлению большого количества технологий программирования таких систем, библиотек для параллельных вычислений; созданию новых языков программирования [10, 11], ориентированных исключительно на параллельные вычисления или добавления их поддержки в уже существующие популярные языки. [12, 13] С другой стороны усложнение аппаратной компоненты требует от пользователя вычислительной системы более глубоких знаний принципов работы системы для достижения приемлемой производительности. Одним из факторов, наи-

более значительно влияющих на скорость вычисления, является правильное соотношение зернистости задачи с зернистостью системы. В то время как пользователь может более точно оценить возможность разбиения задачи, то учёт особенностей вычислительной системы лучше возложить на других специалистов. Таким образом, возникает необходимость в программной системе, которая предполагает описание алгоритмов вычисления, разделения на подзадачи и объединения в терминах предметной области. Это позволило бы параллельно выполнять данные алгоритмы на произвольной системе, но при этом пользователю не требуется углубляться в тонкости аппаратной организации системы. Данный подход может упростить использование гетерогенных ресурсов, в частности грид-инфраструктуры.

Применяемые подходы к описанию вычислительных задач

Одной из наиболее простых в использовании технологий, реализующих близкий к предложенному подход, является OpenMP [14, 15]. Данная технология определяет ряд директив компилятора, обрабатываемых препроцессором, для ручного указания участков исходного кода, допускающих параллельное выполнение. Поддержка спецификации OpenMP 3.0 реализована в большинстве современных компиляторов. [16, 17, 18, 19] Идеология работы с данной технологией предполагает аннотирование исходного кода программы на языках C, C++ или Fortran, в том числе и написанного ранее, так чтобы обеспечить параллельное выполнение некоторых частей программы, именуемых секциями, или итераций циклов. При проектировании данной технологии были учтены статистические данные [20] о том, что выполнение циклической обработки занимает в среднем 80% времени выполнения программы. Исходя из чего основные усилия были применены к созданию простого механизма описания возможности параллельного выполнения итераций цикла. На данный момент OpenMP позволяет аннотировать циклы произвольного уровня вложенности, указывать на данные, разделяемые всеми итерациями, и эффективным алгоритмом выполнять свёртку результатов с использованием предопределённых ассоциативных операторов. Кроме поддержки со стороны компиляторов, технология OpenMP предполагает предоставление конечному пользователю разделяемой библиотеки времени выполнения, которая обеспечивает непосредственно создание параллельных потоков выполнения, планирование и балансировку нагрузки подзадачами (секциями, итерациями) этих потоков. Она также имеет программный интерфейс (API), позволяющий пользователю частично контролировать процесс выполнения, однако это не является обязательным. Технология OpenMP рассчитана для использования в системах с общей памятью и её планировщик, использующий принцип разделения работ и единую очередь готовых к выполнению подзадач, имеет ряд режимов, связанных с этой особенностью [14, 21]. OpenMP позволяет пользователю обеспечить параллельное выполнение произвольных задач при помощи аннотирования исходного кода, преимущественно циклических частей. Наиболее важным преимуществом данной технологии является отсутствие необходимости в существенной модификации исходных кодов, что позволяет достаточно быстро изменить коды ранее написанных вычислительных библиотек. Однако, данная технология применима только в системах с общей памятью, в то время как наиболее производительными на данный момент являются системы с локальной памятью, в частности кластерные.

Существует ряд разработанных библиотек исходного кода, позволяющих решать с применением параллельных вычислений, некоторые научные задачи. К ним можно отнести используемые в Центре суперкомпьютерных вычислений НТУУ «КПИ» [22] программные пакеты Gromacs, fftw, Gamess и другие. Будучи специализированными средствами для решения конкретного класса задач, они предоставляют достаточно простой и в то же время эффективный подход к их параллельному решению, однако все они спроектированы с учётом особенностей конкретных задач, что приводит к невозможности обобщения технических решений, принятых при их разработке, на произвольные описываемые пользователем задачи.

К наиболее обобщённым из используемых пакетов программного обеспечения можно отнести комплекс Lapack [23] и библиотеку Ouverture [24]. Данные программные средства предоставляют программный интерфейс для решения задач линейной алгебры и численного решения дифференциальных уравнений в частных производных, используя особенности кластерных вычислительных систем и технологии MPI [25]. Библиотека Ouverture, в частности

содержит в себе компоненты A++ и P++ для описания работы с матрицами и тензорами в объектно-ориентированном стиле на языке C++. Компоненты имеют идентичный интерфейс, однако второй из них использует MPI для обеспечения параллельного выполнения операций.

Приведенные программные пакеты широко используются в вычислениях и обеспечивают существенное ускорение решения поставленных задач, но вместе с тем требуют от пользователя явного сведения задачи к заложенным в них абстракциям, что хотя и возможно для большого количества задач, может потенциально приводить к накладным расходам на преобразование в процессе работы программы. Кроме того данные преобразования могут потребовать значительного изменения исходного кода программ, если таковой был ранее написан. В последнее время ведущий производитель процессоров компания Intel анонсировала создание специализированных систем с общей памятью, использующих процессоры с большим количеством ядер (до 80) [26, 27], которые могут быть использованы для увеличения производительности существующих вычислительных комплексов. Аналогичный подход был применён при развитии вычислений общего назначения на видеоускорителях, например технология CUDA [28], однако в отличие от него технология Intel Manycore [29] использует традиционную x86 архитектуру процессора и набор инструкций. Последнее позволило спроектировать и разработать ряд высокоуровневых средств для решения пользовательских задач на параллельных вычислительных системах. Эти средства получили название Array Building Blocks и Thread Building Blocks [21].

Технология Array Building Blocks (ArBB) ориентирована на параллельное выполнение операций над массивами в языке C++ с использованием специализированной библиотеки и JIT-компиляции [30]. Такой подход позволяет при первоначальном компилировании преобразовывать операции над объектами ArBB в некоторую метаинформацию для библиотеки времени выполнения и JIT-компилятора ArBB, который в свою очередь непосредственно перед выполнением может сгенерировать последовательность исполняемых инструкций для конкретного типа используемого процессора, например с применением векторизации по технологиям AVX и SSE [31]. Согласно произведённым измерениям уменьшение времени вычисления подавляющего большинства задач перекрывает накладные расходы работы JIT-подсистемы технологии. ArBB предполагает работу с одно-, двух- и трёхмерными массивами с использованием как предопределённых, так и определяемых пользователем операций. Сущности библиотеки могут быть семантически представлены как векторы, матрицы и тензоры третьего порядка, в терминах которых может быть описана задача. Следовательно, данная технология с точки зрения пользователя осложняется тем же, что и ранее рассмотренный пакет Ouverture. ArBB позволяет генерировать поток векторных инструкций для систем с общей памятью, однако не поддерживает работу в системах с локальной памятью, что ограничивает её распространение. Использование векторизации позволяет избавится от чересчур мелкого уровня зернистости векторно-матричных операций, на которые ориентирована данная технология.

Более обобщённой библиотекой для работы с параллельными вычислениями от Intel является Thread Building Blocks (TBB) [32]. Данная библиотека предоставляет программный интерфейс с активным использованием шаблонов языка C++ и задействует дополнительный планировщик в разделяемой библиотеке времени выполнения для обеспечения эффективного распараллеливания. Отличительной особенностью данной библиотеки является высокий уровень абстракции: пользователь описывает в задачу в терминах некоторого набора действий, а также принципы разделения на подзадачи и объединения результатов вычислений. При этом отсутствует необходимость явного описания параллельных вычислений. Во время работы программы библиотека времени выполнения автоматически выполняет разбиение задачи на необходимое число подзадач, с учётом различных стратегий разбиения и планирования, обеспечивает параллельное вычисление результатов подзадач и их объединение в окончательное решение.

В процессе работы используется планирование по запросам (work-stealing scheduling) [33, 34] и множество очередей подзадач для каждого процессора. Такой подход достаточно эффективен для задач общего вида и особенно эффективен для задач обладающих естественным параллелизмом, т. е. независимых по данным [21].

Библиотека TBB использует при планировании вычислений особенности архитектур вы-

числительных систем с общей памятью, и, следовательно, не может быть легко адаптирована для работы в распределённых системах, однако концептуальное описание произвольных задач может быть обобщено для их поддержки. Также стратегии разделения задач в ТВВ не предполагают контроля зернистости за исключением определения минимальной неделимой подзадачи пользователем. Данная библиотека предполагает изначальное описание задачи в своих абстракциях, но вместе с тем не препятствует использованию готовых библиотек для решения подзадач [35].

Описание задач в терминах абстрактных операций

Для достижения поставленной выше цели варьирования зернистости задач в зависимости от оптимальной для используемого аппаратного обеспечения, предлагается описывать вычислительные задачи в терминах некоторых абстрактных операций над ними, так как использование примитивов параллельного программирования напрямую приводит к явному выбору зернистости задачи в процессе программирования.

Задача, вычислимая ЭВМ, может быть представлена в виде некоторой последовательности операций $\langle t_i \rangle$, выполнение которых позволяет получить требуемый результат. Многие алгоритмы не являются строго последовательными в силу того, что они могут быть разбиты на ряд независимых последовательностей операций, или представлены в виде графа задачи $G = (T_i, E)$, где узлы T_i – последовательности операций, а ребра E – зависимости между ними. При распараллеливании граф представляется в ярусно-параллельной форме, где каждый ярус L_k содержит множество независимых между собой последовательностей операций. В дальнейшем будем рассматривать множества операций в рамках одного яруса или задачу, которая может быть представлена в таком виде. Данный подход может быть применен без потери общности к произвольным представленным в виде графа задачам путём применения известных алгоритмов сопоставления графа задачи графу вычислительной системы.

Пусть задача T_0 описывается некоторым набором операцией, подлежащих выполнению. Определим оператор разбиения $split\ T \to \{T_i\}$, выполняющий разбиение задачи на подзадачи с непересекающимися, т.е. независимыми между собой, операциями. Обозначим их подзадачами первого порядка $T_1^{\langle i \rangle}$. Используя структурную рекурсию определим данный оператор таким образом, что применение его к подзадаче i-го порядка приводит к получению множества подзадач $(i+1)$-го порядка. Для сохранения структурности будем считать исходную задачу подзадачей 0-го порядка. Обозначим критерий остановки рекурсии

$$split\ T_i^{\langle a,b,\ldots,k \rangle} = \{T_{(i+1)}^{\langle a,b,\ldots,k,0 \rangle}\},$$

$$card\ \{T_{(i+1)}^{\langle a,b,\ldots,k,0 \rangle}\} = 1.$$

Разбиение подзадачи, приводящее к получению вырожденного множества (кардинальное число, соответствующее количеству элементов, равно единице) приводит к остановке рекурсии, т.к. дальнейшее разбиение невозможно.

Введём понятие пути разбиения $\langle a,b,\ldots,k \rangle$, который демонстрирует, что данная подзадача была получена из a-той подзадачи 1-го порядка, b-той подзадачи 2-го порядка и т.д.

Для непосредственного выполнения последовательности вычислительных операций введем оператор $compute\ T_i^{\langle a,b,\ldots,k \rangle} \to R_i^{\langle a,b,\ldots,k \rangle}$, выполняющий фактически преобразование из подзадачи i-го порядка, в частичный результат того же порядка. Этот оператор сохраняет значения порядка и пути разбиения.

Окончательный результат может быть получен путём комбинирования частичных результатов, для которого вводится оператор $merge\ \{R_{(i+1)}^{\langle a,b,\ldots,k,p \rangle}\} \to R_i^{\langle a,b,\ldots,k \rangle}$, выполняющий корректное слияние всех частичных результатов $(i+1)$-го порядка в частичный результат i-го порядка. Введение ограничения на слияние в терминах равенства путей разбиения $\langle a,b,\ldots,k \rangle$ и необходимости использования всех частичных результатов, позволяет сделать данный оператор неассоциативным.

Принцип построения системы варьирования зернистости

Пользователь описывает задачу, подлежащую решению, как последовательность кортежей

$$\langle (split, compute, merge, T_0, :: T, :: R) \rangle,$$

где *split* – оператор декомпозиции на подзадачи, *merge* – оператор композиции частичных результатов, *compute* – оператор вычисления, описанный так, что может быть применён к подзадаче произвольного порядка, T_0 – начальная задача, $::T$ и $::R$ – типы задачи и результата соответственно, в случае использования типизированной абстракции, в частности – языков программирования; элементы последовательности представляют собой ярусы графового представления задачи.

Непосредственно вычисление сводится к выполнению для каждого элемента последовательности следующих действий.

1. Разбиение выполняется до тех пор, пока не будет получена подзадача, удовлетворяющая требованиям относительно зернистости. Разбиение может производиться до различной глубины в случае неравномерного деления задачи. В случае же чрезмерного мелкого разделения подзадач, некоторые из них могут быть сгруппированы для выполнения на одном ресурсе.
2. Непосредственное выполнение подзадач или групп подзадач.
3. Комбинирование частичных результатов до получения результата исходной задачи. Комбинирование целесообразно выполнять на одном из ресурсов, которые использовались при вычислении для уменьшения накладных расходов на пересылки данных.

Данный подход позволяет достаточно эффективно представлять, в частности, так называемые задачи с параллелизмом по данным. Например, векторно-матричные операции, обладающие естественным параллелизмом, предполагающим мелкое зерно распараллеливания, могут использовать в качестве оператора разделения простейшее разбиение на n равных частей путём изменения индексов, а в качестве оператора слияния – объединение последовательных блоков.

Выводы

На сегодняшний день существует ряд технологий, призванных упростить программирование параллельных систем для решения научных задач путём избавления пользователя от необходимости явно учитывать особенности различных архитектур таких систем. Большинство существующих средств ориентированы на системы с общей памятью, в то время как наиболее распространёнными и наиболее производительными системами являются распределённые и системы с локальной памятью. Для последних существует ряд программных комплексов, обеспечивающих параллельное решение определённого класса задач с использованием существующих технологий.

Наиболее удобным подходом к описанию вычислительных задач является перечисление данных и действий, которые необходимо выполнить над ними, наряду с принципами разбиения их на независимые в вычислениях подзадачи и объединения частичных результатов. Такой подход может незначительно ограничить круг решаемых задач или потребовать их адаптации, однако позволяет абстрагировать архитектуру конкретной вычислительной системы и обеспечить параллельное выполнение на произвольной системе.

Эффективность таких вычислений, однако, будет существенно зависеть от оптимальности выбора зерна распараллеливания. Реализация вышеописанных принципов в виде библиотеки поддержки параллельного программирования для распределённых систем должна обязательно учитывать возможность варьирования зернистости задачи. Субоптимальные значения могут быть получены для различных классов задач на основе статистического, эволюционного или онтологического подхода.

Список литературы

1. Moore G. E. Cramming more components onto integrated circuits [Текст] // Electronics. – 1965. – Т. 38, No 8.
2. Decision no 1982/2006/ec of the European Parliament and of the Council of 18 December 2006 concerning the Seventh Framework Programme of the European Community for research, technological development and demonstration activities (2007-2013) [Text] // Official Journal of the Europian Union. – 2006. – Vol. 149. – Pp. 1–41.
3. Постанова Кабінету Міністрів України № 1020 «Про затвердження Державної цільової науково-технічної програми впровадження і застосування грід-технологій на 2009–2013 роки.» – 2009.
4. Geoffray P., Pham C., Tourancheau B. A software suite for high-performance communications on clusters of SMPs [Text] // Cluster Computing. – 2002. – Vol. 5, no. 4. – Pp. 353–363. – ISSN 1386-7857.

5. The grid: blueprint for a new computing infrastructure [Text] / Ed. by Ian Foster, Carl Kesselman. – San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. – ISBN 1-55860-475-8.
6. Sutter H. The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software [Text] // Dr. Dobb's Journal. – 2005. – Vol. 30, no. 3.
7. Sutter H., Larus J. Software and the concurrency revolution [Text] // Queue. – 2005. – Vol. 3, no. 7. – Pp. 54–62. – ISSN 1542-7730.
8. Amdahl Gene M. Validity of the single processor approach to achieving large scale computing capabilities [Text] // Proceedings of the April 18-20, 1967, spring joint computer conference. – AFIPS '67 (Spring). – New York, NY, USA: ACM, 1967. – Pp. 483–485.
9. Software as a service for data scientists [Text] / Bryce Allen, John Bresnahan, Lisa Childers et al. // Commun. ACM. – 2012. – Vol. 55, no. 2. – Pp. 81–88. – ISSN 0001-0782.
10. Armstrong Joe. Programming Erlang: Software for a Concurrent World [Text]. – Pragmatic Bookshelf, 2007. – ISBN 193435600X, 9781934356005.
11. Armstrong Joe. A history of Erlang [Text] // Proceedings of the third ACM SIGPLAN conference on History of programming languages. – HOPL III. – New York, NY, USA: ACM, 2007. – Pp. 6–1–6–26.
12. International Organization for Standartization (ISO). ISO/IEC 14883, Standard for Programming Language C++: Tech. rep.: 2011.
13. Code Gen Options – Using the GNU Compiler Collection (GCC) [Electronic resource]. – Access mode: http://gcc.gnu.org/onlinedocs/gcc-4.6.2/gcc/Code-Gen-Options.html. – Last access: 15.10.2011. – Title from the screen.
14. OpenMP 3.0 Complete Specification [Electronic resource]. – Access mode: http://www.openmp.org/mp-documents/spec30.pdf. – Last access: 04.10.2010. – Title from the screen
15. Board OpenMP Architecture Review. OpenMP Application Program Interface Version 3.1 [Text]. – 2011.
16. Lattner Chris. Llvm and Clang: Advancing compiler technology [Text] // Proceedings of FOSDEM 2011: Free and Open Source Developers European Meeting. – 2011. – Pp. 234–239.
17. Lattner Chris. Llvm and Clang: Next generation compiler technology [Text] // Proceedings of BSD conference. – 2008. – Pp. 63–67.
18. Novillo D. Openmp and automatic parallelization in GCC [Text] // Proceedings of the GCC developers' summit. – 2006. – Pp. 132–141.
19. Marowka Ami. Performance of openmp benchmarks on multicore processors [Text] // Proceedings of the 8th international conference on Algorithms and Architectures for Parallel Processing. – ICA3PP '08. – Berlin, Heidelberg: Springer-Verlag, 2008. – Pp. 208–219.
20. Aiken A., Nicolau A. Optimal loop parallelization [Text] // SIGPLAN Not. – 1988. – Vol. 23, no. 7. – Pp. 308–317. – ISSN 0362-1340.
21. Засоби паралельного програмування [Текст] / С.Г. Стіренко, Д.В. Грибенко, О.І. Зіненко, Михайленко А.В. – К., 2011. – 154 с.
22. Центр суперкомп'ютерних обчислень НТУУ «КПІ» [Електронний ресурс]. – Режим доступу: http://hpcc.org.ua/. – Останнє звернення: 15.04.2011. – Назва з екрану.
23. Anderson E., Bai Z., Bischof C. LAPACK Users' Guide [Text]. – 1992.
24. Brown David, Henshaw William, Quinlan Daniel. Overture: An object-oriented framework for solving partial differential equations [Text] // Scientific Computing in Object-Oriented Parallel Environments / Ed. by Yutaka Ishikawa, Rodney Oldehoeft, John Reynders, Marydell Tholburn. – Springer Berlin / Heidelberg, 1997. – Vol. 1343 of Lecture Notes in Computer Science. – Pp. 177–184. – ISBN 978-3-540-63827-8.
25. MPI: A message-passing interface Standard, version 2.2 [Text]. – Stuttgart, Germany: High Performance Computing Center Stuttgart (HLRS), 2009. – 648 pp.
26. Larrabee: a many-core x86 architecture for visual computing [Text] / Larry Seiler, Doug Carmean, Eric Sprangle et al. // ACM Trans. Graph. – 2008. – Vol. 27, no. 3. – Pp. 18:1–18:15. – ISSN 0730-0301.
27. Evaluation and improvements of programming models for the intel scc many-core processor [Text] / C. Clauss, S. Lankes, P. Reble, T. Bemmerl // Proceedings of International conference on High Performance Computing and Simulation (HPCS), 2011. – 2011. – Pp. 525–532.
28. Kirk D. Nvidia CUDA software and gpu parallel computing architecture: Tech. rep.: 2007.
29. Intel Manycore Testing Lab [Electronic resource]. – Access mode: http://software.intel.com/en-us/articles/intel-many-core-testing-lab/. – Last access: 15.04.2011. – Title from the screen.
30. Aycock John. A brief history of just-in-time [Text] // ACM Comput. Surv. – 2003. – Vol. 35, no. 2. – Pp. 97–113. – ISSN 0360-0300.

31. Intel SSE4 Programming Reference D91561-001 2007 [Electronic resource]. – Access mode: http://softwarecommunity.intel.com/isn/Downloads/IntelSSE4ProgrammingReference.pdf. – Last access: 08.10.2010. – Title from the screen.
32. Intel thread building blocks documentation [Electronic resource]. – Access mode: http://threadingbuildingblocks.org/documentation.php. – Last access: 08.05.2012. – Title from the screen.
33. Slaw: a scalable locality-aware adaptive work-stealing scheduler for multi-core systems [Text] / Yi Guo, Jisheng Zhao, Vincent Cave, Vivek Sarkar // SIGPLAN Not. – 2010. – jan. – Vol. 45, no. 5. – ISSN 0362-1340.
34. Blumofe Robert D., Leiserson Charles E. Scheduling multithreaded computations by work stealing [Text] // J. ACM. – 1999. – sep. – Vol. 46, no. 5. – Pp. 720–748. – ISSN 0004-5411.
35. Bws: balanced work stealing for time-sharing multicores [Text] / Xiaoning Ding, Kaibo Wang, Phillip B. Gibbons, Xiaodong Zhang // Proceedings of the 7th ACM european conference on Computer Systems. – EuroSys '12. – New York, NY, USA: ACM, 2012. – Pp. 365–378. http://doi.acm.org/10.1145/2168836.2168873.

УДК 004.75

АМОНС О.А.,
ХМЕЛЮК В.С.,
ЧАЛИЙ О.І.,
КИРИЧЕК О.О.

АТРИБУТИВНИЙ ПОШУК ДОКУМЕНТІВ В РОЗПОДІЛЕНИХ СИСТЕМАХ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

Робота присвячена огляду та аналізу алгоритмів пошуку документів в розподілених системах. Порівнюються алгоритми пошуку. Запропоновано модифікувати вибрані алгоритми пошуку та порівняння запитів на пошук документів. Пропонується схема програмної реалізації процесу атрибутивного пошуку документів в розподіленій системі електронного документообігу.

The paper is devoted to the review and analysis of algorithms for document retrieval in distributed systems. Search algorithms are compared. Modification to the selected algorithms for queries to find documents search and comparison is proposed. Software implementation design of process of attributed document retrieval in a distributed electronic document workflow system is suggested.

1. Вступ

У міру зростання об’єму накопичувачів і, відповідно, архівів документів, проблема ефективності пошуку інформації набуває все більш гострий характер, а падіння продуктивності ієрархічних файлових систем як засобу впорядкування стає все більш помітним. На передній план виходять такі проблеми каталогізації, як необхідність дисциплінованого підходу до іменування і розміщення файлів, вдумливому складання назв, а також значні трудовитрати по розбору вже існуючого хаосу. Швидкий розвиток мережевих технологій потребує постійного покращення рівня організації комунікації. Для впорядкування комунікаційних структур використовують розподілені системи, які керують множиною об’єднаннях в мережу комп’ютерів представляючи їх єдиним цілим. Розподілені системи мають широкий спектр властивостей: від можливості маніпулювання інформаційними потоками між віддаленими точками на планеті, до контролю кластерів відносно недалеко розташованих машин. На даний момент доступні централізовані та децентралізовані розподілені системи. Вибір типу системи головним чином залежить від кількості машин в мережі та об’єму оброблюваної інформації. Головною перевагою децентралізованих розподілених систем перед системами, що мають явно виражений центральний вузол – відмовостійкість. При чому, з розвитком мережевої інфраструктури зростає важливість цієї властивості, цим самим стимулюючи активний розвиток даного класу розподілених систем. Система повинна створювати транзитивне замикання інформаційного простору, потрібно переходити від автоматизованого пошуку інформації і від автоматизованого виконання бізнес-процесів до автоматичного. Користувач системи повинен тільки поставити завдання і отримувати результат, не беручи участі у проміжних етапах пошуку, оцінки та аналізу інформації. Головним завданням такої системи є управлінням інформацією.

Процес пошуку інформації є невід’ємною частиною функціонуванні будь-якої системи. Не є винятком і клас розподілених систем. Саме завдяки ефективним алгоритмам пошуку знаходять потрібну інформацію, що може знаходиться на N-вузлах у вигляді k-частин. При використанні модифікованих алгоритмів пошуку можна досягти збільшення продуктивності процесу пошуку на 15-18%, зменшити навантаження мережі зайвими запитами на 15%. Також на 30% збільшується швидкість реагування на запит, що відіграє важливу роль у функціонуванні розподілених систем [1]. Однак для організації продуктивного пошуку інформації необхідно підбирати алгоритми пошуку в залежності від конкретної розподіленої системи.

В даній статті пропонується використання модифікованого методу пошуку документів в розподіленій системі електронного документообігу (СЕДО), який базується на основних засадах алгоритму пошуку в ширину з ітеративним заглибленням та алгоритму ISM.

Враховуючи специфічні особливості формування документів в СЕДО та розмежування прав на доступ будемо використовувати поряд з

модифікаціями вище зазначеного алгоритму ще й статистичну інформацію.

Сформований запит буде порівнюватися з запитами, що вже виконувалися на вузлі. Та в залежності від того до якого запиту він буде подібний, буде формуватися множина вузлів, до яких направлятиметься запит. Величина глибини слідування запиту задаватиметься користувачем.

Для зменшення часу пошуку документів та вдосконаленню роботи з документами необхідно адаптувати систему пошуку документів до СЕДО.

2. Актуальність проблеми

Новий рівень розвитку мережевого інформаційного простору обумовлює потребу створення і розвитку прогресивних моделей інформаційного простору, інформаційних потоків, мережевого пошуку. В зв'язку з цим виникає інтерес до підходів, що базуються на розумінні інформації, як деякої міри впорядкованості деякої системи і, відповідно, до статистичних методів її обробки. Таким чином, актуальність проблеми обумовлюється потребою створення алгоритму пошуку документів для системи електронного документообігу, який дозволятиме якомога ефективніше виконувати пошук документів в розподіленій системі електронного документообігу.

3. Огляд існуючих алгоритмів пошуку

На сьогоднішній день існує велика кількість різноманітних алгоритмів пошуку інформації в розподілених системах. Продуктивність системи залежить від правильної побудови її функціонування, тобто потрібно вибрати такий алгоритму, який буде максимально підходити до структури системи. Не є винятком і процеси пошуку. Розглянемо деякі алгоритми пошуку в розподілених системах.

Алгоритм пошуку в ширину (BFS, Breadth-first search) – метод обходу та розмітки вершин графу. Вся система представляється у вигляді графу, де вузол це окрема машина. Пошук в ширину виконується в такому порядку: вершині, з якої починається обхід приписується номер 1, сусіднім вершинам – 2 і т. д. Потім по черзі розглядуються всі суміжні вершини.

Вузол-ініціатор генерує запит який адресується всім ближчим сусідам. Коли вузол в мережі отримує запит на пошук то виконується пошук в його локальному індексу і повинен обов'язково генерує запит-відповідь щоб повернути результат. Якщо будь-який вузол, який оброблював запит отримує запита-відповіді більш ніж від одного вузла, то він може виконувати операцію завантаження інформацію з найбільш доступного ресурсу.

Недоліком використання даного алгоритму є надмірне перенавантаження мережі зайвими запитами і як результат вузол з найменшою пропускною здатністю може спричинити зменшення швидкодії мережі, а перевагою являється те, що збільшується ймовірність отримання позитивної відповіді при перегляді значної частини мережі. [2,3].

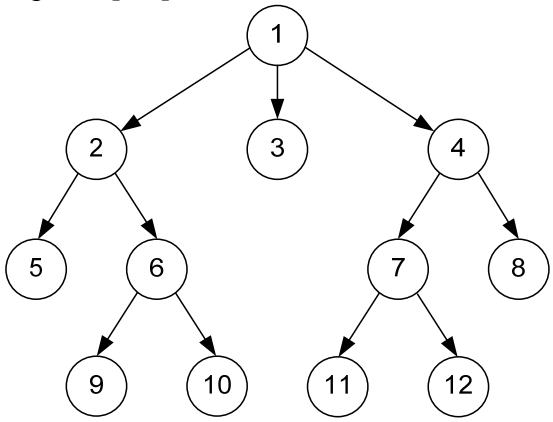


Рис. 1. Порядок обходу дерева в ширину

Алгоритм пошуку в глибину (DFS, Depth-first search) – основна ідея алгоритму полягає у тому, що для кожної не пройденої вершини необхідно знайти всі не пройдені суміжні вершини і повторити пошук для них [2,3].

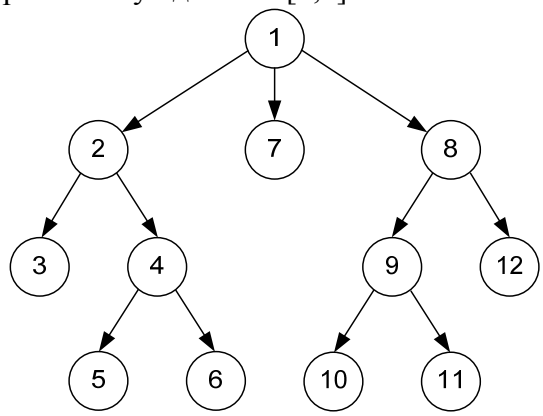


Рис. 2. Порядок обходу дерева в глибину

Алгоритм випадкового пошуку в глибину (RBFS, Random Breadth First Search) – модифікація алгоритму пошуку в ширину, суть якого полягає в тому, що вузол-ініціатор пересилає запит тільки випадково вибраним вузлам в мережі. Перевага даного алгоритму полягає в тому, що не потрібно знати глобальну інформа-

цію про стан контенту мережі, вузол може отримувати локальні рішення так швидко, як це необхідно. Недоліком являється той факт, що даний алгоритм є ймовірнісним, тому деякі великі сегменти мережі можуть стати недоступними [2, 3].

Алгоритм пошуку A («Астар»)* – це алгоритм пошуку по першому найкращому співпадінні на графі, який знаходить маршрут з найменшою метрикою від однієї вершини (початкової) до іншої (кінцевої). Порядок обходу вершин визначається евристичною функцією «відстань + метрика», f(x)=g(x) + h(x). Суть цього алгоритму заключається в тому, що A* крок за кроком розглядає всі шляхи від початкової вершини до кінцевої, доки не знайде мінімальний. Спочатку він розглядає ті маршрути, які «здаються» ведуть до кінцевої вершини, а при виборі вершини – враховує весь пройдений до неї шлях. A* розглядає вузли суміжні з початковим (ініціатором) таким чином, щоб метрика була найменшою, потім він розкривається. На кожному етапі алгоритм оперує множиною шляхів із початкової точки до всіх ще не розкритих вершин графу, яка знаходиться в пріоритетній черзі. Пріоритет визначається як f(x)=g(x) + h(x). Ця дія виконується доки значення f(x) кінцевої функції вершини не буде меншим, чим будь-яке значення в пріоритетній черзі або доки все дерево не буде переглянутим [4].

Інтелектуальний пошуковий механізм (ISM, Intelligent Search Mechanism) – даний алгоритм пошуку використовується в мережах де кожен вузол може зберігати деяку спеціалізовану інформацію. Суть даного метода полягає в тому, що вузол-ініціатор генерує повідомлення, що описує запит на пошук, знаходить множину найбільш підходящих вузлів, що можуть повернути результат, використовуючи профайлер та механізм ранжування, та надсилає запит на пошук лише цим вузлам. Профайлер – файл, який зберігає інформацію про останні запити в стек в форматі: Запит, Вузол. Коли стек заповнюється відбувається заміна останнього запита, який не використовуюся найдовше. Ранжування базується на понятті рангу релевантності. Враховуючи ранг релевантності вузол формує «представлення» про сусідні вузли, вибираються вузли до яких запит буде надходити першим. Ранг релевонтності визначає найбільший ранг вузла, який повертає найбільше результатів. Крім того в даному методі використовується параметр α, за допомогою якого можна збільшити вагу запитів найбільш подібних початковому. Таким чином можна сказати, що якість процесу пошуку за допомогою даного алгоритму залежить від сусідніх вузлів до вузла, який задає запит. Головна проблема даного алгоритму полягає тому, що запити можуть зациклюватися і не досягати деяких частин в мережі. На даний момент існують модифікації даного алгоритму, що дозволяють позбутися цієї проблеми [3].

Алгоритм «напрямлений пошук в ширину та евристика найбільшої кількості результатів у минулому» (>RES) – суть даного методу полягає в тому, що кожний вузол пересилає запит підмножині своїх вузлі, які будуються на основі деякої загальної статистики. Запит в методі вважається задовільним якщо видається z результатів (z – деяка константа). Даний алгоритм дуже схожий до ISM, вузол-ініціатор пересилає запити до вузлів, що повернули найбільші результати для останніх запитів. Таким чином даний алгоритм в результаті експериментів змінювався від пошуку в ширину до первинного пошуку в глибину. Також варто зауважити, що алгоритм >RES використовує більш простішу інформацію про вузли. Недоліком являється те, що даний алгоритм не аналізує параметри вузла, зміст яких пов'язаний з запитом. Перевагою даного алгоритму являється те, що він направляє запити в великі сегменти мережі, а також захватує сусідні вузли, які менш навантажені [3].

Алгоритм «довільних блукань» (RWA, Random Walkers, Algorithm) – суть даного алгоритму полягає в тому, що кожний вузол випадковим чином пересилає повідомлення з запитом одному із сусідніх вузлів і готовий знову приймати інші повідомлення. Даний алгоритм схожий до *RBFS* , але в *RBFS* кожний вузол пересилає запит тільки частині сусідів. До того ж в *RBFS* відбувається експоненціальне збільшення повідомлень, що пересилаються в мережі, коли в RWA – лінійне. RBFS та RWA не використовують ніяких правил щодо вибору найбільш релевантного вузла для пересилки пошукового запиту. Ще однією методикою, подібної RWA, є «адаптивний імовірнісний пошук» (Adaptive Probabilistic Search, APS) [5]. У APS кожен вузол розгортає на своїх ресурсах локальний індекс, що містить значення умовних ймовірностей для кожного сусіда, який може бути вибраний для обробки наступного запиту. Головна відмінність від RWA в даному випадку – це те, що в APS вузол використовує в якості зворотнього зв'язку результати попередніх пошуків (у вигляді умовних ймовірностей) замість повністю випадкових переходів. Тому метод APS часто дає кращі результати, ніж RWA [3, 5].

Табл. 1. Порівняння алгоритмів пошуку

Алгоритм	Переваги	Недоліки
Пошук в глибину	Швидко відбувається процес пошуку, якщо обрана правильна гілка пошукового графу, не велике навантаження на мережу, висока швидкість роботи	Потрібно вибрати оптимальну глибину зупинки пошуку, не висока релевантність результатів, висока ймовірність випадання запиту
Пошук в ширину	Збільшується ймовірність отримання позитивної відповіді при перегляді значної частини мережі, не високе споживання ресурсів	Надмірне перенавантаження мережі зайвими запитами. Пошук проходить в усіх напрямках.
ISM	Пошук проводиться на релевантних вузлах. Ведення статистики у вигляді профайлера. Можливість змінювати ранг релевантності вузла в залежності від запиту. Збільшення швидкості та ефективності процесу пошуку за рахунок зменшення кількості повідомлення між вузлами та кількості опитуваних вузлів	Існує ймовірність зациклювання запита в мережі, завжди існує велика частина мережі до якої не доходять запити. Якість пошуку залежить від сусідніх вузлів
RBFS	Не потрібно знати глобальну інформацію про стан контенту мережі, вузол може отримувати локальні рішення так швидко, як це необхідно.	Даний алгоритм є ймовірнісним, тому деякі великі сегменти мережі можуть стати недоступними.
Алгоритм пошуку A*	Завжди знайде оптимальний шлях до цілі. При правильному налаштуванні дає гарні результати в швидкодії.	Складність алгоритму і час пошуку залежить від евристичної моделі мережі.
>RES	Огляд великої кількості сегментів мережі. Використання простої статистичної інформації для вибору вузлів	Відсутність можливості аналізувати вузли, які зберігають інформацію, що пов'язана із запитом
RWA	Запит пересилається тільки частині сусідніх вузлів. Збільшення кількості запитів в мережі відбувається за лінійним законом	Вибір вузлів відбувається випадковим чином, без яких-небудь правил. Переглядається невелика кількість сегментів мережі. Перенавантаження мережі запитами. Невисока релевантність результатів

Таким чином, проаналізувавши все вище сказане, можна сказати, що всі розглянуті алгоритми пошуку дозволяють вирішити лише деякі проблеми, тобто на даний момент не існує такого пошукового алгоритму, який був би чимось універсальним, широковживаним і дозволив би проводити процес пошуку максимально ефективно, незалежно від структури системи. Припускається, що саме структура системи в цілому повинна «підказувати» розробникам, який саме алгоритм варто використати.

Наступним етапом виконано порівняння алгоритмів пошуку – це виділення основних параметрів, по яких порівнюємо та зведення все в одну таблицю (табл. 2).

Для всіх параметрів порівняння алгоритмів пошуку використовували наступну міру: <0; 0.6; 1>. Тобто, діапазон «0 – 0.6» – відповідає позначці «-», а діапазон «0.6 – 1» – відповідає позначці «+» в табл. 2.

Формули розрахунків параметрів наведені далі в описі математичної моделі.

В табл. 2 зібрано характеристики розглянутих алгоритмів. Ці алгоритми можна згрупувати наступним чином:

- ISM, >RES, A* – характеризуються як найбільш стійкі та продуктивні алгоритми серед усіх розглянутих, але при роботі споживають багато ресурсів системи;
- RWA та RBFS – менш продуктивні, існує велика ймовірність отримання не релевантних результатів, але швидкість роботи на досить високому рівні та не велике споживання системних ресурсів;
- Пошук в глибину, пошук в ширину – ці алгоритми дещо схожі між собою, але все-таки мають ряд відмінних властивостей, які показують ефективність того чи іншого алгоритму в різних умовах.

Керуючись цими факторами, властивостями алгоритмів із Таблиці 2 та обмеженнями СЕДО ми вирішили використовувати деякі властивості алгоритму пошуку в глибину з ітеративним заглибленням та алгоритму ISM.

Табл. 2. Порівняння алгоритмів пошуку

Алгоритм	Синтаксичний аналіз запиту	Використання статистичної інформації	Релевантність результатів	Високе навантаження на мережу	Висока ймовірність випадіння запиту	Висока швидкість роботи	Велике споживання ресурсів
Пошук в глибину	-	-	-	-	-	+	+
Пошук в ширину	-	-	+	+	+	-	-
ISM	+	+	+	-	-	+	+
RWA	-	-	-	-	-	+	-
>RES	+	+	-	-	-	+	+
A*	+	-	+	+	-	+	+
RBFS	-	-	-	-	+	-	-

4. Модифікований алгоритм пошуку документів

В розроблюваній підсистемі атрибутивного пошуку документів, використовується алгоритм пошуку в ширину із ітеративним заглибленням. Величина заглиблення встановлюється на вибір користувача. Суть даного методу полягає у тому, що вузол-ініціатор запиту виконує пошук у себе і потім передає запит одному із сусідів. Вузол, який отримав запит виконує пошук у себе на вузлі і передає запит своїм сусідам на величину d, вказану вузлом-інціатором і повертає потік знайдених результатів. Далі вузол-ініціатор передає запит іншому сусідові який виконує аналогічні дії як і всі інші вузли-сусіди вузла-ініціатора.

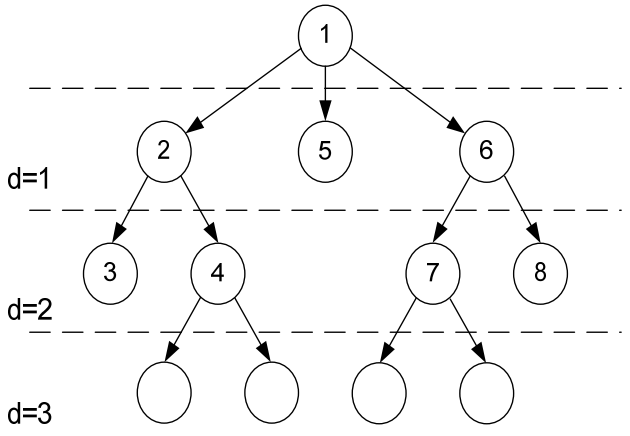


Рис. 3. Порядок обходу дерева в ширину з ітеративним заглибленням (d=2)

Пропонується модифікувати цей алгоритм шляхом використання статистичної інформації. Суть модифікації полягає в тому, що при кожному виконанні запиту на будь-якому вузлі зберігається інформація про структуру самого запиту та про відповідь, яка була відправлена. При надходженні нового запиту на вузол, спо чатку аналізується статистичні дані вузла, а сам запит «заморожується». Якщо новий запит буде схожим на будь-який запит, що вже виконувався на цьому вузлі, то буде сформовано «корисну множину» вузлів. До такої множини включатимуться ті вузли, які повертали відповідь на найбільш схожі запити. Пошук буде виконуватися вже не по всіх сусідах вузла, а тільки по тим, які «можуть» містити потрібну інформацію. Таким чином кожний вузол системи формує представлення про інформацію, яка зберігається на ньому. Схожість оцінюється алгоритмами наведеними нижче.

Розглянемо процес порівняння запитів. Розробимо алгоритм визначення подібності двох запитів, для цього весь процес порівняння розіб'ємо на два етапи: на першому етапі визначимо чи є один запит підмножиною іншого, а на другому на скільки обидва запити подібні.

5. Алгоритм визначення, чи є один запит підмножиною іншого

Якщо кожний елемент множини A належить і множині B то можна говорити, що множина A є підмножиною B, тобто:

$$A \subseteq B \qquad (5.1)$$

Виходячи з цього поняття, сформуємо словник всіх атрибутів D=<індекс, атрибут>.

Використовуючи цей еталонний словник D формуємо дві множини A та B таким чином, щоб множина A включала всі індекси які відпо-

відають використаним атрибутам у першому запиті, а множина B – у другому.

Виділимо такі етапи:

1) Формуємо дві множини, множина A – всі атрибути першого запиту, множина B – всі атрибути другого запиту.
2) Визначаємо потужності сформованих множин m(A) та m(B).
3) Порівнюємо знайдені потужності, таким чином визначаючи яка множина містить більше елементів. Якщо m(A)=m(B) то переходимо до пункту (6).
4) Виконуємо операцію перетину множин A і B. Як результат отримуємо множину спільних елементів.
$$S = A \cap B \qquad (5.2)$$
5) Визначаємо підмножини a=S на множині A, b=S на множині B.
6) Підраховуємо коефіцієнт подібності множин (запитів) K_s. Якщо хоч одна із множин A\S та B\S не є пустою, то припиняється процес порівняння запитів, оскільки робиться припущення, що два порівнюваних запити не є схожими. А якщо A\S та B\S пусті, то припускається, що два порівнювані запити максимально подібні по структурі. Якщо одна з множин є пустою, а інша – ні, то для продовження процесу порівняння потрібно щоб коефіцієнт подібності Ks був більше 0,6.
$$K_S = \frac{m(S)}{m(L)} \qquad (5.3)$$
m(S) – потужність множини S
m(L) – потужність не пустої множини серед двох порівнюваних множин A та B.
7) Виконуємо операцію сортування по індексам в кожній множині. Встановляємо взаємно однозначну відповідність між підмножинами a та b, або між множинами A та B, якщо m(A)=m(B).
8) Порівнюємо попарно всі елементи.

6. Алгоритм визначення чи є один запит подібний іншому (попарного порівняння елементів)

Атрибути документу можна умовно розділити на ті, які користувач заповнює сам (наприклад, номер документу, короткий зміст і т.д.) та на атрибути які користувач вибирає із списку (наприклад, журнал, тип документу, тематика та ін.). В залежності від того, до якого типу належить вибраний атрибут будемо формувати дві множини атрибутів з різними ваговими коефіцієнтами. Керуючись тим, що співпадіння слів більш вагоме чим співпадіння літер виконаємо наступні дії. Атрибути документу розділимо умовно на дві множини: множина A_1, множина B_1. До множини B_1 належать атрибути, що мають ваговий коефіцієнт 2, тобто ті атрибути, які користувач вибирає із списку. А до множини A_1 належать всі інші атрибути, їх ваговий коефіцієнт – 1, ті, що користувач вводить самостійно. Таким чином процес аналізу подібності елементів можна розділити на два етапи. На першому етапі відбувається формування множин із задіяних атрибутів, визначення максимально можливого коефіцієнту N.
$$N = r * 100\% \qquad (6.1)$$
r – кількість атрибутів

На другому етапі відбувається безпосередньо порівняння елементів, по сформованим множинам. Оскільки в множині A_1 знаходяться атрибути, значення яких користувач вводить самостійно, то спочатку потрібно привести введений текст до канонічного виду і потім схожі атрибути порівнюються з використанням «алгоритму шинглів»[6] (або просто визначається перетин цих двох підмножин). Подібність введеної інформації визначається коефіцієнтом подібності, який дорівнює ваговому коефіцієнту (при використанні «алгоритму шинглів») або визначається як:
$$s = \frac{k}{n} * 100\% \qquad (6.2)$$
k – кількість однакових символів\слів
n – загальна кількість символів\слів

Значення коефіцієнту s перемножується на ваговий коефіцієнт множини A_1 і записується в загальний коефіцієнт K.

Множина B_1 складається із атрибутів, значення яких не вводяться користувачем, а вибираються із запропонованого списку. Таким чином, аналогічно і з визначенням коефіцієнту подібності на множині A_1 на множині B_1 виконуються ті ж самі операції:
$$s_1 = \frac{k_1}{n_1} * 100\% \qquad (6.3)$$
k_1 – кількість однакових слів
n_1 – загальна кількість слів

Коефіцієнт s_1 перемножується із ваговим коефіцієнтом множини B_1 і потім додається до загального коефіцієнту K. Аналогічно цей про-

цес повторюється для всіх атрибутів множини B_1. Таким чином при співпадінні елементів на множині $B_1(s_1>40\%)$ робимо висновок, що два порівнювані атрибути є елементами одного дерева.

Результуючий коефіцієнт подібності елементів визнається як:

$$M = \frac{K}{N} * 100\% \qquad (6.4)$$

K – глобальний ваговий коефіцієнт

N – максимально можливий ваговий коефіцієнт

M – коефіцієнт подібності

На основі проведених випробувань в пошукові системі СЕДО, рекомендується встановлювати граничне значення коефіцієнту подібності не менше чим 75%.

Але все-таки даний алгоритм має ряд як переваг, так і недоліків.

Недоліки:

- Складність реалізації механізму аналізу запитів;
- Пошук виконується по всім таблицям в базах даних вузла;
- При великому завантаженні одного із вузлів, значно підвищується час пошуку.
- Організація додаткових заходів щодо очищення статистичних даних на вузлах, які не використовувалися або використовувалися в незначній мірі.

Переваги:

- Зменшується навантаження на мережу при подібності запитів;
- Зменшується час пошуку в цілому;
- Дає можливість вибору вузлів пошуку.
- Потужна система аналізу запитів.

Використання цього модифікованого алгоритму породжує такі проблеми, як:

- Кожен вузол повинен вести свою статистику;
- Затрачується додатковий час на аналіз статистичних даних;
- Якщо при аналізі статистичних даних не знайдено схожих запитів з оброблюваним, то виконується новий пошук по всім таблицям і як результат час пошуку збільшується;
- Відбувається накопичення статистичної інформації.

7. Математична модель модифікованого алгоритму пошуку

За допомогою математичної моделі можна промоделювати як сам процес пошуку документів, так і роботу самої системи пошуку. Розгляд математичної моделі розпочнемо із визначення основних понять.

Отже, в загальному вигляді процес пошуку документів можна представити у вигляді впорядкованої та скінченної сукупності елементів (кортежу) IR=<D, Q, R(d,q)>, де D – множина документів на вузлі, а Q – множина представлення інформаційної потреби (запиту);

R(d,q) – функція ранжування:

- зв'язує документ d із множини D та запит q із множини Q;
- визначає порядок документів відповідно до запиту q.

Релевантний документ – документ, зміст якого максимально правильно задовольняє інформаційний запит, тобто семантична відповідність пошукового запиту та пошукового образу документа. Для визначення релевантності найчастіше використовують TF-IDF метод [7].

Пертинентність – відповідність отриманих в результаті пошуку документів інформаційним потребам користувача [7].

Для оцінювання релевантності результатів пошуку основуємося на загальній моделі [3], та доповнимо її оцінюванням подібності документів запиту. Отже, введемо наступні позначки

Табл. 3. Типи документів

Документи	Видані	Не видані
Релевантні	D_{gr}	D_{nr}
Не релевантні	D_{gnr}	D_{nn}

За допомогою табл. 3 розраховувати показники інформаційного пошуку варто наступним чином.

Коефіцієнт повноти (Recall):

$$R = \frac{D_{gr}}{D_{gr}+D_{nr}} \qquad (7.1)$$

Коефіцієнт точності (Precision):

$$P = \frac{D_{gr}}{D_{gr}+D_{gnr}} \qquad (7.2)$$

Коефіцієнт правильності (Accuracy):

$$A = \frac{D_{gr}+D_{nn}}{D_{gr}+D_{gnr}+D_{nr}+D_{nn}} \qquad (7.3)$$

Коефіцієнт помилки (Error) :

$$E = \frac{D_{gnr}+D_{nr}}{D_{gr}+D_{gnr}+D_{nr}+D_{nn}} \qquad (7.4)$$

Значення F-міри (F-measure):

$$F = \frac{2}{\frac{1}{P}+\frac{1}{R}} \quad (7.5)$$

Коефіцієнт середньої точності (Average precision):

$$ArgPrec = \frac{1}{k}\sum_{i=1}^{k} P(i) \quad (7.6)$$

k – кількість документів, релевантних деякому запиту.

i – номер релевантного запиту документа.

P(i) – точність i-ого релевантного документу.

Подібність документів запиту q на вузлі P_i

$$P_{sim}(P_i, q) = \sum Q_{sim}(q_j, q)^{\alpha} \quad (7.7)$$

P_i – і-тий вузол;

q – пошуковий запит;

q_j – відповідь від вузла P_i;

Q_{sim} – функція визначення подібності запитів;

α – коефіцієнт «підсилення» більш подібних запитів:

α > 0 – розглядається один сусід

α = 0 – розглядається К-сусідів

α = 1 – подібність по всім відповідям однакова.

Ваговий коефіцієнт в статистиці

$$R = \frac{R_d}{N_d} \quad (7.8)$$

Час потрібний на формування запиту (обробка вузлом візуальних елементів)

$$t_e = t_{eo} * n_e \quad (7.9)$$

t_{eo} –час, затрачуваний на обробку одного елементу

n_e – кількість заповнених елементів

8. Реалізація модифікованого алгоритму пошуку

Спочатку користувач заповнює поля пошуку, вибирає атрибути і при натисненні на кнопку «Почати пошук» відбувається відправлення даних на сервер застосувань, де відбувається формування словників вибраних елементів із надісланих даних. На основі цієї інформації відбувається безпосередньо формування запиту, його аналіз та розбиття на підзапити. На кожному вузлі відбувається порівняння виконуваного запиту із статистичною інформацією по вище описаному алгоритму.

Отже, якщо запит подібний будь-якому запиту, що вже виконувався на вузлі, формується список вузлів до яких буде перенаправлятися запит на пошук документів. Після виконання пошуку на вузлах через Менеджер БД виконується повернення набору даних до Серверу застосувань, де вже виконується обмеження прав доступу та об'єднання результатів Серверу застосувань надсилає агреговані результати до Форми пошуку, де користувач може у зручному вигляді переглядати знайдені результати та виконувати подальші маніпуляції із знайденими документами.

Але існує й інший сторона медалі. При обробці вузлом запита він «заморожується», тобто запит не розповсюджуються далі, залишаються в середині деякої множини вузлів. Заморожені запити отримують відповіді із потоку відповідей, що проходить через ці вузли, ініційованого «діючими» запитами. При ретельному виборі множини «заморожених» запитів, отримавши відповіді із потоку даних, збільшується достовірність формування правильних відповідей, та зменшується час протікання цього процесу, навіть при перенавантаженій системі. Робимо припущення, що якщо запити схожі, то і відповіді мають бути схожими в межах деякої метрики. Таким чином, коли в системі виникають два, а то й більше схожих запитів в межах деякої метрики, використовуючи модифікований метод пошуку значною мірою скорочується час виконання запитів у всій системи, зникають перенавантаження на вузлах системи, зменшується кількість зайвих повідомлень в мережі чим покращується ефективність роботи всієї системи.

Вузли підключаються до системи та покидають її динамічно, в будь-який час. Для цього в системі потрібно встановлення додаткового серверу S-i, який буде підтримувати цю динаміку. Його функції полягають у тому, що коли вузол підключається до системи, сервер визначає з якими іншими вузлами він буде обмінюватися даними, потім стартує протокол з'єднання. Сервер такого типу бере участь тільки у формуванні і зміні системи, а не для передачі і обробки запитів.

На рис. 4 представлено неповну діаграму класів розподіленої СЕДО. Основний клас, що описує документ – це DocumentReestration. Він містить основні атрибути та посилання на інші таблиці.

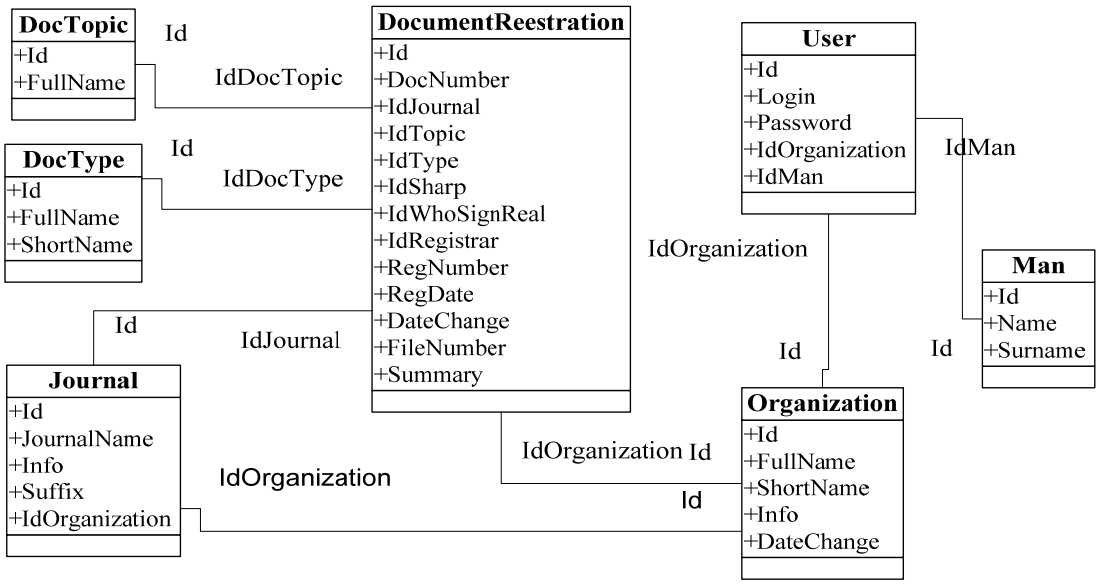


Рис. 4. Діаграма класів

9. Експериментальне дослідження

В процесі розробки системи, до отримання кінцевих результатів ефективності, ми отримали проміжні результати, які є не менш важливими для оцінки процесу пошуку та для майбутніх модифікацій. Таким чином, велику увагу було приділено процесу порівняння запитів: створеного запиту та запитів із статичної інформації кожного із вузлів. Тестування проводилося на 20 вузлах.

Також на рис. 5 графічно показано як змінювався час пошуку документів в залежності від кількості вузлів в мережі. А на рис. 6 показано взаємозв'язок між кількістю запитів, що генеруються у мережі від кількості вузлів даної мережі.

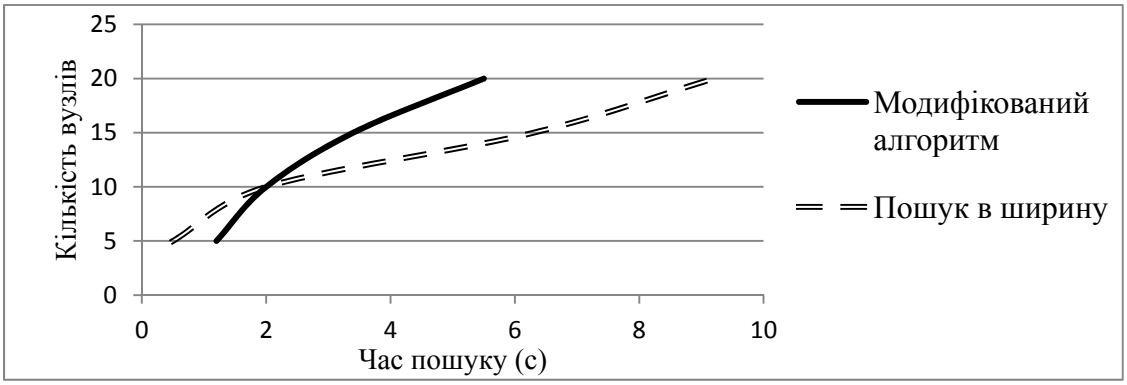


Рис.5. Залежність часу пошуку документів від кількості вузлів

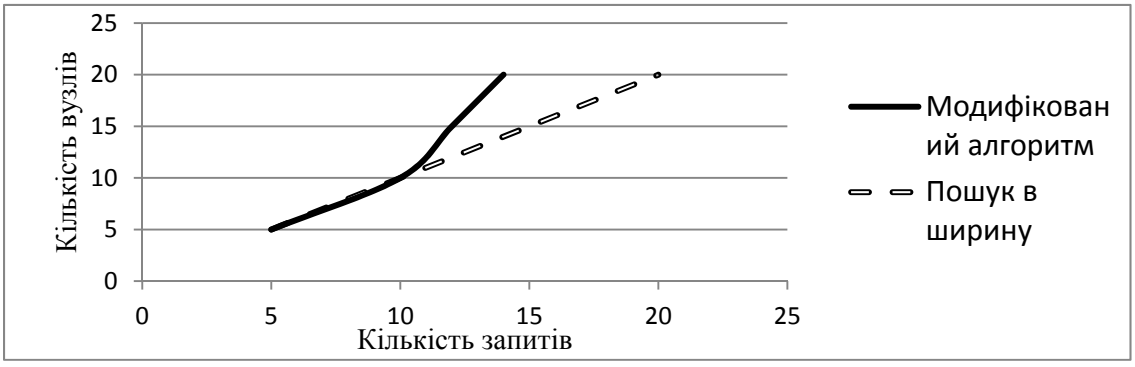


Рис.6. Залежність кількості запитів від кількості вузлів

Кількість запитів в мережі при використанні пошуку в ширину лінійно залежить від кількості сусідніх вузлів, а при використанні модифікованого методу пошуку кількість запитів зменшується завдяки розповсюдженню запитів тільки до тих вузлів, що можуть містити потрібну інформацію з високим рівнем ймовірності.

10. Висновок

Проаналізувавши найбільш популярні алгоритми пошуку документів в розподілених мережах, та враховуючи структуру СЕДО-КПІ [8], існуючу пошукову систему [9] та властивості платформи швидкої розробки застосувань SmartBase [8, 10] було запропоновано модифікувати існуючу систему шляхом адаптування та комбінування алгоритму ISM та алгоритму пошуку в ширину з ітеративним заглибленням.

Таким чином, запропонований модифікований метод пошуку документів в розподілених системах електронного документообігу дозволяє значно покращити процес пошуку документів. Покращилася продуктивність системи при збільшені навантаження як на всю систему, так і на окремі вузли. Такого результату вдалося досягти завдяки використанню «заморожування» запитів. Запит залишається в межах деякої множини вузлів. В цей час він виконується на самому вузлі та порівнюється з тими, що вже виконувалися для визначення корисної множини вузлів із всіх сусідів для подальшого перенаправлення. Також завдяки аналізу подібності запитів, який проводиться кожним вузлом окремо, скоротився час пошуку та множина сусідніх вузлів, яким направляється запит.

Список літератури

1. Телятніков О. О. Моделі та алгоритми оптимізації розподілених баз даних комп'ютерних інформаційних систем : автореф. дис. на здобуття нак. ступеня д-ра тех. наук : спец. 05.13.06 "Технічні науки"/ Телятніков Олександр Олегович; Донецький національний університет – Д., 2005.
2. Кормен Томас. Алгоритмы. Построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М. : Вильямс, 2005. – 1296с.
3. Ландэ Дмитрий Владимирович. Интернетика: Навигация в сложных сетях: модели и алгоритмы / Д. В. Ландэ, А. А. Снарский, И. В. Безсуднов. – М.: ЛИБРИКОМ , 2009. – 264с
4. Рассел С. Дж., Искусственный интеллект: современный подход / С. Дж. Рассел, П. Норвиг; пер. с англ. К.А. Птицына. – М: Вильямс, 2006. – 1408с.
5. Zeinalipour-Yazti D. Information Retrieval in Peer-to-Peer Networks / D. Zeinalipour-Yazti, V. Kalogeraki, D. Gunopulos //IEEE CiSE Magazine, Special Issue on Web Engineering. – 2004. – P. 1-13.
6. Jun Fan A Fusion of Algorithms in Near Duplicate Document Detection / Jun Fan, Tiejun Huang // Lecture Notes in Computer Science. New Frontiers in Applied Data Mining. – 2012. – Vol.7104/2012. – P. 234-242.
7. Wen Zhang Comparative study of TF*IDF, LSI and multi-words for text classification / Wen Zhang, Taketoshi Yoshida, Xijin Tang A. // Expert Systems with Applications. – 2011. – Vol.38. – P. 2758–2765.
8. Теленик С.Ф. Алгебри для автоматичного проектування схем генерації і оброблення електронних документів / С.Ф. Теленик, В.С. Хмелюк, І.О. Безпалий, І.В. Клепач // Інформатика та системні науки (ІСН-2010). – 2010. – С. 185-188.
9. Теленик С.Ф., Пошук і реферування в системі електронного документообігу / С.Ф. Теленик, О.Ю. Шкабура, Н.О. Подригайло// Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. – 2009. – №51.– С. 177-184.
10. Теленик С.Ф. Подход к построению бизнес-процессов в адаптивной технологии SmartBase / С.Ф. Теленик, В.С. Хмелюк, К.А. Крижова // Вісник національного технічного університету «ХПІ». –2007.– №7.– С. 86-100.

УДК 004. 056.5

МАРКОВСЬКИЙ О.П.,
РЯБИКІНА В.О.,
СЕМЕНЮК Ю.В.

ЕФЕКТИВНИЙ МЕТОД КОРЕКЦІЇ “ПАЧКИ” ПОМИЛОК У КАНАЛАХ ЗІ СПЕКТРАЛЬНОЮ МОДУЛЯЦІЄЮ

Запропоновано новий метод для корекції пачки помилок в каналах зі спектральною модуляцією, особливістю якого є використання математичних операцій без міжрозрядних переносів. Описано запропоновані процедури кодування та корекції “пачки” помилок. Виклад проілюстровано чисельним прикладом. Виконано порівняльний аналіз запропонованого методу і кодів Ріда-Соломона з позицій обчислювальної складності. Доведено, що запропонований метод забезпечує суттєве прискорення кодування, виявлення та корекції “пачки” помилок, а також спрощення схеми апаратної реалізації.

The new method for burst error correction in spectral modulation channels is proposed, feature of one is use of mathematical operations without carry. The proposed burst of errors coding and correction procedures are presented. The presentation is illustrated by numerical example. The comparative analysis of proposed method and Reed-Solomon codes from the standpoint of computation complexity is performed. It has been shown that proposed method provides a significant acceleration bust errors coding, detection and correction and simplifies hardware implementation.

Вступ

Розвиток технологій передачі даних в комп’ютерних та телекомунікаційних системах пов’язаний з проблемою забезпечення високої надійності в процесі передачі.

Для ущільнення каналів та збільшення обсягу даних, які вони здатні передавати, використовують схеми модуляції з ефективним використанням наданого спектру. Прикладом таких схем модуляції може бути QAM (Quadrature Amplitude Modulation). Цим схемам притаманна висока швидкість передачі за умови високої якості каналу [1]. Данні передаються не окремими бітами як, під час використання MSK (Minimum Shift Keying) та BPSK (Binary phase-shift keying), а символами, які можуть складатись з багатьох біт, так QAM256 один символ складається з 8 біт. Кожному символу співвідноситься один сигнал, що безпосередньо передається по каналу.

Якщо в процесі передачі на сигнал діє зовнішня завада, то, відповідно, спотворюється весь символ, на який модулюється цим сигналом. В останні роки, в зв’язку зі багатократним зростанням швидкості передачі, за час дії зовнішніх завад по каналу передається декілька символів. Відповідно, типовою помилкою стає спотворення “пачки” символі – тобто послідовності суміжних символів на час передачі яких діє зовнішня завада [1].

Чим вища швидкість передачі даних, тим більша довжина “пачки” помилок від однієї перешкоди. Для існуючих методів корекції символьних помилок збільшення їх кількості сильно впливає на зростання обчислювальної складності процесів виправлення. Вважаючи на те, що для переважної більшості систем корекція має виконуватися в реальному часі, об’єктивна тенденція збільшення довжини “пачки” спотворених символів потребує експоненційного зростання обчислювальних ресурсів потрібних для корекції в реальному часі.

Разом з тим, швидкісні канали все більше використовується не тільки в спеціалізованих системах, а і в мобільних пристроях. Ці пристрої використовують батарейне живлення, тому надзвичайно обмежені в обчислювальних ресурсах. З іншого боку батарейне живлення мобільних термінальних пристроїв накладає суттєві обмеження на потужність канальних сигналів, що негативно впливає на достовірність передачі даних і, відповідно, потребує використання ефективних корегуючих кодів.

Таким чином, проблема створення ефективних в плані обчислювальної складності методів для корекції домінуючого в каналах зі спектральним ущільненням типу помилок – а саме “пачки”, є актуальною для сучасного етапу розвитку технології передачі даних.

Аналіз методів корекції "пачки" помилок в каналах зі спектральним ущільненням

В сучасних комп'ютерних та телекомунікаційних системах використовують спеціальні види модуляції, які дозволяють ефективніше використовувати частотні властивості каналу. Особливістю таких видів модуляції є те, що символ, який передається належить алфавіту з M символів, тому одночасно передається $k = \log_2 M$ бітів за один символьний інтервал. Це дозволяє в k раз збільшити швидкість передачі інформації.

На сьогодні, розроблено і активно використовується декілька видів такої модуляції, які орієнтовані на різні типи каналів. Параметри більшості з них фіксовані відповідними протоколами передачі цифрової інформації.

Найбільш відомим видом спектрально ефективної модуляції є квадратуно-амплітудна модуляція (QAM). Існують різні модифікації QAM які відрізняються кількістю біт з яких складається символ.

Під впливом зовнішньої завади може бути пошкоджено один або "пачка" символів, що залежить від часу дії завади та частоти слідування канальних сигналів. Виходячи з існуючої тенденції збільшення частоти слідування канальних сигналів має місце зростання довжини "пачки" пошкоджених символів.

В роботі розглядається передача блока m-розрядних символів, кожен з яких модулюється канальним сигналом. Вважається що за час передачі блока, можлива дія лише однієї зовнішньої завади, причому кількість спотворених нею символів лежить в інтервалі від одного до $2 \cdot q - 1$ (параметр q залежить від типу каналу, умов та швидкості передачі).

Для виправлення помилок в каналах зі спектральною модуляцією найбільш часто використовують різновиди циклічних кодів, таких як коди Файра, коди Міласа, або коди Ріда-Соломона. Найбільшим недоліком цих кодів є їхня висока обчислювальна складність, так як для знаходження помилок необхідно розв'язувати систему нелінійних рівнянь. Аналітичного способу її розв'язання не існує, тому використовуються технології перебору. Технологія перебору відрізняється в залежності від коду. Саме необхідність виконувати перебір зумовлює високу обчислювальну складність.

Найбільшого поширення для корекції пачок помилок в каналах зі спектральною модуляцією набули коди Ріда-Соломона. Ці коди дозволяють виправляти h спотворених символів з використанням $2 \cdot h$ контрольних символів. При виникненні пачки помилок коди Ріда-Соломона не враховують те, що спотворені символи є суміжними, тобто при корекції пачки значна частина процедури виправлення є надлишковою.

Динамічне зростання швидкостей передачі даних має наслідком зміщення акценів в оцінці ефективності корегуючих кодів. Зокрема, на сьогодні, більш важливим критерієм є обчислювальна складність, а ніж кількість надлишкової інформації в блоці передаваємої інформації. З цих позицій найбільш суттєвою вадою кодів Ріда-Соломона є висока обчислювальна складність, яка значно ускладнює корекцію помилок в реальному часі.

Інший підхід до корекції "пачки" помилок базується на використанні контрольних сум. Добре відомо, що контрольні суми є найшвидшим методом контролю та виправлення помилок. Зокрема, запропонована спеціальна модифікація зважених контрольних сум для швидкого знаходження помилок у каналах з імпульсно кодовою модуляцією [3]. Проте вона прямо не може бути використана для виявлення та корекції пачок помилок в каналах зі спектральним ущільненням.

Метою дослідження є прискорення процесів виявлення та корекції одиничної "пачки" помилок передачі даних в каналах зі спектральною модуляцією за рахунок розробки методу, що враховує особливості вказаного класу помилок та базується на використанні зважених контрольних сум, для обчислення яких використовується операція множення без переносів.

Метод швидкої корекції пачки помилок

Запропонований метод дозволяє виконувати корекцію "пачки" помилок з максимальною довжиною $2 \cdot q - 1$ пошкоджених символів. В основі методу лежить використання зважених контрольних сум. Для зменшення обчислювальної складності при обчисленнях зважених контрольних сум використовуються спеціальні різновиди математичних операцій множення без переносів [4], а також поліноміальне ділення.

Множення без міжрозрядних переносів визначається наступним чином: нехай X та Y m-розрядні двійкові числа, що визначаються як:

$$X = \{x_0, x_1, ..., x_{m-1}\} = x_{m-1} + 2 \cdot x_{m-2} + ... + 2^{m-1} \cdot x_0,$$
$$x_i \in \{0,1\} \forall i \in \{0,1,...,m-1\},$$
$$Y = \{y_0, y_1, ..., y_{m-1}\} = y_{m-1} + 2 \cdot y_{m-2} + ... + 2^{m-1} \cdot y_0,$$
$$y_i \in \{0,1\} \forall i \in \{0,1,...,m-1\},$$

тоді їхній добуток без міжрозрядних переносів $P = X \otimes Y$ є $2 \cdot m$-розрядне двійкове число, що визначається як:

$$P = \{p_0, p_1, ..., p_{m-1}\} = p_{2 \cdot m-1} + 2 \cdot p_{2 \cdot m-2} + ... +$$
$$+ 2^{2 \cdot m-1} \cdot p_0 = X \cdot y_{m-1} \oplus 2 \cdot X \cdot y_{m-2} \oplus ... \oplus$$
$$\oplus 2^{m-1} \cdot X \cdot y_0, p_i \in \{0,1\} \forall i \in \{0,1,...,m-1\}.$$

Наприклад, якщо m = 4: $X = \{1,0,1,0\} = 10_{10}$ $Y = \{1,0,0,1\} = 9_{10}$, тоді $P = X \otimes Y = 1010 \oplus$ $\oplus 1010000 = 01011010 = 90_{10}$.

Операція поліноміального ділення визначається як ділення $2 \cdot m$-розрядного поліному на m-розрядний. При виконанні операції формуються частка та остача, які позначаються як *Q(P/Y)* та *R(P/Y)* відповідно, де:

$$P = \{p_0, p_1, ..., p_{m-1}\} = p_{2 \cdot m-1} + 2 \cdot p_{2 \cdot m-2} + ... +$$
$$+ 2^{2 \cdot m-1} \cdot p_0, p_i \in \{0,1\} \forall i \in \{0,1,...,m-1\}.$$
$$Y = \{y_0, y_1, ..., y_{m-1}\} = y_{m-1} + 2 \cdot y_{m-2} + ... + 2^{m-1} \cdot y_0,$$
$$y_i \in \{0,1\} \forall i \in \{0,1,...,m-1\}.$$

Наприклад, якщо m = 4: $P = \{0,1,0,1,1,1,1,0\} =$ $= 94_{10}, Y = \{1,0,0,1\} = 9_{10}$, тоді $Q(P/Y) = 1010 =$ $= 10_{10}, R(P/Y) = 0100 = 4_{10}$.

Запропонований метод передбачає ділення інформаційного блоку B на n кадрів F, кожен довжиною q m-розрядних символів, де q – обраний параметр коду, отже:

$$B = \{F_0, F_1, ..., F_{n-1}\}, F_i = \{X_{i \cdot q}, X_{i \cdot q+1}, ..., X_{i \cdot q+(n-1)}\},$$
$$X_j = \{x_{j,0}, x_{j,1}, ..., x_{j,m-1}\}.$$

Використовуючи кадри F, передавач обчислює контрольний код, який складається з $2 \cdot q$ компонент. Компоненти контрольного коду обчисленого передавачем позначаються $C_{Tk}, S_{Tk}, k \in \{0,1,...,q-1\}$. Перші q компонент контрольної суми обчислюються за формулою:

$$C_k = F_{0,k} \oplus F_{1,k} \oplus ... \oplus F_{n-1,k},$$
$$k \in \{0,1,...,q-1\} \quad (1)$$

Наступні q компонент зваженої контрольної суми і обчислюються за формулою:

$$S_k = F_{0,k} \otimes 1 \oplus F_{1,k} \otimes 2 \oplus ... \oplus F_{n-1,k} \otimes n,$$
$$k \in \{0,1,...,q-1\} \quad (2)$$

Після обчислення контрольного коду передавачем, інформація передається на приймач в наступній послідовності:

$B, C_{T0}, S_{T0}, C_{T1}, S_{T1}, ..., C_{Tq-1}, S_{Tq-1}$.

Приймач за прийнятим блоком B обчислює контрольні коди $C_{Rk}, S_{Rk}, k \in \{0,1,...,q-1\}$ за формулами (1) та (2) відповідно. З обчислених та прийнятих контрольних кодів обчислюються різниці між кодами приймача і передавача:

$$\Delta_k = C_{Rk} \oplus C_{Tk}, \delta_k = S_{Rk} \oplus S_{Tk},$$
$$k \in \{0,1,...,q-1\} \quad (3)$$

Зі значень різниць визначається наявність помилок та їх тип. Можливі наступні часткові випадки:

1) $\Delta_k = 0, \delta_k = 0, k \in \{0,1,...,q-1\}$, в цьому випадку блок передано правильно.

2) $\exists k \in \{0,1,...,q-1\} : (\Delta_k = 0, \delta_k \neq 0) \vee (\Delta_k \neq 0$ $, \delta_k = 0)$ – означає, що помилка відбулась у контрольних кодах, блок даних передано правильно і корекція не потрібна.

Якщо попередні умови не виконались, то в даних є помилки, які необхідно виправити. Необхідним є ввести означення помилок 1, 2 та 3 типів.

Помилка з позицією l вважається помилкою першого типу якщо більше нема помилок з позиціями $l' = l \pm q \cdot n, n \in N$.

Якщо існує лише одна помилка з позицією $l' = l \pm q \cdot n, n \in N$, то помилка вважається помилкою другого типу.

Якщо таких помилок більше – то це помилка третього типу.

В межах однієї пачки помилок довжиною $2 \cdot q - 1$ помилки 3 типу неможливі. Тому вони в запропонованому методі не розглядаються.

Для помилок першого типу виконується умова $R(\delta_k / \Delta_k) = 0$, де k позиція помилки у кадрі, якщо $R(\delta_k / \Delta_k) \neq 0$ – вважається, що помилка на позиції k – помилка другого типу.

Наступний крок – знаходження множин індексів помилок обох типів. Для цього, використовуючи обчислені значення Δ_k та δ_k, для помилок першого типу будується множина A:

$$A = \{k \mid \forall k \in \{0,1,...,q-1\} : \Delta_k \neq 0, \delta_k \neq 0, R(\delta_k / \Delta_k) = 0\}, \quad (4)$$

Для побудови множини B помилок другого типу використовується формула:

$$B = \{k \mid \forall k \in \{0,1,...,q-1\} : \Delta_k \neq 0, \delta_k \neq 0, R(\delta_k / \Delta_k) \neq 0\} \quad (5)$$

З побудованих множин визначаються помилки. Розглядаються 2 випадки:

1) Множина $B = \varnothing$ – робиться висновок, що відбулись лише помилки першого типу, які виправляються за формулою (6).

$$\forall k \in A : l(k) = (Q(\delta_k / \Delta_k) - 1) \cdot q + k, \quad X_{l(k)} = X_{Rl(k)} \oplus \Delta_k \quad (6)$$

Після корекції за формулою (6) блок даних вважається виправленим.

2) Множина $B \neq \varnothing$ – робиться висновок, що присутні помилки обох типів. Важливо перевірити умову $|A| + |B| \neq q$, що означає не виконання умови належності всіх помилок до однієї пачки або умови максимальної довжини пачки і виправлення помилок запропонованим методом неможливе.

З умови $|A| + |B| = q$, слідує що в даних, які отримав приймач відбулась пачка помилок довжиною $|A| \cdot 2 + |B|$. Знаходження позиції p першої помилки в пачці реалізується за формулою :

$$p = \min((Q(\delta_k / \Delta_k) - 1) \cdot q + k) - |B|, \forall k \in . \quad (7)$$

Оскільки знайдено початок пачки, можлива корекція. Для виправлення помилок першого типу використовується формула (6), для помилок другого типу наступна формула:

$$\forall t \in N, t \geq p, t < p + |B| :$$
$$k(t) = t \bmod q, f(t) = \left\lfloor \frac{t}{q} \right\rfloor + 1,$$
$$y(t) = Q\left(\frac{\delta_{k(t)} \oplus (\Delta_{k(t)} \otimes (f(t)+1))}{f(t) \otimes (f(t)+1)}\right), \quad (8)$$
$$X_t = X_{Rt} \oplus y(t),$$
$$X_{t+q} = X_{Rt+q} \oplus y(t) \oplus \Delta_{k(t)}$$

Після виконання всіх необхідних операцій блок даних вважаться виправленим або робиться висновок що пошкодження надто сильні і корекція неможлива.

Описана процедура корекції ілюструється наведеним нижче прикладом в якому параметр коду q = 4, нехай початковий інформаційний блок складається з 12 символів по 4 біти кожен:

$X_0 = \{1,0,0,1\}, X_1 = \{0,1,1,1\}, X_2 = \{0,1,1,0\},$
$X_3 = \{1,0,0,1\}, X_4 = \{0,1,0,0\}, X_5 = \{1,1,0,0\},$
$X_6 = \{0,1,1,1\}, X_7 = \{1,0,1,1\}, X_8 = \{0,1,0,1\},$
$X_9 = \{1,0,0,0\}, X_{10} = \{1,0,1,1\}, X_{11} = \{0,0,1,1\}$

Контрольні коди обчислені передавачем:

$C_{T0} = 1001 \oplus 0100 \oplus 0101 = 1000,$
$C_{T1} = 0111 \oplus 1100 \oplus 1000 = 0011,$
$C_{T2} = 0110 \oplus 0111 \oplus 1011 = 1010,$
$C_{T3} = 1001 \oplus 1011 \oplus 0011 = 0001,$
$S_{T0} = 1001 \otimes 1 \oplus 0100 \otimes 2 \oplus 0101 \otimes 3 = 1110,$
$S_{T1} = 0111 \otimes 1 \oplus 1100 \otimes 2 \oplus 1000 \otimes 3 = 0111,$
$S_{T2} = 0110 \otimes 1 \oplus 0111 \otimes 2 \oplus 1011 \otimes 3 = 10101,$
$S_{T3} = 1001 \otimes 1 \oplus 1011 \otimes 2 \oplus 0011 \otimes 3 = 11010,$

В рамках прикладу вважається, під час передачі блоку під дією завади було спотворено 7 символів:

$X_2 = \{0,1,0,1\}, X_3 = \{1,1,0,0\}, X_4 = \{0,0,0,1\},$
$X_5 = \{0,1,0,1\}, X_6 = \{0,0,0,0\}, X_7 = \{0,1,0,0\},$
$X_8 = \{1,1,1,1\}.$

Контрольні коди обчислені приймачем

$C_{R0} = 1001 \oplus 0001 \oplus 1111 = 0111,$
$C_{R1} = 0111 \oplus 0101 \oplus 1000 = 1010,$
$C_{R2} = 0101 \oplus 0000 \oplus 1011 = 1110,$
$C_{R3} = 1100 \oplus 0100 \oplus 0011 = 1011,$
$S_{R0} = 1001 \otimes 1 \oplus 0001 \otimes 2 \oplus 1111 \otimes 3 = 11010,$
$S_{R1} = 0111 \otimes 1 \oplus 0101 \otimes 2 \oplus 1000 \otimes 3 = 10101,$
$S_{R2} = 0101 \otimes 1 \oplus 0000 \otimes 2 \oplus 1011 \otimes 3 = 11000,$
$S_{R3} = 1100 \otimes 1 \oplus 0100 \otimes 2 \oplus 0011 \otimes 3 = 00001,$

З формули (3) обчислюються різниці:

$\Delta_0 = C_{T0} \oplus C_{R0} = 1000 \oplus 0111 = 1111,$
$\Delta_1 = C_{T1} \oplus C_{R1} = 0011 \oplus 1010 = 1001,$
$\Delta_2 = C_{T2} \oplus C_{R2} = 1010 \oplus 1110 = 0100,$
$\Delta_3 = C_{T3} \oplus C_{R3} = 0001 \oplus 1011 = 1010,$
$\delta_0 = S_{T0} \oplus S_{R0} = 01110 \oplus 11010 = 10100,$
$\delta_1 = S_{T1} \oplus S_{R1} = 00111 \oplus 10101 = 10010,$
$\delta_2 = S_{T2} \oplus S_{R2} = 10101 \oplus 11000 = 01101,$
$\delta_3 = S_{T3} \oplus S_{R3} = 11010 \oplus 00001 = 11011.$

Аналіз отриманих результатів показує, що всі $\Delta_k \neq 0, \delta_k \neq 0, k \in \{0,1,2,3\}$ отже необхідно знайти остачі від ділення:

$R(\delta_0 / \Delta_0) = R(10100/1111) = 0101,$

$R(\delta_1 / \Delta_1) = R(10010/1001) = 0000,$

$R(\delta_2 / \Delta_2) = R(1101/0100) = 0001,$

$R(\delta_3 / \Delta_3) = R(11011/1010) = 0101,$

Зі знайдених залишків, за формулами (4) і (5) на стороні приймача формуються множини $A = \{1\}$ і $B = \{0,2,3\}$. Оскільки $B \neq \varnothing$, то це значить, що “пачка” містить помилки обох типів. Наступним кроком є перевірка умови $|A| + |B| = q$: для прикладу, що розглядається $1 + 3 = 4$, це означає, що для виправлення помилок застосується формула (7), яка дозволяє визначити позицію першого спотвореного символу “пачки” наступним чином p=(Q(10010/1001)-1)·4+1-3=2.

Далі з використанням формули (6) виконується корекція для помилки 1 типу: $l(1) = (Q(10010/1001) - 1) \cdot 4 + 1 = 5$, X_5=X_{R5}⊕Δ_1= 0101⊕1001=1100. Корекція помилок другого типу виконується за формулою (8):

$t \in \{2,3,4\}$:

$t = 2, k(2) = 2, f(2) = 1,$

$y(2) = 0011,$

$X_2 = 0101 \oplus 0011 = 0110,$

$X_6 = 0000 \oplus 0011 \oplus 0100 = 0111,$

$t = 3, k(3) = 3, f(3) = 1,$

$y(3) = 0101,$

$X_3 = 1100 \oplus 0101 = 1001,$

$X_7 = 0100 \oplus 0101 \oplus 1010 = 1011,$

$t = 4, k(4) = 0, f(4) = 2,$

$y(4) = 0101,$

$X_4 = 0001 \oplus 0101 = 0100,$

$X_8 = 1111 \oplus 0101 \oplus 1111 = 0101.$

Отже приймач повністю відновлює пошкоджені завадою символи.

Оцінка ефективності

Запропонований метод дозволяє виправляти одну в блоці “пачку”, що містить від 1 до $2 \cdot q - 1$ суміжних спотворених символів з використанням K контрольних розрядів, причому:

$$K = q \cdot (2 + \frac{\log_2 n}{m}) \qquad (9)$$

Для виправлення аналогічного класу помилок з використання кодів Ріда-Соломона, потрібно $K_{RS} = 2 \cdot (2 \cdot q - 1) = 4 \cdot q - 2$ контрольних символів.

Для кодів Ріда-Соломона найбільша ефективність досягається якщо кількість L символів в блоці та їх розрядність m пов’язані умовою: $2^L = m$. Якщо прийняти таку умову для порівняння запропонованого методу, то, приймаючи до уваги, що L=n·q, формула (9) трансформується до наступного виду:

$$K = q \cdot (2 + 1 - \frac{\log_2 q}{m}) = 3 \cdot q - \frac{q \cdot \log_2 q}{m} \qquad (10)$$

Цілком очевидно, що значення K і K_{RS} близькі. Наприклад, якщо інформаційний блок складається з $L = 64$ 4-бітних символів і вирішується задача корекції “пачки”, яка містить від одного до 7-ми спотворених символів, то параметр коду $q = 4$, кількість кадрів n=L/q=16. Для вирішення цієї задачі в запропонованому методі потрібно K=4·(2+log$_2$16/4)=12 контрольних символів, тобі, як використання кодів Ріда-Соломона вимагає 4·4-2=14 контрольних символів.

Основною перевагою запропонованого методу є значне, в порінянні з кодами Ріда-Солмона, зменшення обчислювальної складності за рахунок того, що врахлвується характер помилок, а саме: той факт, що спотворені символи є суміжними. Це дозволяє спростити програмну а апаратну реалізацію.

Для кодів Ріда-Соломона обчислення (4·q-2)-х контрольних символів на передавачі вимагає виконання L циклів, в кожному з яких реалізується (4·q-2) операцій множення m-розрядних символів на полі Галуа.

Якщо вважати, що операція множення на полях Галуа m-розрядних чисел вимагає, в середньому, 3·m логічних операцій, то обчислювальна складність кодування для кодів Ріда-Соломона становить O(12·q^2·m·n).

Компоненти контрольного коду при використанні запропонованого метод обчислюються за формулами (1) та (2). Відповідно при обчисленні кожної компоненти згідно з (1) потрібно виконати $q \cdot (n - 1)$ логічних операцій, а при обчисленні компоненти по формулі (2) необхідно виконати $q \cdot (n - 1)$ логічних операцій та $q \cdot n$ операцій множення без переносів. Враховуючи, що операція множення без переносів в середньому потребує $2.5 \cdot m$ логічних операцій, то загальна обчислювальна складність для знаходження контрольного коду визначається як

$O(5 \cdot m \cdot q \cdot n)$. Таким чином, обчислювальна складність кодування в запропонованому методі в $2.4 \cdot q$ менша в порівнянні з використанням кодів Ріда-Соломона. Так в рамках наведеного вище прикладу q=4 і, відповідно застосування розробленого методу дозволяє зменшити обчислювальну складність в 38 разів. Практично це означає, що використання запропонованого методу значно спрощує проблему контролю в темпі передачі даних.

Корекція спотворених символів з використанням кодів Ріда-Соломона включає наступні процедури:

1) Розв'язання системи $2 \cdot q$-1 символьних рівнянь для визначення коефіцієнтів вектору локатора помилок.

2) Локалізація позицій $2 \cdot q$-1 спотворених символів. Для цього потрібно знайти ненульові значення вектору локатора помилок шляхом перебору L можливих варіантів.

3) Розв'язання системи $2 \cdot q$-1 символьних рівнянь для локалізації спотворених бітів для кожного із символів "пачки".

Обчислювальна складність виправлення помилок, головним чином, визначається складністю процедури локалізації позицій спотворених символів, що основана на переборі L можливих варіантів. при цьому, для кожного з варіантів необхідно виконати $2 \cdot q$-1 операцій множення m-розрядних чисел на полі Галуа GF(2^m). Відповідно, обчислювальна складність корекції помилок для кодів Ріда-Соломона становить $O(6 \cdot q^2 \cdot n \cdot m)$.

Для запропонованого методу операція корекції має найбільшу обчислювальну складність за умови коли зовнішня завада спотворила $2 \cdot q - 1$ суміжних символів. При цьому виникає одна помилка першого типу та q помилок другого типу. Для виправлення однієї помилки першого типу за формулою (6) потрібно виконати лише одну логічну операцію, для виправлення кожної помилки з q помилок другого типу за формулою (7) потрібно виконати операцію ділення без переносів, дві операції множення без переносів та арифметичної суми, а також чотири логічних операцій. Оскільки операція ділення без переносів потребує, як і множення, в середньому $2.5 \cdot m$ логічних операцій, загальна обчислювальна складність корекції блоку визначається як $O(5 \cdot m \cdot q)$

Це означає, що використання запропонованого методу дозволяє реалізувати корекцію помилок за час, що, на відміну від кодів Ріда-Соломона, не залежить від довжини інформаційного блоку. При цьому обчислювальна складність зменшується приблизно в $q \cdot n$ разів. Так, в рамках наведеного вище прикладу для n=16 і q=4 використання розробленого методу дозволяє в 64 рази прискорити процес корекції помилок в порівнянні з кодами Ріда-Соломона.

Висновки

В результаті виконаних досліджень запропоновано новий метод корекції однієї в інформаційному "пачки" помилок, що виникають при передачі блоку даних в каналах зі спектральною модуляцією.

За рахунок врахування особливостей вказаного типу помилок а також застосування зважених контрольних сум на основі множення без переносів вдалося досягти значного зменшення, в порівнянні з кодами Ріда-Соломона, обчислювальної складності як процесу кодування, так і процесу корекції помилок. використання запропонованого методу дозволяє реалізувати корекцію помилок за час, що, на відміну від кодів Ріда-Соломона, не залежить від довжини інформаційного блоку.

Використання запропонованого методу дозволяє вирішити важливу для сучасних комп'ютерних мереж та телекомутаційних систем задачу виявлення та виправлення в реальному часі помилок передачі даних, зумовлених зовнішніми завадами в бездротових каналах.

Список літератури

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Изд. дом "Вильямс".– 2004.– 1104 С.
2. Морелос-Сагароса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. – М.: Техносфера, – 2005. – 320 С.
3. Марковский А.П., Клименко И.А., Иванов А.Н. "Способ эффективной коррекции "пачки" ошибок в каналах с кодово-импульсной модуляцией" // Проблеми інформатизації та управління. Збірник наукових праць: Випуск 2(33).-К.,НАУ.– 2011.– С.143-151.
4. Bardis N.G., Doukas N., Markovskyi O. Burst Error Correction Using Binary Multiplication without Carry // Proceedings of MILCOM – 2011, Baltimore, USA, November 2011. CA, ISSN: 2155-7578, pp. 1783 – 1787, – 2011.

УДК 004.052.42

МАРКОВСКИЙ А.П.,
УВАРОВ Н.В.,
УВАРОВА Н.В.

ГАРАНТИРОВАННОЕ ОБНАРУЖЕНИЕ ЧЕТЫРЕХКРАТНЫХ ОШИБОК С ИСПОЛЬЗОВАНИЕМ МИНИМАЛЬНОГО ЧИСЛА КОНТРОЛЬНЫХ РАЗРЯДОВ

В статье описан метод гарантированного обнаружения четырехкратных ошибок передачи данных в симметричном двоичном канале. Особенность разработанного метода заключается в использовании теоретически минимального числа контрольных разрядов. Метод основывается на использовании весовых коэффициентов, сформированных на основе специального нелинейного функционального преобразования. Показано, что использование таких коэффициентов для вычисления контрольных разрядов обеспечивает гарантированное обнаружение четырехкратных ошибок при минимально возможной длине контрольного кода. В статье проанализированы свойства специальных нелинейных преобразований. Определены условия, которым должны соответствовать как само преобразование, так и ее дифференциал.

The article is dedicated to the method of the guaranteed detection of quadruple errors during transmission via symmetrical binary channel. The main feature of the proposed method is the theoretically minimal length of the error detection code. The method is based on the use of the weight coefficients formed on the basis of a special non-linear functional transformation. It is shown that use of such coefficients in control code calculation provides guaranteed detection of quadruple errors with the minimal length of the error detection code. Also in the article the properties of such non-linear transformation has been analyzed. The conditions to be met for both the transformation itself and its differential are determined.

Введение

В условиях углубления процессов информационной интеграции и расширения использования распределенных компьютерных систем возрастает роль средств передачи данных. Важная роль в прогресса средств передачи и обмена информации принадлежит технологиям контроля возникающих ошибок, частью которых являются способы обнаружения битовых искажений различной кратности.

В современных условиях объективно существуют факторы, педалирующие рост кратности возникающих ошибок. В частности, динамичный рост скорости передачи данных имеет следствием рост числа ошибок, вызванных межсигнальной интерференцией [1]. Расширяющееся использование беспроводных средств передачи данных существенно повысило интенсивность внешних электромагнитных помех, что приводит к увеличению вероятности возникновения ошибок. Отмечается [1] и тенденция роста длины контролируемых блоков, данных, которые составляют в настоящее время тысячи байт. Ясно, что с ростом блины блока возрастает вероятность появления многократных битовых искажений.

В отличие от систем связи или цифрового телевидения, при передаче данных между компонентами компьютерных систем необходимо обеспечить качественно более высокий уровень надежности. Соответственно, ошибка передачи данных в компьютерных системах имеет значительно большую цену в сравнении с системами связи.

Таким образом, проблема повышения эффективности обнаружения ошибок большей кратности, возникающих при передаче данных является актуальной в реалиях современного этапа развития компьютерных технологий.

Анализ кодов, используемых для обнаружения ошибок

Для обнаружения ошибок передачи данных в компьютерных системах преимущественно используются:

– циклические избыточные коды (CRC – Cyclic Redundancy Codes);

– контрольные суммы (CS – Check Sum).

И контрольные суммы, и CRC относятся к средствам блокового контроля.

При использовании этих кодов, к блоку из m битов $B=\{b_1,b_2,\ldots,b_m\}$, $\forall l=1,\ldots,m$: $b_l\in\{0,1\}$ добавляется k контрольних разрядов

$\{b_{m+1},b_{m+2},\ldots,b_{m+k}\}$. Совокупность всех возможных блоков вместе с зависящими от них контрольными разрядами образует множество Ω разрешенных кодов. Получение приемником неразрешенного кода означает ошибку. Обнаруживающие возможности кодов характеризуются минимальной величиной кодового расстояния d_{min} между разрешенными кодами, которое определяется следующей формулой [1]:

$$d_{min} = \min_{X,Y\in\Omega} \sum_{j=1}^{m+k} (x_j \oplus y_j).$$

Для обнаружения ошибок кратностью h необходимо и достаточно, чтобы выполнялось условие [1]:

$$d_{min} > h. \qquad (1)$$

Выражение (1) устанавливает теоретические границы числа необходимых контрольных разрядов.

Для гарантированного выявления 4-кратных ошибок, которые возникают при передаче m-битового блока, минимальное количество k контрольных разрядов должно быть таким, что минимальное хемингово расстояние между $(m+k)$-битовыми кодами равнялась пяти. Таким образом, можно показать, что: $k = \lfloor 2 \cdot log_2 m \rfloor$.

Наиболее распространенным в вычислительной технике способом контроля правильности передачи данных является CRC.

Сущность контроля с использованием циклических избыточных кодов состоит в том, что контролируемый блок B представляется в виде полинома $P(B)$ степени $m+k$:

$$P(B)=b_1 \cdot x^k + b_2 \cdot x^{k+1} + \ldots + b_{m-1} \cdot x^{k+m-1} + b_m \cdot x^{k+m} \qquad (2)$$

Контрольный код $R(B)$ вычисляется как остаток от деления полинома $P(B)$, определяемого (2), на образующий полином $Q(X)$ степени k циклического избыточного кода.

По основному критерию эффективности – надежности обнаружения ошибок, циклические коды превосходят контрольные суммы. Ошибки при передаче блока данных не обнаруживается, если полином $E(X)$, соответствующий вектору ошибки, делится на образующий полином CRC $Q(X)$ без остатка. Показано [2], что все полиномы $E(X)$, соответствующие указанным ниже ошибкам, не делятся на специальным образом выбранный базовый полином $Q(X)$, а, следовательно, они обнаруживаются гарантированно:

1. Все искажения битов $b_1,b_2,\ldots,b_m$ нечетной кратности, если базовый полином $Q(X)$ может быть представлен в виде произведения полиномов: $Q(X) = (x+1) \cdot S(X)$;

2. Все двукратные искажения битов контролируемого блока, если базовый полином $Q(X) = q_0 + q_1 \cdot x + q_2 \cdot x^2 + \ldots + q_k \cdot x^k$ содержит не менее трех ненулевых компонент: $\sum_{i=0}^{k} q_i \geq 3$;

3. Группа ошибок, локализованных в рамках k разрядов.

Для остальных ошибок показано [2], что остаток $R(B)$ представляет собой результат хеширования блока B данных в пространство 2^k всех возможных контрольных кодов. Соответственно, вероятность P_{CRC} того, что эти ошибки не будут обнаружены с использованием CRC с образующим полиномом $Q(X)$ степени k, определяется как: $P_{CRC} = \frac{1}{2^k}$.

Наряду с высокой надежностью, CRC обладает рядом недостатков, наиболее важным из которых является принципиально последовательный характер вычисления контрольного кода, что обуславливает существование ограничений на скорость выполнения операций, связанных с контролем ошибок [3]. Этот недостаток особенно актуален в современных условиях быстрого роста скоростей передачи.

Как и все способы блокового контроля, CRC обладает низкой скоростью реакции на возникновение ошибки передачи данных, что ограничивает его использование в компьютерных системах реального времени [3].

Важным достоинством контроля передачи блоков информации в компьютерных сетях с использованием CS, по сравнению с другими методами блокового контроля, является простота реализации и высокая скорость [4].

Традиционно, контрольная сумма S блока данных, представляющего собой n k-разрядных символов $D_1,D_2,\ldots D_n$, формируется в виде поразрядной суммы по модулю 2 всех символов блока: $S = D_1 \oplus D_2 \oplus \ldots \oplus D_n$.

Вполне очевидно, что при использовании традиционной контрольной суммы, обнаруживаются все ошибки нечетной кратности.

Основным достоинством контрольных сумм является отсутствие ограничений на скорость реализации вычислительных операций, связан-

ных с контролем ошибок. Это обусловлено тем, что структура вычислений контрольных сумм допускает возможность широкого распараллеливания при аппаратной реализации.

Как следует из проведенного анализа, CRC и CS имеют ограниченный класс гарантированно обнаруживаемых классов ошибок. Для расширения этого класса необходимо существенно изменить организацию этих средств контроля. Вместе с тем, использование кодов Хемминга позволяет гибко расширять класс гарантированно обнаруживаемых ошибок [2].

Тем не менее, q-мерными кодами Хемминга принципиально не может быть обнаружены ошибки кратности 2^q, которые образуют куб в q-мерном пространстве. Это означает, что двумерными контрольными кодами Хемминга не может быть обнаружены 4 ошибки, которые образуют прямоугольник на плоскости. При применении трехмерных кодов Хемминга (q=3), наименьшая кратность ошибки, для которой не обеспечивается гарантированное обнаружение, равна 2^3=8. Так, не обнаруживаются ошибки, точки локализации которых в трехмерном пространстве образуют куб. То есть, при использовании трехмерных контрольных кодов Хемминга, гарантированно могут быть обнаружены ошибки кратностей 2,4 и 6.

Поэтому для гарантированного обнаружения четырех ошибок необходимо наличие $k_3 = 3 \cdot (\sqrt[3]{m})^2$ контрольных разрядов. Основным недостатком контрольных сумм Хемминга является быстрый рост контрольных разрядов, необходимых для гарантированного обнаружения многократных ошибок, при увеличении их кратности. Так, для гарантированного обнаружения 4-кратных ошибок при передаче 1024 бит, использование кода Хемминга требует 305 контрольных разрядов, что резко снижает эффективность контроля ошибок.

В таблице 1 приведены значения числа контрольных разрядов, которые обеспечивают гарантированное обнаружение 4-кратных ошибок. Вполне очевидно, что приведенные в таблице 1 значения существенно превышают теоретический минимум числа контрольных разрядов, устанавливаемого формулой (1).

Это свидетельствует о невысокой эффективности кодирования контрольной информации кодами Хемминга для случая 4-кратных ошибок.

Таким образом, использование кодов Хемминга не решает в достаточной мере эффективно актуальную в современных условиях проблему расширения класса гарантированно обнаруживаемых ошибок

Табл. 1. Количество контрольных разрядов кода Хемминга для гарантированного обнаружения 4-кратных ошибок

Длина кода	Число контрольных разрядов	Длина кода	Число контрольных разрядов
64	48	1024	305
128	76	2048	485
256	121	4096	768
512	193	8192	1223

Метод гарантированного обнаружения 4-кратных ошибок с использованием взвешенных контрольных сумм

Пусть контролируемый блок B состоит из m бит: $B=\{b_1,b_2,\ldots,b_m\}$, $b_l \in \{0,1\}$, $l=1,\ldots,m$.

Код W_j весового коэффициента j-го бита b_j контролируемого блока предлагается формировать в виде 3-х компонент. Первой компонентой является n-разрядный ($n=\log_2 m$) двоичный код $X_j=\{x_{1j},\ldots,x_{nj}\}$, $\forall i \in 1,..,n$: $x_{ij} \in \{0,1\}$ номера j такой, что $j=x_{1j}\cdot 2^{n-1}+x_{2j}\cdot 2^{n-2}+\ldots+x_{(n-1)j}\cdot 2+x_{nj}$,. Вторая компонента – это n-разрядный результат булевого преобразования $F(X_j)$ над кодом X_j. Третья компонента – единичный бит. Таким образом, W_j имеет вид: $W_j=<X_j, F(X_j), 1>$.

Функциональное преобразование $F(X)$ выбирается таким образом, чтобы для любой пары битов блока сумма по модулю 2 их весовых коэффициентов не повторялась. Формально это означает, что для любых 4-х различных номеров битов $q,l,g,r \in \{1,\ldots.m\}$ всегда выполняется неравенство:

$$W_q \oplus W_l \neq W_g \oplus W_r . \qquad (3)$$

Описанный выбор функционального преобразования $F(X)$ иллюстрируется следующим примером. Если блок содержит 15 бит (m=15), то одним из преобразований $F(X)$, при котором выполняется (3), является преобразование, представленное в таблице 2.

Если выбрать, например, четыре номера: q=5, l=7, g=12 и r=14, то их весовые коэффициенты равны соответственно:

$W_q = <5,11,1>$, $W_l = <7,10,1>$, $W_g = <12,9,1>$ и $W_r = <14,4,1>$.

Легко убедиться, что для выбранных номеров неравенство (3) выполняется:

$$W_q \oplus W_l = <2,1,0> \neq W_g \oplus W_r = <2,13,0>.$$

Табл. 2. Пример преобразования F(X), для которого выполняется (3)

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$F(X)$	0	1	2	3	7	11	12	10	13	14	5	8	9	6	4	15

При использовании взвешенных контрольных сумм, контрольный код C блока данных на приемнике и передатчике вычисляется в виде суммы по модулю два произведений битов на соответствующие им весовые коэффициенты:

$$C = \bigoplus_{j=1}^{m} b_j \cdot W_j . \qquad (4)$$

Таким образом, число k_c разрядов контрольного кода C совпадает с разрядностью весовых коэффициентов и определяется формулой:

$$k_c = 2 \cdot n + 1 = 2 \cdot log_2 m + 1 . \qquad (5)$$

Контрольный код C состоит из 3-х компонент: $C=<C_X, C_F, C_p>$, где C_X – сумма по модулю 2 номеров единичных битов передаваемого блока, C_F – сумма по модулю 2 результатов функционального преобразования $F(X)$ номеров единичных битов, а C_p – сумма по модулю 2 битов блока, или бит четности.

В процессе получения блока, приемник формирует свой контрольный код C_R и, по завершению передачи, вычисляет сумму по модулю 2 сформированного и принятого от приемника контрольного кода C_S: $\Delta = C_R \oplus C_S = <\Delta_X, \Delta_F, \Delta_p>$. Если разность Δ контрольных кодов приемника и передатчика равна нулю, то считается, что блок передан без ошибок.

Если в процессе передачи ошибочно передано нечетное количество битов, то значения бита четности контрольных кодов приемника и передатчика отличны и, соответственно, $\Delta_p=1$, то есть $\Delta \neq 0$. Это значит, что битовые ошибки нечетной кратности гарантированно обнаруживаются.

При ошибочной передаче 2-х бит, имеющих номера q и l, код Δ разности контрольных кодов приемника и передатчика равен сумме соответствующих весовых коэффициентов: $\Delta = W_q \oplus W_l = <X_q \oplus X_l, F(X_q) \oplus F(X_l), 0>$. Поскольку первая компонента Δ являет собой сумму по модулю 2 двух различных кодов: $\Delta_X = X_q \oplus X_l$, то $\Delta_X \neq 0$. Это означает, что при двукратной ошибке первая компонента Δ не равна нулю, следовательно, такая ошибка гарантированно обнаруживается.

При ошибочной передаче 4-х битов, номера которых равны q, l, g и r, разность контрольных сумм приемника и передатчика $\Delta = W_q \oplus W_l \oplus W_g \oplus W_r$. Поскольку, согласно (3), $W_q \oplus W_l \neq W_g \oplus W_r$, то $\Delta \neq 0$, что означает гарантированное обнаружение любой 4-кратной ошибки.

Выводы

В результате проведенного исследования разработан метод гарантированного обнаружения четырехкратных ошибок передачи данных в каналах, теоретической моделью которых является двоичный симметричный канал.

Главное отличие предложенного метода от существующих – использование теоретически минимального числа контрольных разрядов. При вычислении контрольных разрядов используются весовые коэффициенты, формируемые с помощью специального нелинейного функционального преобразования. Исследованы свойства таких преобразований.

Вычисление контрольных разрядов легко может быть распараллелено, что обеспечивает возможность контроля ошибок в темпе передачи данных. Предложенный метод может быть эффективно использован для контроля ошибок передачи данных в высокоскоростных каналах между компонентами компьютерных систем, работающих в режиме реального времени.

Список литературы

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Изд. Дом "Вильямс".– 2004.– 1104 С.
2. Klove T., Korzhik V. Error Detecting Codes: General Theory and Their Application in Feedback CommunicationSystems. Norwell, MA: Kluwer, 1995. – 433 p.
3. Nikolaos G. Bardis, Athanasios Drigas and Oleksandr P. Markovskyi, Performance Increase of Error Control Operation on Data Transmission", 3rd International Conference on New Technologies, Mobility and Security, IEEE, *IEEE COMMUNICATIONS SOCIETY*, Egypt – Cairo, 20-23 December 2009.
4. Самофалов К.Г., Марковский А.П., Мулки Яссин Ахмед Ал Бадайнех. Обнаружение и исправление ошибок передачи данных с использованием взвешенных контрольных сумм // Проблеми інформатизації та управління. Збірник наукових праць: Випуск 3(14). – К.,НАУ. – 2008. – С.121– 128.

УДК 004.942

ПОГРЕБНЮК І.М.,
ТОМАШЕВСЬКИЙ В.М.

МОДЕЛЮВАННЯ СЦЕНАРІЇВ АДАПТИВНОГО НАВЧАННЯ З ВИКОРИСТАННЯМ МЕРЕЖ ПЕТРІ

Запропонована інформаційна технологія проектування адаптивних навчальних систем, яки базуються на моделях студентів на основі параметрів рівня підготовки та когнітивних особливостей з використанням карт прогалин знань при вивченні навчального матеріалу. Для моделювання процесу адаптивного навчання та тестування запропонована багаторівнева мережа Петрі, яка використовується як функціональна модель і формує унікальні сценарії навчання для кожного студента.

Developed is the information technology of planning of the adaptive educational systems, which are based on the models of students on the basis of the level of preparation parameters and on cognitive features (maps of gaps in knowledge during the process of studying the educational material are being used). A multilevel Petri net which is used as a functional model and which forms the unique scenarios of studies for every student is offered for the design of process of adaptive studies and testing.

Вступ

Сучасні тенденції розвитку освіти вимагають гнучкості, динамічності й індивідуалізації навчального процесу. Закономірно, що ці зміни впливають і на технології дистанційного навчання. Інтеграція моделей, методів, технологій експертних систем з навчальними дистанційними системами в рамках єдиної архітектури інтегрованої експертної системи, що поєднує в собі взаємодіючі логіко-лінгвістичні, математичні, імітаційні й деякі інші види моделей, сприяє виникненню нових адаптивних та інтелектуальних навчальних середовищ.

На сьогодні існує безліч чудових прикладів Web-систем для організації дистанційного навчання, серед яких Blackboard, WebCt, Moodle, IBM LearningSpace, проте вони не використовують модель студента, що знижує якість навчального процесу й не дозволяє організувати адаптивне навчання. Необхідно підкреслити, що саме модель користувача враховує механізм адаптації при формуванні адаптивних сценаріїв навчання та тестування. Варто згадати також про науковий напрям у дослідженнях «штучний інтелект у навчанні» – нова методологія психологічних, дидактичних і педагогічних досліджень з моделювання поведінки людини в процесі навчання на основі методів інженерії знань. У зв'язку з цим перспективними й актуальними є розробки адаптивних інтелектуальних навчальних систем на базі методів штучного інтелекту й Інтернет-технологій.

У даній статті розглядаються наступні завдання:

- створення моделі студентів на основі параметрів рівня підготовки та когнітивних особливостей (процесів запам'ятовування та забування);
- розробка моделі подачі навчального матеріалу в дистанційній адаптивній навчальній системі в залежності від кривих забування користувачів;
- створення технології побудови індивідуальних сценаріїв навчання на основі карт прогалин знань користувачів і технології побудови інтелектуальних модулів дистанційної адаптивної навчальної системи з використанням мереж Петрі.

1. Моделі студентів на основі параметрів рівня підготовки та когнітивних особливостей

Процеси запам'ятовування і забування відіграють у процесі навчання важливу роль. Пам'ять є одним з найважливіших психічних процесів, що реалізує засвоєння знань.

Закон забування осмисленого матеріалу представимо апроксимуючою логарифмічною функцією виду

$$f(t) = a \cdot \ln(t) + b,$$

де t – час, що минув з моменту повного оволодіння матеріалом; a, b – параметри, які характеризують індивідуальні характеристики пам'яті студента і визначаються методом найменших квадратів за індивідуальною статистикою на основі 3-4 тестувань протягом певного часу.

Зважаючи на величину достовірності апроксимації, для апроксимації даних у зоні засвоєння (рис. 1) найбільше підходить експоненціаль-

на залежність обсягу засвоєного матеріалу від часу (рис. 2).

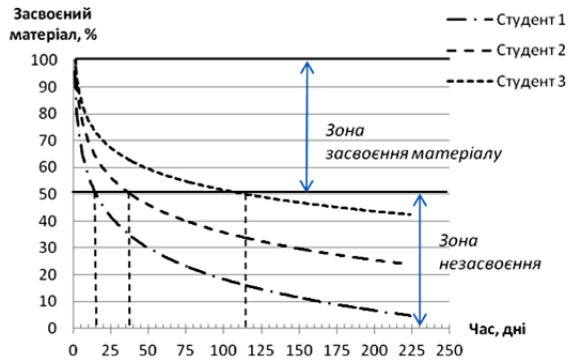


Рис. 1. Індивідуальні криві забування студентів

Експоненціальні рівняння кривих забування студентів мають вигляд:

Студент 1: $f(t) = 104{,}92 \cdot e^{-0{,}051t}$

Студент 2: $f(t) = 99{,}19 \cdot e^{-0{,}019t}$

Студент 3: $f(t) = 94{,}254 \cdot e^{-0{,}006t}$

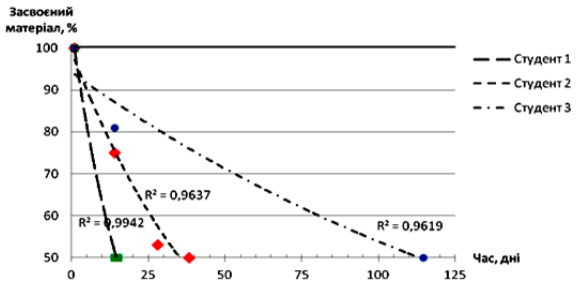


Рис. 2. Експоненціальна апроксимація експериментальних даних

На основі індивідуальних кривих забування студентів адаптивна система формує календарний план повторення тем, визначаючи час, коли кожному студенту необхідно повторити вивчений матеріал. У певний день студенту надається нагадування з посиланням на тему для повторення.

2. Карти прогалин знань користувачів

Методи, які сьогодні застосовуються для запам'ятовування інформації, мають невисоку ефективність («заучування», лінійне конспектування, багаторазове повторення). Тому виникла нова техніка запам'ятовування інформації, яка з'явилась на стику психології та інформатики – майндмеппінг (mind mapping), що перекладається як «карта розуму» або «ментальна карта», або «інтелект-карта». Це принципово новий спосіб аналітичного представлення інформації, заснований на графічному відображенні асоціативних або логічних зв'язків. Методика створення інтелект-карт є альтернативним методом наочного представлення теоретичного матеріалу. Формально інтелект-карта є моделлю знань викладача певної теми й навчального курсу загалом і своєрідним відображенням предметної області. Використання інтелект-карт дозволяє охопити всю ситуацію в цілому, а також утримувати одночасно у пам'яті велику кількість інформації для знаходження зв'язків між окремими елементами, запам'ятовування інформації та відтворення її навіть через довгий період часу. Формалізація інтелект-карт у семантичні моделі даних надає можливості автоматизації процесів контролю знань за допомогою порівняння знань студентів з еталонними інтелект-картами та побудови карт прогалин знань. Саме карти прогалин знань (рис. 3) є якісним показником засвоєння знань студентами.

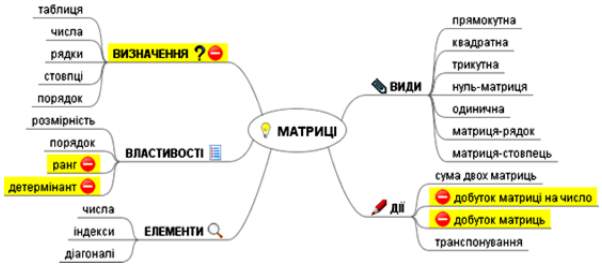


Рис. 3. Карта прогалин знань студента з теми «Матриці»

3. Технологія побудови індивідуального сценарію адаптивного тестування користувачів

Під адаптивним тестуванням розуміють широкий клас методик тестування, які передбачають зміну послідовності подання завдань у самому процесі тестування з врахуванням відповідей студента на вже надані завдання. В основі адаптивного підходу лежить індивідуалізація процедури відбору завдань тесту, яка за рахунок оптимізації складності завдань відповідно до рівня підготованості студентів забезпечує генерацію ефективних тестів [1]. Іншими словами, основна ідея адаптивного тестування полягає в тому, що тестові завдання необхідно адаптувати за складністю до рівня підготовки студента, при цьому підбір завдань витікає з міркувань, що слабким студентам не варто давати складні завдання, тому що з більшою ймовірністю вони не зможуть виконати їх правильно. Також і легкі завдання неефективно давати сильним студентам.

Під сценарієм адаптивного тестування будемо розуміти індивідуальний набір тестових завдань з різними рівнями складності, кожний тест якого обирається для кожного студента в залежності від його відповіді на попереднє запитання.

У цілому алгоритм комп'ютерного адаптивного тестування складається з наступних етапів:

1. З банку завдань вибирається відповідне за параметром завдання.
2. Вибране завдання пред'являється студенту, який відповідає на нього правильно або неправильно.
3. Оцінка тестованої здатності оновлюється на підставі цієї відповіді.

Як описано у роботі [2] попередні три кроки повторюються доти, поки згідно з певним критерієм оцінка вимірюваної якості визнається задовільною, і відбувається вихід з тестів.

Моделі на основі ймовірнісних критеріїв, які належать до сучасної теорії тестування IRT, успішно використовуються при адаптивному тестуванні. На наш погляд, для оцінки параметрів рівня знань і складності завдань у адаптивному дистанційному тесті найбільш підходить однопараметрична модель Г. Раша:

$$P_j(\theta)=\frac{e^{1,7(\theta-\beta_j)}}{1+e^{1,7(\theta-\beta_j)}};$$

$$P_i(\beta)=\frac{e^{1,7(\theta_i-\beta)}}{1+e^{1,7(\theta_i-\beta)}},$$

де θ (латентний параметр, який визначає рівень підготовки студента) і β (латентний параметр, який визначає складність завдання тесту) – незалежні змінні для першої і другої функцій відповідно.

Однопараметрична модель Раша є однією із сімейства логістичних кривих. Звичайно у процесі розробки тесту оцінюють обидва латентні параметри θ і β, які відповідно називають логітами рівня знань і логітами рівня складності завдань.

Логіт рівня знань – натуральний логарифм відношення частки правильних відповідей студента на всі завдання тесту, до частки неправильних.

Логіт рівня складності завдання – натуральний логарифм відношення частки неправильних відповідей на завдання до частки правильних відповідей на це завдання по множині студентів.

Оцінка цих параметрів проводиться з припущення нормальності розподілу емпіричних даних тестування по множині як студентів, так і завдань тесту. Нормально розподіленими вважаються і значення латентних змінних.

Оскільки при реалізації комп'ютеризованого тестування на основі моделі Раша є деякі труднощі з обробкою даних, необхідно її модифікувати.

Модифікація моделі Раша запропонована в роботі [3]:

1. Для оцінки параметрів складності завдань і рівня підготовки використовують матрицю результатів тестування, яка задається таблицю, у рядках якої розташовані студенти, у стовпцях – номера завдань. Далі підводяться підсумки по кожному рядку та стовпцю, і матриця впорядковується та аналізується. При застосуванні адаптивних тестів, які за певним критерієм обираються з банку тестових завдань, виникають труднощі, адже матриця заповнюється не повністю. Кожен студент тестується за своїм індивідуальним сценарієм. Одне і те ж запитання може мати різний порядковий номер, кількість запитань у тесті також індивідуальна для кожного студента, через що тестові завдання мають різну кількість відповідей на них. Тому аналіз рівня підготовки та складності завдань необхідно проводити окремо. Номер тестового завдання визначається як номер даного тесту в банку тестових завдань.

2. Частки правильних і неправильних відповідей розраховуються відносно кількості завдань, на які давав відповіді студент, а не загальної кількості запитань у тесті:

$$p_i=\frac{x_i}{n},$$

де p_i – частка правильних відповідей i-го студента; x_i – кількість правильних відповідей; n – кількість завдань, на які відповідав i-й студент; i – номер студента.

Аналогічно для складності завдань відносно кількості студентів, які на нього відповіли, а не загальної кількості студентів:

$$p_j=\frac{R_j}{N},$$

де p_j – частка правильних відповідей на j-е завдання; R_j – кількість студентів, що виконали j-е завдання вірно; N – число студентів, які виконували j-е завдання; j – номер тестового завдання.

Для об'єктивної оцінки складності завдань необхідно накопичити статистику по кожному завданню.

3. Модель Раша для підбору тестових завдань застосовується з моменту появи неправи-

льної або правильної відповіді, і для цього питання обчислюється параметр рівня знань. Для випадків, коли студент підряд правильно або неправильно відповідає на запитання тесту, то зміна рівня складності відбувається таким чином:

- при правильній відповіді на запитання ймовірність відповіді на завдання більш високого рівня складності прирівнюється до 0.7 і відбувається перехід до завдань більшого рівня складності, якщо рівень складності не найвищий; у разі найвищого рівня складності студент залишається на цьому рівні;
- при правильній відповіді на запитання ймовірність відповіді на завдання більш високого рівня складності прирівнюється до нуля та відбувається перехід до завдань нижчого рівня складності, якщо рівень складності не найнижчий; у разі найнижчого рівня складності студент залишається на цьому рівні.

4. Якщо студент правильно (неправильно) відповів на усі запитання тесту застосовується модель наведена у п. 3.

Об'єктивно оцінити складність завдань при невеликих вибірках неможливо, тому далі пропонується алгоритм переходу від дихотомічної шкали рівнів знань до інтервальної шкали логітів.

Оцінки латентних параметрів складності завдань звичайно лежать в інтервалі (-5; 5) і мають декілька знаків після коми, а також можуть приймати від'ємні значення, що дозволяє розробити формули перетворення шкал логітів. Лінійним перетворенням надається більша перевага, оскільки вони зберігають інтервальний характер шкали. Найбільш поширеним є перетворення, запропоноване Чопіном:

$$\theta_i = 50 + 4{,}55\theta,$$
$$\beta_j = 50 + 4{,}55\beta. \qquad (1)$$

У результаті цих перетворень отримуємо додатні значення параметрів θ і β, що розташовані в інтервалі (30;70), які округлюються до цілих.

Отже, з формул (1) можна вивести обернені для переходу від звичайної шкали завдань з різними рівнями у логіти.

Різниця між границями інтервалу (30,70) дорівнює 40. Припустимо, викладач обирає 4 рівня складності для тестових завдань ($z_j = \overline{1,4}$). Тоді найлегшому рівню складності z_j буде відповідати інтервал від (30;40), z_j – (40;50), z_j – (50;60), z_j – (60,70).

З формули (2.4) маємо

$$\beta = \frac{\beta_j - 50}{4{,}55}. \qquad (2)$$

Для шкалювання звичайних рівнів складності в логіти пропонуємо брати у якості β_j ліві границі інтервалів z_j тобто найлегше завдання у шкалі логітів за формулою (2) буде мати наступне значення:

$$\beta = \frac{30 - 50}{4{,}55} = -4{,}3956.$$

Таким чином, враховуючи п. 2, для випадку нестачі статистики (невелика кількість студентів, велика кількість тестових завдань) після розподілу викладачем тестових завдань за рівнями складності ($z_j = \overline{1,z}$) і перетворення їх у логіти складності завдань, необхідно визначити лише значення логітів рівня знань студентів θ.

4. Модель адаптивного дистанційного навчального курсу на основі мереж Петрі

Для моделювання процесу адаптивного навчання та тестування зручно використовувати теорію мереж Петрі [4].

Об'єктами моделі є студенти, які вивчають навчальний курс. Вузли мережі Петрі інтерпретуються як навчальні фрейми. Переходи маркерів показують прогрес користувача у вивченні курсу. Кожний перехід відповідає певному етапу навчального процесу. Спрацьовування переходу інтерпретується як виконання деякого навчального завдання. Навчальним фреймом є будь-який тип завдань, які студент повинен виконати під час навчання.

Формально модель процесу навчання описується кортежем $M = \langle P, T, F, C, M_0, cf, h, s \rangle$,

де $P = \{p_1, p_2, ..., p_n\}$ – множина вузлів p_i, кожному вузлу відповідає етап процесу навчання (вивчення теоретичного блоку, виконання тестового завдання), n – кількість вузлів;

$T = \{t_1, t_2, ..., t_k\}$ – множина переходів t_i, перехід відповідає меті навчання, спрацювання переходу інтерпретується як виконання навчального завдання, k – кількість переходів;

$F = P \times T \cup T \times P$ – відношення інцидентності, яке визначає множину дуг, спрямованих від вузлів до переходів і від переходів до вузлів;

$C = \{c_1, c_2, ..., c_v\}$ – множина кольорів c_i мережі Петрі, v – кількість кольорів;

$M_0 : P \to C^{()}$ – функція, яка задає початкову розмітку мережі Петрі;

$cf : F \to C$ – функція, яка задає вирази для дуг мережі Петрі;

$h : F \times M$ – функція, що задає правила спрацювання переходів;

$M = P \times C^{()}$ – множина всіх можливих розміток мережі Петрі;

$s : S \times T \to (-\infty, +\infty)$ – функція, яка задає статистичні дані S для кожного переходу.

Визначення кольору об'єктів мережі задається як

$$c_i = \langle Id, b, l, t_z \rangle, ,$$

де $Id = \{1,..., m\}$ – компонента кольору для ідентифікації студента, m – число студентів; b – компонента кольору для підрахунку загальної кількості балів студента; l – компонента кольору для врахування рівня підготовки студента; t_z – компонента кольору для врахування часу забування навчального фрейму.

Статистичні дані для кожного переходу задаються як

$$S = \{m, th, fr, type, p, status, kw, Rz, t_z, t_{\max}, z\},$$

де m – номер модуля; th – номер теми; fr – номер навчального фрейму теми; $type$ – тип навчального фрейму, який може приймати значення

$$type_j = \{"теоретичний", "тест", "завдання"\};$$

p – пріоритет навчального фрейму; $status$ – стан, у якому перебуває студент відносно навчального фрейму, який може приймати значення

$$status(type) = \begin{cases} \{"Вивчено", "Повторено", "Не вивчено"\}, якщо\ type="теоретичний"; \\ \{"Пройдено", "Пройдено\ з\ помилками", "Не\ пройдено"\}, \\ якщо\ type="тест"; \\ \{"Виконано", "Не\ виконано"\},\ якщо\ type="завдання" \end{cases}$$

kw – ключове поняття навчального фрейму;

Rz – бал за виконання навчального фрейму;

t_z – час забування студентом навчального фрейму; $t_{\max}$ – максимальний час на виконання навчального фрейму; z – рівень складності навчального фрейму.

Розглянемо навчальний курс, що містить сім модулів (M1-M7). На рис. 4 наведено модель цього курсу у вигляді розфарбованої мережі Петрі верхнього рівня. Кожний модуль представлений у вигляді переходу і відповідного вузла [5].

Рис. 4. Модель навчального курсу (мережа Петрі верхнього рівня)

Кожному переходу відповідає мережа Петрі 2-ого рівня (рис. 5), яка описує виконання відповідного модуля, тобто кожен з семи модулів містить теоретичний матеріал, який розбито на блоки – окремі незалежні частини (M1.1, …, M1.N, де N – кількість блоків (M1.1 – перший блок першого модуля)), а також тестові завдання з кожного модуля (MK1, …, MK7).

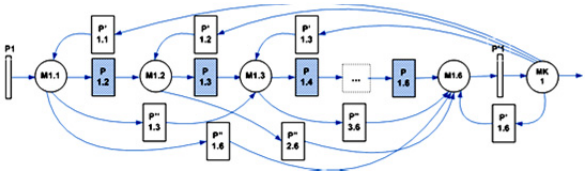


Рис. 5. Модель модуля M1 (мережа Петрі нижнього рівня)

Модуль M1 містить 6 теоретичних блоків (M1.1, …, M1.6) і тестовий контроль MK1. Переходи детерміновані та залежать від об'єму навчального матеріалу модулів та індивідуальних особливостей пам'яті студента.

Після проходження модулів M1.1, …, M1.6 студент повинен пройти тестовий блок MK1, який складений таким чином, що включає завдання з кожного блоку M1.1, …, M1.6. На основі результатів тестування MK1 визначається рівень засвоєння кожного теоретичного модулю. При недостатньому засвоєнні кожного модуля M1.1, …, M1.6 студент повертається на початок модуля 1 – M1.1 і розпочинає вивчення цього модуля спочатку. При недостатньому засвоєнні деяких модулів M1.1, …, M1.6 для студента формується індивідуальний сценарій подальшого вивчення наступного модуля з додаванням блоків повторення недостатньо засвоєних з M1.1, …, M1.6. Таким же чином моделюються всі інші модулі дисципліни.

Поточні модульні тести дозволяють здійснити перевірку знань студента блоків теоретичного матеріалу та сформувати індивідуальний сценарій навчання для кожного студента, на основі якого здійснюється адаптація системи подачі нового матеріалу та блоків повторення забутого матеріалу.

Далі розглянемо більш детально модель процесу адаптації, тобто переходу студента з одного рівня складності вивчення матеріалу на інший.

Під адаптацією в теорії управління розуміють [6] «процес зміни параметрів і структури системи, а можливо, і дій, що управляють, на основі поточної інформації з метою досягнення визначеного, звичайно оптимального, стану системи при початковій невизначеності й умовах роботи, що змінюються». Застосовуючи це визначення до процесу навчання, можна сказати, що адаптація в навчальній системі – процес зміни параметрів і структури моделі об'єкту (студента) і навчальних дій на основі поточної інформації, що отримується в ході навчання, з метою досягнення оптимального стану об'єкту при його початковій невизначеності в середовищі, що змінюється. Початкова невизначеність пов'язана з майже повною відсутністю в навчальній системі інформації про студента. При цьому, якщо взяти за основу мережу Петрі верхнього рівня і зосередитися лише на моделюванні процесу переходу між рівнями складності, то ми отримаємо звичайну мережу Петрі, зображену на рис. 6.

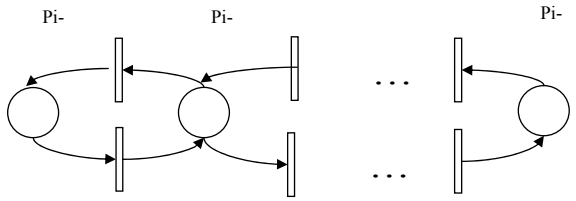


Рис. 6. Процес зміни рівня складності тестових завдань

На основі вище наведених положень адаптивна дистанційна система формує унікальні сценарії тем і їх блоків (фреймів) для кожного студента, тобто вибір наступного навчального матеріалу з бази знань залежить від певних характеристик моделі студента (рівня підготовки, результатів тестування, карти прогалин знань, часу забування).

5. Проектування дистанційної адаптивної навчальної системи

Дистанційна адаптивна навчальна система DAOS (Distance Adaptive Open System) – це складна, багатокомпонентна система, що має тенденцію до безперервних змін і динамічного розвитку. Ефективна робота такої системи не може спиратися лише на емпіричний досвід і інтуїцію розробників. Необхідно застосування системного підходу до адаптивного дистанційного навчання з врахуванням залежності складових системи – підсистем і компонентів. При проектуванні системи DAOS необхідно розглядати ряд моделей з метою якнайповнішого уявлення про проектуємо систему дистанційного навчання не лише з точки зору поточних вимог і можливостей, але і перспектив розвитку.

У зв'язку з цим виділимо наступні складові дистанційної адаптивної навчальної системи:

1) структурна;
2) функціональна;
3) інформаційно-технологічна;
4) еволюційна.

Структурна модель системи базується на структурі організації, у якій вона розроблена і застосовується. Компоненти системи відповідають підрозділам, що беруть участь у процесі дистанційного навчання. Наприклад, приймальна комісія, навчальний відділ, кафедри.

Функціональна модель призначена для вивчення особливостей роботи (функціонування) системи і її призначення у взаємозв'язку з внутрішніми і зовнішніми елементами. Функціональна модель DAOS поділяє систему на компоненти за функціональними ознаками. Наприклад, теоретичне навчання, тестування, спілкування тощо.

Інформаційно-технологічна модель може відноситися як до визначення потоків інформації в системі, так і до технологій, використовуваних у процесі обстеження і технічного проектування системи.

Еволюційна модель відображає розвиток системи в часі, а також перспективи розвитку системи в майбутньому.

Більшість пакетів дистанційного навчання розробляються на основі функціональної моделі (рис. 7), яка й використовується у дистанційній адаптивній навчальній системі DAOS.

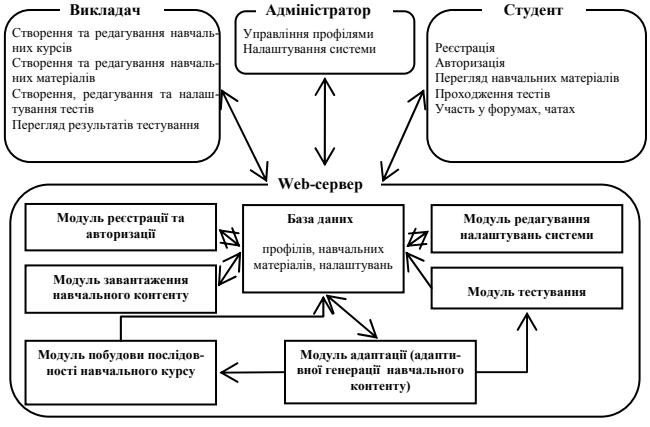


Рис. 7. Функціональна модель дистанційної адаптивної навчальної системи

6. Реалізації дистанційної адаптивної навчальної системи

Система DAOS заснована на трирівневій архітектурі та містить клієнт, сервер додатків і базу даних.

Компоненти системи:

- клієнт – довільний браузер;
- сервер додатків базується на Java технологіях для підтримки крос-платформеності;
- база даних – PostgreSQL, MySQL або Apache Derby.

Як протокол прикладного рівня використовується HTTP/HTTPS.

Клієнт виконує наступні функції:

1. Адміністративні завдання: управління користувачами, складання тестів, експортування або імпортування даних, уведення супровідного теоретичного матеріалу.
2. Підтримка тестування.
3. Підтримка інтерактивності системи та аутентифікації користувачів.
4. Виведення результатів проміжних повідомлень.

Функції сервера додатків:

1. Побудова на основі завантажених моделей правил проходження тестів (черговість, вибір типа тестів, вибір рівня складності, перехід з одного рівня складності на інший, врахування інформації про результати вже пройдених тестових завдань).
2. Оброблення та збереження в базу даних результатів вивчення і повторення супровідного теоретичного матеріалу та результатів тестування.
3. Формування індивідуальних сценаріїв вивчення теоретичного матеріалу на основі результатів тестування та кривих забування.
4. Генерування звітів про успішність, побудова графіків, таблиць.

База даних призначена для зберігання:

- інформації про користувачів;
- інформації про моделі (сценарії навчання);
- результатів тестування;
- налаштувань системи;
- супровідного теоретичного матеріалу і тестів з варіантами відповідей.

Робота з системою розпочинається з реєстрації і в подальшому авторизації, після чого викладач і студент можуть увійти до системи. Профілі викладачів в систему вносить адміністратор. Після створення адміністратором профілю «Викладач» (автор навчальних курсів) отримує обліковий запис з правами викладача. Новий користувач «Студент» може бути зареєстрований безпосередньо на сайті. Коли доступ до інформації буде дозволений, користувачі системи можуть проглянути список навчальних курсів, тестів і супровідного навчального матеріалу по них.

Викладач створює навчальний матеріал і завантажує в систему через спеціально розроблений інтерфейс.

Для вивчення навчальних курсів або проходження тестів студенту необхідно обрати (рис. 8) факультет, освітньо-кваліфікаційний рівень (бакалавр, магістр), напрям підготовки, викладача, навчальний курс і вид навчального контенту (лекції, лабораторні, практичні, тести тощо).

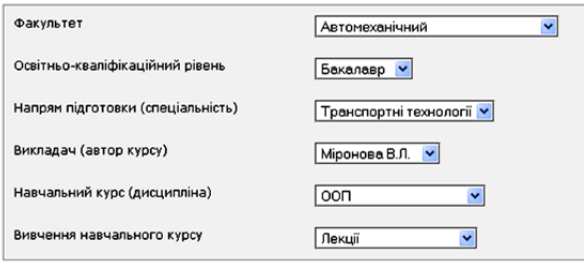


Рис. 8. Вибір навчального курсу

Інтерфейс перегляду та додавання навчального контенту для різних груп користувачів різний. Для групи користувачів «Студент» є лише можливість перегляду навчальних матеріалів, а для групи «Викладач» додана можливість їх редагування.

Користувач групи «Викладач» після входу в систему отримує доступ до бази даних навчальних матеріалів і статистики. Для кожного запитання тесту в системі тестування зберігається статистична інформація відносно відповіді на нього.

Робота студента з системою починається з авторизації, після вибору навчального курсу можна приступати до процесу тестування. Закриття програмного забезпечення або розрив зв'язку означатиме не проходження тесту.

Викладач має можливість отримати з сервера системи тестові завдання, відредагувати їх або розробити новий тест (рис. 9) і зберегти його в базу даних.

Висновки

Запропонована інформаційна технологія створення адаптивних індивідуальних сценаріїв навчання користувачів, яка враховує індивідуа-

льні характеристики запам'ятовування інформації користувачами а також результати тестування та виконання тематичних, модульних контрольних робіт.

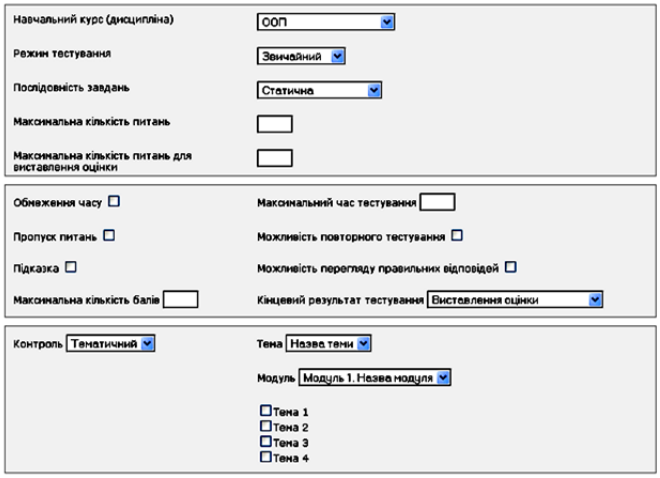


Рис. 9. Налаштування тесту

Використання карти прогалин знань з тем і інформації про кількість повторно пройдених навчальних фреймів дозволило розробити ефективний алгоритм побудови сценаріїв адаптивного тестування.

Для моделювання процесу адаптивного навчання та тестування запропонована багаторівнева мережа Петрі, яка використовується як функціональна модель дистанційної навчальної системи. Адаптивна система формує унікальні сценарії навчання для кожного студента, в залежності від певних характеристик моделі студента (рівня підготовки, результатів тестування, карти прогалин знань, часу забування).

У подальших дослідженнях при експлуатації системи варто отримати статистичні експериментальні оцінки для порівняння з теоретичними результатами.

Список літератури

1. Звонников В.И., Челышкова М.Б. Современные средства оценивания результатов обучения. – М.: Издательский центр «Академия», 2007. – 224 с.
2. Адаптивное тестирование как метод контроля знаний студентов. [Електронний ресурс]. – Режим доступу: http://psysoft.su/articles/72-adaptive-tests-as-method-of-control
3. Погребнюк І.М. Модель оцінки знань при адаптивному тестуванні // Шоста науково-практична конференція з міжнародною участю „Математичне та імітаційне моделювання систем. МОДС 2011. Тези доповідей. – Чернігів. – 2011. – С. 378-382.
4. Федорук П.И. Организация процесса индивидуализированного обучения на базе адаптивной системы дистанционного обучения и контроля знаний EduPro // Information models of knowledge – ITHEA, Kiev – Sofia, 2010. – С. 335-341.
5. Доррер А.Г. Динамическое моделирование процесса интерактивного обучения. / А.Г. Доррер // Материалы Всероссийской научно-практической конференции «Лесной и химический комплексы: проблемы и решения». – Красноярск: СибГТУ. – 2005. – С.253-258.
6. Растригин Л.А. Адаптивное обучение с моделью обучаемого. [Текст] / Л.А. Растригин, М.Х. Эренштейн. – Рига: Зинатне, 1988. – 160 с.

УДК 004.04+ 004.4242

ЯСІНСЬКИЙ В.В.,
БОЛДАК А.О.

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ЗБОРУ, ПОПЕРЕДНЬОЇ ОБРОБКИ ТА АНАЛІТИЧНОГО ОПРАЦЮВАННЯ РЕЗУЛЬТАТІВ МОНІТОРИНГУ ЯКОСТІ ПІДГОТОВКИ ФАХІВЦІВ

В роботі розглядається інформаційна система для збору, попередньої обробки та аналітичного опрацювання результатів моніторингу якості підготовки фахівців у ВНЗ, засоби якої використовуються в інституті моніторингу якості освіти НТУУ «КПІ» для виявлення латентних залежностей, оцінки узгодженості теоретичних моделей з наявними результатами моніторингу та уточнення моделей еволюції системи залишкових знань, що суттєво зменшує невизначеність при прийнятті рішень, пов'язаних з управлінням такою великою навчальною системою як НТУУ «КПІ»

In the work the information system of pretreatment and analytical processing of monitoring results of training quality in institution of higher education is considered. Its product are used in the institute of education quality monitoring NTUU KPI for identify latent relationships, monitoring compliance of existing theoretical models with monitoring results and refining models of the evolution of residual knowledge. This (This approach) significantly reduces the uncertainty when making decisions, associated with management of such large educational system as NTUU KPI

Вступ

Пошук можливих шляхів модернізації сучасної вищої школи в умовах становлення інформаційного суспільства зумовлює важливість науково-практичної проблеми системного формування концептуальних засад такої методології управління навчальним процесом, яка б дозволила на основі синергетичного підходу принципово по-новому вибудувати саму стратегію підготовки фахівців у вищих навчальних закладах.

В контексті вирішення сформульованої проблеми особливе місце належить «колективній» системі навчальних знань – інтегральному продукту діяльності великої навчальної системи, в якій інформаційна взаємодія її структурно взаємозв'язаних і функціонально взаємозалежних компонент (студентське, викладацьке, соціальне середовища та підсистема комплексного моніторингу якості підготовки фахівців) визначена як ключовий фактор синергії формування системи «колективних» залишкових знань [1]. Фундаментальна роль, відведена підсистемі комплексного моніторингу якості підготовки фахівців в управлінні великою навчальною системою, пов'язана з необхідністю оцінки стану системи «колективних» залишкових знань при формуванні інформаційних потоків взаємодії та управляючих впливів в тому числі. Зрозуміло, що адекватність такої оцінки є визначальною для ефективного управління великою навчальною системою.

Мета роботи полягає у підвищенні ефективності та адекватності оцінки стану системи «колективних» залишкових знань з боку підсистеми комплексного моніторингу якості підготовки фахівців за рахунок розробки та використання інформаційної системи для збору, попередньої обробки та аналітичного опрацювання результатів моніторингу.

Вимоги до інформаційної системи

Архітектура розроблюваної інформаційної системи повинна враховувати вимоги, які до неї висуваються. Такі вимоги можна віднести до двох категорій: перша категорія пов'язана з цільовим призначенням системи та вимогами користувачів, друга – визначає можливості швидкої розробки та компоновки як інформаційної системи в цілому, так і її компонент[2].

У відповідності до цільового призначення системи, пов'язаного з адекватною оцінкою системи «колективних» залишкових знань, до неї висуваються вимоги щодо здійснення оцінки узгодженості теоретичних моделей [3] з результатами експериментальних досліджень та можливості розширення базових теоретичних моделей за рахунок дослідження та включення різноманітних важко формалізованих факторів, в тому числі тих, що обумовлені «конфліктами»

та «мотиваціями», які виникають у великій навчальній системі на мікро- та макрорівні.

Тому в розроблюваній системі передбачається наявність засобів чисельного моделювання задач теорії крайових ефектів в композитних системах із застосуванням концепції локальних факторів [4-5], які призначені для дослідження процесів виникнення та розвитку «конфліктів» та «мотивацій» на макрорівні. Такі засоби призначені для розв'язку наступних задач.

1. Побудова дискретної моделі вихідної задачі з використанням методів дискретизації континуальної моделі (метода скінчених елементів, метода скінчених різниць і т. д.)
2. Апріорний аналіз властивостей дискретної моделі (точність, збіжність, похибка апроксимації і т. д.)
3. Розв'язання дискретної задачі з використанням чисельних методів. При цьому вибір метода здійснюється, виходячи з властивостей оператора або матриці дискретної задачі.
4. Аналіз результатів розрахунків з подальшим уточненням алгоритму системи чисельного моделювання і параметрів моделювання вихідної дискретної моделі в рамках обчислювального експерименту.

З метою забезпечення можливості побудови адекватних моделей процесів виникнення та розвитку «конфліктів» та «мотивацій» на мікрорівні в системі передбачені засоби імітаційного моделювання багатоетапних переговорних процесів із заданими пріоритетами, інтересами, правами та можливостями сторін з використанням методології ігор з ієрархічною структурою Геймейєра-Мойсєєва та принципу максимального гарантованого результату.

Можливості системи, пов'язані з виявленням латентних залежностей, оцінкою узгодженості теоретичних моделей з наявними результатами моніторингу та уточненням моделей еволюції знань, забезпечуються засобами аналітичного опрацювання даних. Такі засоби призначені для вирішення задач технологічного забезпечення повного життєвого циклу даних, від їх попередньої обробки до аналізу і візуалізації.

З боку користувача розроблювана система повинна відповідати наступним вимогам:

• бути здатною обробляти великі об'єми інформації та обробляти дані у «сирому» форматі, також система повинна мати розвинуті засоби взаємодії з різноманітними джерелами та сховищами даних;

• система повинна бути доступною широкому кругу користувачів різних категорій та як найкраще задовольняти їх вимоги до функціональності системи та інтерфейсу користувача;

• система повинна мати засоби автоматизації генерації звітів з використанням текстової та графічної інформації, які можуть експортуватися в різні формати документів.

Системні вимоги полягають в наступному:

• система повинна за потребою забезпечувати збільшення обчислювальних ресурсів та об'ємів збереження даних;

• внутрішні механізми обробки повинні бути приховані від користувача, який може мати доступ тільки до кінцевого сервісу, що надається йому системою;

• архітектура та програмні механізм системи повинні передбачати можливості додавання нової функціональності та можливості реконфігурації системи без перекомпіляції її вихідного коду.

Таким чином, сформульовані вимоги до інформаційної системи для збору, попередньої обробки та аналітичного опрацювання результатів моніторингу дозволяють говорити про неї як про відкриту, розподілену та складну програмну систему.

Сервіс-орієнтована архітектура інформаційної системи

Вибір сервісно-орієнтованої архітектури [6] обумовлено тим, що вона передбачає використання модульного підходу до розробки програмного забезпечення, основаного на використанні слабко зв'язаних сервісів $\{S_i\}$, $i=\overline{1,n}$, що об'єднані загальним комунікаційним механізмом $\tilde{I}_G$. Це дозволяє багаторазово використовувати сервіси для вирішення різноманітних прикладних задач. Тобто, $(\{S_i\},\tilde{I}_G)$ – є замкненою відносно інтерфейсу $\tilde{I}_G$ системою, визначеною на множині сервісів $\{S_i\}$.

В інформаційній системі, побудованій за цими принципами, для користувача загальний комунікаційний механізм (системна шина) реалізує концепцію єдиної точки доступу до додатків. Це передбачає створення диспетчера $\tilde{I}_G \leftrightarrow \boldsymbol{D}$, який несе відповідальність за доставку запитів та результатів їх обробки [7].

Таким чином, основу розроблюваної інформаційної системи складає програмний каркас, який забезпечує функції комунікаційного середовища, та в рамках якого визначаються уніфіковані програмні інтерфейси і протоколи взаємодії програмних підсистем. В цьому випадку ключовим аспектом є ефективна реалізація програмних механізмів, які забезпечують уніфікацію підсистем, уніфікацію методів доступу до даних, можливість їх хешування тощо. При реалізації таких механізмів доцільно використовувати типові рішення рівня структурного проектування – шаблони [8].

Так для приховування внутрішніх механізмів підсистем застосовано шаблон «Фасад» $\boldsymbol{F}$: $\tilde{I}_G \leftrightarrow \boldsymbol{F} \leftrightarrow \tilde{I}_S \leftrightarrow S$, $\tilde{I}_G \subseteq \tilde{I}_S$, де $\tilde{I}_S$ – внутрішній інтерфейс підсистеми S. «Фасад» використано на двох рівнях: для оформлення інтерфейсу окремих блоків в складних підсистемах та для оформлення інтерфейсу підсистем в цілому. Для інтеграції існуючих підсистем використано шаблон «Адаптер» $\boldsymbol{A}$: $\tilde{I}_G \leftrightarrow \boldsymbol{A} \leftrightarrow \tilde{I}_S \leftrightarrow S$, $\tilde{I}_G \cap \tilde{I}_S = \varnothing$, де в якості клієнта виступає диспетчер системної шини, а в якості об'єкта, що адаптується, – існуюча підсистема S.

Уніфікація методів доступу до даних здійснюється з використанням шаблону «Міст» $\left\{\boldsymbol{B_j}\right\}$: $\tilde{I}_B \leftrightarrow \left\{\boldsymbol{B_j}\right\} \leftrightarrow \left\{\tilde{I}_{D_j}\right\} \leftrightarrow \left\{D_j\right\}, \quad j = \overline{1,k}$ який передбачає відокремлення абстрактного інтерфейсу $\tilde{I}_B$ та реалізації методів доступу $\left\{B_j\right\}$, що забезпечує можливість розробки декількох реалізацій, що не будуть відрізнятися з точки зору системи у сенсі протоколу взаємодії з джерелами даних $\left\{D_j\right\}$.

Необхідність повторної передачі та обробки великих об'ємів даних примушує використовувати спеціальні механізми хешування даних та результатів запитів, для реалізації яких використано шаблон «Замісник».

Модульна організація програмного забезпечення

Зрозуміло, що описаний на архітектурному рівні програмний каркас визначає найбільш загальні риси системи та її універсальні можливості, які задовольняють системні вимоги та вимоги користувачів. Сама ж функціональність системи, яка залежить від її цільового призначення, буде формуватися в результаті інтеграції різноманітних спеціалізованих підсистем.

Враховуючи ієрархічність структури програмного забезпечення та ґрунтуючись на концепції «швидкої» розробки програм, пропонується багаторівнева модульна організація програмного забезпечення такої інформаційної системи:

- рівень подання моделей даних, які спільно використовуються програмними засобами вищих рівнів для специфікації даних і результатів;
- рівень програмної реалізації методів попередньої обробки та аналітичного опрацювання даних, які можуть використовуватись для вирішення різноманітних задач;
- рівень програмної реалізації стадій та організації загального процесу обробки та аналітичного опрацювання даних для конкретних прикладних задач;
- рівень програмної реалізації засобів чисельного та імітаційного моделювання;
- рівень програмної реалізації спільних механізмів для доступу до даних, їх візуалізації, генерації звітів тощо;
- рівень програмної реалізації засобів побудови інтерфейсу користувача та засобів комунікації.

При наявності достатнього спектра реалізованих програмних засобів на кожному з рівнів така модульна організація дозволяє фактично «збирати» з програмних компонентів необхідні підсистеми та додатки, зводячи до мінімуму необхідність у розробці вихідного програмного коду.

Висновки

Інформаційна система для збору, попередньої обробки та аналітичного опрацювання результатів моніторингу якості підготовки фахівців у ВНЗ розроблена та впроваджена в Інституті моніторингу якості освіти Національного технічного університету України «Київський політехнічний інститут» (НТУУ «КПІ»).

Засоби цієї системи використовуються для виявлення латентних залежностей, оцінки узгодженості теоретичних моделей з результатами моніторингу та уточнення моделей еволюції знань, що суттєво зменшує невизначеність при прийнятті рішень, пов'язаних з управлінням такою великою навчальною системою як НТУУ «КПІ».

Так в результаті аналітичного опрацювання даних щосеместрового комплексного моніторингу якості підготовки фахівців в НТУУ «КПІ» (2005–2011р.р.) та моніторингу якості знань слухачів системи доуніверситетської підготовки НТУУ «КПІ» (1991–2011р.р.) (обсяг вибірки перевищив 50 000 реєстрацій), експериментально підтверджено наявність самоорганізації процесу еволюції «колективних» залишкових знань у великій навчальній системі [9-10]. Також експериментально досліджено властивості еволюції системи залишкових знань в контексті педагогічної практики та обґрунтовано можливість створення нових навчальних технологій, орієнтованих на гарантований рівень «колективних» залишкових знань.

Список літератури

1. Ясінський В. В. Системне моделювання процесів накопичення і дисипації знань http://journal.iasa.kpi.ua/pravila-oformlennya-statei-dlya-zhurnaluТекст]/ В. В. Ясінський // Системні дослідження та інформаційні технології.– 2007. – №3. – с. 111 – 121.
2. Болдак А.О. Архитектура информационной системы интеллектуальной обработки данных[Текст]/ Болдак А.О., Пустовит М.А // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2011. – № 53. – с.68 -73.
3. Ясинский В. В. Исследование процессов самоорганизации в образовательных системах на основе метода синергетического моделирования [Текст]/ В. В. Ясинский // Кибернетика и системный анализ. – 2010. – №2. – с. 161 – 174.
4. Коханенко Ю.В., Ясінський В.В. До питания про визначення напруженого стану у композитах, армованих прямокутними волокнами, на основі сіткового підходу // Доповіді НАНУ. – 1996. – №4. – С. 50 – 52.
5. Быстров В.М., Коханенко Ю.В., Ясинский В.В. Проблемно-ориентованная среда программирования и разработки дискретных моделей для некоторых классов задач механики деформируемого тела // Проблемы программирования. – 1998. – Вып.4. – С. 102 – 108.
6. Сервисная шина предприятия // Википедия. [2011-2011]. Дата обновления: 04.04.2011. URL: http://ru.wikipedia.org/?oldid=33340781 (дата обращения: 04.04.2011) http://www.w3.org/TR/ws-gloss/
7. Дональд Фергюсон, Марша Стоктон Модель программирования SOA для реализации Web-сервисов, Часть 1: Введение в модель программирования SOA [Электронный ресурс] // Фергюсон, Стоктон URL: http://www.ibm.com/developerworks/ru/library/ws-soa-progmodel/
8. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. // СПб: Питер, 2001. – 368 с.
9. Ясінський В. В. Дослідження еволюції залишкових знань у складних навчальних системах [Текст]/ В.В. Ясінський, А.О. Болдак // Системні дослідження та інформаційні технології.– 2012. – №4. – с. 120 – 132.
10. Ясінський В. В. Експериментальні дослідження властивостей залишкових знань у складних навчальних системах [Текст]/ М.З. Згуровський, В.В. Ясінський, А.О. Болдак // Матеріали III Всеукраїнської науково-практичної конференції "Інформатика та системні науки" (ІСН-2012), с. 115 – 118, 1 – 3 березня 2012р., Україна, Полтава.

УДК 004.04+ 004.4242

БОЛДАК А.О.,
ЄФРЕМОВ К.В.

ПРЕДМЕТНО-ОРІЄНТОВАНА МОВА АНАЛІТИЧНОЇ ОБРОБКИ ДАНИХ

В роботі розглядається предметно-орієнтована мова аналітичної обробки даних, використання якої для створення сценаріїв обробки даних не потребує від експертів знання інструментальних мов програмування, що суттєво знижує трудомісткість процесу розробки аналітичних звітів. Запропонована мова впроваджується в Світовому центрі даних з геоінформатики та сталого розвитку НТУУ «КПІ».

Work contains review of declarative domain-specific language for data mining, using of which doesn't demand knowledge of tool programming languages for development of data processing scenarios and that really reduces labor inputs in analytical reports development. Proposed language is implemented at ICSU World Data Center for Geoinformatics and Sustainable Development hosted by NTUU "KPI".

Вступ

Засоби аналітичної обробки даних [1], в основі якої лежать різноманітні методи статистичної обробки та аналізу даних, є потужним інструментом, що сьогодні входить до складу систем підтримки прийняття рішень. Нажаль, не дивлячись на те, що для більшості інструментальних мов програмування розроблено спеціалізовані бібліотеки для аналітичної обробки даних [2-4], ефективність використання такої обробки в межах універсальних засобів підтримки прийняття рішень залишається низькою. Одною з причин такого становища є те, що, з одного боку, розробка нових та(або) модифікація існуючих сценаріїв обробки даних здійснюється засобами інструментальної мови з використанням імперативної та об'єктно-орієнтованої парадигми програмування. З іншого боку, експерти, які повинні визначати сутність аналітичної обробки даних, а, відповідно, і розробляти сценарії, не мають належного досвіду програмування та вимушені звертатися за допомогою до програмістів, що, в свою чергу, збільшує трудомісткість створення та модифікації сценаріїв.

З огляду на зазначене вище, мета роботи полягає у зниженні трудомісткості створення та модифікації сценаріїв аналітичної обробки даних за рахунок розробки, реалізації та використання предметно-орієнтованої мови, яка не потребує від експерта спеціальних знань та навичок програмування та достатня для опису процесу перетворення даних.

Процес розробки нової предметно-орієнтованої мови складається з етапу створення абстрактного синтаксису, на якому визначається множина семантичних одиниць, етапу розробки синтаксису, на якому формально визначається подання семантичних одиниць синтаксичними, графічними або змішаними засобами, та етапу розробки правил трансформації абстрактного подання в таке, що виконується інструментальним оточенням.

Визначення семантичних одиниць

Сценарій аналітичної обробки даних є моделлю обчислювального процесу та може розглядатися як поглинаючий ланцюг Маркова з дискретними станами та автоматним часом [5]. Вважається, що в такій моделі перехід із одного стану до іншого здійснюється миттєво. Тому моменти часу, в які здійснюється перехід, пов'язують з подіями, а проміжки часу між подіями – з активністю (процесом). У відповідності до цього, сценарій може бути описаний як послідовністю подій, так і послідовністю процесів. Слід зауважити, що в моделі, орієнтованій на опис подій, припускається наявність формального апарату опису процесів, а в моделі, орієнтованій на опис процесів, навпаки, – певна формалізація подій. Використання обмеженого алфавіту подій з чітко визначеною в межах протоколу взаємодії процесів семантикою забезпечує підходу до опису сценаріїв аналітичної обробки даних, орієнтованих на процеси, переваги, пов'язані з тим, що модель подається як сукупність взаємодіючих процесів та більш адекватно (ніж сукупність подій) відображає структуру перетворень даних, які необхідно здійснити в сценарії.

Таким чином, сценарій аналітичної обробки даних є сукупність взаємодіючих процесів, кожен з яких може розглядатися як «чорна скринька»

$$P = \left\langle X, Y, X \xrightarrow{F} Y \right\rangle,$$

де X – множина вихідних даних, Y – множина результатів, $X \xrightarrow{F} Y$ – відображення, вихідних даних в результати, яке здійснює процес.

Процеси можуть використовувати дані та генерувати результати декількох типів, тому множини вхідних та вихідних даних можна подати як:

$$X = X_1 \times X_2 \times ... \times X_n$$
$$Y = Y_1 \times Y_2 \times ... \times Y_m \quad ,$$

де $X_i, i = \overline{1,n}$, $Y_j, j = \overline{1,m}$ визначають різні за семантикою та(або) типом вихідні дані і результати відповідно. Елементи вихідних даних $X_{k,i}, i = \overline{1,n}$ для процесу P_k можуть бути отримані від інших процесів P_l, які генерують $Y_{l,j}, j = \overline{1,m}$, при цьому повинна виконуватись умова суміщення типів даних

$$Y_{l,j} \subseteq X_{k,i}. \qquad (1)$$

Активація процесу P_k можлива лише у випадку, коли всі вхідні дані для процесу P_k готові, тобто отримані від процесів-попередників P_l, та при цьому не порушується умова суміщення типів даних (1). Отже, для організації взаємодії процесів достатньо однієї події готовності даних, яка ініціюється процесами після генерації вихідних даних.

Зрозуміло, що з моменту генерації результатів до моменту появи відповідної події готовності даних та активації наступного процесу існує певний проміжок часу, протягом якого проміжні дані повинні зберігатися. Слід також зауважити, що особливістю статистичної обробки даних є використання агрегатних перетворень (наприклад обчислення середнього значення, стандартного відхилення тощо), коли результат залежить від серії значень, що входять до статистичної вибірки. З огляду на це, процеси з'єднуються за допомогою черг (потоків) даних, до яких поступають результати та з яких вибираються вихідні дані у вигляді пакетів – елементів даних, з якими пов'язано тип даних. Відносно процесів черги даних можна розділити на вхідні, з яких процеси вибирають пакети вихідних даних, та вихідні, в які процеси поміщають пакети результатів обробки. Оскільки операції вибірки та поміщення в чергу ідентифікуються типом черги відносно процесів, черги можуть подаватися у неявній формі, за допомогою поняття вхідних та вихідних портів, які асоціюються з процесами. Таким чином, процеси з'єднуються в мережу (сукупність взаємодіючих процесів) за допомогою портів, а кожна пара «вихідний порт – вхідний порт» асоціюється з чергою (потоком) пакетів.

У відповідності до наявності вхідних і вихідних портів процеси можуть відноситись до трьох категорій: процеси-джерела даних, які не мають вхідних портів та призначені для вибірки даних з різних джерел способом, що виходить за рамки описуваної моделі; процеси-звіти, які не мають вихідних портів та асоціюються з виходами процесу аналітичної обробки даних; процеси-перетворення, які мають як вхідні, так і вихідні порти та призначені для перетворення даних. Оскільки процес не має іншого способу отримання даних, як вибірка пакетів з вхідних портів, виникає потреба у визначенні особливого типу процесів-джерел з постійним потоком пакетів – процесів-констант, які слугують для налаштування інших процесів.

Пакети, які є елементами потоків даних, що підлягають обробці, відносяться до певних типів даних, які є кількісними та (або) якісними оцінками властивостей деякої сукупності об'єктів $O = \{o_i\}, i = \overline{1,n}$, де o_i – ідентифікуюче значення номінальної шкали для $i-$ого об'єкту. Такі дані є відображенням виду:

$$O \xrightarrow{I_j} X^j, j = \overline{1,m}, \qquad (2)$$

де $I_j, j = \overline{1,m}$ – відображення, визначені на множині об'єктів ($D(I_j) = O$), з областями значень, що відповідають областям визначення показників X^j, тобто $E(I_j) = D(X^j)$.

Дані виду (2) можна подати в вигляді таблиці «об'єкт-властивість» [6]:

$$X = \left(x_{i,j}\right)_{i=1,j=1}^{n,m}, \qquad (3)$$

в якій рядок X_i відповідає набору значень, що характеризують властивості об'єкту o_i, а стовпчик X^j задає значення j-ого показника для всієї сукупності об'єктів. Така таблиця є однією з форм опису відношення, заданого в просторі показників

$$X \subseteq O \times E(X^1) \times ... \times E(X^m).$$

Даними для процесів можуть бути як таблиці виду (3), так і їхні перетини типу X_i та X^j, а також окремі значення $x_{i,j}$.

Таким чином, дані, що інкапсулюються в пакеті, можуть належати як до простих типів (значень), так і до структурних типів (масивів та таблиць).

Слід також зауважити, що дані, які містяться у вхідному пакеті, можуть бути структурно перетворені (бути декомпозовані, або, навпаки, об'єднані в структури) та пов'язані з іншими пакетами результуючого потоку. Також особливістю визначеного процесу обробки даних є неможливість втручання в нього, поки вхідні потоки даних не буде вичерпано. Тому, якщо необхідно виконувати аналіз результатів, то він здійснюється на окремій стадії обробки, на яку додаткову інформацію може бути передано завдяки використанню метаданих, що пов'язані з даними.

На рис.1 мовою UML [7] подано загальну схему відношень між семантичними одиницями предметно-орієнтованої мови аналітичної обробки даних, яка може бути використана для розробки відповідних інструментальних засобів.

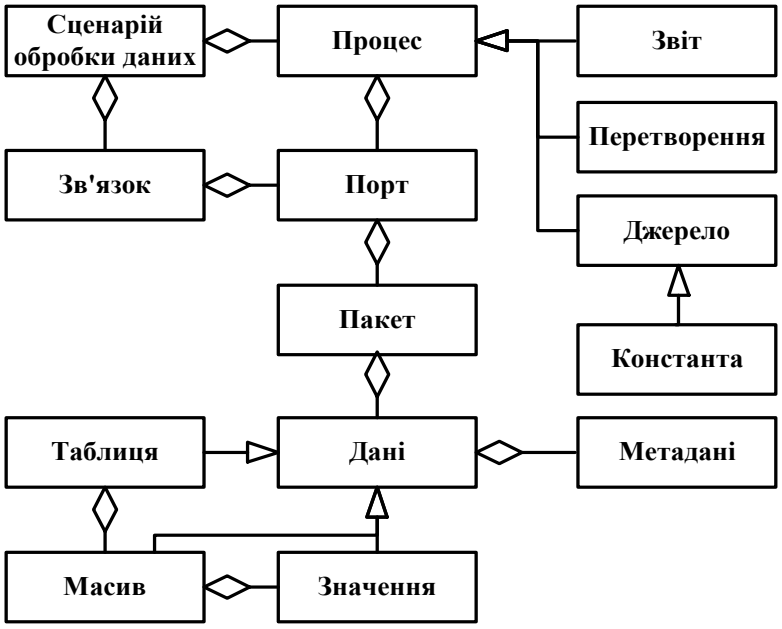


Рис.1. Схема відношень між семантичними одиницями предметно-орієнтованої мови аналітичної обробки даних.

Визначення синтаксису

Сценарій аналітичної обробки даних можна подати як граф, в якому множина вершин відповідає множені процесів, а дуги задають зв'язки між вихідними та вхідними портами. Незалежно від того, якими синтаксичними, графічними або змішаними засобами буде подано модель процесу аналітичної обробки даних, треба вирішити задачу ідентифікації класів процесів, екземплярів процесів, портів, тощо.

Кожен з процесів, які входять до складу сценарію, є екземпляром деякого класу, який визначає сукупність вхідних та вихідних портів, сутність обробки даних, що здійснюється процесом цього класу, та реалізацію класу процесу інструментальними засобами. Ідентифікація цих сутностей здійснюється у відповідності до наведених нижче правил, заданих за допомогою EBNF [8]:

```
process ::= process_name ':' (   process_type|
'{'stream'}'
process_name ::= ID
process_type ::= ID
ID ::=('a..z' | 'A..Z' | '_')('a..z' | 'A..Z'
 | '0..9' | '_')*
```

Синтаксичне правило «process» визначає операцію породження екземпляра «process_name» класу процесу «process_type». Альтернативне правило задає специфікацію процесу-константи, для якого необхідно задати

вихідний потік «stream», який асоціюється зі стандартним вихідним портом OUT.

```
stream ::= packet (';' packet)*
packet ::= metadata (',' data)*
metadata ::= 'META{' statement (','
statement)* '}'
data ::= 'DATA{' item (',' item)* '}'
statement ::= ID '=' value
item ::= value
value ::= (INT | FLOAT | STRING | structure)
STRING ::= ' " '  ~ ' \" '  ' " '
structure ::= '{' value (',' value)* '}'
```

Як бачимо з наведених правил, потік даних є послідовністю пакетів, кожен з яких може містити як метадані, так і самі дані. Метадані, в свою чергу, є мапою зі строковими ключами, а дані можуть бути як простими, так і структурованими.

Специфікація з'єднань використовує імена вхідних і вихідних портів, які належать процесам, тому повна специфікація порту включає ім'я процесу.

```
connection ::= input_port'>'output_port
input_port ::= process_name '.'port_name
output_port ::= process_name '.'port_name
port_name ::= ID
```

Остаточно, сценарій описано як мережу, задану множиною примірників процесів і зв'язків між портами.

```
network::= network_name':NETWORK{'g_stat  (';'
g_stat)* '}'
g_stat ::= process | connection
```

Нижче наведено приклад опису процесу стандартизації даних, отриманих в результаті SQL-запиту до бази даних. При цьому над даними виду (3) здійснюється перетворення [9]:

$$C_{norm}(x_{i,j}) = \frac{x_{i,j} - a}{b}, \qquad (4)$$

де $x_{i,j}$ – значення з таблиці (3), a – параметр, який задає зсув; b – параметр, який визначає масштаб нормування. Параметри a і b обчислюються згідно з формулами:

$$a = \overline{X^j} = \frac{1}{n}\sum_{i=1}^{n} x_{i,j}, \qquad (5)$$

$$b = \sigma(X^j) = \sqrt{\frac{\sum_{i=1}^{n}(x_{i,j} - \overline{X^j})^2}{n}} \qquad (6)$$

де $\overline{X^j}$ – середнє значення, а $\sigma(X^j)$ – стандартне відхилення показника X^j.

Лістинг 1. Декларативний опис процесу стандартизації даних.

```
STANDARTIZATION:NETWORK{P1:
{META{url="DBCONNECTION",sql="SQLQUERY"}};
P2:SQL; P3:AVG; P4:STDDEV; P5:AT;P6:TABLE;
P1.OUT>P2.PREFS;
P2.VAR>P3.SERIE;
P2.VAR>P4.SERIE;P2.VAR>P5.IN;
P3.VALUE>P5.SHIFT;P4.VALUE>P5.SCALE;
P5.OUT>P6.VAR;}
```

У наведеному прикладі опису сценарію обробки даних використано наступні процеси: P1 – процес-константа, який генерує вихідний потік, що містить один пакет з метаданими, які описують строку доступу до бази даних (url) та SQL-запит (sql); P2 – процес-джерело, який здійснює SQL-запит та налаштовується через порт PREFS, а його порт VAR містить потік пакетів з перетинами типу X^j, які відповідають стовпцям таблиці (3); P3 та P4 – процеси-перетворення, які обчислюють значення за формулами (5) і (6) відповідно; P5 –процес-афінне перетворення, яке здійснюється за формулою (4); P6 – процес, який з потоку змінних, що надходять через вхідний порт VAR, формує таблицю даних виду (3).

Зрозуміло, що наведений вище опис носить декларативний характер і може бути поданий за допомогою мови XML[10]:

Лістинг 2. XML-опис процесу стандартизації даних.

```
<?xml version="1.0" encoding="utf-8"?>
<network name="STANDARTIZATION">
   <process name="P1">
   <stream>
       <packet>
          <meta key="url">
             DB CONNECTION
          </meta>
          <meta key="sql">
             SQL QUERY
          </meta>
       </packet>
 </stream>
 </process>
 <process name="P2" class="SQL"/>
 <process name="P3" class="AVG"/>
 <process name="P4" class="STDDEV"/>
 <process name="P5" class="AT"/>
 <process name="P6" class="TABLE"/>
 <connection from="P1.OUT" to="P2.PREFS"/>
 <connection from="P2.VAR" to="P3.SERIE"/>
 <connection from="P2.VAR" to="P4.SERIE"/>
 <connection from="P2.VAR" to="P5.IN"/>
 <connection from="P3.VALUE" to="P5.SHIFT"/>
 <connection from="P4.VALUE" to="P5.SCALE"/>
 <connection from="P5.OUT" to="P6.VAR"/>
</network>
```

Найбільш цікавою з точки зору експертів є графічна форма подання сценарію аналітичної обробки даних, яка інструментальними засобами може бути автоматично перетворена до XML-опису. Зображена на рис.2 схема сценарію для процесу стандартизації даних демонструє такі переваги графічної форми подання, як її наочність, простоту та точність. Так на цій схемі кожному процесу відповідає умовне графічне позначення (УГП), яке ідентифіковане ім'ям та класом процесу, вхідні та вихідні порти зображено їх ідентифікаторами, що розташовані в межах УГП процесу з лівого та правого боку відповідно. Зв'язки зображено лініями зв'язку між портами у відповідності до загальних правил побудови структурних та функціональних схем.

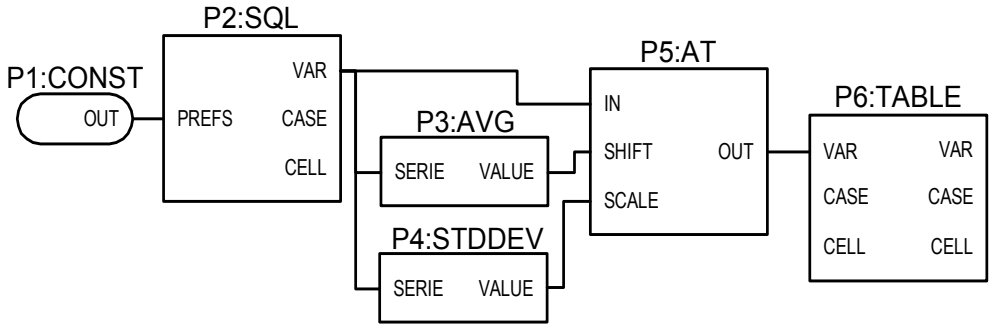


Рис.2. Схема процесу стандартизації даних.

Саме графічна форма подання є найбільш прийнятною до використання експертами з аналітичної обробки даних.

Опис процесу аналітичної обробки даних засобами імперативної інструментальної мови програмування передбачає наявність реалізації класів процесів та класу мережі, який містить всю інформацію про склад моделі та відіграє роль фасаду, в якому передбачені операції для створення процесів та зв'язків. У цьому випадку опис сценарію є лінійним алгоритмом додавання до моделі складових опису, як це показано в Лістингу 3.

Лістинг 3. Імперативна форма опису процесу стандартизації даних.

```
Network n = newNetwork( ”STANDARTIZATION”);
n.add(n.constant(
”P1”,”META{url=\”DBCONNECTION\”,sql=\”SQL
QUERY\”}”));
n.add( n.process(”P2”,SQL.class));
n.add( n.process(”P3”,AVG.class));
n.add( n.process(”P4”,STDDEV.class));
n.add( n.process(”P5”,AT.class));
n.add( n.process(”P6”,TABLE.class));
n.add( n.connection(”P1.OUT”, ”P2.PREFS”));
n.add( n.connection(”P2.VAR”, ”P3.SERIE”));
n.add( n.connection(”P2.VAR”, ”P4.SERIE”));
n.add( n.connection(”P2.VAR”, ”P5.IN”));
n.add( n.connection(”P3.VALUE”, ”P5.SHIFT”));
n.add( n.connection(”P4.VALUE”, ”P5.SCALE”));
n.add( n.connection(”P5.OUT”, ”P6.VAR”));
```

Реалізація засобів автоматизації процесу аналітичної обробки даних

Серед розглянутих засобів опису процесу аналітичної обробки даних найбільш наближеною до потреб експертів є подання у вигляді графічної схеми. З метою забезпечення користувачів інструментальними засобами на основі бібліотеки MXGraph [11] розроблено спеціалізований графічний редактор з набором класів процесів, що може розширюватись. Цей редактор дозволяє створювати схеми процесів та зберігати їх описи у декларативному виді (див. Лістинг 1) та у виді XML (див. Лістинг 2).

В якості засобів моделювання процесу аналітичної обробки даних використано каркас, розроблений на основі бібліотеки FBP [12], яка надає сукупність java-класів для реалізації парадигми програмування, орієнтованого на процеси [13].

Перехід від декларативного опису до імперативної форми (Лістинг 3) здійснюється з використанням синтаксичного аналізатора та семантичного процесора, розробленого засобами ANTLR [14]. Аналогічний перехід від XML-опису реалізовано за допомогою семантичного процесора на основі SAX-парсера [15].

Предметно-орієнтована мова аналітичної обробки даних та інструментальні засоби її реалізації впроваджуються у Світовому центрі даних з геоінформатики та сталого розвитку [16] і використовуються для розрахунку моделей оці-

нювання сталого розвитку в глобальному та регіональному контексті, розрахунку моделі оцінювання загроз для регіонів України, тощо [17].

Висновки

Розроблено проблемно-орієнтовану графічну мову аналітичної обробки даних, використання якої не потребує спеціальних знань та навичок програмування в межах імперативної та об'єктно-орієнтованої парадигми, що дає можливість експертам з аналітичної обробки даних розробляти нові та модифікувати існуючі сценарії безпосередньо, без залучення програмістів. Досвід впровадження розроблених засобів автоматизації аналітичної обробки даних у Світовому центрі даних з геоінформатики та сталого розвитку показав, що можливість подання процесу аналітичної обробки даних у графічному вигляді, автоматична трансформація до виду, що може виконуватися в інструментальному середовищі, наявність інтерактивних засобів налагодження процесу аналітичної обробки даних дозволяють знизити трудомісткість розробки процедур попередньої обробки даних,моделей оцінювання, процедур формування аналітичних звітів з використанням результатів обробки даних великого обсягу.

Список літератури

1. Анализ данных и процессов / А.А.Барсегян, М.С.Куприянов, И.И.Холод и др. 3-е изд. перераб. и доп. СПб.: БХВ-Петербург.– 2009.– 512 с.
2. M.F.Hornick, E.Marcadé, S.VenkayalaJavaDataMining: Strategy, Standard, and Practice.–CA, US:Morgan Kaufmann.– 2007.– 520 p.
3. IMSL Numerical Libraries [Електронний ресурс] – Режим доступу: http://www.roguewave.com/products/imsl-numerical-libraries.aspx
4. OracleDataMining[Електронний ресурс] – Режим доступу: http://www.oracle.com/ru/products/database/options/advanced-analytics/index.html
5. Кельберт М. Я., Сухов Ю. М. Вероятность и статистика в примерах и задачах. Т. II: Марковские цепи как отправная точка теории случайных процессов и их приложения.– М.: МЦНМО.– 2009. – 295 с.
6. Айвазян С.А. и др. Прикладная статистика: Исследование зависимостей: Справ, изд. / С. А. Айвазян, И. С. Енюков, Л. Д. Мешалкин; Под ред. С. А. Айвазяна. – М.: Финансы и статистика, 1985. – 487 с.
7. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS. 2-е изд. / Пер. с англ.; Под общей редакцией проф. С. Орлова – СПб.: Питер.– 2006. – 736 с.
8. Information technology. Syntactic metalanguage. Extended BNF. [Електронний ресурс] ISO/IEC 14977:1996. – Режим доступу: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26153
9. Згуровский М.З., Болдак А.А. Системное согласование даннях разной природы в мультидисциплинарных исследованиях // Кибернетика и систем. анализ. – 2011., N 4. – С. 51–64.
10. Extensible Markup Language (XML) 1.0 (Fifth Edition)[Електронний ресурс] – Режим доступу: http://www.w3.org/TR/REC-xml/.
11. The most popular Java Graph Visualization Library[Електронний ресурс] – Режим доступу: http://www.jgraph.com/jgraph.html.
12. Java Implementation of FBP Concepts (JavaFBP) [Електронний ресурс] – Режим доступу: http://www.jpaulmorrison.com/fbp/#JavaFBP.
13. J.P.Morrison Flow-based programming : a new approach to application development.–Unionville, Ont.,CA : J.P. Morrison Enterprises.– 2010.– 336p.
14. P.Terence Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages (1st ed.), Raleigh, N.C., US: Pragmatic Bookshelf. –2010.–374p.
15. D.Brownell SAX2.– Sebastopol, CA : O'Reilly.– 2002.– 228p.
16. Світовий центр даних з геоінформатики та сталого розвитку [Електронний ресурс] – Режим доступу: http://wdc.org.ua/
17. Аналіз сталого розвитку – глобальний та регіональний контексти: моногр. /Міжнар. рада з науки (ICSU) [та ін.]; наук. кер. М.З.Згуровський. – К.:НТУУ «КПІ», 2010.– Ч. 2. Україна в індикаторах сталого розвитку.–220 с.

УДК 519.854.2

ШЕВЧЕНКО К.Ю.

АЛГОРИТМ ГІЛОК ТА МЕЖ ДЛЯ СТАТИСТИЧНИХ ДОСЛІДЖЕНЬ НОВОГО ПДС-АЛГОРИТМУ РОЗВ'ЯЗАННЯ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ЗВАЖЕНОГО ЗАПІЗНЕННЯ ВИКОНАННЯ РОБІТ НА ОДНОМУ ПРИЛАДІ

Зроблений огляд сучасного стану досліджень задачі мінімізації сумарного зваженого запізнення виконання робіт на одному приладі. Пропонується ефективний алгоритм гілок та меж її розв'язання, призначений для використання у статистичних дослідженнях характеристик нового ПДС-алгоритму.

Provided here is a state-of-the-art survey of research results on minimizing total weighted tardiness of processing jobs on a single machine. We suggest an effective branch-and-bound algorithm for solving this problem and which is intended to be used in statistical studies of characteristics of new PDC-algorithm.

Вступ

В багатьох прикладних сферах діяльності задоволення директивних строків виконання завдань та уникання штрафів за запізнення є найважливішою метою планування. Вартість запізнення доставок, яку формують незадоволення клієнту та упущена вигода, різна не тільки для різних замовлень, а й для різних клієнтів. Таким чином повне врахування стратегічних пріоритетів компанії потребує використання інформації про важливість клієнтів під час складання розкладів виконання завдань на виробничих дільницях, що обумовлює оперування зваженими штрафами за запізнення (див. напр. [1]). Іншим приводом до використання критеріїв, в яких фігурує зважене запізнення – різні пріоритети устаткування (машин), що складають фонди підприємства.

В цій статті розглядається задача мінімізації сумарного зваженого запізнення виконання множини робіт на одному приладі в рамках теорії календарного планування. Один прилад являє собою найпростіше машинне середовище. Таке середовище важливе за декількома причинами. По-перше, воно є частковим випадком усіх інших середовищ. Моделі на єдиній машині часто мають властивості, яких не мають ні елементарні середовища, що складаються з паралельних машин, ні багатостадійні конвеєрні середовища. Результати, які отримані для однієї машини (приладу), можуть служити потужними евристиками для побудови розкладів у більш складних машинних середовищах. Також на практиці, задачі операційного планування в більш складних машинних середовищах часто можуть декомпозуватись на підзадачі, кожна з яких моделюється одним приладом. Складне машинне середовище, в якому єдина машина складає вузьке місце відносно критеріїв продуктивності, в моделі календарного планування може бути замінене однією цією машиною.

Лоулер (Lawler) показав, що розглянута задача є NP-складною в сильному розумінні зведенням до 3-розбиття та запропонував псевдополіноміальний алгоритм динамічного програмування для розв'язання її часткового випадку [2].

Найбільш ефективними існуючими точними методами розв'язання загального випадку задачі є методи математичного програмування, що базуються на перегляді області визначення. Ці методи використовують так звані правила переваги (правила домінування), застосування яких полягає в тому, що в ході розв'язання будується відношення часткового порядку на множині робіт, яке відповідає деякому оптимальному розкладу та залежить від конкретних значень параметрів поточного екземпляру задачі.

Найпершим запропонованим і найпростішим правилом домінування є лема Ельмаграбі [3]. Додаткові правила домінування для задачі мінімізації зваженого запізнення були започатковані Ріннуйєм Каном (Rinnooy Kan) та ін. [4]. Актурк та Їлдірім наводять правило домінування для робіт, обробка яких починається після певного моменту часу, який визначається з параметрів цих робіт [5]. Канет (Kanet) зовсім нещодавно навів більш ефективні правила домінування, до яких зводяться більшість правил Ріннуйя Кана та ін., а також запропонував зо-

всім нові правила для пар робіт, ваги яких не узгоджені [6].

В даній статті приділяється увага результатам, які можуть бути використані у алгоритмі, що побудований за схемою гілок та меж. Для того, щоб повністю описати алгоритм гілок та меж, потрібно визначитись у виборі наступних елементарних процедур (що разом складають стратегію алгоритму гілок та меж – для задачі мінімізації):

- *процедури галуження* конструктивного пошуку, тобто способу розбиття множини допустимих розв'язків у вершині дерева пошуку;
- *процедури оцінювання знизу* множин допустимих *розв'язків* у вузлах дерева пошуку.

В деяких випадках виділяють окрему процедуру оцінювання зверху множин допустимих розв'язків. В даній роботі ця процедура стандартна та полягає у фіксуванні рекорду – найкращого значення критерію листового вузла дерева пошуку.

Відомі нижні оцінки Швімера (Shwimer) мінімаксу запізнення [7], Гелдерса (Gelders) та Клайндорфера (Kleindorfer) [8, 9], обчислення якої зводиться до розв'язання транспортної задачі лінійного програмування, Ріннуйя Кана та ін. [4] розв'язання задачі про призначення, Фішера (Fisher) [10] субградієнтного спуску, Поттса (Potts) та Ван Вассенгова (Van Wassenhove) лагранжевої релаксації [11], Актурка (Akturk) та Їлдіріма (Yildirim) на основі їх нового правила домінування [5], Хугевена (Hoogeveen) і Ван де Вельде (Van de Velde) [12] як посилення оцінки Поттса та Ван Вассенгова застосуванням фіктивних змінних, ін. Незалежні дослідження Абдул-Раза (Abdul-Razaq) та ін. [13] показали, що найбільш ефективною є нижня оцінка Поттса та Ван Вассенгова через її дуже просте обчислення за лінійним методом підбору множників, найбільш строгою – оцінка Гелдерса та Клайндорфера.

Нижня оцінка Поттса та Ван Вассенгова потребує використання деякої евристики для побудови наближеного розкладу в кожному вузлі дерева пошуку. Найбільш ефективною за результатами обчислювальних експериментів [5] виявилась евристика очевидного штрафу за запізнення (apparent tardiness cost, ATC), що була запропонована Вепсалайненом (Vepsalainen) та Мортоном (Morton) в [14] та яка дає близькі до оптимальних послідовності робіт.

В [15] представлений новий підхід до розв'язання окремих NP-складних задач календарного планування та дозволяє знаходити точні розв'язки задач набагато більших розмірностей, ніж було можливо до цього будь-яким іншим методом. Його суттю є попереднє дослідження властивостей відповідних класів важкорозв'язних задач, доведенні положень та правил, які дозволяють використати єдиний принцип обчислень при побудові так званих ПДС-алгоритмів (алгоритмів із поліноміальною та декомпозиційною складовими). Для кожного такого алгоритму виводяться логіко-аналітичні умови, які свідчать про можливість розв'язання задачі поліноміальним підалгоритмом («p-умови»), та відповідно «d-умови», при виконанні яких в процесі розв'язання довільної індивідуальної задачі вона строго декомпозується на підзадачі меншої розмірності. Ці умови перевіряються під час розв'язання екземпляру задачі з класу, для якого побудовано ПДС-алгоритм.

Один варіант ПДС-алгоритму розв'язання досліджуваної задачі був запропонований в [16]. Для оцінки ефективності цього, а також будь-якого іншого нового алгоритму, виконують статистичну обробку даних, отриманих в ході обчислювальних експериментів над побудованими реалізаціями алгоритмів. Характеристики ефективності нових алгоритмів важливо порівнювати із аналогічними для відомих алгоритмів, але при цьому останні мають враховувати усі поточні результати вивчення відповідного класу задач. Коректність алгоритмів перевіряється на наборах розв'язаних індивідуальних задач. Ресурс [17] містить базу індивідуальних задач мінімізації сумарного зваженого запізнення та їх оптимальних (найкращих відомих) розв'язків розмірності до 100.

Для задачі мінімізації сумарного зваженого запізнення подібні експерименти щодо алгоритмів гілок та меж проводились в роботі [13] та не враховували отримані пізніше результати, такі як [4], [18], [5]. В [6] автор наголошує на тому, що із запровадженням нових правил домінування оцінка Гелдерса та Клайндорфера може конкурувати із оцінкою Поттса та Ван Вассенгова через те, що остання є доволі слабкою навіть після посилення за рахунок введення фіктивних змінних за зразком [12].

Дана стаття носить методологічний характер, та описує алгоритм гілок та меж, що використовує нові правила домінування та модифіковані процедури оцінювання. В якості нижніх оцінок оптимального значення критерію множин допустимих розв'язків використовуються два варіанти процедур. Процедура Поттса та Ван Васенгова будується на локально оптимальній послідовності відносно попарних перестановок робіт. Для процедури Гелдерса та Клайндорфера визначений найефективніший алгоритм обчислення (результати обчислювальних експериментів до статті не увійшли). Нарешті, наведено опис ходу розв'язання екземпляру задачі потужністю n = 15 робіт за допомогою сформульованого алгоритму.

Формальна постановка задачі

Тут і далі будуть використовуватись наступні позначення (велика літера в позначенні параметру робіт означає, що значення цього параметру залежить від позиції роботи в розкладі, інакше параметр роботи не залежить від конкретного розкладу; без обмеження загальності розглядаємо лише цілі часові параметри).

J – множина усіх робіт, які призначаються на прилад.

σ – деякий допустимий розклад, тобто послідовність робіт із J.

n – кількість робіт у розкладі, нумерація не залежить від їх розміщення у послідовності, $|J| = n$.

[j] – номер роботи, що стоїть в допустимому розкладі на позиції j.

p_j – тривалість роботи із номером j, усі роботи складаються із однієї операції.

d_j – директивний строк виконання роботи із номером j.

w_j – вага роботи із номером j, $\forall j \in J\ w_j > 0$; величина w_i / p_i називається пріоритетом роботи.

C_j – момент завершення обробки роботи j.

$S_j = d_j - C_j$ – резерв часу роботи j; залежність резерву часу від моменту початку обробки роботи j записується як $S_j(t) = d_j - p_j - t$.

$T_j = \max\left(0, C_j - d_j\right)$ – запізнення роботи j.

$P(A)$, де $A \subseteq J$ – сумарна тривалість обробки робіт з множини A, тобто $P(A) = \sum_{i \in A} p_i$.

$\overline{A} = J$, A, $A \subseteq J$.

В стандартній нотації теорії розкладів досліджувана задача позначається як $1 \| \sum w_i T_i$. Роботи поступають на прилад одночасно в момент $t = 0$, переривання робіт не дозволяються, час на переналадку приладу не залежить від послідовності обробки робіт, є постійним та вважається, що він включений в тривалості робіт. Усі роботи складаються з однієї операції. Якщо в тексті зустрічається посилання «робота j», мається на увазі робота із номером j.

При складанні розкладів на одному приладі широко застосовують відношення передування однієї роботи іншою. Якщо робота j передує роботі k, це записується як $j \to k$ (при цьому роботи не обов'язково суміжні). Множина робіт, щодо яких відомо, що вони передують роботі $j \in J$ буде позначатись $B_j \subseteq J$, а множина робіт, для яких з'ясовано, що вони слідують за роботою j, позначатиметься $A_j \subseteq J$. Слід зазначити, що $j \notin A_j,\ j \notin B_j$. A_j, B_j називаються *множинами домінування*.

Досліджується задача мінімізації сумарного зваженого запізнення, тобто задача пошуку такого розкладу n робіт, для якого

$$\sum_{i=1}^{n} w_i T_i = \sum_{i=1}^{n} w_i \max\left(0, C_j - d_j\right) \to \min. \quad (1)$$

Цей критерій є регулярним (це неспадна функція від моментів завершення обробки робіт $C_1, \ldots, C_n$), а отже, як відомо з елементарної теорії розкладів, оптимальний розклад відносно регулярного критерію не містить інтервалів очікування, і всі роботи виконуються одна за одною. З цього випливає, що областю допустимих розв'язків задачі $1 \| \sum w_i T_i$ є усі можливі перестановки потужності n.

Характеристики складності екземплярів задач

Розглядаючи випадкові величини ваги роботи w, тривалості p та директивного строку d для певного класу екземплярів задач мінімізації сумарного зваженого запізнення для багатьох застосувань можна вважати, що w та p незалежні так само, як незалежні w та d. Натомість p та d можуть бути залежними, розподіленими за де-

яким сумісним розподілом із щільністю $f_{pd}(x,y) = f_p(x) f_{d/p}(y/x)$ [4].

Позначимо через $\lambda_d(p)$ довжину інтервалу, на якому $f_{d/p}(y/p) > 0$. Введемо наступні величини. Відносним діапазоном директивних строків класу екземплярів задач називається випадкова величина

$$R(p) = \frac{\lambda_d(p)}{np}. \qquad (2)$$

Далі, фактором запізнення називається випадкова величина

$$t(p) = 1 - \frac{M(d/p)}{np}. \qquad (3)$$

Якщо $f_{d/p}(y/p)$ – щільність рівномірного розподілу, то R та t стають детермінованими. Як показали численні обчислювальні експерименти [13, 18, 5, 4] в цьому випадку середні характеристики ефективності алгоритмів гілок та меж залежать від значень величин R та t. Особливо складними виявляються екземпляри задач із значеннями цих величин, близькими до $R = t = 0{,}6$ або $R = 0{,}4$ та $t = 0{,}8$. Через це R та t називаються характеристиками складності екземплярів задач.

Одним із способів отримання оцінок характеристик складності для конкретного екземпляру задачі (за гіпотези, що умовний розподіл d для усіх значень p рівномірний) може бути використання наступних співвідношень:

$$\hat{R} = \frac{d_{\max} - d_{\min}}{P(J)}, \qquad (4)$$

$$\hat{t} = 1 - \frac{\sum_{i=1}^{n} d_j}{nP(J)}, \qquad (5)$$

де $d_{\min} = \min_{i\in\{1,\ldots,n\}}(d_i)$, $d_{\max} = \max_{i\in\{1,\ldots,n\}}(d_i)$.

Перестановки суміжних робіт

Розрізняють декілька типів відношень передування. Введемо означення.

Означення 1. Відношенням глобального передування роботою i роботи j $i \to j$ називається транзитивне відношення на множині J, в якому для кожної пари робіт існує оптимальний розклад призначення робіт в заданому порядку.

Означення 2. Відношенням безумовного передування роботою i роботи j $i \prec\prec j$ називається відношення на множині J, що задає порядок слідування суміжних робіт, при перестановці місцями яких значення критерію завжди збільшується. Для цього відношення властивість транзитивності не виконується.

Означення 3. Відношенням передування роботою i роботи j, залежним від часу (відношенням умовного передування) $i \prec j$, називається відношення на множині J, яке задається так само, як і відношення безумовного передування, але залежить від моментів початку обробки робіт.

Актурк та Їлдірім [5, 18] запропонували алгоритм, який для будь-якої початкової послідовності робіт знаходить локально оптимальну послідовність відносно перестановок суміжних робіт. В такій послідовності перестановка місцями суміжних робіт може призводити лише до збільшення значення критерію оптимізації.

Означення 4. Критичною точкою для пари суміжних робіт i та j називається момент часу t_{ij} такий, що якщо $t = \min\{C_i - p_i, C_j - p_j\} \ge t_{ij}$, то $i \prec j$ $(j \prec i)$, інакше $j \prec i$ $(i \prec j)$.

Виберемо пару робіт $i, j \in J$ таку, що виконується

$$\begin{aligned} &(d_i < d_j) \vee \\ &(d_i = d_j \wedge p_i < p_j) \vee \\ &(d_i = d_j \wedge p_i = p_j \wedge w_i \ge w_j). \end{aligned} \qquad (6)$$

За таких позначень Актурк та Їлдірім показали [18], що критичні точки для робіт i та j обмежуються наступними:

$$t_{ij}^1 = \frac{w_i d_i - w_j d_j}{w_i - w_j} - (p_i + p_j) \qquad (7)$$

при $d_j - (p_i + p_j) \le t_{ij}^1 \le d_i - p_i$,

$$t_{ij}^2 = d_j - p_i - p_j\left(1 - \frac{w_i}{w_j}\right) \qquad (8)$$

при $\max\{d_i - p_i, d_j - (p_i + p_j)\} \le t_{ij}^2 < d_j - p_j$,

$$t_{ij}^3 = d_j - p_j - p_i\left(1 - \frac{w_j}{w_i}\right) \qquad (9)$$

при $d_j - p_j \le t_{ij}^3 < d_i - p_i$.

Якщо відповідна критична точка лежить в означених границях, то кажуть, що ця критична точка справджується (інакше – не справджується). Справедлива наступна теорема, яка дозволяє визначати критичні точки, які справджуються для пари суміжних робіт замість того, щоб перевіряти це безпосередньо.

Теорема 1 [5]**.** Для пари суміжних робіт $i, j \in J$ такої, що виконується (6), позначивши $t = \min\{C_i - p_i, C_j - p_j\}$, справедливі наступі твердження:

1) якщо $d_i < d_j$, $p_i w_j < p_j w_i$ та $p_i(w_j - w_i) > (d_j - d_i)w_i$, тоді справджуються критичні точки t_{ij}^1, t_{ij}^3, та $j \prec i$ при $t_{ij}^1 \le t \le t_{ij}^3$, та $i \prec j$ інакше;

2) якщо $d_i = d_j$, $w_i < w_j$, $p_i w_j < p_j w_i$, то справджується лише критична точка t_{ij}^3, та $j \prec i$ при $t \le t_{ij}^3$, та $i \prec j$ інакше (замість умови $w_i < w_j$ можна використовувати $p_i \le p_j$);

3) якщо $d_i < d_j$, $p_i w_j \ge p_j w_i$ та $p_i(w_j - w_i) > (d_j - d_i)w_i$, то справджується лише критична точка t_{ij}^1, причому $i \prec j$ при $t \le t_{ij}^1$, та $j \prec i$ інакше;

4) якщо $d_i \le d_j$, $p_i w_j \le p_j w_i$ та $p_i(w_j - w_i) \le (d_j - d_i)w_i$, тоді $i \prec\prec j$;

5) якщо $d_i = d_j$ та $p_i w_j \ge p_j w_i$, тоді робота $j \prec\prec i$;

6) якщо $d_i < d_j$, $p_i w_j > p_j w_i$ та $p_j(w_j - w_i) \le (d_j - d_i)w_j$, тоді справджується лише критична точка t_{ij}^2 та при $t \le t_{ij}^2$ $j \prec i$, і $i \prec j$ після критичної точки.

Означимо матрицю T критичних точок, елементами (i, j) якої можуть бути наступні символи:

- «⟶» якщо $i \to j$;
- «$\prec\prec$» якщо $i \prec\prec j$;
- «індекс критичної точки», якщо для пари робіт *i,j* виконується умова (6) та справджується єдина критична точка із цим індексом;
- «13», якщо для пари робіт *i,j* реалізується випадок 1 теореми 1;
- «-» у всіх інших випадках.

З теореми 1 можна отримати універсальну процедуру обчислення елементу (i, j) матриці критичних точок (при цьому обчислюється також симетричний відносно головної діагоналі елемент (j, i)). Вона є наступною:

1. Якщо $d_i = d_j$, то:
   1.1) якщо $p_i w_j \ge p_j w_i$, то $T_{ji} :=$ «$\prec\prec$», $T_{ij} :=$ «–»;
   1.2) інакше якщо $w_i \ge w_j$, то $T_{ij} :=$ «$\prec\prec$», $T_{ji} :=$ «–»;
   1.3) в противному випадку $T_{ij} :=$ «3», $T_{ji} :=$ «–».
2. Інакше якщо $p_i w_j \le p_j w_i$, то:
   2.1) якщо $p_i(w_j - w_i) \le (d_j - d_i)w_i$, то $T_{ij} :=$ «$\prec\prec$», $T_{ij} :=$ «–»;
   2.2) інакше $T_{ij} :=$ «13», $T_{ij} :=$ «–».
3. Інакше:
   3.1) якщо $p_j(w_j - w_i) \le (d_j - d_i)w_j$, тоді $T_{ij} :=$ «3», $T_{ji} :=$ «–»;
   3.2) інакше $T_{ij} :=$ «1», $T_{ji} :=$ «–».

Елементи «⟶» заповнюються окремо та дана процедура для них не виконується, на діагоналі ставляться символи «-».

Алгоритм знаходження локально оптимальної послідовності відносно перестановок суміжних робіт може бути сформульованим наступним чином (задано початкову послідовність σ):

1. Для $i = \overline{1, n-1}$, $j = \overline{i+1, n}$ якщо $[i] \to [j]$ та $i > j$ або $[j] \to [i]$, та $i < j$, тоді переставити місцями роботи $[i]$ та $[j]$.
2. $k := 1, t := 0$. Поки $k \le n-1$ виконувати:
   2.1) $i := [k]$, $j = [k+1]$, *swap* := false;
   2.2) якщо $T_{ji} =$ «$\prec\prec$», поміняти місцями роботи *i* та *j, swap* := true;
   2.3) інакше якщо умова (6) виконується, $T_{ij} =$ «13», $d_j - p_i - p_j < t$ і $t_{ij}^1 < t \le t_{ij}^3$, переставити місцями роботи *i* та *j*, *swap* := true;
   2.4) інакше якщо умова (6) виконується, $T_{ij} =$ «1» («2») та $t > t_{ij}^1 (t > t_{ij}^2)$, переставити місцями роботи *i* та *j*, *swap* := true;
   2.5) інакше якщо умова (6) не виконується, $T_{ji} =$ «13», $d_j - p_i - p_j < t$ і $t < t_{ij}^1 \vee t_{ij}^3 < t$, переставити місцями роботи *i* та *j*, *swap* := true;
   2.6) інакше якщо умова (6) не виконується, $T_{ji} =$ «1» («2») та $t \le t_{ij}^1 (t \le t_{ij}^2)$, переставити місцями роботи *i* та *j*, *swap* := true;
   2.7) якщо *swap* = true та $k > 1$, $t := t - p_{k-1}$, $k := k - 1$;
   2.8) інакше $t := t + p_k$, $k := k + 1$.

3. Кінець, отримана локально оптимальна послідовність.

Цей алгоритм має часову складність $O(n^3)$ подібно до алгоритму із [5] і відрізняється тим, що може знаходити локальний оптимум на частковій послідовності робіт з *J*.

Правила домінування

Часто кажуть, що важкорозв'язні задачі бракують структури, тобто при проведені конструктивного пошуку в області допустимих рішень відсутні співвідношення між параметрами задачі, які б дозволяли достатньо скорочувати цю область по мірі конструювання часткової послідовності. Правила домінування якраз і містять такі співвідношення, і в окремих випадках їх застосування дозволяє відносно легко отримати оптимальну послідовність. Твердження про те, що при виконанні певного співвідношення між параметрами задачі одна робота обов'язково передує іншій хоча б в одному оптимальному розкладі, називається правилом домінування.

Окрім правил домінування відомі елементарні факти, що стосуються часткових випадків індивідуальних задач. Далі наведемо їх у вигляді переліку лем.

Лема 1 [5]. Якщо в задачі $1||\sum w_iT_i$ роботи упорядкувати за неспаданням директивних строків (побудувати EDD-розклад), і щонайбільше одна робота при цьому буде запізнюватись, то така послідовність оптимальна.

Лема 2 [19]. Якщо в задачі $1||\sum w_iT_i$ при сортуванні робіт за незростанням пріоритетів (WSPT-розклад) всі роботи запізнюються, або жодна робота не запізнюється, то отримано оптимальний розклад.

З лем 1 та 2 випливає наступний наслідок: якщо для задачі $1||\sum w_iT_i$ EDD-розклад збігається із WSPT-розкладом, то ці розклади оптимальні.

Лема 3 (Ельмаграбі [3]). Якщо для задачі $1||\sum w_iT_i$ $\exists k: d_k \ge P(\sigma)$, то робота із номером *k* в оптимальній послідовності призначається останньою.

Лема 4 [2, 4]. Якщо в задачі $1||\sum w_iT_i$ знайдуться дві роботи $j,k \in J$ із $d_j \le d_k$, $p_j \le p_k$ та $w_j \ge w_k$, то існує оптимальна послідовність, в якій $j \to k$.

Правила домінування додатково враховують множини домінування. Виберемо дві роботи $j,k \in J, j \ne k$: відомі A_j, A_k, B_j, B_k, причому $j \notin A_k, B_k$, $k \notin A_j, B_j$. Відомі правила домінування сформулюємо у вигляді теорем.

Теорема 2 [4] Для задачі $1||\sum w_iT_i$ та пари робіт $j,k \in J, j \ne k$, для яких відомі A_j, A_k, B_j, B_k, $j \notin A_k, B_k$, $k \notin A_j, B_j$ якщо $d_k \ge P\left(\overline{A_j}\right)$, то $j \to k$.

Можна бачити, що з цієї теореми випливає лема Ельмаграбі.

Теорема 3 [6]. Для задачі $1||\sum w_iT_i$ та пари робіт $j,k \in J, j \ne k$, для яких відомі A_j, A_k, B_j, B_k, $j \notin A_k, B_k$, $k \notin A_j, B_j$ виконуються наступні твердження:

1) якщо $p_j \le p_k, w_j \ge w_k$ та

$$d_j \le \max\left\{d_k, \frac{w_j - w_k}{w_j}\left(P(B_j \cup B_k) + p_j + p_k\right) + \frac{w_k}{w_j}d_k\right\},$$

або

$$d_j \le \frac{w_j - w_k}{w_j}\left(P(B_j \cup B_k) + p_j + p_k\right) + \frac{w_k}{w_j}\left(P(B_k) + p_k\right),$$

тоді $j \to k$;

2) якщо $p_j \le p_k$, $w_j < w_k$, та

$$d_k \ge \frac{w_k - w_j}{w_k}P(\overline{A_j}) + \frac{w_j}{w_k}d_j$$

і одночасно

$$d_k \ge \frac{w_k - w_j}{w_k}P(\overline{A_j}) + \frac{w_j}{w_k}\left(P(\overline{A_j \cup A_k}) - p_k\right),$$

тоді $j \to k$;

3) якщо $p_j \le p_k$, $w_j < w_k$, та

$$d_j \le \frac{w_j - w_k}{w_j}P(\overline{A_j}) + \frac{w_k}{w_j}\left(P(B_k) + p_k\right)$$

і одночасно

$$p_j \le \frac{w_j - w_k}{w_j}\left(P(\overline{A_j}) - P(B_k)\right) + \frac{w_k}{w_j}p_k,$$

тоді $j \to k$;

4) якщо $w_j \ge w_k$, та

$$d_k \le \min\Big\{d_j, \frac{w_k - w_j}{w_k}\big(P(B_j \cup B_k) + p_j + p_k\big) + \frac{w_j}{w_k} d_j\Big\},$$

і одночасно

$$d_k \ge P(\overline{A_j}) - \frac{w_j}{w_k} p_k,$$

тоді $j \to k$;

5) якщо $w_i \ge w_k$, та якщо

$$d_j \le \frac{w_j - w_k}{w_j}\big(P(B_j \cup B_k) + p_j + p_k\big) + \frac{w_k}{w_j}\big(P(B_k) + p_k\big)$$

і одночасно

$$p_k \ge \frac{w_k}{w_j}\big(P(\overline{A_j}) - P(B_k) - p_k\big),$$

тоді $j \to k$;

6) якщо $w_i < w_k$, та якщо

$$d_k \ge \frac{w_k - w_j}{w_k} P(\overline{A_j}) + \frac{w_j}{w_k} d_j$$

і одночасно

$$d_k \ge P(\overline{A_j}) - \frac{w_j}{w_k} p_k,$$

тоді $j \to k$.

Можна бачити, що лема 4 є наслідком першого випадку теореми 3.

Теорема 4 [5]. Для задачі $1||\sum w_i T_i$ якщо для пари робіт $i, j \in J$ справедлива умова (6), $d_i \le d_j$, $p_i > p_j$, $w_i < w_j$, то при

$$t = \min\big\{C_i - p_i, C_j - p_j\big\} > t_{ij},$$

де t_{ij} – одна з критичних точок, що можуть справджуватись в даному випадку (t_{ij}^1 або t_{ij}^2), $j \to i$.

Теореми 2-4 стверджують відомий на даний час набір глобальних правил домінування (передування) для задачі $1||\sum w_i T_i$.

Процедура галуження

В методі гілок та меж здійснюється пошук в глибину. Кожний вузол дерева пошуку відповідає частковому розкладу σ_η, який складається із робіт, призначених на останні $\eta \le n$ позицій. Позначимо множину робіт, що належать σ_η через $S(\eta)$, та $U(\eta) = J$, $S(\eta)$ (конкретні часткові розклади однієї потужності для описання процедури галуження розрізняти не має необхідності). Галуження відбувається для $\eta < n$ шляхом призначення деякої роботи $k \in U(\eta)$ безпосередньо перед усіма роботами з $S(\eta)$, таким чином отримуючи частковий розклад $\sigma_{\eta+1}$. Щодо роботи *k* приймаються наступні рішення:

1) $\exists j \in U(\eta) : k \in B_j \Rightarrow$ галуження не виконується;

2) для пари робіт *k*, *j*, $j \in S(\eta)$ справджуються умови теореми 4 $\Rightarrow$ галуження не виконується;

3) $k \in A_j \ \forall j \in U(\eta) \Rightarrow$ робота k безумовно фіксується на позиції $n - \eta$;

4) після призначення $\forall j \in U(\eta)$ $A_j := A_j \cup \{k\}$, та множини домінування оновлюються для усіх робіт з $U(\eta)$ – будується транзитивне замикання відношення передування наступним чином:

$$A_i := A_i \cup \{k\} \cup A_k \quad \forall i \in B_j \cup \{j\},$$
$$B_i := B_i \cup \{j\} \cup B_j \quad \forall i \in A_k \cup \{k\}.$$

Таким чином уникаються циклічні перевірки, і будується антирефлексивне відношення.

Для того, щоб враховувати усі зміни множин домінування, після додавання дуги в граф глобального передування, заданий матрицею суміжності *M,* виконуються наступні дії.

1. $i := 1$, $j := 1$.
2. Якщо $i \ne j$ та $M_{ij} = 0 \vee M_{ji} = 0$, перевірити послідовно усі правила домінування для пари робіт *i* та *j*.
3. Якщо виконується хоча б одне з них, задати $M_{ij} := 1$, знайти транзитивне замикання *M*, перейти на крок 1.
4. Інакше якщо $j = n, i := i + 1, j := 1$, інакше $j := j + 1$.
5. Якщо $i > n$, кінець, інакше перейти на крок 2.

Аналогічна логіка будується для будь-якого іншого представлення графу відношення глобального передування.

Нижні оцінки мінімуму критерію для множини робіт

В загальному випадку задача оцінювання має наступний вигляд. У деякому вузлі дерева конструктивного пошуку отримано частковий розклад σ_η, який складається із η призначених робіт. Нижня оцінка цього вузла *LB* буде складатись з двох компонент, а саме

$$LB = \sum_{i\in\sigma_\eta} w_i T_i + LB^*, \qquad (10)$$

де $LB^* \le \sum_{\{i:i\notin\sigma_\eta\}} w_i T_i$. В цьому розділі розглянемо дві процедури оцінювання: лагранжевої релаксації Поттса та Ван Вассенгова, а також транспортну оцінку Гелдерса і Клайндорфера.

Введемо оцінку на основі підходу лагранжевої релаксації. Задача $1||\sum w_i T_i$ може бути записана наступним чином:

$$\sum_{i=1}^{n} w_i T_i \to \min$$

при обмеженнях

$$T_i \ge 0, \quad i = \overline{1,n}, \qquad (11)$$

$$T_i \ge C_i - d_i, \quad i = \overline{1,n}, \qquad (12)$$

де мінімізація проводиться по всім наборам моментів закінчення обробки робіт. Відкинемо обмеження (12). Перетворена відповідно до принципу лагранжевої релаксації задача тоді буде мати наступний вигляд:

$$\min_{(T_i,C_i)} \sum_{i=1}^{n} \left((w_i-\mu_i)T_i + \mu_i(C_i-d_i)\right) \to \max \qquad (13)$$

при обмеженнях

$$T_i \ge 0, \quad i = \overline{1,n},$$

де $\mu \ge 0$ – вектор множників Лагранжа. Позначимо функцію Лагранжа як $L(\mu)$, виключення з області допустимих розв'язків точок, в яких $L(\mu) = -\infty$, еквівалентно накладанню додаткових обмежень

$$\mu_i \le w_i, i = \overline{1,n}. \qquad (14)$$

Нехай задана деяка послідовність робіт σ^*, причому робота *i* в цій послідовності завершується в момент C_i^*. Після спрощення перетворення задача отримує наступний вигляд [11]:

$$L(\mu) = \sum_{i=1}^{n} \mu_i \left(C_i^* - d_i\right) \to \max \qquad (15)$$

при обмеженнях

$$\frac{\mu_i}{p_i} \ge \frac{\mu_{i+1}}{p_{i+1}}, i = \overline{1,n-1} \qquad (16)$$

$$0 \le \mu_i \le \hat{w}_i, \ , i = \overline{1,n}, \qquad (17)$$

де величини

$$\hat{w}_i = p_i \min_{h\in\{1,\ldots,i\}} \left\{\frac{w_h}{p_h}\right\} \qquad (18)$$

називаються *зведеними вагами робіт*. Вибір послідовності σ^* є довільним, обчислювальні експерименти показали, що дана оцінка буде тим більш строгою, чим ближче до оптимальної є σ^*.

Розклад σ^* пропонується будувати наступним чином: спочатку знаходиться послідовність із застосуванням евристики очевидного штрафу за запізнення (АТС), в якій роботи послідовно призначаються на прилад у порядку спадання залежних від часу пріоритетів, що обчислюються для $\forall i \in J$ за формулою:

$$I_j(t) = \frac{w_j}{p_j} \exp\left(-\frac{|U| \cdot \max(0, d_j - p_j - t)}{2P(U)}\right) \qquad (19)$$

де U – множина ще не призначених робіт. Після цього на знайденій послідовності шукається локальний оптимум відносно перестановок суміжних робіт.

Для розв'язання задачі (15), (17) запропоновано так званий метод підбору множників, який будує оптимальний вектор множників Лагранжа за лінійний час, що обумовлене простою структурою розв'язку. Наведемо остаточну процедуру обчислення, її обґрунтування можна знайти в [11]. Спочатку сформулюємо теорему.

Теорема 5 [11]. Нехай

$$\Lambda_0 = 0, \Lambda_j = \sum_{i=1}^{j} p_i L_i, \ j = \overline{1,n}.$$

Задамо множину цілих чисел $V = \{v_1, \ldots, v_r\}$ таку, що $v_0 = 0$, а v_k – це найменше число таке, що

$$v_k > v_{k-1} \Rightarrow \Lambda_{v_k} > \Lambda_{v_{k-1}}.$$

Тоді оптимальним розв'язком задачі (15), (17) буде вектор множників Лагранжа μ^*, компоненти якого обчислюються наступним чином:

$$\mu_i^* = \mu_{i+1}^* \frac{p_i}{p_{i+1}}, \quad i \notin V, i \ne n, \qquad (20)$$

$$\mu_i^* = \hat{w}_i, \quad i \in V, \qquad (21)$$

$$\mu_n^* = 0, \quad \text{якщо n} \notin V. \qquad (22)$$

Теорема 5 стверджує, що якщо пронумерувати роботи в порядку їх слідування відповідно до використаної евристики σ^*, вклад робіт із номерами $v_{i-1}+1, \ldots, v_i$ до нижньої оцінки буде складати $\hat{w}_{v_i}(\Lambda_{v_i} - \Lambda_{v_{i-1}})/p_{v_i}$. Таким чином для кожного елементу $v_i \in V$ до нижньої оцінки додається означений член, а отже вона обчислюється за лінійний час від кількості робіт в *J*.

Ідея транспортної нижньої оцінки полягає в тому, що відкидається вимога про заборону переривань робіт. Замість простору Π усіх можливих перестановок довжини n розглядається простір S усіх розкладів, в яких роботи можуть перериватись (але заборона інтервалів очікування все ще має місце). Таким чином маємо

$$\min_{\Pi} \sum_{j\in J} w_j \max\left(0, C_j - d_j\right) \geq \geq \min_{\mathsf{S}} \sum_{j\in J} w_j \max\left(0, C_j - d_j\right). \quad (23)$$

Розіб'ємо часовий інтервал $\left[0; P(J)\right]$ на v відрізків, вибравши з цією метою v моментів часу

$$\mathrm{T} = \left(\tau_1, \tau_2, \ldots, \tau_v \mid 0 = \tau_1 \leq \tau_2 \leq \ldots \leq \tau_v \leq P(J)\right).$$

Довжину відрізка i позначимо через s_i. Введемо невідомі x_{ij} – це обсяг часу, який робота j обробляється в часовому інтервалі i. Розглянемо наступну транспортну задачу лінійного програмування (ТЗЛП):

$$L(x, \mathrm{T}) = \sum_{i=1}^{v}\sum_{j=1}^{n} c_{ij} x_{ij} \to \min \quad (24)$$

при обмеженнях

$$\sum_{j=1}^{n} x_{ij} = s_i,\ i = \overline{1, v}, \quad (25)$$

$$\sum_{i=1}^{v} x_{ij} = p_j,\ j = \overline{1, n}, \quad (26)$$

$$x_{ij} \geq 0 \quad i = \overline{1, v},\ j = \overline{1, n}. \quad (27)$$

Задача (24) – (27) моделює задачу $1 \mid pmpt \mid \sum w_i T_i$, де поле *pmpt* вказує про можливі переривання робіт, що надалі буде називатись *послабленою задачею*. Обмеження (25) відповідають умові, що загальний обсяг часового інтервалу, що розподіляється по роботах, точно рівний його довжині. Подібним чином обмеження (26) стверджують, що загальний обсяг часу обробки, який виділяється кожній роботі, повинен бути рівним її тривалості. Обмеження (27) також обов'язкові, оскільки компоненти невідомого вектору x позначають час, і тому цей вектор має бути напівдодатним.

Єдиним невизначеним питанням залишається вибір ваг c_{ij} цільової функції. Обґрунтуємо такий вибір у вигляді теореми.

Теорема 6 [8]. Якщо в задачі (24) – (27) компоненти вектора ваг c обрати наступним чином:

$$c_{ij} = \begin{cases} \left(1 + \left\lfloor \dfrac{\tau_i - d_j}{p_j} \right\rfloor w_j\right) \text{при } \tau_i \geq d_j, \\ 0, \text{ інакше,} \end{cases} \quad (28)$$

то оптимальний розв'язок послабленої задачі з такими вагами буде нижньою оцінкою оптимального розв'язку відповідної задачі $1 \| \sum w_i T_i$ для будь-якого розбиття T інтервалу $\left[0; P(J)\right]$. Тут $\lfloor a \rfloor$ позначає найбільше додатне ціле число, що $\leq a$.

Наведемо ще один результат, який вказує на те, що вибір розбиття T напряму впливає на ефективність обговорюваної оцінки.

Теорема 7 [8]. Для задачі (24) – (27), в якій ваги обираються відповідно до виразу (28), розглянемо два розбиття часового інтервалу $\left[0; P(J)\right]$

$$\mathrm{T} = \left(\tau_1, \tau_2, \ldots, \tau_n, \tau_{n+1}, \ldots, \tau_v\right)$$

та

$$\mathrm{T}' = \left(\tau'_1, \tau'_2, \ldots, \tau'_n, \ldots, \tau'_{v+p}\right)$$

такі, що

$$\tau_i = \tau'_i,\ i = \overline{1, n},$$
$$\tau_n = \tau'_n \leq \tau'_{n+1} \leq \tau'_{n+2} \ldots \leq \tau'_{n+p+1} = \tau_{n+1},$$
$$\tau_i = \tau'_{i+p},\ i = \overline{n+1, v},$$

тобто в розбитті T′ між точками τ_n та τ_{n+1} знаходяться додаткові проміжні точки, і тому T′ продукує більше інтервалів, ніж T. Тоді $L(x, \mathrm{T}) \leq L(x', T')$.

Чим більше інтервалів утворює розбиття, тим більш строгою є дана нижня оцінка. В той же час із збільшенням кількості інтервалів, які утворюються в результаті розбиття, збільшується складність розв'язання послабленої задачі. Саме у виборі конкретного розбиття відображається компроміс між строгістю оцінки та складністю її обчислення.

Для того, щоб алгоритм гілок та меж, що використовує транспортну оцінку Гелдерса та Клайндорфера, давав найкращі результати, послаблена задача у кожному вузлі дерева пошуку повинна розв'язуватись якомога більш ефективним з точки зору кількості обчислень методом. До того ж бажано, щоб складність алгоритму, що розв'язує послаблену задачу, не залежала від характеру вхідних даних, а лише від розмірності задачі, тобто щоб цей алгоритм був строго поліноміальним. Більше того, в послабленій задачі x_{ij} – не обов'язково цілі числа, тому і з

цього положення вибір строго поліноміального алгоритму є більш привабливим (інакше для обчислень слід використовувати представлення чисел з фіксованою точністю).

Послаблена задача є ТЗЛП (24) – (27), і в сучасній практиці математичного програмування її розв'язують, зводячи до задачі знаходження потоку мінімальної вартості [20]. Слід відмітити, що стандартні методи розв'язання ТЗЛП як задачі лінійного програмування загального вигляду є набагато менш ефективними, ніж спеціалізовані.

Згідно із [20], строго поліноміальні алгоритми розв'язання ТЗЛП, найкращі на сьогодні за часовою складністю – це запропоновані в [21], [22], [23]. Але обчислювальна практика показала, що великі приховані константи виконання для алгоритмів [21] та [23] обмежують вибір алгоритмом Орліна масштабування пропускних здатностей без стягування вершин [22].

Відповідно до (10) під час конструктивного пошуку задачу оцінювання потрібно розв'язувати кожного разу як частковий розклад змінюється. Хоча при русі до листових вузлів дерева пошуку розмірність транспортної задачі зменшується, в деяких випадках раціонально знаходити розв'язок не у всіх, а лише в певних ключових вузлах дерева пошуку. Для інших часткових розкладів можна отримувати дещо слабшу оцінку простішим методом, опишемо його.

При розв'язанні задачі (24) – (27) можна отримати набори двоїстих змінних u_i^* та v_j^*, $i=\overline{1,v}$, $j=\overline{1,n}$, які являють собою потенціали вершин транспортної мережі, та відповідають оптимальному потоку в ній (див напр. [23]). При цьому виконується

$$\sum_{i=1}^{v}\sum_{j=1}^{n} c_{ij}x_{ij}^* = \sum_{i=1}^{v} s_i u_i^* + \sum_{j=1}^{n} p_j v_j^*, \qquad (29)$$

де x_{ij}^* позначає оптимальний розв'язок транспортної задачі. Можна легко показати, використовуючи теорему 7, що

$$L(x,T,\eta) = \sum_{i=1}^{I-1} s_i' u_i^* + \sum_{j=1}^{\eta-1} p_j v_j^* \qquad (30)$$

– це нижня оцінка оптимального значення критерію на підмножині розкладів, які починаються із робіт $J\setminus\{\sigma_\eta\}$. Якщо нижня оцінка (30) не призводить до виключення із розгляду вузла дерева пошуку, для цього вузла все одно слід розв'язати задачу (24) – (27).

Приклад розв'язання екземпляру задачі

Згенеруємо екземпляр задачі із характеристиками складності $R=t=0,6$ потужністю $n=15$. Для цього візьмемо n реалізацій $w_i \in_U [1;10]$, $p_i \in_U [1;100]$,

$$d_i \in_U \left[P(J)\cdot(1-t-R/2);\, P(J)\cdot(1-t+R/2)\right].$$

Тоді $Md = P(J)\cdot(1-t)$, $\lambda_d = R\cdot P(J)$. Беремо цілу частину від отриманих значень. Дані екземпляру наведені в табл. 1.

Табл. 1. Згенерований екземпляр задачі

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
w_i	1	2	6	4	9	2	8	4	1	2	10	2	1	6	7
p_i	17	46	6	79	52	88	96	54	47	87	78	100	62	27	85
d_i	301	468	98	246	419	557	362	505	347	506	425	500	410	177	329

Значення критерію для поданого порядку виконання робіт $f=11735$, горизонт планування $C_{\max}=924$. Складемо деякі базові розклади: розклад EDD (сортування за неспаданням директивних строків), розклад WSPT (сортування за незростанням пріоритетів), розклад ATC та на основі нього локально оптимальний відносно перестановок суміжних робіт (LM-розклад). Матриця критичних точок T наведена в табл. 2. Отримані наступні послідовності:

$$\sigma_{EDD} = (3,14,4,1,15,9,7,13,5,11,2,12,8,10),$$
$$f_{EDD} = 4731;$$
$$\sigma_{WSPT} = (3,14,5,11,7,15,8,1,4,2,10,6,9,12,13),$$
$$f_{WSPT} = 3566;$$
$$\sigma_{ATC} = (3,14,4,15,7,5,11,8,1,2,10,6,9,12,13),$$
$$f_{ATC} = 2548;$$
$$\sigma_{LM} = (3,14,4,15,7,5,11,1,8,2,10,6,9,12,13),$$
$$f_{LM} = 2494.$$

Табл. 2. Матриця критичних точок

Роботи	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-	≺≺	-	-	2	≺≺	1	2	≺≺	≺≺	2	≺≺	≺≺	-	1
2	-	-	-	-	-	≺≺	-	2	-	≺≺	-	≺≺	-	-	-
3	≺≺	≺≺	-	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺
4	2	≺≺	-	-	2	≺≺	2	2	≺≺	≺≺	2	≺≺	≺≺	-	2
5	-	≺≺	-	-	-	≺≺	-	≺≺	-	≺≺	≺≺	≺≺	-	-	-
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	-	≺≺	-	-	2	≺≺	-	≺≺	-	≺≺	2	≺≺	≺≺	-	-
8	-	-	-	-	-	≺≺	-	-	-	≺≺	-	-	-	-	-
9	-	2	-	-	2	2	1	2	-	2	2	≺≺	≺≺	-	-
10	-	-	-	-	-	≺≺	-	-	-	-	-	-	-	-	-
11	-	≺≺	-	-	-	≺≺	-	≺≺	-	≺≺	-	≺≺	-	-	-
12	-	-	-	-	-	2	-	1	-	2	-	-	-	-	-
13	-	2	-	-	1	2	-	2	-	2	1	2	-	-	-
14	≺≺	≺≺	-	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	≺≺	-	≺≺
15	-	≺≺	-	-	2	≺≺	2	≺≺	≺≺	≺≺	2	≺≺	≺≺	-	-

Локально оптимальний розклад був отриманий перестановкою робіт 8 та 1, що починаються в розкладі АТС в момент часу $t = 423$. Як видно з таблиці, для робіт справджується критична точка t_{18}^2, значення якої $t_{18}^2 = 447,5$, що і обумовлює перестановку. Сумарне зважене запізнення робіт в LM-розкладі є попередньою верхньою оцінкою оптимального значення критерію.

Далі, побудуємо відношення глобального передування для робіт, починаючи із порожніх множин домінування. Результат обчислення наведений в табл. 3. Для кожної пари робіт, що знаходяться у відношенні, вказана теорема та номер твердження, з якого це випливає. Пара може знаходитись у відношенні також за транзитивністю, в цьому випадку над стрілкою відношення позначки не ставляться.

Табл. 3. Матриця відношення глобального передування

Роботи	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1						$\rightarrow^{T3.3}$			$\rightarrow^{T3.1}$	$\rightarrow^{T3.3}$		$\rightarrow^{T3.3}$	$\rightarrow^{T3.1}$		
2						$\rightarrow^{T3.1}$				$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$	$\rightarrow^{T3.1}$		
3	$\rightarrow^{T3.1}$	$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$	$\rightarrow^{T3.2}$	$\rightarrow$	$\rightarrow^{T3.2}$	$\rightarrow^{T3.1}$	$\rightarrow$	$\rightarrow$	$\rightarrow^{T3.2}$	$\rightarrow$	$\rightarrow$	$\rightarrow^{T3.1}$	$\rightarrow^{T3.2}$
4						$\rightarrow^{T3.1}$				$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$	$\rightarrow^{T3.4}$		
5		$\rightarrow^{T3.4}$				$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$	$\rightarrow$		
6												$\rightarrow^{T3.1}$	$\rightarrow^{T3.5}$		
7		$\rightarrow^{T3.4}$			$\rightarrow^{T3.6}$	$\rightarrow^{T3.4}$		$\rightarrow^{T3.4}$		$\rightarrow^{T3.4}$	$\rightarrow^{T3.6}$	$\rightarrow^{T3.1}$	$\rightarrow^{T3.4}$		
8						$\rightarrow^{T3.1}$				$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$	$\rightarrow^{T3.1}$		
9												$\rightarrow^{T3.3}$	$\rightarrow^{T3.1}$		
10						$\rightarrow^{T3.1}$						$\rightarrow^{T3.1}$	$\rightarrow^{T3.5}$		
11		$\rightarrow^{T3.4}$				$\rightarrow^{T3.1}$		$\rightarrow^{T3.4}$		$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$	$\rightarrow$		
12													$\rightarrow^{T3.5}$		
13															
14	$\rightarrow^{T3.4}$	$\rightarrow^{T3.1}$		$\rightarrow^{T3.1}$	$\rightarrow^{T3.2}$	$\rightarrow$	$\rightarrow^{T3.2}$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$		$\rightarrow^{T3.2}$
15	$\rightarrow^{T2}$	$\rightarrow^{T3.4}$		$\rightarrow^{T2}$	$\rightarrow$	$\rightarrow^{T3.1}$	$\rightarrow^{T3.2}$	$\rightarrow$	$\rightarrow^{T3.4}$	$\rightarrow^{T3.1}$	$\rightarrow$	$\rightarrow$	$\rightarrow$		

Можна легко бачити, що роботи 12 та 13 призначаються в оптимальному розкладі останніми і тому виключаються з розгляду, оскільки їм передують усі інші роботи. Подібним чином підпослідовність робіт (3,14,15) призначається в тому самому оптимальному розкладі на початку.

Таким чином, в корені дерева пошуку $\eta = 2$, $\sigma(2) = (12,13)$. Щоб остаточно визначитись із структурою кореневого вузла, сформуємо множину кандидатів, які можуть призначатись безпосередньо перед $\sigma(2)$.

Роботи 3, 14, 15 вже були виключені. Можна не включати до кандидатів роботи 1, 2, 4, 5, 7, 8, 10, 11, оскільки вони належать B_6. Таким чином кандидатами на галуження з кореня є роботи 6 та 9. Хід подальших обчислень та дерево пошуку зображене на рис. 1.

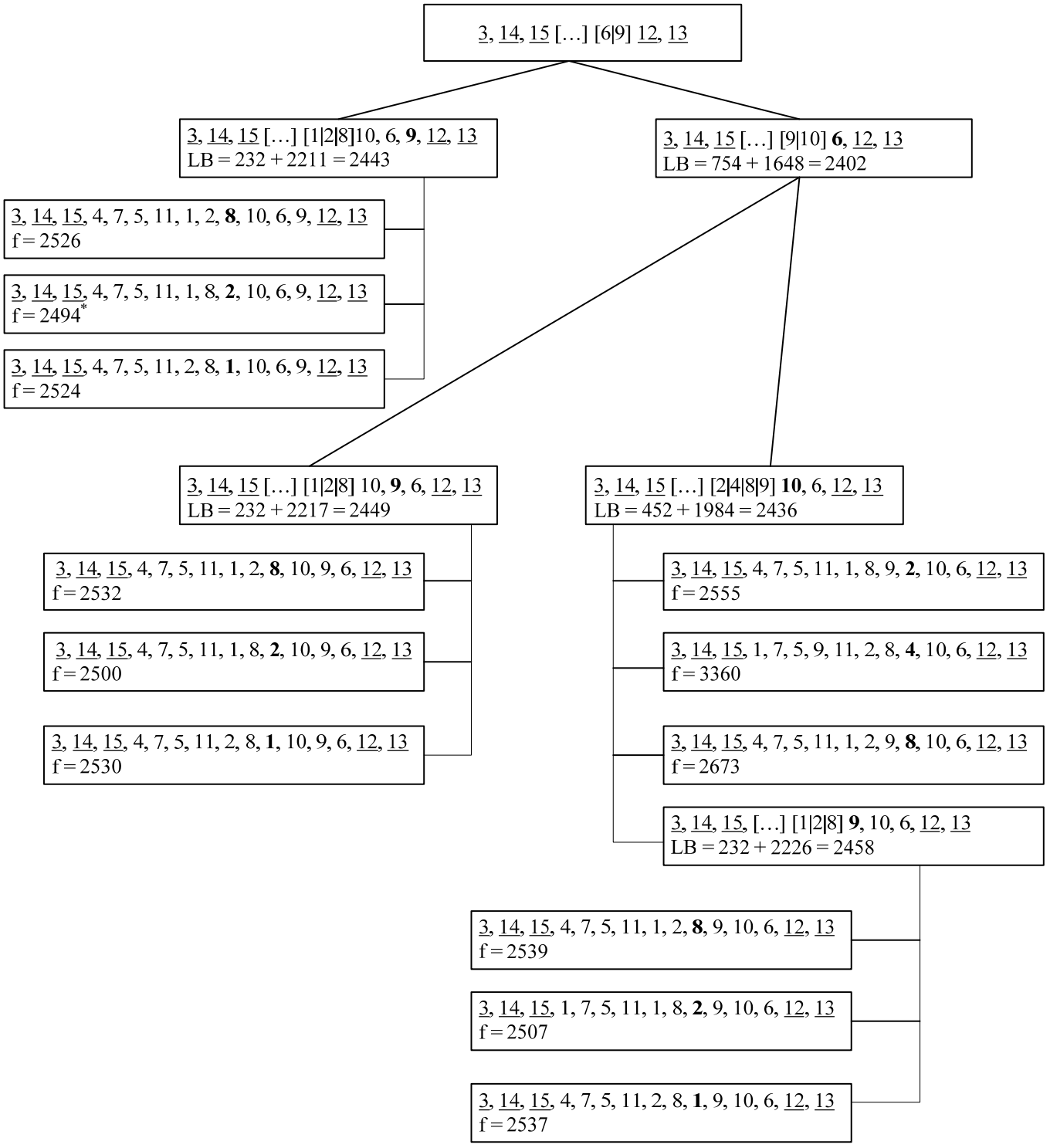


Рис. 1. Дерево пошуку

Підкреслення номерів робіт означає їх безумовну фіксацію на початку або у кінці розкладу. В квадратних дужках зазначені кандидати на галуження в кожному вузлі.

Для оцінки вузлів була використана процедура Гелдерса та Клайндорфера, оскільки на невеликих за потужністю екземплярах оцінка Поттса та Ван Вассенгова часто нульова. Це компенсується швидкодією її обчислення для задач більшої розмірності. Після кожного галуження оновлюється матриця відношення глобального передування, в кожному вузлі зберігаються лише пари робіт, які були додані до відношення, обчисленого на початку. Це дозволило вже на 3-4 рівні заглиблення отримати конкретні розклади у листових вузлах. Таким чином послідовність σ_{LM} є оптимальною.

Висновки

Зроблений огляд останніх досягнень в області дослідження задачі мінімізації сумарного зваженого запізнення виконання робіт одним приладом, та був сформульований на їх основі ефективний алгоритм гілок та меж, який призначений для статистичних досліджень характеристик нових алгоритмів, таких, як запропонований ПДС-алгоритм для цієї задачі.

Даний алгоритм широко використовує 10 відомих глобальних правил домінування, більшість з яких були встановлені нещодавно та не згадувались в емпіричних дослідженнях, дві процедури нижньої оцінки оптимального значення критерію, одна з яких є найбільш строгою серед відомих із регульованою складністю обчислень, а інша – обчислюється за лінійний час та дає приріст ефективності на задачах великої розмірності, процедуру отримання локально оптимальної послідовності з метою одержання строгої верхньої оцінки оптимального значення критерію та удосконалення нижньої оцінки на основі лагранжевої релаксації.

Подальші дослідження включатимуть реалізацію запропонованого алгоритму та проведення обчислювальних експериментів для порівняння продуктивності згаданого ПДС-алгоритму із продуктивністю описаного в цій статті.

Список літератури

1. Jensen J.B., Philipoom P.R., Malhotra M.K. Evaluation of scheduling rules with commensurate customer priorities in job shops // Journal of Operations Management. – 1995, Vol.13. – P.213-228.
2. Lawler E.L. A "Pseudopolynomial" Algorithm for Sequencing Jobs to Minimize Total Tardiness // Annals of Discrete Mathematics. – 1977, Vol. 1. – P.331-342.
3. Elmaghraby S.E., The One Machine Sequencing Problem with Delay Costs // Journal of Industrial Engineering. – 1967, Vol. 19. – P.105-108.
4. Rinnooy Kan, A.H.G., Lageweg B.J., Lenstra J.K. Minimizing Total Costs in One-machine Scheduling // Operations Research. – 1975, Vol. 23. – P.908-927.
5. Akturk M.S., Yildirim M.B. A New Lower Bounding Scheme for the Total Weighted Tardiness Problem // Computers Operations Research. – 1998, Vol. 24 (4). – P.265-278.
6. Kanet, J.J. New precedence theorems for one-machine weighted tardiness // Mathematics of Operations Research. – 2007, Vol. 32. – P.579–588.
7. Shwimer J. On the N-Job, One-Machine, Sequence-Independent Scheduling Problem with Tardiness Penalties: a Branch-and-Bound Solution // Management Science. – 1972, Vol. 18. – P.301-313.
8. Gelders L., Kleindorfer P.R. Coordinating Aggregate and Detailed Scheduling Decisions in the One-Machine Job Shop I – Theory // Operations Research. – 1974, Vol. 22. – P.46-60.
9. Gelders L., Kleindorfer P.R. Coordinating Aggregate and Detailed Scheduling Decisions in the One-Machine Job Shop II – Computation and Structure // Operations Research. – 1975, Vol. 23. – P.312-324.
10. Fisher M.L. A Dual Algorithm for the One-machine Scheduling Problem // Mathematical Programming. – 1976,Vol. 11. – P.229-251.
11. Potts C.N., Van Wassenhove L.N. A Branch and Bound Algorithm for Total Weighted Tardiness Problem // Operations Research. – 1985, Vol. 33. – P.363-377.
12. Hoogeveen J.A., Van de Velde S.L. Stronger Lagrangian Bounds by Use of Slack Variables: Applications to Machine Scheduling Problems // Mathematical Programming. – 1995, Vol. 70. – P.173-190.
13. Abdul-Razaq T. S., Potts C. N., Van Wassenhove L. N. A Survey of Algorithms for Single Machine Total Weighted Tardiness Scheduling Problems // Discrete Applied Mathematics. – 1990, Vol. 26(2–3). – P.235–253.

14. Vepsalainen A.P.J., Morton T.E. Priority Rules for Job Shops with Weighted Tardiness Costs // Management Science. – 1987, Vol. 39. – P.626-632.
15. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка. – 2010. – 573 с.
16. Павлов А.А., Мисюра Е.Б. Новый подход к решению задачи «Минимизация суммарного взвешенного опоздания при выполнении независимых заданий с директивными сроками одним прибором» // Системні дослідження та інформаційні технології. – 2002, № 2. – С.7-32.
17. OR-Library: Weighted Tardiness [Електронний ресурс] // Режим доступу: http:// people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html.
18. Akturk M.S., Yildirim M.B. A New Dominance Rule for the Total Weighted Tardiness Problem // Production Planning & Control: The Management of Operations – 1999, Vol.10 (2). – P.138-149.
19. Rachamadugu R.M.V. A Note on Weighted Tardiness Problem // Operations Research – 1987, Vol.35. – P.450-452.
20. Schrijver A. Combinatorial Optimization: Polyhedra and Efficiency, vol. A – Springer, Berlin. – 2004. – 647 p.
21. Kleinschmidt P., Schannath H. A strongly polynomial algorithm for the transportation problem, Mathematical Programming. – 1995, Vol. 68. – P.1-13.
22. Orlin J.B. A Faster Strongly Polynomial Minimum Cost Flow Algorithm // Operations Research – 1993. – Vol 41 (2). – P.338-350.
23. Tokuyama T., Nakano J. Efficient algorithms for the Hitchcock transportation problem, SIAM Journal on Computing. – 1995, Vol. 24. – P.563-578.

УДК 004.75

ПЕТРЕНКО А.І.,
БУЛАХ Б.В.,
ЧЕКАЛЮК В.В.

АВТОМАТИЗАЦІЯ ВИКОНАННЯ ПОТОКІВ РОБІТ У ГРІДІ ІЗ ЗАЛУЧЕННЯМ ГРІД-СЕРВІСІВ

В даній статті розглядається архітектура системи автоматизованого виконання потоків робіт (англ. workflows) у гріді, складених з веб-сервісів та грід-сервісів. Описано механізм взаємодії компонентів системи, вказано деталі її практичної реалізації. Проведено оцінку часу виконання потоків робіт в рамках даної архітектури, наведено результати експериментів на тестовій інфраструктурі.

This article describes the architecture of the software system for automated workflow execution in grid environment for workflows composed of web services and grid services. The interactions between the components of this system are described and the details of its implementation are given. The workflow execution time is evaluated and the experimental results are provided.

1. Вступ

Грід-обчислення як технологія експлуатації віддалених обчислювальних ресурсів, наданих у спільне використання учасникам певних об'єднань користувачів (віртуальних організацій), вже давно успішно застосовуються для рішення різноманітних задач науки та інженерії високої обчислювальної складності [1]. Однак навіть сьогодні більшості користувачів (зокрема й українського гріду) доступна лише базова функціональність грід-середовища, що дозволяє віддалено запускати поодинокі обчислювальні задачі, контролювати стан їх виконання та вивантажувати результати обчислень. Для організації виконання складних сценаріїв обчислень, що вимагають скоординованого виконання у гріді десятків-сотень задач, такі можливості є, як правило, недостатніми. Можна стверджувати, що нині лишаються актуальними проблеми розробки і подальшого розвитку таких прикладних грід-середовищ, які б надавали достатньо гнучкості для організації власних обчислювальних сценаріїв користувачами, при цьому не вимагаючи від них додаткової підготовки, яка наразі необхідна при безпосередній роботі з базовими засобами службового програмного забезпечення проміжного рівня (ПЗПР).

Серед найбільш поширених дистрибутивів ПЗПР, що керують українськими грід-ресурсами, а саме – Nordugrid ARC та EGEE gLite різних версій, лише gLite надає можливість опису «композитних задач», складених з атомарних грід-задач. Натомість пропонується дослідити альтернативний підхід, що полягає у використанні принципів сервісно-орієнтованої архітектури (СОА [2]), зокрема – використати композицію грід-сервісів для організації гетерогенних обчислювальних сценаріїв.

2. Постановка задачі

Під обчислювальним сценарієм будемо далі розуміти попередньо визначену послідовність виконання обчислень, спрямовану на отримання певного визначеного результату. Під обчислювальним потоком робіт будемо розуміти такий обчислювальний сценарій, який можна представити у вигляді N відокремлених кроків обчислень $S_i \mid i = 1..N$, що виконуються у певній послідовності відносно один одного [3]. Якщо кроками потоку є грід-задачі, такий потік будемо називати потоком грід-задач [4]. Задачами даної статті є: а) дослідити архітектуру системи проектування та виконання потоків грід-задач, у якості кроків яких виступають грід-сервіси; б) оцінити часову ефективність виконання таких потоків відносно виконання обчислень на локальному ресурсі.

3. Організація потоків грід-задач

Як вже зазначалося вище, базових засобів таких ПЗПР, як Globus Toolkit чи ARC, не вистачає для повноцінного проектування та виконання потоків грід-задач. Однак, як правило, кожне відоме ПЗПР надає засоби для розробки власних клієнтів та планувальників у вигляді бібліотек та інструментарію розробника (SDK)

під різні мови програмування (напр. засоби libarcclient з проекту ARC).

Певним винятком є можливості ПЗПР gLite по запуску взаємозалежних грід-задач як єдиної «супер-задачі» засобами планувальника Condor DAGman. Такі сценарії описуються на мові опису задач Job Description Language (JDL), підтримуваній gLite. Втім, рішення, що базуються на даному функціоналі, дієві лише для gLite-інфраструктур.

З часів появи ПЗПР Globus Toolkit 3 чимале поширення дістала сервісно-орієнтована архітектура ПЗПР. Нині в тій чи іншій мірі усі дистрибутиви ПЗПР використовують веб-сервіси (або їх варіант – грід-сервіси) як програмний інтерфейс до своїх підсистем. Тож напрошується рішення, що використовуватиме як грід-сервіси, так і веб-сервіси як складові етапи (елементи) S_i потоку робіт. До переваг такого підходу порівняно з планувальником грід-задач типу DAGman можна віднести: можливість організації «змішаних» сценаріїв обчислень, складених не лише з грід-задач; сумісність з численними стандартами на веб-сервіси та їх композиціювання, і, відповідно, – з наявним інструментарієм.

4. Грід-сервіс

Згідно опису відкритої архітектури грід-сервісів OGSA [5], грід-сервісом може називатися лише такий програмний компонент, що відповідає усім вимогам саме цієї архітектури. Надалі дозволимо більш вільне трактування грід-сервісу як веб-сервісу, що надає доступ до грід-ресурсів (керує ними).

Характерною проблемою при реалізації грід-сервісів є довготривалі транзакції. Сервіси у гріді часто мають справу з тривалістю операцій, що перевищує допустимі межі для «таймауту» з'єднання. Серед можливих шляхів вирішення даної проблеми були розглянуті наступні: механізм оповіщень для асинхронного повідомлення клієнту про завершення операції, що дещо ускладнює архітектуру та підсилює її зв'язність, та циклічне опитування сервісу про стан виконання розпочатої операції.

Для практичної перевірки даного підходу було розроблено веб-сервіс, здатний ініціювати виконання обчислень у гріді. Його інтерфейс складають наступні операції (не включаючи службових операцій, що вирішують такі питання, як делегування прав тощо):

- *submitJob*: запуск обчислень. Повертає унікальний ідентифікатор задачі, який слугуватиме ключем для подальшої роботи із задачею;
- *cancelJob*: скасування виконання задачі;
- *getJobStatus*: отримання поточного статусу задачі;
- *getJobInfo:* отримання розширеної діагностичної інформації про задачу та грід-ресурс;
- *getJobOutput:* отримання результатів виконання грід-задачі.

Сам перелік операцій не є унікальним і співзвучний з інтерфейсами таких веб-сервісів, як WM-Proxy з ПЗПР gLite та A-REX з ARC. Розроблений сервіс відрізняє орієнтація на виконання конкретних обчислень з області моделювання електронних схем, що забезпечуються функціоналом комплексу NetALLTED [6]. Механізм роботи грід-сервісу спрощено проілюстровано на рис.1.

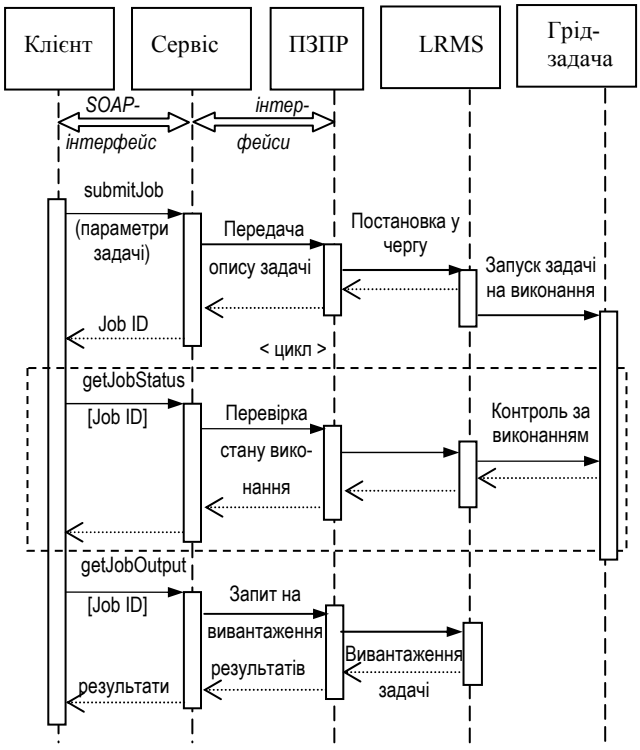


Рис.1. Спрощена діаграма послідовності взаємодії з грід-сервісом

Задля наочності дана діаграма не відображає усіх операцій, пов'язаних із запуском задачі.

Оцінимо час виконання задачі через грід-сервіс. Загальний час на отримання результатів від виконання обчислень на віддаленому ресурсі можна представити як:

$$T = \sum_{i=1}^{s} I_i + E + \sum_{j=1}^{d} O_j \qquad (1)$$

де I_i – час на i-ту транзакцію вхідних даних між хостами на шляху від джерела даних до обчислювального ресурсу (s – загальна кількість транзакцій), O_j – час на j-ту транзакцію вихідних даних між хостами на шляху від обчислювального ресурсу до сховища результатів (d – загальна кількість транзакцій), E – час на виконання обчислень на обчислювальному ресурсі.

У випадку, коли запуск обчислень передбачає взаємодію із додатковим службовим ПЗ (ПЗПР типу локальних менеджерів ресурсів кластерів або ПЗПР гріду), з'являються додаткові затримки. Окремо слід врахувати, що інформація про статус задачі оновлюється у інфосистемі ПЗПР з певним періодом τ (залежно від налаштувань ПЗПР). Тоді реально спостережений час виконання E^* можна оцінити як:

$$E \le E^* < E + \tau \qquad (2)$$

Тоді оцінка (1) приймає вигляд:

$$T = \sum_{i=1}^{s} I_i + P + E + \sum_{k=1}^{c} \tau_k + \sum_{j=1}^{d} O_j \qquad (3)$$

де P – час на підготовку до виконання обчислень конкретним ПЗПР, h – кількість рівнів ієрархії інформаційної системи ПЗПР, τ_k – період оновлення інформації про стан виконання обчислень на k-му рівні інфосистеми (оцінка є граничною, для найгіршого випадку співвідношення (2)).

Перейдемо до оцінки часу виконання грід-задачі. Для типового сценарію роботи в гріді маємо наступне.

1. Завантаження даних на віддалений ресурс (stage in). Час виконання i-тої транзакції:

$$I \approx D_I / W_I \qquad (4)$$

де D_I – об'єм вхідних даних, W_I – пропускна здатність мережі між джерелом даних та грід-ресурсом. Аналогічно, для вивантаження даних (stage out) час виконання j-тої транзакції:

$$O \approx D_O / W_O \qquad (5)$$

де D_O – об'єм вхідних даних, W_O – пропускна здатність мережі між ресурсом та кінцевим сховищем даних.

2. Накладні витрати на підготовку до виконання складають:

$$P = T_a + T_Q + T_g \qquad (6)$$

де T_a – часові витрати на авторизацію та делегування (клієнт делегує ПЗПР право діяти від його імені), вимірюються секундами, T_Q – час на постановку та очікування задачі у черзі локальної системи керування ресурсом (LRMS), T_g – інші накладні витрати, що не залежать від даних самої задачі (створення каталогів сесії тощо). T_Q для ресурсу зі спільним доступом – випадкова величина, що залежить від його поточної завантаженості. Обираючи для моделювання грід-ресурсу ту чи іншу модель СМО та її параметри згідно статистичних спостережень за ресурсом, можна наближено оцінити середній час очікування заявки у черзі.

3. Фактичний час виконання задачі визначається продуктивністю ресурсу R, алгоритмом виконання задачі A, причому може залежати і від розміру вхідних даних:

$$E = f(A, R, D_I) \qquad (7)$$

4. Інфосистема гріду, як правило, розрахована на тривалі задачі і має період оновлення τ_G ~30 с (залежно від налаштувань).

При оцінці часу виконання обчислень через грід-сервіс слід врахувати те, що недоліком SOAP-протоколу є зависоке відношення кількості службових символів до корисної інформації та можливість передачі лише символів, що відображаються (алфавіт, цифри, пунктуація та деякі спецсимволи). Зважаючи на це, бінарні дані передаються, як правило, у кодуванні Base64, що додатково збільшує розмір даних приблизно на третину. Також до витрат на кодування-декодування даних XML слід додати витрати на шифрування повідомлень при безпечному SSL-з'єднанні, що істотно зростуть у випадку використання шифрування самих даних повідомлення (WS-Security).

У випадку, коли дані прямо передаються сервісу через SOAP, слід скоригувати (4) та (5):

$$I \approx K_{WS} D_I / W_I, O \approx K_{WS} D_O / W_O \qquad (8)$$

де K_{WS} – коефіцієнт, що враховує накладні витрати на кодування/шифрування.

5. Потік робіт з грід-сервісів

Однією з переваг грід-сервісів над іншими інтерфейсами до ПЗПР є те, що перші краще придатні для композиції внаслідок відкритості

та стандартності своїх WSDL-описів, кращої стандартизації самої композиції (так зване «оркестрування»), наявності численних засобів розробки та розгортання, як комерційних, так і відкритих.

Архітектуру запропонованої системи виконання потоків задач складають:

– **сервіс керування потоками**, що має набір операцій, загалом аналогічних операціям грід-сервісу (за виключенням того, що сервіс керування потоками оперує описом та даними для усього потоку, а не окремого грід-сервісу): *submitTask, cancelTask, getTaskStatus, getTaskInfo, getTaskOutput*.

– **ПЗ автоматичного виконання потоків** грід- та веб-сервісів. Сервіс керування приймає опис потоку на певній мові та перекладає його на стандартну мову виконання потоку веб-сервісів, що підтримується цими засобами. Одним з таких стандартів є мова WS-BPEL 2.0. Ця мова описує так званий «бізнес-процес», що зовні поводить себе як «віртуальний» веб-сервіс. Таким чином, потік робіт представляється одним «композитним» веб-сервісом. Серед некомерційних BPEL-засобів нині доступні такі продукти, як Apache Ode, OW2 Orchestra та ін. Останній і було використано у практичній реалізації.

– **грід-сервіси**, з якими взаємодіє ПЗ автоматичного виконання потоку згідно отриманого BPEL-опису.

– **службові компоненти**: реєстр доступних грід-сервісів та їх метаданих, сховище грід-сертифікатів та ін.

Перші три компоненти розробленої архітектури виконання потоків робіт, складених з веб- та грід-сервісів, представлені на рис.2.

6. Оцінка ефективності рішення

«Чистий» (без урахування накладних витрат) час виконання потоку сервісів повністю визначається конфігурацією потоку G (що зазвичай задається графом):

$$E_W = f(T_i, I_i, O_i, G), i = 1..N \qquad (9)$$

де T_i – час виконання i-го етапу потоку робіт. Для оцінки E_W можна дотримуватись наступної процедури. Розглянемо довільний потік робіт з N кроків як послідовно сполучені M під-потоків, кожен з яких містить кілька паралельних гілок. Тривалість виконання кожного з M під-потоків визначається найбільшою тривалістю виконання з усіх паралельних гілок. Рекурсивно застосовуючи дане розбиття, можна отримати оцінку часу виконання для довільного потоку.

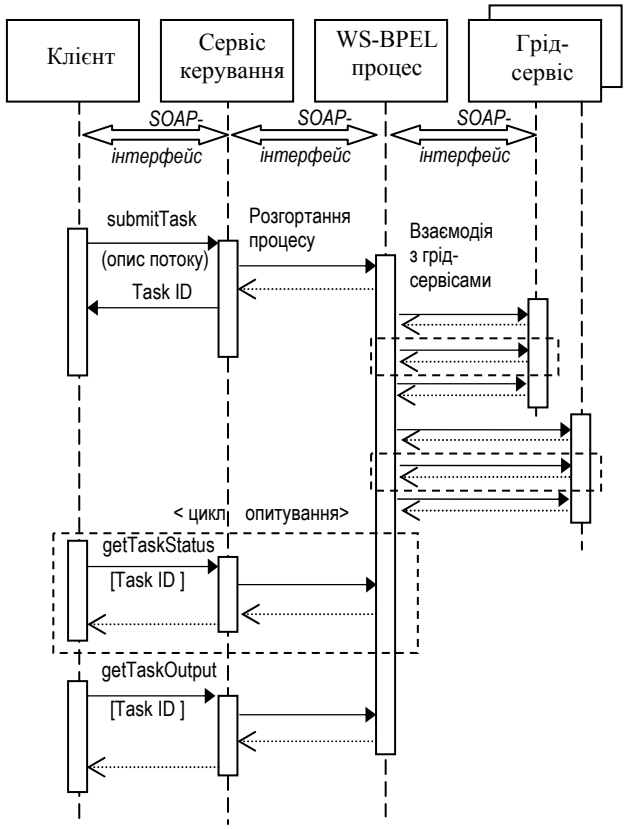


Рис.2. Взаємодія компонентів архітектури системи керування виконанням потоків робіт

Оцінимо тепер накладні витрати на виконання потоку. Витрати на пересилку даних визначаються аналогічно до (8), однак за випадку, коли використовується BPEL-процес, кількість пересилань може зрости приблизно у 2 рази. Аналогічно до міркувань щодо грід-сервісів, тут також слід урахувати інтервал опитування стану виконання потоку τ_W, а також інтервал опитування BPEL-процесом грід-сервісів τ_S. Накладні витрати також збільшаться за рахунок часу на розгортання BPEL-процесу T_{BP}.

Для експериментальної оцінки ефективності запропонованого підходу у гріді вирішувалась задача моделювання та оптимізації схемотехнічного рішення за допомогою функціоналу пакету NetALLTED. Подальші уточнення та припущення справедливі для задач моделювання на схемотехнічному рівні.

Експеримент проводився на виділеній тестовій грід-інфраструктурі з наступними характеристиками. Основним компонентом є тестовий

кластер, який складається з головного вузла та чотирьох робочих вузлів. Кожен робочий вузол має 2 чотириядерні ЦП Intel Xeon E5345 2,33 ГГц, 8 Гб оперативної пам'яті та жорсткий диск ємністю 500 Гб.

Оцінка часу виконання проводилася для потоків, складених з задач одного з кількох умовних класів: короткотривалі (E ~ 0,05 хв. на вищевказаних ресурсах), середньої тривалості (E ~ 7 хв.) та довготривалі (E ~ 35 хв.), коригування часу виконання відбувалось за рахунок зміни параметрів процедури оптимізації.

Грід-задачі було виділено в окрему високопріоритетну чергу (під час проведення експерименту у черзі не було сторонніх задач). Грід-ресурси керуються ПЗПР Nordugrid ARC 0.8.3.

Розглядався чи не найвигідніший у порівнянні з послідовним виконанням випадок з M = 1 та N паралельними гілками (тобто, одночасний запуск N задач моделювання електричної схеми з різними наборами вхідних параметрів; для оцінки часу при плануванні послідовного потоку робіт див. також [7]). Позначимо символом «′» параметри виконання потоку на одному локальному вузлі, а «′′» – параметри потоку грід-сервісів. Тоді з формул (3-9) для вищеописаного сценарію час виконання 1 етапу потоку:

$$T_1' = E' \tag{10}$$

$$T_1'' = E'' + P + \sum_{i=1}^{s} I_i + \sum_{j=1}^{d} O_j + \tau_G \tag{11}$$

Тоді при обмеженнях на ресурси у n ядер на локальному вузлі та n_G вільних ядер у гріді повний час виконання потоку:

$$T' = \begin{cases} E' \mid N \le n \\ E'N/n \mid N > n \end{cases} \tag{12}$$

$$T'' = (E'' + P + \tau_G + \sum_{i=1}^{s} I_i + \sum_{j=1}^{d} O_j)\lceil N/n_G \rceil + \\ + \sum_{i=1}^{s''} I_i'' + \sum_{j=1}^{d''} O_j'' + T_{BP} + \tau_S + \tau_W \tag{13}$$

Враховуючи, що об'єм вхідних та вихідних даних запущених задач є нехтовно малим відносно пропускної здатності мережі (що є поширеним сценарієм для задач схемотехнічного моделювання), а шифрування не застосовувалось, накладними витратами на пересилку даних можна знехтувати. Тоді з (12) та (13) слідує, що для такого типу задач виграш у часі від запуску потоків грід-сервісів порівняно з локальним виконанням можна отримати при $N > n$ за умови:

$$E' \frac{N}{n} > \left\lceil \frac{N}{n_G} \right\rceil (E'' + P + \tau_G) + T_{BP} + \tau_S + \tau_W \tag{14}$$

Експеримент проводився за наступних умов (дані отримані на основі налаштувань та попередніх тестів окремих компонентів системи): T_{BP} = 30 с, P = 1 хв., τ_G = 30 с, τ_S = 10 с, τ_W = 12 с. З (14) слідує, що при N = 16 виграш у часі можна отримати при E > 3 хв., при N = 32 та N = 64 маємо виграш для задач довше 1 хв. (мається на увазі, що $E' = E'' = E$).

Результати виконання потоків грід-задач, що загалом підтвердили ці оцінки, зведені у табл.1-3 та проілюстровані на графіках (рис.3). Слід зазначити, що теоретичні оцінки враховували максимально можливі затримки інфосистеми, а також не враховували випадкового характеру затримок P.

При $N > n$ = 8 для задач середньої та довгої тривалості виграти у часі відносно рішення на потоках грід-задач не дозволило навіть використання для локальних обчислень потужніших ЦП (Intel Xeon E5440 @ 2.83ГГц, також 2 ЦП по 4 ядра на вузол, локальне виконання пришвидшилось на ~25%).

Поряд із запуском потоків грід-сервісів проводились запуски тих самих наборів задач за допомогою bash-скриптів та утиліт командного рядка ПЗПР. Даний механізм запуску дозволяє скоротити накладні витрати на T_{BP}+ τ_S+τ_W ~ 0,5..1 хв., однак є заскладним для безпосереднього використання грід-користувачами, і програє у гнучкості запропоновану рішенню при побудові прикладних грід-середовищ, дружніх до користувача.

7. Висновки

Представлено архітектуру системи виконання потоків грід-задач, що використовує сервісно-орієнтований підхід та базується на окремих незалежних функціональних одиницях – грід-сервісах. Особливостями даного рішення є: використання мови WS-BPEL 2.0 у якості внутрішнього формату опису потоку, що дозволяє спиратися на існуючі стандартні BPEL-засоби при виконанні потоку робіт; використання грід-сервісів, що сумісні з WS-стандартами та дозволяють створення гібридних потоків робіт, складених як з грід-сервісів, так і звичайних

веб-сервісів. Було розроблено робочий прототип системи, що дозволив виявити деякі недоліки даного рішення, на які слід звернути увагу при подальшому вдосконаленні системи, а саме: надмірні об'єми внутрішнього інформаційного обміну, та, відповідно, суттєві накладні витрати при передачах великих об'ємів даних.

Було розроблено та інтегровано в дану архітектуру грід-сервіс для рішення задач оптимального моделювання схемотехнічних рішень, для яких вказані недоліки не є вагомими.

Хоча головними перевагами даної архітектури, що автоматизує виконання багатокрокових обчислень у гріді, є, в першу чергу, гнучкість, відповідність стандартам, використання існуючої інфраструктури, а не мінімізація часу розрахунків, було проведено оцінку часу виконання потоків робіт у даній архітектурі, додатково підтверджену експериментально. Слід зазначити, що порівняльні експерименти проводились за умов відсутності конкуруючих задач, що відрізняється від реальних умов динамічного грід-середовища, що слід брати до уваги при застосуванні отриманих оцінок.

Проведені дослідження дозволяють стверджувати, що дана архітектура найкраще підходить до вирішення довготривалих обчислювальних задач. Накладні витрати на організацію виконання потоку робіт роблять недоцільним запуск короткотривалих задач із тривалістю виконання порядку кількох хвилин (для яких, як правило, достатньо потужностей локальних ресурсів). Дане рішення з автоматизованого виконання складних обчислювальних сценаріїв може розглядатися як складова архітектури спеціалізованих високорівневих грід-середовищ та грід-порталів.

Табл. 1. Результати тестів для короткотривалих задач

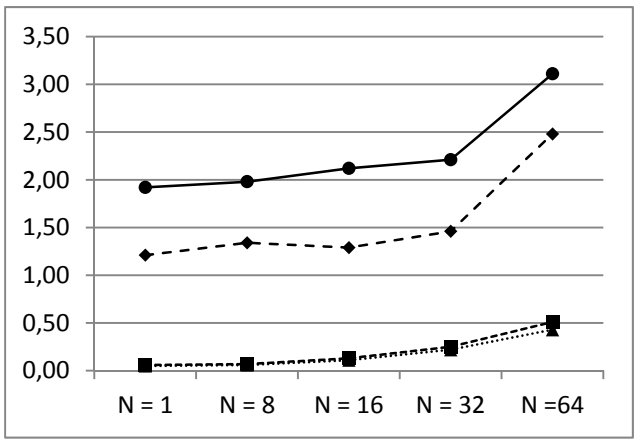
Спосіб запуску	**Середній час виконання потоку задач, хв.**				
	N = 1	N = 8	N = 16	N = 32	N =64
Локальне виконання @ 2.33 ГГц	0,06	0,07	0,13	0,25	0,51
Локальне виконання @ 2.83 ГГц	0,05	0,06	0,11	0,22	0,43
Запуск у грід (скрипт)	1,21	1,34	1,29	1,46	2,48
Запуск потоку сервісів	1,92	1,98	2,12	2,21	3,11

Табл. 2. Результати тестів для задач середньої тривалості

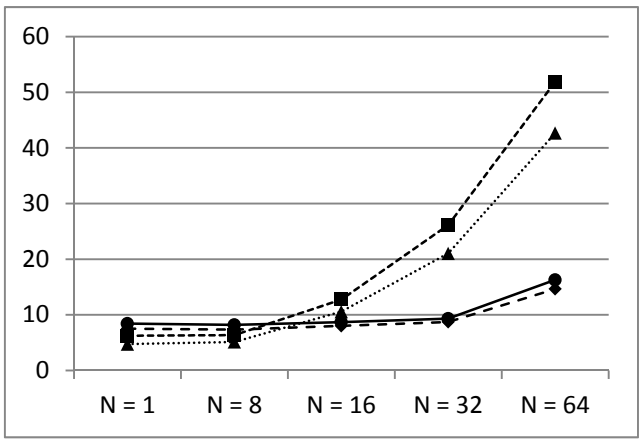
Спосіб запуску	**Середній час виконання потоку задач, хв.**				
	N = 1	N = 8	N = 16	N = 32	N = 64
Локальне виконання @ 2.33 ГГц	6,25	6,36	12,75	26,13	51,89
Локальне виконання @ 2.83 ГГц	4,73	5,12	10,58	21,04	42,66
Запуск у грід (скрипт)	7,51	7,32	8,01	8,72	14,64
Запуск потоку грід-сервісів	8,42	8,18	8,69	9,31	16,27

Табл. 3. Результати тестів для довготривалих задач

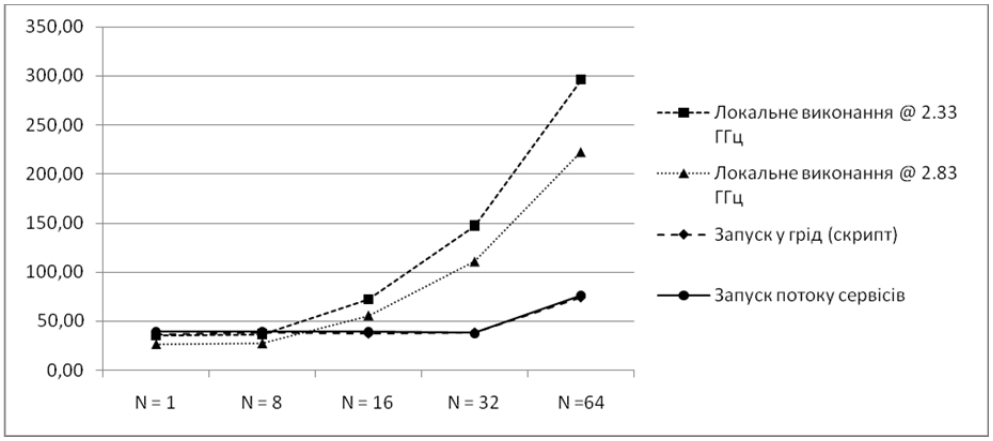
Спосіб запуску	**Середній час виконання потоку задач, хв.**				
	N = 1	N = 8	N = 16	N = 32	N =64
Локальне виконання @ 2.33 ГГц	35,33	36,58	71,97	147,15	295,73
Локальне виконання @ 2.83 ГГц	26,69	27,65	55,92	110,79	222,21
Запуск у грід (скрипт)	36,18	37,82	36,8	38,42	74,13
Запуск потоку сервісів	38,85	38,91	38,87	37,81	75,91



а



б



в

Рис. 3. Час виконання потоку грід-задач (хвилин): а – короткотривалих, б – середньої тривалості, в – довготривалих

Список літератури

1. Згуровський М. З. Grid-технології для е-науки і освіти / Згуровський М. З., Петренко А. І. // Наукові вісті НТУУ «КПІ». – 2009. – №2. – С. 10–17 .
2. Erl T. Service-Oriented Architecture: Concepts, Technology & Design / Erl T. – New York : Prentice Hall/PearsonPTR, 2005. – 792 p.
3. Workflows for e-Science. Scientific Workflows for Grids / Edited by I.J. Taylor, E. Deelman, D.B. Gannon, M. Shields. – Guildford : Springer, 2007. – 530 p.
4. Yu J. A Taxonomy of Workflow Management Systems for Grid Computing / Yu J., Buyya R. // Journal of Grid Computing. – 2005. – Vol. 3, № 3. – P. 171–200.
5. Talia D. The Open Grid Services Architecture: Where the Grid Meets the Web // IEEE. Internet Computing. – 2002. – Vol. 6, № 6. – P. 67–71.
6. Petrenko A. ALLTED – a computer-aided engineering system for electronic circuit design / Petrenko A., Ladogubets V., Tchkalov V., Pudlowski Z. – Melbourne : UICEE, 1997. – 205 p.
7. Franz D. A Workflow Engine for Computing Clouds / Franz D., Tao J., Marten H., Streit A. // Cloud Computing, GRIDs, and Virtualization : 2nd International Conference «CLOUD COMPUTING 2011», 25-30 Sep. 2011, Rome, Italy : proc. – 2011. – P. 1–6. – ISBN : 9780470940105.

УДК 51.74: 519.254

БАРАНОВСЬКИЙ О.М.,
КАЧИНСЬКИЙ А.Б.

ПОБУДОВА ФАЗОВИХ ПОРТРЕТІВ ВІДОБРАЖЕНЬ ОКРЕМИХ ІНФОРМАЦІЙНИХ ПОТОКІВ МЕРЕЖІ ІНТЕРНЕТ

Дана робота присвячена побудові фазових просторів відображень інформаційних потоків мережі Інтернет, що відображають прояви громадянської активності, пов'язані з реформуванням податкової системи України. Для проведення візуального аналізу поведінки визначених інформаційних потоків застосовано метод затримки аргументів та побудовано аттрактори отриманих реалізацій. В ході роботи підтверджено наявність фрактальних властивостей у розглянутих інформаційних потоках, визначені окремі характеристики їх поведінки.

This work is about phase space construction of Internet information flows, which show signs of civil activity associated with the Ukraine tax system reform. For the information flows behavior visual analysis the method of delayed arguments was used. Attractors obtained implementations. During the work confirmed the presence of fractal properties in the discussed information flows, identified specific characteristics of their behavior.

1. Вступ

На прикладі проявів громадянської активності населення під час реформування податкової системи України в попередніх роботах [1,2] були досліджені окремі властивості інформаційних потоків мережі Інтернет. Авторами проведена робота по моделюванню інформаційних потоків в системі контент-моніторингу, отриманню та первісній обробці кількісних даних, визначенню природи даних потоків. Було зроблено висновок про нелінійну динамічну (хаотичну) природу даних потоків та наявність в розглянутих інформаційних потоках довгої пам'яті.

2. Постанова задачі

Динамічна система характеризується станом системи в визначений момент часу та її динамікою (правил зміни стану системи в часі). Для динамічних систем прийнятим уявленням розвитку процесу в часі є побудова «портрета» у фазовому просторі (тобто просторі, координатами якого є змінні стану). Нелінійна динамічна система характеризується дивним аттрактором – притягаючою множиною в фазовому просторі, в якій розташовані хаотичні траєкторії. **Основною задачею** є вибір та обґрунтування методу відновлення аттраторів, а також побудова фазових портретів відповідних інформаційних потоків.

3. Вибір та обґрунтування методу

Розмірністю вкладення *m* називається найменша ціла розмірність простору, що містить весь аттрактор. Вона відповідає кількості незалежних змінних, що однозначно визначає рух динамічної системи. Важливою кількісною характеристикою аттрактора, що містить інформацію про ступінь складності поведінки динамічної системи, є кореляційна розмірність D_c.

Стан системи описується параметрами:

$$X^1(t), X^2(t), \dots, X^d(t) \qquad (1)$$

де верхні індекси вказують на нумерацію компонент. Набір параметрів стану в момент *t* формує вектор в *d*-мірному просторі, який має назву фазового. Цей вектор змінюється в часі та напрямку:

$$\dot{X}(t) = \frac{dX(t)}{dt} = F(X) \qquad (2)$$

а хід векторів даного фазового простору визначає фазову траєкторію, до того поле швидкостей *F(t)* буде дотичним до цієї траєкторії. Для автономних систем така траєкторія є самонеперетинаючею. Подібна графічна реалізація в часі дає безпосередню інформацію про динаміку та надає можливості візуального аналізу поведінки системи. Знаючи *F(t)*, стан системи в поточний момент часу можна визначити інтегруванням системи рівнянь (2).

Для відомої динамічної системи m та D_c легко визначити – відомі усі компоненти вектора *X(t)*, що описує поведінку системи (так, для системи Лоренца D_c=2,05, *m*=3). В даному випадку автори мають справу з реальними процесами та їх спостереженнями – часовими рядами з визначеним кроком. Вхідні дані – згладжені ряди кількості публікацій, зафіксовані системою контент-моніторингу у визначений день дослідження, що відповідають заданій вербаль-

ній моделі. Для процесів, що досліджуються, вимірювання всіх компонент, що характеризують систему, неможливо – вони не всі відомі. Однак Такенс показав [4], що можна відновити деякі властивості аттрактора (наприклад, розмірність вкладення m і кореляційну розмірність D_c) по тимчасовій послідовності однієї з компонент вектора. Цей підхід був вперше запропонований в 1980 році [4]. Дослідження показали, що можна отримати задовільну картину аттрактора невеликої розмірності, якщо замість змінних X, що входять в рівняння динамічної системи, використовувати m – мірні вектора, отримані з елементів часового ряду. У [5] доведена теорема, що дозволяє реконструювати фазову траєкторію по методу відображення затримуючих аргументів:

$$X_i = \left(U_i, U_{i+\tau}, \dots, U_{i+(m-1)\tau}\right) \quad (3)$$

де m – розмірність вкладення та τ – затримка по часу (реальна затримка по часу визначається як $\tau\Delta t$).

4. Опис методики

Методика заснована на побудові псевдоаттрактору, де в якості компонент вектора служить сама виміряна послідовність, але узята з деякою тимчасовою затримкою τ. Оскільки компоненти вектора, що характеризує динамічну систему, незалежні, то в якості τ береться перше значення, при якому автокореляційна функція збігається до 0 (або досягає мінімуму). Оскільки заздалегідь розмірність вкладення m невідома, то процедура зводиться до наступного:

Послідовно збільшують розмірність фазового простору і додають компоненти псевдовектора;

При кожному m = 2, 3, ... обчислюють кореляційну розмірність D_c і будують залежність $D_c(m)$. Спочатку при додаванні нових компонент псевдовектора кореляційна розмірність зростає. Це означає, що ми ще не досягли потрібної кількості вимірів, і, відповідно, потрібної складності, ступінь якої характеризує D_c;

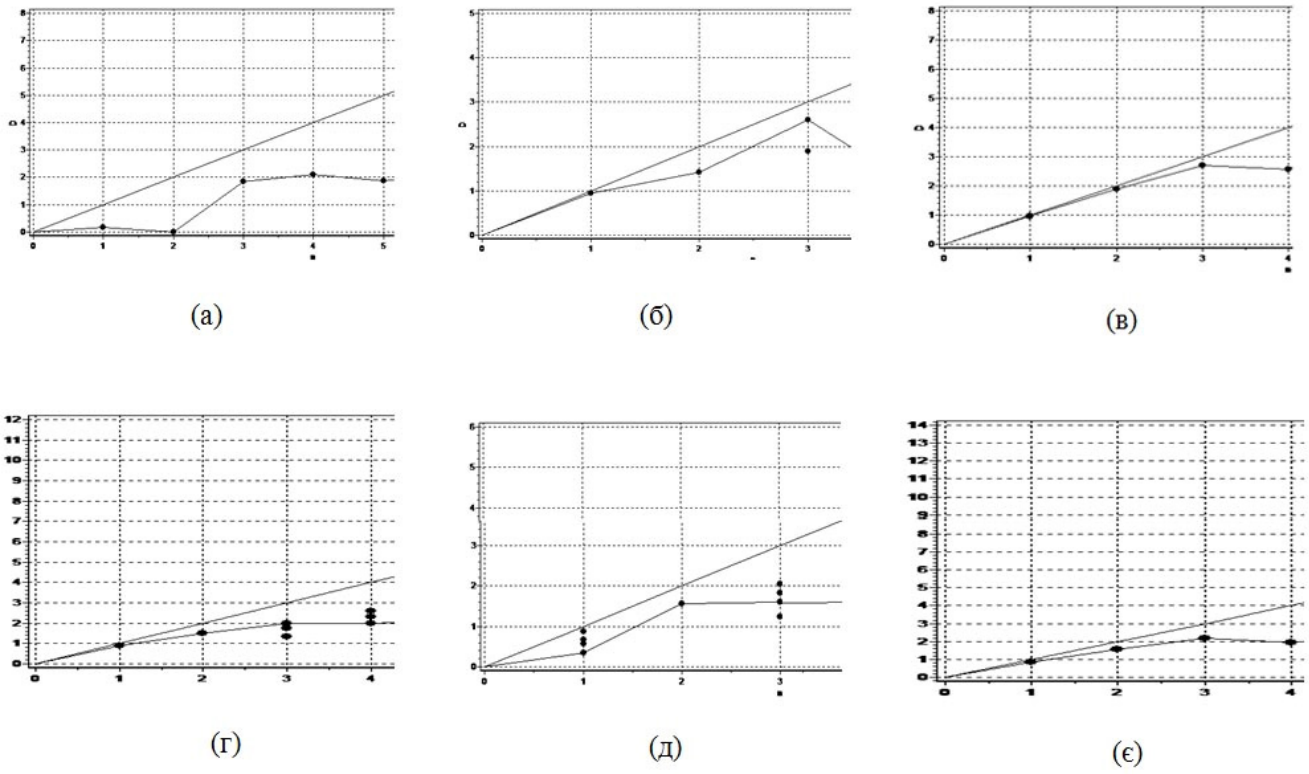


Рис. 1. Кореляційні розмірності рядів. а – «Бойкот», б –«Демонстрації», в –« Лобіювання», г – «Тиск та протест», д –« Судові процеси», є –« Страйк».

Починаючи з деякої розмірності m простору, кореляційна розмірність D_c досягає насичення і перестає змінюватися. Значення m, при якому це відбувається, є оцінкою мінімальної розмірності вкладення, а значення D_c оцінкою кореляційної розмірності аттрактору.

5. Результати застосування

Описана вище методика була застосована до отриманих в [1,2] рядів даних. Послідовність побудови фазових траєкторій наступна:

1. Визначення розмірності фазового простору (розмірності вкладення) за допомогою кореляційного інтегралу;
2. Побудова фазової траєкторії у просторі визначеної на попередньому кроці розмірності.

На рис. 1 представлена кореляційна розмірність та розмірність фазового простору для отриманих рядів при автоматично знайденій оптимальній затримці («Бойкот» – 6, «Демонстрації» – 12, «Лобіювання» – 8, «Тиск та протест» – 9, «Судові процеси» – 6, «Страйк» – 8).

Кореляційна розмірність D_c ряду, що відповідає виду політичної активності «Бойкот» дорівнює 2.127, розмірність фазового простору m – 3; для ряду «Демонстрації» D_c=2.599, m=3; «Лобіювання» D_c=4.627, m=3; «Тиск та протест» D_c=2.996, m=3; «Судові процеси» D_c=2.278, m=3; «Страйк» D_c=2.246, m=3. Як видно з результатів для всіх розглянутих часових рядів розмірність фазового простору (розмірність вкладення) дорівнює 3.

Для порівняння, кореляційна розмірність гаусівського «шуму» 9.743, розмірність фазового простору 15. Кореляційна розмірність для загального Броунівського руху 2.541, розмірність фазового простору 7 [6].

На рис. 2 побудовані аттрактори рядів, що досліджуються, які дозволяють зробити висновок про поведінку відповідних процесів.

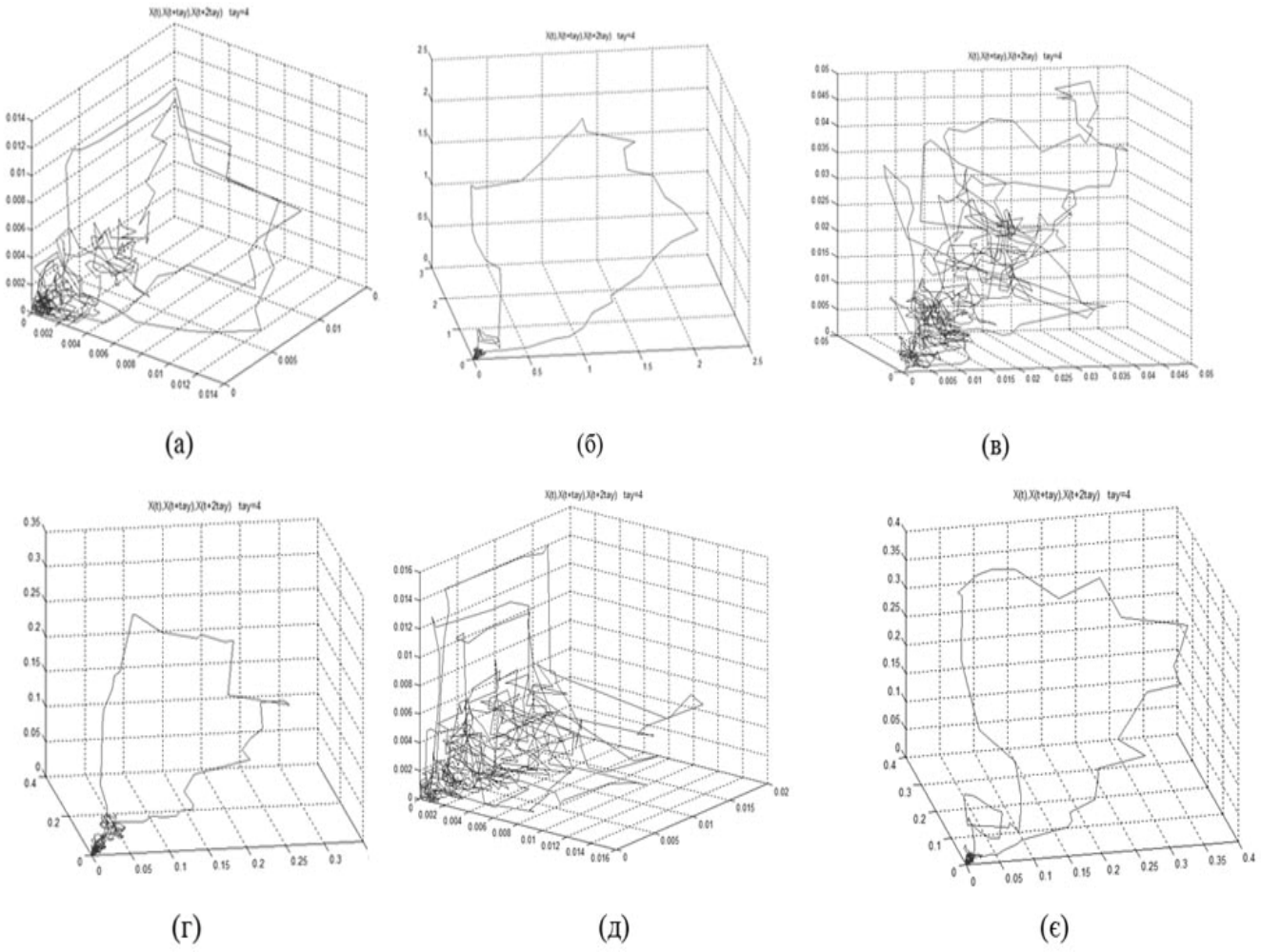


Рис. 2. Аттрактори рядів. а – «Бойкот», б –«Демонстрації», в –« Лобіювання», г – «Тиск та протест», д –« Судові процеси», є –« Страйк».

При побудові фазових просторів значення τ для кожного випадку обчислювалось окремо.

6. Висновки

Отримані результати підтвердили, що інформаційні потоки, що досліджуються, не є випадковими процесами та простіші, ніж випадкові процеси. На це вказує порівняння обчислених показників з кореляційною розмірністю та розмірністю фазового простору гаусівського «шуму» та броунівського руху.

Усі досліджені процеси мають розмірність фазового простору (розмірність вкладення) рівну 3, що вказує на наявність трьох основних компонент, які формують динаміку даних систем. Таким чином можна зробити висновок, що на досліджені інформаційні потоки, які відображають прояви громадянської активності під час реформування податкової системи України в електронних засобах масової інформації, впливають три сили. Виявлення характеру впливу цих сил на процес відображення є темою для наступних досліджень авторів.

Візуальний аналіз отриманих фазових траєкторій дозволяє зробити висновок про спорідненість (схожість поведінки) інформаційних потоків, що відображають «Лобіювання» та «Судові процеси», а також «Демонстрації», «Тиск та протест» та «Страйк». В першому випадку поведінка системи характеризується великою амплітудою та достатньо великою кількістю «стрибків» від потенційної області рівноваги системи. В другому випадку ми маємо теж велику амплітуду, але невелику кількість окремих «стрибків». Для «Демонстрацій», «Тиску та протесту» та «Страйку» система набагато сильніше намагається повернутись в явно виражену область рівноваги, ніж в першому випадку. Щодо проявів «Бойкоту», то поведінка цієї системи є середньою між розглянутими випадками, та характеризується явною областю рівноваги, великою амплітудою і середньою кількістю «стрибків».

Список літератури

1. Качинський А.Б., Добровольський Є.Л., Барановський О.М., Ланде Д.В. Прикладні аспекти застосування моделей взаємозв'язку "ЗМІ -соціум – державна політика" на прикладі реформування податкової системи України //Інформаційна безпека людини, суспільства, держави. – К.: Національна академія Служби безпеки України, №2 (2), 2011г., с. 140-146
2. Барановський О.М., Качинський А.Б., Добровольський Є.О., Ланде Д.В. Характеристики інформаційних потоків, пов'язаних з реформуванням податкової системи України //Правова інформатика. – К.: Науково-дослідний центр правової інформатики Національної академії правових наук України, №2 (34), 2012г., с. 89-95
3. Петерс Э. Хаос и порядок на рынках капитала. Новый аналитический взгляд на циклы, цены и изменчивость рынка. – Москва: Мир, 2000. – 277 с.
4. Packard N.H., Crutchfield J.P., Farmer J.D., Shaw R.S. Geometry from a time series. //Phys. Rev. Lett., 45, 1980, p. 712-716
5. Takens F. Detecting strange attractors in turbulence. //Vol. 898 of Lecture Notes in Mathematics. Springer, Berlin, 1981, p. 366-381
6. Давыдов А.А. Системный подход в социологии: новые направления, теории и методы анализа социальных систем. – М.: КомКнига, 2005 – 328 с.

УДК 004

РОЛИК А.И.

ТЕНДЕНЦИИ И ПЕРСПЕКТИВЫ РАЗВИТИЯ УПРАВЛЕНИЯ ИНФОРМАЦИОННЫМИ ТЕХНОЛОГИЯМИ

Проанализированы основные тенденции развития управления ИТ. Определены факторы, которые окажут влияние на развитие управления ИТ в ближайшие годы. Предложены иерархия уровней корпоративной ИТ-инфраструктуры и пять универсальных процессов, общих для всех категорий команд управления ИТ.

Main trends in IT management were analyzed. The factors, that will impact the development of IT management in coming years, were defined. A hierarchy of levels of the corporate IT infrastructure and five universal processes that are common to all categories of IT management teams were proposed.

Введение

Бизнес рассматривает информационные технологии (ИТ) в качестве источника повышения своей производительности и улучшения конкурентоспособности. Эффективность всех бизнес-процессов повышается за счет автоматизации, обеспечиваемой ИТ-услугами. Опрос в конце 2010 г. почти трех тысяч руководителей крупных копаний, проведенных Forrester Research, подтвердил, что ИТ является одним из основных элементов бизнес-модели, способствующим снижению издержек бизнеса и ускорению бизнес-инноваций [1]. В то же время ведущие специалисты в области автоматизации бизнеса отмечают резкое снижение отдачи от вложений в ИТ, а экономический кризис заставил предприятия сокращать ИТ-бюджеты и остро ставить вопрос об экономичности ИТ, хотя до кризиса ИТ были единственной отраслью, для которой не существовало понятия экономической эффективности [2].

Увеличивающееся количество ИТ-услуг, необходимых для автоматизации бизнес-технологий (БТ), усложнение распределенных приложений, возрастающее количество компонентов ИТ-инфраструктуры (ИТИ), а также количество и сложность информационно-коммуникационных технологий (ИКТ), которые необходимо поддерживать ИТ-подразделению, приводит к снижению эффективности работы ИТ-организаций, повышению экономических и человеческих затрат на обслуживание ИТИ. Для повышения эффективности функционирования ИТИ и автоматизации процессов обслуживания создаются системы управления ИТ-инфраструктурой (СУИ) [3–5]. Увеличение бизнес-спроса на ИТ-услуги, сложность и дороговизна современных ИТ, а также постоянное совершенствование бизнес-процессов и вызванная этим необходимость разработки и внедрения новых ИТ приводят к чрезмерному усложнению СУИ, являющейся продуктом системной интеграции многообразных подходов и порой несовместимых решений от различных производителей.

Возрастающая сложность управления ИТ, сопровождающаяся увеличением затрат на операционную деятельность ИТ-подразделения, делает актуальной задачу поиска новых подходов к управлению ИТ. Анализу основных тенденций развития управления ИТ и посвящена данная статья.

Анализ проблем управления ИТ

Исследования Forrester Research [6], проведенные в начале 2012 г. и анализ материалов других аналитических агентств показали, что несмотря на понимание важности ИТ для бизнеса, осознание ИТ-руководителями места и роли ИТ-подразделения в организационной структуре предприятия, которые сводятся к необходимости обеспечивать максимальную эффективность БТ с наименьшими затратами на ИТ, повсеместно отмечаются следующие проблемы:

- существует несогласованность приоритетов бизнеса с приоритетами ИТ. Очень часто бизнес и ИТ считают критичным совершенно разные вещи. В то время как бизнес интересует совокупность скоординированных сервисов, ИТ предлагает множество разрозненных ресурсов и технологий, а также приложений, решающих отдельные задачи;
- бизнес хочет осознавать ценность инвестиций и отдачу от внедрения ИТ. Для

удовлетворения запросов и потребностей бизнеса, ИТ-организации должны использовать модели операционной деятельности, предполагающей представление услуг, а не просто поддержание функционирования приложений. Управление ИТИ должно претерпеть изменения, делая акценты на службу каталогов, порталы самообслуживания клиентов, автоматизацию и координацию операционной деятельности. Эффективная организация и автоматизация рутинных процессов управления ИТИ позволит высвободить персонал для решения стратегических и инновационных задач создания перспективной архитектуры ИТ;

- отсутствует интеграция процессов управления ИТ. Даже такие родственные процессы, как управление устранением неисправностей, проблемами, инцидентами, изменениями все еще слабо связаны. Позитивным является переход от накопления технологий к накоплению процессов и сервисов. Следующим шагом должны стать координация и интегрирование процессов и услуг [6];
- особую актуальность приобретают процессы, улучшающие удовлетворенность заказчиков ИТ-услуг, что особенно важно для облачных технологий, когда акценты работы ИТ-подразделения смещаются в сторону процессов, связанных с обеспечением качества обслуживания заказчиков, таких как, например, управление портфелем услуг и управление взаимоотношениями с заказчиками;
- бизнес и ИТ-руководители заинтересованы в повышении отдачи от ИТ. По опросу почти трех тысяч ИТ-руководителей крупнейших компаний основным приоритетом 2012 года является существенное повышение эффективности ИТ, которое наряду с оптимизацией затрат позволит перевести операционную деятельность ИТ-подразделения на модель сервис-провайдера услуг, так как это необходимо бизнесу [6].

Исследования в области управления ИТИ сосредоточены главным образом на том, как ИТ-подразделение должно быть организовано и управляемо. Сюда включаются вопросы создания корпоративной ИТИ, спецификация метрик для ИТ-функций, управление проектами, разработка и поддержание стратегического ИТ-плана, а также создание организационной структуры ИТ-подразделения.

Рынок ПО управления ИТ и ИТИ один из наиболее динамичных и многогранных рынков ИТ-индустрии, значимость которого существенно возрастает с годами. Аналитики Forrester [7], считают, что под категорию систем управления (СУ) ИТ подпадают все программные продукты, осуществляющие мониторинг, обнаружение и идентификацию любого аномального поведения ИТИ. Сюда относят ПО, осуществляющее: управление серверами, приложениями, сетями, работой конечных пользователей, событиями, производительностью баз данных, инструментарий автоматизации ЦОД и пр. Под эту категорию подпадает и ПО, осуществляющее управление самой ИТИ (управление ресурсами, изменениями и конфигурацией), поддержку ее функционирования и эксплуатации (планирование работ, управление потоками работ), управление коммуникациями как внутри ИТ-подразделения, так и с бизнес подразделениями (служба поддержки, управление качеством обслуживания, управление бизнес-сервисами). Таким образом, управление ИТИ рассматривается как составляющая управления ИТ в целом.

Существенную концептуальную роль в сфере управления ИТ безусловно играет библиотека ITIL (IT Infrastructure Library), обобщающая передовой опыт. В соответствии с положениями ITIL ИТ-подразделение превращается в сервисную организацию, являющуюся партнером бизнеса с понятными для бизнес-подразделений функциями, а СУИ должны поддерживать процессное управление ИТ, автоматизировать выполнение операций и, самое главное, предоставлять зависимость бизнес-сервисов от ИТИ. Отличительной чертой библиотеки ITIL является ориентация на ИТ-операции. При правильном внедрении ITIL позволяет улучшить качество предоставления ИТ-сервисов, снизить продолжительность простоев ИТ-ресурсов, ускорить разрешение проблем и обеспечить высокий уровень безопасности. В настоящее время ITILv3 является фактическим стандартом ITSM (IT service management) и принят или принимается 60% организаций, предоставляющих ИТ-услуги [6]. По опросу Forrester, проведенному в 2012 г., 85% ИТ-директоров отмечают, что применение ITIL позволяет значительно повысить продуктивность сервисов, улучшить качество ИТ-сервисов (83%), повысить престиж бизнеса (65%) и существенно сокра-

тить затраты на операционную деятельность (41%) [6]. По мнению ИТ-директоров выгоды, получаемые от внедрения методологии ITIL, с лихвой окупают существенные затраты времени и человеческих ресурсов на реализацию процессного подхода. Так, только за счет построения службы поддержки пользователей компании Finisar в соответствии с ITIL степень удовлетворенности клиентов возросла с 33 до 95%, а затраты на поддержание работоспособности ИТИ сократились с 4 до 2,4% от доходов компании [8].

Поиск путей повышения производительности и предсказуемости работы ИТ привел к появлению двух центральных столпов очередной волны в сфере ПО управления ИТ: управления бизнес-сервисами (Business Service Management, BSM) и базы данных управления конфигурациями (Configuration Management Database, CMDB), позволяющей создавать упорядоченное описание объектов технологической инфраструктуры для бесшовной поддержки процессов ITSM. База CMDB необходима при осуществлении контроля за ресурсами, поскольку содержит историю всех выполненных операций управления ИТИ. CMDB является основой ITIL и представляет собой отражение всех ИКТ, автоматизированных систем, маршрутизаторов, серверов, персональных компьютеров (ПК) и т. д., содержит записи всех изменений, связанных с ресурсами ИТИ, историю инцидентов, выявленных при работе этих ресурсов, с описанием места ресурсов в ИТИ. При проведении модернизации ИТИ или внесении существенных изменений в конфигурационные файлы, которые могут привести к временной неработоспособности бизнес-сервисов, CMDB может помочь понять, на каких бизнес-процессах это скажется. Появляется возможность оптимизировать время проведения модернизации с учетом графика работ и загруженности бизнеса для сокращения возможных негативных последствий от приостановки предоставления сервиса. В версии ITILv3 роль BSM и CMDB существенно возросла. Однако разрекламированные широкие возможности CMDB-продуктов, которые должны были привести к появлению монолитных и всеобъемлющих хранилищ данных о конфигурациях компонентов ИТИ с последующей чуть ли не автоматической реализацией процессов управления, не оправдались [9].

Несмотря на популярность ITIL внедрение рекомендаций сопряжено с рядом трудностей. Так, некоторые эксперты отмечают существенное увеличение сложности ITILv3 по сравнению с ITILv2, обусловленной тем, что библиотека стремится быть максимально универсальной и дать ответы на все вопросы. Для внедрения методологии ITIL организации вынуждены приглашать внешних специалистов, это объясняется тем, что книги последней версии ITIL создавались консультантами или производителями, зарабатывающими на жизнь внедрением описанных методик [8]. ИТ-консультанты рекомендуют обращаться к интеграторам, которые решат проблемы заказчика на основе конкретных продуктов. В то же время для заказчика выбор и настройка конкретных инструментов управления ИТ-услугами часто оказывается гораздо дороже лицензий на интегрируемые продукты.

Бизнес развивается очень динамично, а реализация ITIL сопряжена с длительным поэтапным внедрением всех процессов и требует много времени на достижение ими стадии постоянного совершенствования. На внедрение новых процессов у большинства ИТ-организаций уходят годы и еще годы требуются для того, чтобы оценить достигнутые результаты [8]. При этом СУИ, построенная в соответствии с ITIL, не любит внесения изменений.

Несмотря на то, что ITIL является обобщением лучшего опыта управления ИТ, библиотека не содержит рекомендаций или советов по практическому применению обобщенного в ней опыта и не содержит методик оптимальной декомпозиции процессов [10]. Фактически ITIL не внедряется в организации, а используется как методология проведения преобразований в ИТ-организациях, на основе которой разрабатывая собственные процедуры, исходя из принципов ITIL [8].

Основной недостаток ITIL пользователи видят в том, что применение ITIL оправданно для крупных корпораций, финансовых или банковских структур, телекоммуникационных компаний с большими ИТ-подразделениями, может быть использовано некоторыми организациями среднего бизнеса и неприменимо для малого бизнеса.

Несмотря на недостатки, аналитики видят ITIL в качестве перспективной основы для организации управления ИТ, а большинство про-

изводителей и разработчиков СУИ приняли процессную модель описания бизнеса и использование СУ, основанной на ITIL.

Экономический кризис заставил бизнес ограничивать бюджет на ИТ и жестко контролировать ИТ-расходы, при этом пересмотреть свое отношение к ИТ и впредь рассматривать управление ИТ по принципу бизнес в бизнесе. Теперь ИТ-подразделение должно руководствоваться экономическими факторами, а не технической целесообразностью. Это принципиально меняет общую картину рынка ПО управления ИТИ.

Целью статьи является определение основных тенденций и перспектив развития управления ИТ на ближайшие годы.

Обобщенная схема ИТ-инфраструктуры предприятия

Современный этап развития систем обработки информации завершает виток спирали эволюции ИТ-индустрии, вернувшись к сосредоточенным центрам обработки информации, которые в 60-х–80-х годах прошлого века существовали в виде вычислительных центров (ВЦ), а в настоящее время – в форме центров обработки данных (ЦОД). При этом мы стали свидетелями начала очередного витка эволюции ИТ, признаком которого является переход к парадигме облачных вычислений.

ВЦ, который часто назывался информационно-вычислительный центром или вычислительным центром коллективного пользования, когда предоставлял вычислительные ресурсы сторонним организациям, представлял собой специализированное подразделение, которое имело комплекс помещений с вычислительной техникой и обслуживающим персоналом, предназначенный для предоставления вычислительных услуг [11]. Как правило, ввод заданий в ЭВМ и вывод результатов осуществлялся через терминалы, расположенные на территории ВЦ. Выполнение задач пользователей обычно осуществлялось в режиме разделения времени, а в случае применения мультипроцессорных систем использовалась параллельная обработка информации.

Появление ПК, уступающих по мощности ЭВМ в ВЦ, но имеющих существенно меньшую стоимость, привело к переходу от централизованной к децентрализованной обработке информации, а развитие сетевых технологий позволило соединять в единое целое совокупность разнородных вычислительных систем для коллективного использования вычислительных ресурсов.

В настоящее время в средних и некоторых крупных предприятиях превалирует тенденция к консолидации вычислительных ресурсов в специализированных помещениях, называемых серверными комнатами, позволяющая существенно сократить эксплуатационные затраты на поддержание парка вычислительной техники, осуществляющей обработку информации и обеспечивающей базу поддержки бизнес-приложений. Серверные помещения стали прообразом специальных зданий с мощной инженерной инфраструктурой, огромным количеством серверов, системами хранения данных и пр., получивших название дата-центров или ЦОД. За рубежом бум создания ЦОД, представляющих собой совокупность информационной, телекоммуникационной и инженерных инфраструктур, пришелся на период 1995–2000 гг.

Альтернативой ЦОД, с точки зрения возможности получения большой вычислительной мощности, можно считать технологию Grid [12], которая использует распределенную инфраструктуру для объединения с помощью телекоммуникационной сети и специализированного ПО множества разнотипных вычислительных ресурсов, а также слабосвязанных, гетерогенных компьютеров в виртуальный суперкомпьютер с огромными вычислительными возможностями. Однако практика доказала, что централизованная обработка данных в ЦОД решает множество задач построения ИТ-систем масштаба предприятия дешевле, проще, надежнее и безопаснее, чем распределенная на разрозненных серверах или Grid-системах.

Таким образом, четко просматриваются витки эволюции телеобработки информации от централизованных систем на основе мэйнфреймов к децентрализованным на основе ПЭВМ, и далее к централизованным на основе ЦОД с использованием клиент-серверных технологий с возможным последующим переходом к децентрализованным на основе GRID или подобных технологий. Что касается сегодняшнего дня, то можно говорить о том, что ИТ-индустрия на пороге революционных преобразований, основой которых является парадигма облачных вычислений. Реализация облачных технологий в настоящее время в основном осуществляется на базе ЦОД.

В то же время, из двух основных подходов к консолидации вычислительных ресурсов – на основе Grid-систем и на базе ЦОД, в данный момент Grid-технологию нельзя рассматривать в качестве основы построения корпоративной ИТИ. Поэтому в настоящее время ввиду эффективности централизованной обработки информации и низких экономических затрат при обслуживании аппаратного и программного обеспечения сосредоточенных серверов, корпоративные и специализированные ИТ-системы строятся по схеме, приведенной на рис. 1.

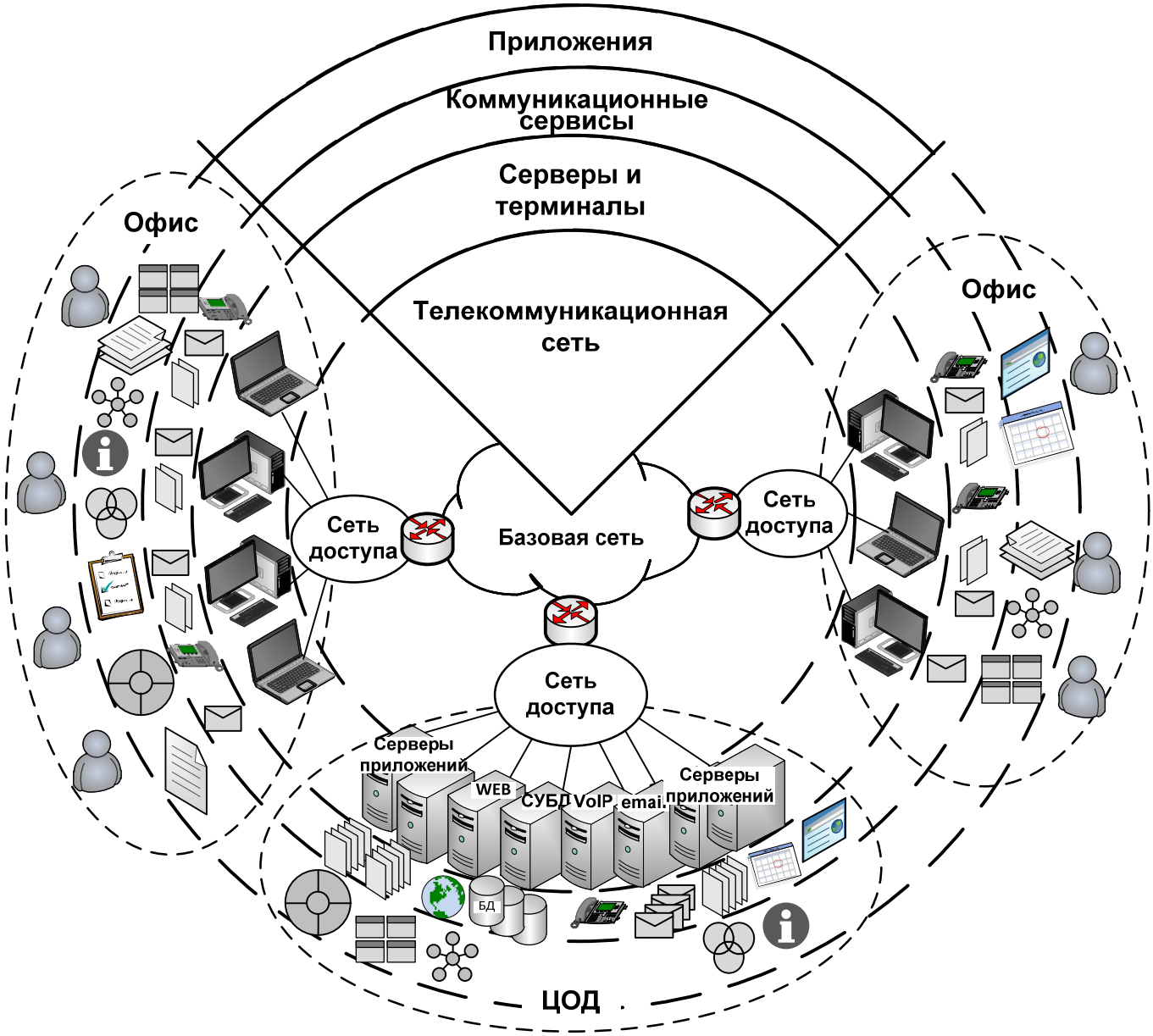


Рис. 1. Обобщенная структура корпоративной ИТ-системы

ИТИ любого предприятия или организации представляет собой комплекс взаимосвязанных структур, систем, объектов и пр., обеспечивающих надлежащее функционирование ИТ-системы. Конкретная схема ИТИ определяется размером организации, характером решаемых бизнес-задач, используемыми ИТ и пр.

Предприятия, ограничивающиеся использованием в локальной сети нескольких бизнес-приложений, установленных на сервер и позволяющих несколько повысить производительность труда сотрудников за счет автоматизации ряда бизнес-процессов, могут нести существенные убытки из-за, например, потери данных вследствие отсутствия системы резервного копирования. ИТИ, в которой отсутствует только часть важных составляющих, будет более эффективной, но величина этого эффекта может

быть незначительной. Максимальный эффект от функционирования ИТИ может быть достигнут только при наличии комплексной полноценной ИТИ, позволяющей рационально выполнять процессы деятельности и эффективно решать бизнес-задачи. В то же время организация такой ИТИ – сложный и длительный процесс, требующий от предприятия серьезных капиталовложений на создание ИТИ и существенных текущих затрат на ее поддержание.

При рассмотрении вопросов, связанных с управлением ИТИ, выделяется различное количество иерархических уровней в зависимости от специфики бизнеса предприятия и характера решаемых СУИ задач. Так, в [13] при рассмотрении метрик операторов телекоммуникационных сервисов (ОТС) выделяется пять иерархических уровней: телекоммуникационной технологии, сети, сервисов, абонентский и телекоммуникационного бизнеса. Следуя модели OSI, [14] концепция системы управления сетями (TMN) рассматривает сетевую логическую архитектуру, включающую в себя пять уровней управления [15] – это уровень сетевых элементов, уровни управления: элементами, сетью, услугами и уровень бизнес-управления. В [16] концепция управления ИТИ рассматривается применительно к иерархической структуре, содержащей три слоя: нижний – управление сетями, средний – управление системами, верхний – управление сервисом ИТ.

Целесообразно рассматривать ИТИ предприятия в виде обобщенной иерархической схемы, приведенной на рис. 2, и представляющей собой совокупность информационно-телекоммуникационной системы (ИТС) и ИТ-подразделения.

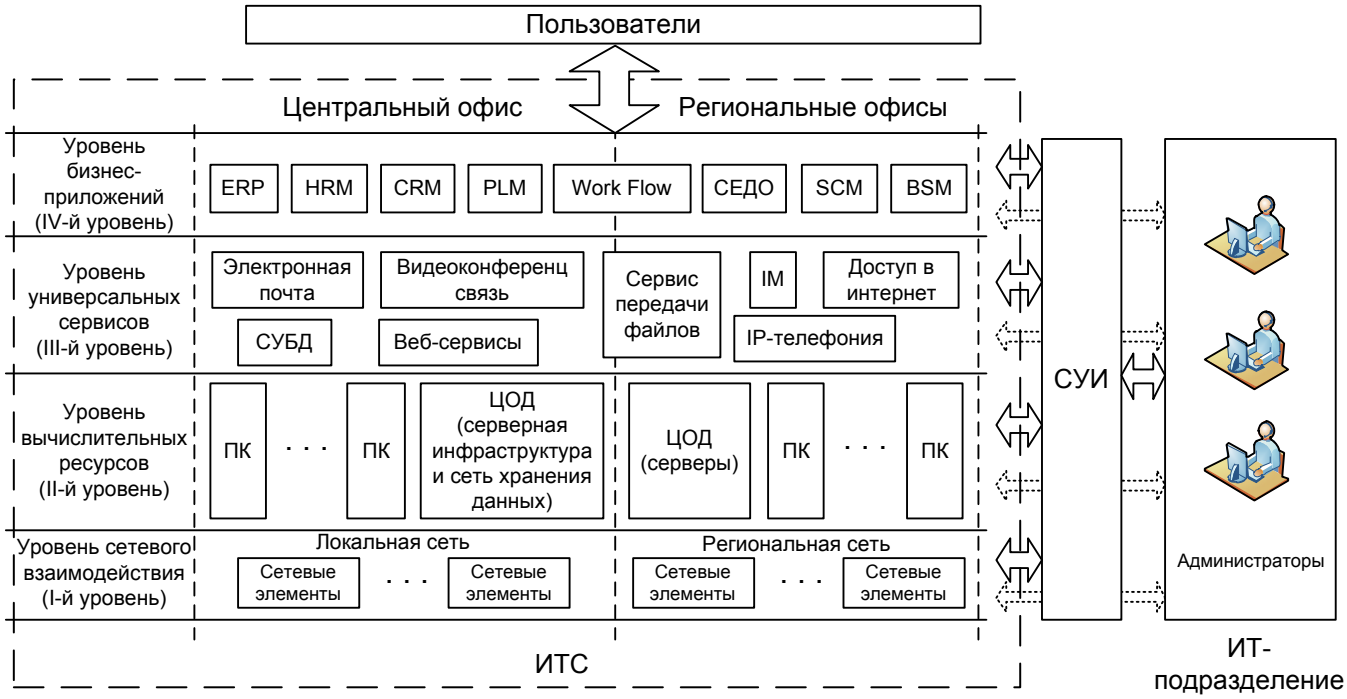


Рис. 2. Обобщенная схема ИТИ предприятия

В ИТС – организационно-технической совокупности ИТ-системы и телекоммуникационной сети [5], предлагается выделить четыре иерархических уровня: бизнес-приложений, универсальных сервисов, вычислительных ресурсов и сетевого взаимодействия. В свою очередь, при необходимости, каждый из четырех уровней может быть разделен на подуровни.

На IV-м иерархическом уровне работают распределенные приложения, имеющие непосредственное отношение к автоматизации выполнения бизнес-процессов или процессов деятельности и предоставляющих ИТ-услуги пользователям. На этом уровне работают системы: планирования ресурсов предприятия (ERP), управления персоналом (HRM), управления взаимоотношениями с клиентами (CRM), управления информацией об изделиях (PLM), управления рабочими процессами (Work Flow), документооборотом (СЕДО), BSM и множество других ИТ, необходимых для успешного ведения бизнеса предприятия. К этому уровню относятся программные продукты, определяющие значимость бизнес-процессов для предприятия на некотором интервале времени и компоненты СУИ, управляющие распределением вычислительных и коммуникационных ресурсов. Так,

например, BSM позволяет настраивать инфраструктурные ресурсы на приоритеты бизнеса.

На III-й иерархический уровень отнесены сервисы, не зависящие от специфики бизнеса предприятия. Такими сервисами и ПО являются: электронная почта, СУБД, видеоконференцсвязь, различные веб-сервисы, сервис передачи файлов, включающий обмен файлами с помощью FTP и пиринговых сетей, IP-телефония, служба мгновенного обмена сообщениями (Instant messenger, IM), средства обеспечения и контроля доступа в интернет и пр.

Уровень вычислительных ресурсов включает в себя терминалы пользователей в виде ПК и пр., а также ресурсы ЦОД. Предприятия малого размера, а также филиалы практически всегда вместо ЦОД используют серверные или кластерные решения, на которых размещают серверные части ПО уровней бизнес-приложений и универсальных сервисов. Все или только часть вычислительных ресурсов могут быто реализованы в виде облачных ресурсов с соответствующими сервисами в виде SaaS, PaaS, IaaS (ПО, платформа и инфраструктура – как услуга) и т.д. При использовании внешних облаков ИТ-подразделение решает организационные вопросы и осуществляет контроль качества предоставления сервисов.

Посредством уровня сетевого взаимодействия обеспечивается доступ пользователей к ресурсам ЦОД и ко всем распределенным вычислительным ресурсам. Доступ сотрудников центрального офиса обычно производится по локальной сети, а сотрудников региональных офисов или сотрудников, находящихся вне компании, – через региональную сеть с использованием средств удаленного доступа.

Управление ИТИ осуществляется с помощью СУИ, предназначенной для ведения инвентаризационных баз данных об информационных, вычислительных и телекоммуникационных активах предприятия, мониторинга состояния и управления поддержанием состояния этих активов на заданном уровне, управления доступом и распределением вычислительных и телекоммуникационных ресурсов, контроля, анализа поведения и поддержки пользователей, планирования и управления закупками ИТ-активов, автоматизации деятельности и ведения отчетности ИТ-подразделения, а также решения множества других задач, перечень которых определяется размерами и особенностями ИТИ. СУИ позволяет не только уменьшить затраты на содержание ИТ-подразделения, но и является средством автоматизации работы администраторов и руководителей ИТ-подразделения. Таким образом, СУИ не только автоматизирует работу администраторов по управлению ИТИ, но и решает задачи автоматизации управления ИТ-подразделением.

Обязательной составляющей ИТИ предприятий среднего и крупного бизнеса является ИТ-подразделение, главной задачей которого является обслуживание внесенных в каталог услуг бизнес-процессов. Подробности предоставления ИТ-услуг регламентируются пакетом соглашений об уровне сервиса (SLA), заключенных между бизнес-пользователями и ИТ-подразделением. В SLA определяются значения ключевых показателей эффективности (KPI) и качества (KQI) – ограниченный набор объективно измеримых параметров, позволяющий оценить результативность работы ИТИ. Показателем качества ИТ-услуг, как правило, является фактическая эффективность в сравнении с критериями, заданными в SLA [17]. Для поддержания значений KPI и KQI на зафиксированном в SLA уровне, администраторы обеспечивают непрерывное функционирование ИТИ, осуществляют обслуживание и ремонт. При этом администраторы могут управлять элементами ИТИ как с помощью СУИ, так и непосредственно. В последнем случае используются фирменные или универсальные СУ отдельными элементами ИТИ (пунктирные стрелки на рис. 2). Стандарт ISO 20000 [18, 19] подчеркивает важную роль СУИ, поддерживающей управление ИТ-услугами, для контроля метрик, характеризующих поведение ИТ-процессов.

В соответствии с ITIL в управлении ИТ-услугами принимают участие минимум два отделения ИТ-подразделения: поддержки сервисов (Service Desk), обеспечивающее поддержку пользователей, и отделение управления ИТИ (ICT Infrastructure Management Team), отвечающее за функционирование ИТС.

Анализ проблем управления ИТ-инфраструктурой

ИТ-инфраструктура (инфраструктура информационных технологий) – организационно-техническая совокупность программных, вычислительных и телекоммуникационных

средств и связей между ними, а также обслуживающего персонала, обеспечивающая надлежащее предоставление пользователям, которыми являются сотрудники предприятия или организации, информационных, вычислительных и телекоммуникационных ресурсов и услуг, необходимых для выполнения ими процессов деятельности и решения бизнес-задач.

ИТИ включает в себя информационные активы, приложения, средства взаимодействия и техническую инфраструктуру. Управление каждой из этих составляющих сопряжено с рядом специфических проблем, обусловленных индивидуальными особенностями: информация – постоянный рост объемов обрабатываемых данных; приложения – необходимость интеграции; средства взаимодействия – необходимость увеличения полосы пропускания, коммуникационных возможностей и поддержки новых сервисов; техническая инфраструктура – возрастающая сложность. Кроме того, вся ИТИ пронизывается средствами обеспечения безопасности. Для управления этими составляющими ИТИ создаются различные СУ.

Первые системы административного управления создавались и продавались производителями и поставщиками сетевого оборудования. Большая часть фирменных систем предназначалась для работы со специфическим аппаратным и программным обеспечением. Модернизация и применение этих систем для управления оборудованием других производителей были или сильно затруднены, или практически невозможны. Во многом это положение сохраняется и по сей день, несмотря на появление дорогостоящих универсальных систем административного управления.

В настоящее время для систем связи производители коммуникационного оборудования выпускают и СУ, ориентированные на управление только отдельным типом оборудования. Фирменные СУ, использующие различные ноу-хау, носят закрытый характер, могут использовать собственные языки программирования и обычно содержат примитивные средства сопряжения с другими СУ, реализуемые, как правило, в виде обмена файлами с данными конфигураций и результатами мониторинга. Такая же картина имеет место и в области компьютерных технологий.

В то же время в современных ИТИ для обеспечения эффективности бизнес-процессов и управления ими используется большое количество разнообразных ИКТ и оборудование разных производителей. ИКТ обеспечивают доступ пользователей к различным ресурсам и одновременно потребляют часть ресурса для собственных нужд.

Все современные БТ имеют различную потребность в разнообразных информационно-коммуникационных ресурсах, причем для повышения их эффективности и, исходя из экономических соображений, для многих бизнес-процессов задействуются одни и те же ресурсы. При этом требуемые объемы ресурса, а также необходимые и запрашиваемые условия предоставления ресурса отдельным приложением могут иметь существенную динамику, независящую от других приложений. В то же время, все ресурсы ИТС ограничены, а при одновременном обращении к общим ресурсам всегда возникает конфликт, при разрешении которого СУ ресурсом не всегда могут отдать предпочтение наиболее важному приложению по причине отсутствия информации о значимости приложений.

Каждая из используемых в ИТИ ИКТ может иметь собственную СУ, работающую независимо от других СУ. При этом разрозненные СУ решают только собственные задачи и, как правило, совершенно не учитывают проблемы, возникающие в других СУ. В этом случае СУ ИТ, непосредственно управляющие отдельными технологиями, оборудованием или средствами и опосредованно – отдельными бизнес-процессами, могут не учитывать значимость других бизнес-процессов и приоритеты приложений. Более того, разрозненные СУ не могут учитывать изменение приоритетов приложений при изменении условий функционирования ИТИ.

Для решения этих проблем необходимо найти комплексное решение, желательно не очень сложное и громоздкое, позволяющее уменьшить проблемы взаимодействия, совместимости различных ИТ, совершенствовать процессы и методы коллективной работы администраторов, ИТ-менеджеров, менеджеров производства, повысить качество работы ИТ, организовать эффективное использование ресурсов и, следовательно, эффективность управления производственными процессами и эффективности функционирования ИТИ.

Таким комплексным решением является СУИ, включающая в себя совокупность методов, средств и мероприятий, направленных на поддержание работоспособности ИТИ. Требования к перспективным СУИ сформулированы

в [3]. СУИ является сетью, наложенной на управляемую ИТИ. Эффективность работы СУИ определяется применяемым в ней информационными, коммуникационными и компьютерными технологиями, а непосредственная зависимость успешности бизнеса от ИТ требует, чтобы управление ИТИ осуществлялось с учетом актуальной важности бизнес-процессов. В настоящее время неизвестны интегрированные СУИ, позволяющие централизованно управлять всеми подсистемами ИТИ, а также ИТС, и представляющие инструментарий ИТ-подразделению для эффективного автоматизированного управления всеми распределенными ресурсами и приложениями.

Основные стандарты, практики и протоколы управления ИТИ

Аспекты построения СУ верхними уровнями предоставления услуг, а также управления ИТ-подразделением, хорошо проработаны в международных стандартах.

Решению вопросов корпоративного ИТ-управления, актуальных для крупных международных компаний и важных для среднего и малого бизнеса, посвящен ряд международных стандартов. В документах ITILv3, IS020000 [18, 19], COBIT [20], Six Sigma [21], рекомендациях M.3050 ITU-T и TM Forum по eTom и др. рассматриваются процессы управления предоставлением ИТ-услуг и вопросы менеджмента ИТ-подразделения (см. табл. 1).

Табл. 1. Основные стандарты, практики и протоколы, применяемые в управлении ИТИ

Организация	Стандарты, практики, протоколы	Особенности
OGC	ITIL	Способы организации работы ИТ-подразделений или организаций, предоставляющих услуги в ИТ-области
ISACA	COBIT	Принципы управления и аудита ИТ
Motorola	Six Sigma	Методика настройки бизнес-процессов с целью минимизации вероятности возникновения дефектов в операционной деятельности
IBM	PRM-IT	Процессная модель управления ИТ
TM Forum	eTOM	Архитектура бизнес-процессов телекоммуникационных компаний
RosettaNet	PIPs	Индустриальные стандарты для B2B коммуникаций
ISO	IS0 20000	Требования к менеджменту ИТ-сервисов
	CMIP	Интеллектуальные агенты, многофункциональная сложная система управления, объектно-ориентированный подход для представления управляемых устройств
IETF	SNMP	Агенты выполняют элементарные функции, управление простое, ориентировано на переменные
ITU-T	TMN	Стандартизируется только архитектура системы управления
DMTF	WBEM	Веб-ориентированное управление сетями и системами
	CIM	Объектно-ориентированная схема объектов управления
SEI	CMMI	Набор рекомендаций в виде практик, реализация которых необходима для совершенствования процессов в организациях разных размеров и видов деятельности. Доработанный вариант модели CMM.
OMG	CORBA	Поддержка разработки, развертывания и функционирования объектно-ориентированных распределенных программных систем

Эффективное руководство бизнесом требует улучшения корпоративного управления, которое в последние годы получает законодательное регулирование. Так, например, в США закон Сарбейнза-Оксли значительно ужесточает режим контроля и регулирования финансовой деятельности и, в частности, размещения активов. Для управления жизненным циклом ИТ-активов могут использоваться процессы управления конфигурациями и изменениями, являющиеся, в соответствии с ITIL, основой управления ИТ-услугами. Стандарт IS0 20000 делает конкретной центральную контролирующую функцию, а соответствие функционирования ИТ-подразделения положениям этого стандарта позволит оценить также и степень зрелости

корпоративного управления. В то же время, в IS0 20000 не рассматриваются аспекты управления инфраструктурой ИТС и распределенными приложениями.

Группа документов GB921 Международной ассоциации TM Forum определяет расширенную карту процессов телекоммуникационных компаний (eTOM), которая используется ОТС в качестве модели платформы предоставления услуг, основанной на концепции бизнес-процессов и стандартах управления сетями.

В совместно разработанном TM Forum и ITSMF документе [22] показано, как eTOM и ITIL могут быть использованы совместно. Несмотря на стратегическое схождение ITIL и eTOM, терминология, область применения и определения eTOM и ITIL существенно различаются. Прямого отображения ITIL на eTOM не существует и ITIL подходит только для высокоуровневых процессов eTOM. В то же время eTOM, концентрируясь на бизнес-процессах ОТС, не уделяет должного внимания сфере управления коммуникационной инфраструктурой.

Обобщение мирового практического опыта, а также анализа международных стандартов и рекомендаций, имеющих отношение к ИТ-управлению, аудиту и безопасности, составляет пакет документов COBIT [20]. Главная задача COBIT состоит в преодолении разрыва в видении целей бизнеса руководством компании и понимании целей бизнеса ИТ-подразделением, осуществляющим поддержку ИТ, способствующих достижению этих целей. Являясь средством аудита ИТ-системы компании, детальным описанием целей и принципов управления, объектов управления, управления ИТ-безопасностью, стандартизацией ИТ-процессов, протекающих в компании на верхних уровнях ИТ-бизнеса, COBIT не уделяет должного внимания вопросам управления ИТИ.

В стандартах ITU-T на TMN [15] предложена концепция и описана архитектура СУ сетями разных уровней и масштабов, предоставляющих различные типы услуг. Концепция TMN призвана объединить функции существующих СУ с добавлением высокоуровневого сервиса для организации единой сетевой структуры, обеспечивающей взаимодействие различных типов управляющих средств и телекоммуникационного оборудования, использующих стандартные протоколы. Универсальность и гибкость, характерная для TMN, стала причиной сложности архитектуры и интерфейсов, и, как следствие, незначительной практической реализации принципов TMN. Концепция TMN, позволяющая объединить практически все сети и сетевые технологии, не затрагивает вопросы управления ИТИ.

Компания IBM предложила свою процессную модель IBM PRM-IT (IBM Process Reference Model for IT) [10], которая концептуально мало чем отличается от ITIL. Модель PRM-IT рассматривает ИТ как важный компонент бизнеса, управление которым аналогично управлению производственными ресурсами.

Основные принципы процессной модели IBM PRM-IT заключаются в следующем:

1) Независимо от организации и технологий, существует некоторый фундаментальный набор процессов, необходимых для управления любой ИТ-средой.

2) Эти процессы не существуют и не выполняются независимо от других процессов, фактически процессы взаимосвязаны и взаимодействуют друг с другом.

3) Не существует единственной, доказуемо правильной декомпозиции процесса, или каких-либо способов демонстрации того, что конкретная реализация ИТ-процесса лучше любой альтернативной реализации. Все зависит от конкретных условий внедрения процесса.

4) Устоявшееся определение термина «лучшая практика» из ITIL представляет собой стандарт де-факто для множества ИТ-процессов, известных как “Service Management”.

Компания IBM уверяет, что следование предложенной модели позволит поэтапно проектировать и внедрять СУИ с использованием инструментария различных вендоров, с непротиворечивым дополнением модели успешными практиками из CMMI, COBIT, ITIL, Lean, Six Sigma, и т.д. Модель позволяет определять роли и распределить ответственность с гарантией их полноты и непротиворечивостью для возможного отражения на любую структуру ИТ-организации.

Модель компании IBM по сути является попыткой объединения ITIL и COBIT, а также других практик.

С аналогичной инициативой совместного использования ITIL и COBIT выступила компания BMC [23], пытаясь реализовать идею слияния на базе собственных продуктов. В настоящее время стандарты развиваются самостоятельно и не противоречат друг другу.

Все перечисленные выше стандарты сосредотачиваются в основном на вопросах организационного управления и не рассматривают детально управление ИТИ на всех уровнях, не предлагают модели, методы и алгоритмы рационального распределения ресурсов ИТС и пр. В настоящее время отсутствуют международные стандарты, специфицирующие комплексное управление ИТС на всех иерархических уровнях. Задачи нижних уровней ИТИ проработаны стандартами лишь по отдельности для сетей серверов, хранилищ данных, отдельных ИКТ и пр. и не рассматривают интегральный подход к управлению ИТИ. В то же время эффективность функционирования ИТИ может быть обеспечена только путем комплексного взаимосвязанного управления всеми иерархическими уровнями ИТИ.

Концепция “IT Management Software 2.0”

Несмотря на то, что имеются зрелые решения ПО управления ИТИ [24], этот рынок, вопреки кризису, продолжает активно развиваться, чему способствует появление новых сфер деятельности, где необходимы ИТ, создание новых ИТ, а также новое видение управления ИТ-подразделением, ИТ и ИТИ как производственно-экономической единицей. Это приведет к новым инструментариям поддержки принятия решений на разных уровнях бизнеса, наряду с тем, что такие сферы ИТ-управления, как управление ИТИ, станут все более массовыми, заставляя производителей соответствующего ПО постоянно совершенствовать компоненты СУИ и делать новые предложения [25], а к концу текущего десятилетия центр тяжести рынка ПО управления ИТ сместится в сторону глобального управления и оптимизации ИТ-ресурсов [26].

Для отслеживания всех этих изменений в области управления ИТ и ИТИ в 2009 г. аналитики Forrester предложили новую систематику категорий ПО управления ИТ, которая получила название “IT Management Software 2.0” (далее – ITMS2) [27]. До этого момента широко использовалась классификация программных решений для управления ИТ, выделяющая пятнадцать основных классов, дифференцируемых по выполняемым функциям [7, 28], включая управление: сетями, серверами, базами данных, событиями, хранилищами данных, работой конечных пользователей, приложениями, ресурсами ИТИ, изменениями и конфигурациями ИТИ, производительностью ИТИ-ресурсов, счетами ИТ-подразделения, персоналом ИТ-подразделения, а также службу поддержки пользователей (Service desk), планирование работ и выполнения операций, SLM и BSM.

Концепция ITMS2 предусматривает группирование основных классов управления в меньшее количество с добавлением новых классов, расширяет функции и роли ИТ-подразделения, декларирует новое видение и отношение к ИТ-подразделению.

ITMS2 основывается на следующих основных положениях:

а) ИТ-подразделение в совокупности с ИТ рассматривается и управляется как экономически ответственная и финансово прозрачная организация;

б) в управлении ИТ применяется ЛИН-модель;

в) организационное строение ИТ-подразделения осуществляется в соответствии с ЛИН-моделью;

г) пятнадцать классических функций управления ИТИ объединяются в три группы;

д) функции управления перегруппировываются в новых категориях управления.

Экономическая ситуация заставила предприятия использовать ЛИН-технологии (Lean Thinking) [29] в управлении ИТ.

ЛИН-модель в ИТ базируется на пяти основных принципах [30] (см. рис.3):

1. Точное определение важности ИТ-продукта или услуги с точки зрения бизнеса. Это базовый принцип при трансформации ИТ в БТ, который реально означает, что должен быть постоянный диалог между бизнесом и ИТ на предмет значимости технологий или инновационного бизнеса, осуществимости и стоимости разнообразных решений. Для этого ИТ-подразделение должно обладать знаниями в технической и финансовой областях.
2. Определение потока создания ценности каждым ИТ-продуктом и услугой. Здесь учитывается знание того, как услуги построены, и стоимость услуги на каждой стадии ее жизненного цикла.

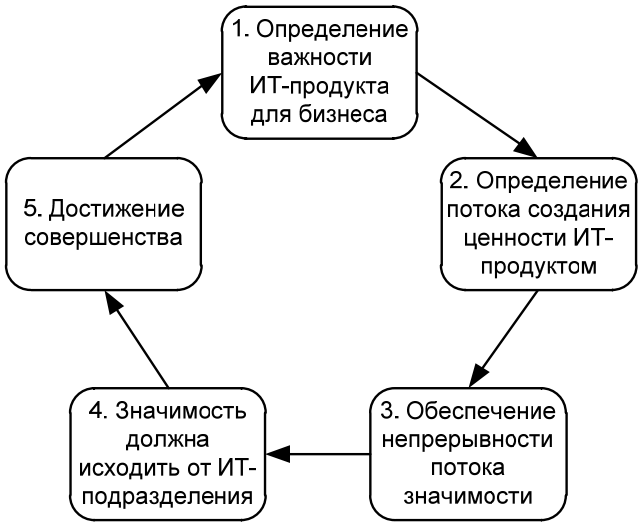


Рис. 3. ЛИН-модель для ИТ

3. Обеспечение непрерывности потока значимости. Процессы следует рассматривать глобально с рациональной точки зрения для исключения потерь и уверенности в том, что важность и стоимость оптимизированы на всех этапах от концепции до предоставления ИТ-услуг.

4. ИТ-подразделение должно предоставлять бизнесу значимость ИТ-продуктов и услуг. БТ занимают на предприятии главенствующее положение, а задача ИТ заключается в автоматизации процессов с целью повышения эффективности БТ. При этом должен соблюдаться баланс между стремлением к максимальному удовлетворению бизнес-целей и стоимостью ИТ и ресурсов, необходимых для достижения целей бизнеса. Этот баланс оценивается соотношением цена/значимость. При управлении ИТ-подразделением как предприятием заказчикам разъясняется полезность от ИТ-продуктов, которую они могут получить в рамках потенциального бюджета.

5. Достижение совершенства. Качество ИТ-сервиса должно быть таким, чтобы количество неполадок при предоставлении ИТ-услуг было минимальным. Этому должно способствовать непрерывное понимание ценности каждого ИТ-процесса и требуемого качества предоставления ИТ-услуг для БТ. В данном случае достойный ответ дает высокотехнологичная методика Six Sigma и другие статистические подходы к оценке качества выполнения бизнес-процессов, применяемые для минимизации вероятности возникновения дефектов в операционной деятельности [31].

Учитывая экспоненциальный рост сложности технологий [27], ИТ-подразделение не сможет выполнять все пять пунктов ЛИН-модели без набора базовых методов, процессов и соответствующего инструментария.

Любая инициатива, направленная на оптимизацию, требует фундаментальных знаний об элементах сервиса, основных характеристиках и структуре затрат. Структура ИТ-сервиса может оптимизироваться на глобальном, стратегическом и тактическом уровнях [27]. При этом тактический уровень оптимизируется локально в ИТ-подразделении, а большая часть принципов управления и процессов имеют тактические и стратегические компоненты, что делает необходимым управление последовательностью операций этих процессов.

С точки зрения ITMS2 организационное строение ИТ-подразделения должно соответствовать подобным структурам предприятий (см. рис. 4).

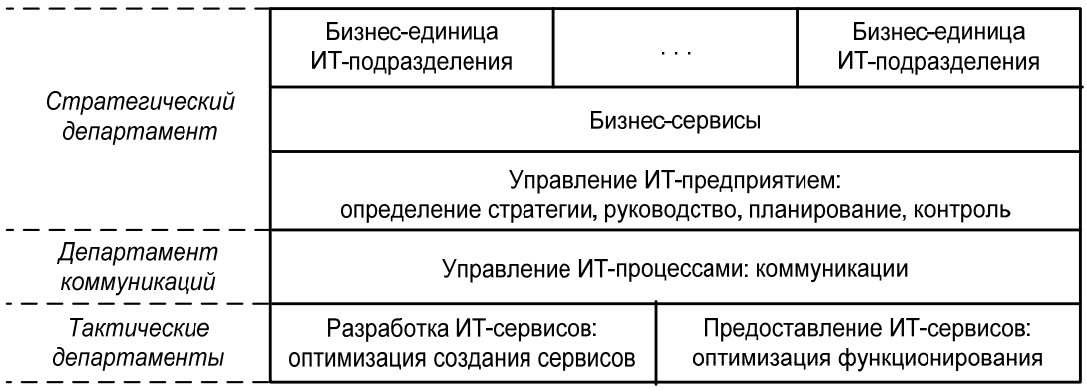


Рис. 4. Схема организационного строения ИТ-подразделения с точки зрения ЛИН-модели

Стратегический департамент занимает центральное место в организационном строении, решая высокоуровневые задачи и выполняя функции: взаимодействия с потребителями ИТ-услуг – бизнес-подразделениями (задачи маркетинга и продажи); контроля расходов и доходов (финансовые задачи); ведения переговоров с поставщиками, вынесения решений по продуктам и определения стратегии производства (задачи планирования и уровня обслуживания). Используемые методики этого департамента очевидны и подобны методикам бизнес-подразделений.

Департамент коммуникаций обеспечивает информационное взаимодействие между другими департаментами. Учитывая то, что рабочие процессы в смежных департаментах носят циклический характер, структурированный обмен информацией между ними имеет важное значение.

Два тактических департамента создают сервисы с учетом требований к ним и выделяемого бюджета, а также обеспечивают их внедрение и сопровождение.

По прогнозам аналитиков компании Forrester [27] к 2013 г. использование процессно-ориентированной модели в ПО управления ИТ, а также повсеместный переход ИТ-подразделений от управления технологиями к предоставлению ИТ-сервисов для бизнеса, должны привести к конвергенции пятнадцати основных категорий продуктов управления ИТ в три основные группы (см. табл. 2) [28]. Существующая экономическая ситуация и движение к ЛИН-ИТ не изменили этой тенденции, а наоборот, добавили необходимость учета соотношения значимость/стоимость ИТ-сервисов, положительно влияющего на процесс конвергенции.

Табл. 2. Категории и группы ПО управления ИТ

Традиционные категории ПО управления ИТ		Группы ПО управления ИТ в соответствии с ITMS2	
Категория	Основные функции	Группа	Основные функции
1	2	3	4
Сетевое управление (Network Management)	Управление сетями и межсетевым взаимодействием	Управление предоставлением и производительностью ИТ-сервисов (IT service delivery and performance management)	Управление доступностью, производительностью и событиями ИТИ. Сквозное управление сервисами. Решение аналитических задач и поддержка принятия решений. Автоматизация процессов и создание информационных панелей. Сопоставление результатов ИТ с результатами ВТ
Управление серверами (Server Management)	Управление производительностью серверов		
СУБД (DBMS Management)	Управление базами данных		
Управление событиями (Event Management)	Мониторинг и анализ событий в ИТИ		
Управление приложениями (Application Management)	Управление функционированием приложений		
SLM и BSM	Управление согласованным уровнем обслуживания. Связывание бизнес-целей и приоритетов с задачами ИТ. Управление сервисами с учетом политики бизнеса		
Управление ИТ для пользователей (End user Management)	Сервис рабочих станций. Повышение эффективности работы приложений пользователей. Управление устранением проблем. Качественный доступ к ресурсам ИТИ		
Управления хранилищами данных (Storage Management)	Обеспечение гарантированного предоставления емкостей в зависимости от потребностей бизнес-процессов. Оптимизация использования имеющихся емкостей		
Служба поддержки (Service desk)	Решение проблем пользователей. Регистрация, анализ и устранение проблем и инцидентов в ИТИ	Управление ИТ сервисами и процессами (IT service management and process management)	Служба поддержки. Управление взаимоотношениями с заказчиками. Управление выполнением потока операций
Управление ИТ-активами (IT asset management)	Управление жизненным циклом вычислительных и коммуникационных ресурсов ИТИ.	Поддержка ИТ-сервисов и управление ресурсами (IT	Среднесрочная и долгосрочная эволюция активов Прогнозирование ресур-
Управление персоналом	Управление трудовыми ресурсами		

(Workforce Management)	ИТ-подразделения	service support and resource management)	сов Инициализация ресурсов и активов Финансовый менеджмент
Управление изменениями и конфигурациями (Change and configuration management)	Организация и оптимизации процессов управления изменениями, релизами и конфигурациями.		
Управление мощностями (Capacity Management)	Экономически эффективное обеспечение ИТ-мощностями, удовлетворяющими текущим и перспективным потребностям бизнеса		
Управление счетами (Billing Management)	Управление финансовыми вопросами ИТ-подразделения		
Планирование работ (Job scheduling)	Планирование и управление исполнением программных задач, которые необходимы как часть ИТ-услуг. Автоматизация запуска задач в определенное время дня, недели, месяца или года.		

Учет значимости ИТ для бизнеса, исходящий из ЛИН-мышления, приводит к следующему отображению категорий ПО управления ИТ в ЛИН-ИТ организационную модель [28, 27], которое позволяет точнее систематизировать инструментарий управления и определить основные направления развития рынка ПО управления ИТ. На рис. 5 показаны основные группы управления ИТ и соответствующие им категории управления. Необходимо отметить, что ITMS2 не только производит конвергенцию существующих категорий, но и рассматривает новые категории, функции многих из которых представляют собой перегруппировку функций из существующих категорий управления.

Рис. 5. Систематизация инструментария управления ИТ по ITMS2 с учетом ЛИН-модели

Широкомасштабное внедрение ITIL и стремление производителей ПО управления ИТ учитывать концепции ITIL привело к некоторой путанице между процессами и инструментариями и, следовательно, к путанице между инструментариями категорий ПО управления ИТ,

составляющими рынок ПО управления ИТ. Этот рынок ориентирован на сервисный подход, а его инструментарий поддерживает сервисные функции. Эти функции только поддерживают процессы управления ИТ, при этом процессами не являются, поскольку они только извлекают из процессов информацию и действуют в соответствии с решениями, вытекающими из процессов. Систематизация на рис. 5 учитывает это различие в трех главных группах ПО управления ИТ, которые должны сформироваться к 2013 г., [28, 27] – управления предоставлением и производительностью ИТ-сервисов, управления ИТ сервисами и процессами, а также поддержки ИТ-сервисов и управления ресурсами.

Управление предоставлением и производительностью ИТ-сервисов. Эта группа призвана объединить аспекты управления ИТИ, связанные с мониторингом и управлением ресурсами, с целью обеспечения заданных показателей качества функционирования ИТИ, и включает в себя весь программный инструментарий, используемый в операционных процессах ИТ-подразделения, связанных с предоставлением ИТ-услуг с необходимым качеством, независимо от того, где располагаются аппаратно-программные средства, предоставляющие эти услуги. Поэтому ПО этой группы должно с одинаковым успехом управлять доставкой и характеристиками локально расположенных сервисов, а также SaaS или IaaS, предоставляемыми как внешними, так и внутренними облаками. Всё ПО категорий управления этой группы в процессе функционирования пользуется едиными данными и моделями сервисов, хранящимися в CMDB, и снабжает информацией продукты SLM/BSM, которые вырабатывают команды, сигналы или изменяют политику управления ИТ-сервисами в зависимости от значимости бизнес-процессов.

Инструментарий этой группы объединен в восемь основных категорий:

– «Мониторинг ИТ-инфраструктуры» представляет собой перегруппировку всего инструментария, осуществляющего сбор данных о функционировании элементов ИТИ. Управление сетями и серверами объединено в одну категорию, поскольку в настоящее время стираются различия между традиционными категориями управления технологиями, а также исчезают формальные различия между управлением устранением неисправностями и управлением производительностью.

– «Информационные панели и аналитика (управление событиями и рабочими характеристиками)» является перегруппировкой инструментария управления событиями, рабочими характеристиками и сигналами тревоги. Современные панели меньше ориентированы на обнаружение отказов в отдельных технологиях, а предназначаются для выявления первопричин отказов в интегрированных бизнес-услугах. Они могут играть роль накопителей, которые снабжают надлежащими показателями другие инструментарии, такие как бизнес-аналитики.

– Категория «Автоматизация процессов и рабочей нагрузки» предназначена для планирования работ и автоматизации запуска вычислительных задач, которые изначально находятся в ЦОД в свернутом состоянии.

– Категория «Управление производительностью, выделением ресурсов и виртуализацией» оперирует с понятиями, которые в настоящее время рассматриваются в тесной взаимосвязи. Кроме того, широкое использование технологии виртуализации позволяет понять, что производительность является общей характеристикой всех компонентов предоставления сервисов (памяти, вычислительных ресурсов, сетей).

– «Управление хранилищами данных» становится неотъемлемой частью СУИ. Особую актуальность эта категория приобретает в связи раскрывшимися преимуществами виртуализации, когда управление хранением данных становится одним из ключевых элементов управления ИТИ [32].

– «APM и BTM». Управление производительностью приложений (APM) и управление бизнес-транзакциями (BTM) относятся к числу важнейших элементов оказания ИТ-услуг. Эта категория должна включать в себя также управление комплексными распределенными приложениями, работающими на мэйнфрэймах или рассредоточенных вычислительных системах, управление обменом сообщениями и мониторинг поведенческих особенностей пользователей. Эта категория формировалась в результате перегруппирования всех компонентов APM в одну категорию, включая контроль поведения пользователей при работе с приложениями, мониторинг J2EE и .NET, мониторинг

серверов, отслеживание транзакций, управление производительностью БД и пр.

– Категория «Исследование и управление конфигурациями» является общей для всего инструментария ИТ-управления. ЛИН-подход не может быть использован без глубокого знания сервисов и их построения. Эти знания нужны не только для управления ИТИ, но и для анализа стоимости и значимости сервисов. Поскольку каждый уровень управления использует свой взгляд на активы (аппаратное обеспечение, ПО и сервисы), то по мнению аналитиков [27], правильным решением будет создание интегрированной системы управления конфигурациями (CMS), а не использование монолитной CMDB. Заполнение CMS требует автоматического исследования приложений, зависимостей и, в конечном итоге, сервисов.

– «Управление безопасностью» приобретает большое значение в современных СУИ. Безопасность рассматривается как обширная дисциплина, охватывающая практически все компоненты ИТИ. Однако, иногда вопросы безопасности могут негативно влиять на доступность и производительность сервисов. Разрешение этих проблем должно обеспечивать управление предоставлением сервисов.

Концепция TMN выделяет пять функциональных областей сетевого управления [15], которые вписываются в функции группы управления предоставлением и производительностью ИТ-сервисов. Этими функциональными областями управления являются управление: производительностью, устранением неисправностей, конфигурацией, расчетами и безопасностью.

Управление ИТ-сервисами и процессами. Эта группа должна стать платформой коммуникаций между различными процессами управления ИТ и вывести на новый уровень средства автоматизации взаимоотношения ИТ-подразделения с заказчиками сервисов. Рабочие процессы этой категории используются не только для поддержки взаимоотношений пользователей с ИТ-подразделением, но и поддержки процессов внутри ИТ-подразделения. Взаимодействие между процессами должно осуществляться с помощью ПО управления потоками работ и управления процессами. Служба поддержки должна в режиме реального времени выявлять проблемы в ИТИ и переадресовать их в центры компетенции, ответственные за решение проблем.

Поддержка ИТ-сервисов и управление ресурсами. Третья группа сосредоточена на управлении ИТ-подразделением. На основе четкого понимания развертываемых сервисов инструментарий этой группы должен способствовать эффективному решению задач планирования сервисов и ИТ-подразделения в соответствии с финансовыми ограничениями, оптимизации активов и управления их жизненным циклом, управления контрактами и взаимоотношениями с поставщиками, оптимизации решений по построению сервисов и их предоставлению. Критерием для оптимизации решений этих продуктов должно являться соотношение цена/значимость [33]. Данная группа призвана также объединить ПО управления планированием работ, активами, мощностями, изменениями и конфигурациями, финансами ИТ. К этой группе относится инструментарий управления сервисами, превосходящий по функционалу продукты первой группы, обеспечивающий оптимальное распределение ресурсов, рабочей нагрузки, затрат и пр. в зависимости от значимости ИТ-сервисов для БТ. Так, например, каталог сервисов обычно содержит иерархическую структуру сервисов, построенную с учетом агрегированных требований пользователей.

Понимание структуры образования стоимости сервисов необходимо для оптимизации портфеля услуг – выбор пути оптимизации услуг производится с учетом потребностей различных бизнес-единиц, решение по разработке или закупке ПО, а также управление всеми ИТ должно быть основано на финансовых соображениях. Этих же соображений ИТ-руководители должны придерживаться при управлении контрактами и взаимоотношениями с поставщиками, определяя стратегию развития, удовлетворяющую потребности бизнес-подразделений. Управление инновациями, независимо от того, исходят ли они от бизнеса или из ИТ-подразделения, должно производиться с оценкой соотношения цена/значимость.

В третьей группе ожидаются наибольшие изменения под воздействием ЛИН-мышления.

Необходимо отметить, что несмотря на перспективность подхода ITMS2 до настоящего времени он не получил должного широкого распространения прежде всего по причине экономического кризиса, несколько затормозившего инвестиции в ИТ. В то же время перегруппирование функций и слияние категорий является

естественным этапом эволюции ПО управления ИТ.

Развитие концепции ITMS2

Повышение конкурентоспособности предприятия за счет владения ИТИ с высокой производительностью и низкими операционными затратами возможно путем внедрения СУИ. В свою очередь сами СУИ создаются с использованием современных ИТ, которые, в данном случае, используются для управления другими ИТ. Перспективная концепция управления ИТ – ITMS2 [27] принята на вооружение многими производителями ПО управления ИТ [34].

По сути ITMS2 предполагает объединение существующих категорий ПО управления ИТ в три группы, с последующей перегруппировкой функций этих категорий внутри новых категорий, образующих группы, с добавлением дополнительных категорий. Так, например, категории сетевого управления и управления серверами объединены в категорию мониторинг ИТИ, которая призвана осуществлять управление не только всем парком сетевого и серверного оборудования, но и других элементов ИТИ. Такое группирование достаточно разнородных элементов вполне оправданно, поскольку для управления этими элементами применяются не только одни и те же подходы к управлению, но и одни и те же механизмы и технологии управления.

Кроме того, использование универсальных протоколов управления, таких как SNMP, который создавался как протокол сетевого управления, позволяют управлять самыми разнородными устройствами, поддерживающими этот протокол. В этом случае удаленное управление любыми аппаратно-программными элементами, подключенными к сети, становится возможным после обеспечения поддержки ими SNMP и создания соответствующих MIB.

Еще одним фактором конвергенции категорий управления, способствующим созданию универсального инструментария, является повсеместное внедрение ИТ для автоматизации управления ИТИ.

Таким образом, одной из характерных черт современного этапа управления ИТ является конвергенция категорий управления с последующим перегруппированием выполняемых функций категорий ПО управления ИТ по признаку однородности. Такое слияние представляется вполне естественным и наталкивает на поиск сходных концептуальных решений, которые могут стать основой создания перспективных подходов к управлению ИТИ.

Учитывая, что каждая категория управления содержит одинаковые процессы или операции, например, мониторинг или анализ, то возникает естественное желание выделить и объединить одинаковые операции родственных процессов из всех категорий управления всех групп с последующей реализацией этих операций в едином для процессов инструментарии, который будет использоваться для всех категорий ITMS2. Такой подход кажется естественным продолжением тенденций развития ПО управления ИТ, рассмотренных в ITMS2.

Эти процессы должны удовлетворять следующим требованиям и свойствам:

– давать ответы на следующие вопросы, касающиеся состояния ИТИ и ее компонентов: «Что сейчас происходит в ИТИ?», «Почему это происходит?», «Что нужно сделать, чтобы сохранить штатное состояние?»; «Что можно сделать, что бы ИТИ функционировала эффективнее?», «Как развивать ИТИ?»;

– возможность работы в реальном или близком к реальному масштабу времени и пр.

При рассмотрении вопросов ИТ-управления выделяют общие циклы управления или вводят в рассмотрение жизненные циклы процессов управления. В процессном подходе часто используется цикл Шухарта-Деминга [35], представляющий собой повторяющийся процесс принятия решения и обозначаемый PDCA (Plan-Do-Check-Act). Цикл управления PDCA является последовательностью действий руководителя по управлению процессом для достижения целей процесса. Цикл включает этапы планирования работ и ресурсов, необходимых для достижения целей, выполнения, проверки и воздействия с целью устранения причин отклонений от запланированного результата. После чего повторяется этап планирования с изменения планов и распределения ресурсов. Цикл PDCA многократно выполняется с различной периодичностью, обычно совпадающей с периодичностью циклов отчетности ИТ-подразделения. При выполнении корректирующих действий длительность цикла может быть сокращена в зависимости от мероприятий

по устранению причин отклонения. Цикл PDCA относится к процессам организационного управления, применим для управления качеством, но совершенно не применим для оперативного управления ИТС.

Методология управления, контроля и аудита информационных систем COBIT [20] делит всю деятельность ИТ-подразделения на четыре сферы, включающие 34 процесса, каждый из которых относится к ИТ-цели. Каждая ИТ-цель привязана к цели бизнеса. Каждый процесс содержит метрики и систему оценки. Стандарт выделяет четыре группы процессов: планирования и организации, приобретения и внедрения, эксплуатации и сопровождения, мониторинга и оценки. COBIT больше подходит для аудита деятельности ИТ-подразделения, чем для организации управления ИТС.

Международный процессно-ориентированный стандарт по управлению ИТ-сервисами ISO 20000 [18, 19] определяет требования к системе управления ИТ, содержащей политики и подходы обеспечения эффективного управления и внедрения ИТ-сервисов. Стандарт определяет требования к 13 процессам, объеденным в пять групп: предоставления сервисов (управление уровнем сервиса, отчетность по предоставлению сервисов, управление непрерывностью и доступностью; бюджетирование и учет затрат для ИТ-сервисов, управление мощностями, управление информационной безопасностью), взаимодействия (управление взаимодействием с бизнесом, управление подрядчиками), разрешения (управление проблемами, управление инцидентами), процессы контроля (управление конфигурациями, управление изменениями) и управления релизами. Процессы ISO 20000 относятся в основном к организационному управлению ИТ-подразделением и только косвенно затрагивают процессы оперативного управления ИКТ.

Методика Six Sigma [21] при реализации проектов использует последовательность этапов DMAIC (define, measure, analyze, improve, control – выявить, измерить, проанализировать, усовершенствовать, проконтролировать). Этапы DMAIC предназначены для определения целей проекта и запросов потребителей, как внутренних, так и внешних; измерения процессов для определения хода выполнения; анализа и определения первопричин дефектов, улучшения процесса сокращением дефектов; контроля дальнейшего протекания процесса. Данная методика совершенствования бизнес-процессов в основном предназначена для минимизации вероятности возникновения дефектов в операционной деятельности при производстве продукции и предоставлении услуг, может быть использована в операционной деятельности ИТ-подразделения, но этапы DMAIC могут лишь частично использоваться для управления ИТИ.

Ни один из выше проанализированных наборов общих процессов различных стандартов не удовлетворяет требованиям к общим процессам управления ИТИ, поэтому предлагается все операции всех категорий управления сгруппировать в пять основных процессов: мониторинг, анализ, управление, оптимизация и планирование (МАУОП). Для каждого из процессов разработать наборы универсальных методов, которые можно применять к разнообразным категориям управления ИТ. Результатом применения такого подхода видится создание универсального инструментария, осуществляющего автоматизацию выполнения функций всех категорий управления по ITMS2 и реализуемого в СУИ (рис. 6). Такой подход, когда выделяется всего пять ключевых процессов, позволит постепенно наращивать функциональность, не включая каждую категорию управления полностью, а используя только необходимые функции. Так можно отследить только ключевые для организации ИТ-ориентиры, способствующие сокращению затрат и оптимизации расходов на содержание ИТИ.

Объединение не функций, как это сделано в ITMS2, а методов, операций, процессов или видов деятельности, таких как планирование, позволит при практической реализации всех категорий ПО управления ИТ использовать наборы универсальных моделей, алгоритмов, методов, модулей и механизмов МАУОП.

Процессы МАУОП можно определить следующим образом:

1. Мониторинг – сбор, измерение, первичная обработка и сравнение информации, отчетность. Отвечает на вопросы: «Что сейчас происходит в ИТ-системе?» и «Что происходило?». Мониторинг для всех категорий целесообразно осуществлять единым инструментарием. Например, использование агента, установленного на сервере, позволяет сразу следить за всеми ИКТ, элементами ИТИ, работой приложений и пр.

2. Анализ – анализ результатов мониторинга, состояния компонентов и причин возникновения неисправностей. Отвечает на вопрос «Почему это происходит?». Набор различных методов анализа, реализованный в универсаль-

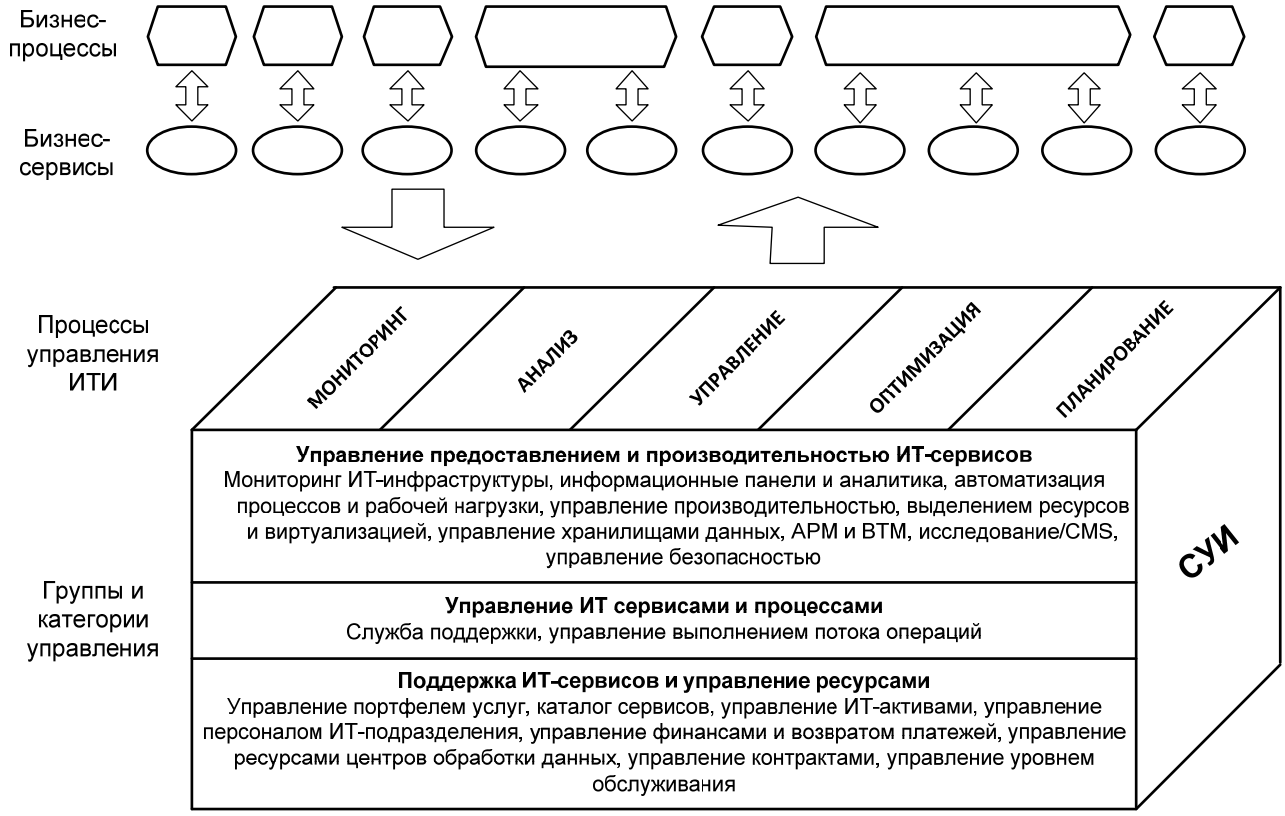


Рис. 6. Место процессов управления ИТИ в СУИ

ном инструментарии, позволит выявлять тенденции поведения разнообразных компонентов ИТИ.

3. Управление – включает все аспекты управления элементами ИТИ, ИКТ, ТКС, предоставлением ИТ-сервисов и пр. Отвечает на вопрос: «Что нужно сделать, чтобы все работало хорошо, работало лучше или хотя бы не хуже?»

4. Оптимизация – оптимизация использования ресурсов ИТИ. Отвечает на вопрос: «Что можно сделать, чтобы стало лучше?»

5. Планирование – прогнозирование поведения ИТ-системы, планирование развития ИТИ. Отвечает на вопросы: «Что произойдет?», «Что мы хотим получить или достичь?» и «Как развивать ИТИ?». Инструментарий планирования также целесообразно объединить для разных процессов. Например, планирование при расширении предприятия вызовет необходимость расширения сети, вычислительных мощностей, привлечение дополнительного ИТ-персонала и пр.

Необходимо учитывать, что при управлении ИТ-подразделением процессы МАУОП часто выполняются вручную, как, например, большая часть процессов управления документацией [17], которая в соответствии с ITIL использует такой показатель, как процент документов, не просматривавшихся в течение года, и другие показатели, измеряемые с периодичностью квартал, полугодие или год. Автоматизация таких процессов часто оказывается нецелесообразной прежде всего по экономическим соображениям. В таких случаях руководство ИТ-подразделения выполняет умственно процессы МАУОП или часть этих процессов, в результате которых выдаются указания на пересмотр политики работы с документацией, приводящей к тому, что, например, создаются дополнительные копии документов, документы подвергаются существенному редактированию или некоторые категории документов убираются далеко в архив или удаляются.

Естественно, что не для всех процессов всех категорий управления ИТИ, особенно нижних уровней иерархии ИТС, всегда необходимо выполнять все пять процессов МАУОП жизненно-

го цикла управления компонентами ИТИ (см. рис. 7). Так, при управлении серверами и сетевым оборудованием цикл управления обычно включает в себя только два процесса – мониторинга и управления (МУ) (рис. 7,в), а управление большинством аппаратно-программных средств функциональных и технологических подсистем удовлетворяется только циклом – мониторинг, анализ и управление (МАУ) (рис. 7,б).

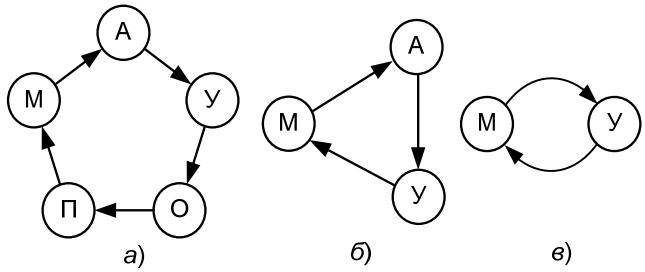


Рис. 7. Жизненные циклы управления а) полный цикл; б) и в) сокращенные циклы

Факторы, влияющие на развитие ИТ

До 2020 г. должны кардинально измениться отношения между бизнесом, технологиями и IT-организациями. Эти изменения обусловлены воздействием на развитие ИТ следующих трех факторов [6]: 1) облачные вычисления, предоставляющие технологии по принципу «как сервис» (“as-a-service”); 2) уполномоченные технически грамотные сотрудники; 3) радикально более сложные ИТ-среды ведения бизнеса. ИТ-организации будут обеспечивать повышение полезности и гибкости ИТ-услуг для увеличения доходов и удовлетворенности пользователей, при этом постоянно снижая затраты на ИТ услуги. Кроме того, изменится роль ИТ-организаций, которые из поставщика ИТ-услуг превратятся в посредника, предлагающего общие и частные услуги. Более подробно три фактора, под действием которых будут развиваться ИТ, представляют собой следующее.

1) Станет нормой парадигма «как услуга», которая будет предоставлять ИТ-решения, готовые для использования бизнесом, кроме того, будут широко внедряться технологии самообслуживания клиентов. Прогнозируемые расходы на SaaS, PaaS, IaaS возрастут с примерно $28 млрд. в 2012 г. до $258 млрд. в 2020 году и составят примерно 45% от общих затрат на ИТ-услуги. Перейдя на «как сервис» технологии можно существенно сократить ИТ бюджет, разгрузить ИТ-специалистов от выполнения рутинных работ, легально использовать большое количество продуктов без платы за лицензии, оплачивая только использованные услуги. Кроме того, существенное сокращение времени выхода на высокопроизводительный режим предоставления ИТ-услуг и быстрое последующее совершенствование ИТ-сервисов склоняет бизнес к переходу от текущего положения дел в ИТ области к широкому использованию облачных технологий. В настоящее время ИТ-подразделения стремятся перестроить корпоративные ИТИ и организовать свою деятельность так, чтобы функционировать по принципу внутреннего сервис-провайдера ИТ-услуг. Сочетание виртуализации, автоматизации и самонастраивающихся технологий рассматривается ИТ-подразделением в качестве основы предоставления ИТИ как-услуги для улучшения адаптируемости, гибкости и снижения затрат. Однако, несмотря на доступность технологий и передового опыта Forrester считает, что только 5% ИТ-организаций имеют достаточно виртуализации, стандартизации, автоматизации и уровня самообслуживания пользователей, чтобы добиться успеха в частных облаках или IaaS [6].

2) К работе с технологиями заказчика будут активно привлекаться технически грамотные специалисты, самостоятельно принимающие решения. Это проще и дешевле, чем содержать штат собственных сотрудников, занимающихся разработкой и обслуживанием ИТ. К 2020 г. количество привлекаемых уполномоченных специалистов составит около 45% штата специалистов ИТ-отдела. Уже в настоящее время 34% таких специалистов считают, что у себя дома они имеют оборудование и технологии лучше, чем на работе, а 48% заявили, что приобретенные ими за собственные средства смартфоны они используют для рабочих целей [6].

3) Постоянно растущая квалификация бизнес-пользователей в сфере ИТ, жесткая конкуренция в бизнесе, повсеместное внедрение ИТ во все сферы человеческой деятельности, позволяет создать и использовать для каждого рода деятельности более сложную бизнес-среду. Расширение рынка ИТ-услуг ожидается в основном за счет появления новых клиентов, которые не смогут позволить себе приобретение продуктов, ориентированных на крупный и средний бизнес. Это потребует инноваций от ИТ-организаций для создания и предложения качественных ИТ-услуг по низким ценам.

Кроме этих трех факторов, которые Forrester считает основными, на развитие ИТ безусловно

окажут влияние и множество других факторов и тенденций, к числу которых следует отнести: широкое использование концепции ITSM, клиент-ориентированный подход, виртуализация ИКТ, стандартизация, автоматизация процессов управления ИТ, самообслуживание и др.

Ключевым при переходе от роли поставщика к роли посредника при предоставлении ИТ-услуг является клиент-ориентированный подход. В настоящее время около 70% ИТ-бюджета выделяется на поддержание текущей деятельности ИТ-подразделения, обслуживание систем и оборудования [6]. Даже при наличии средств, большинство ИТ-организаций сегодня нацелены на обеспечение работоспособности и эффективности ИТИ, а не на поддержку клиентов и повышение их удовлетворенности. Подобно бизнесу, который изначально стремился быть полезным клиентам, ИТ-организации должны следовать этому примеру и строить свою деятельность на принципе максимального удовлетворения интересов пользователей. Бизнес рассматривает и будет рассматривать ИТ как инструмент увеличения доходов и прироста клиентов, а ИТ-организации должны стремиться оправдать ожидания бизнеса.

Повсеместная виртуализация ресурсов ЦОД требует совершенствования ИТ-управления виртуализацией и всей виртуальной средой. Методы и средства управления вируализацией становятся важной частью интегрированных решений СУИ.

Ключевым фактором, оказывающим влияние на развитие ИТ, в ближайшие пять лет будет экономика [1]. Для сокращения расходов и повышения производительности ИТ-организаций в целом, развития ИТИ и операционной деятельности в частности, необходимо мыслить терминами ИТ-индустриализации: рационализации ИТ-процессов и инструментария, которые должны привести к более гибкому, предсказуемому и надежному управлению ИТИ с минимальной стоимостью. На сегодняшний день затраты на управление ИТИ связаны в основном с зарплатой ИТ-персонала. Повысить эффективность работы ИТ-подразделения можно путем автоматизации процессов повседневной деятельности. Кроме того, автоматизация ИТ-процессов открывает новые возможности интеграции разрозненных инструментов ИТ-управления. Поэтому востребованность средств автоматизации продолжает возрастать. При этом ожидается переход от локальной автоматизации отдельных процессов к глобальной автоматизации, которая не только приведет к масштабной экономии, но и позволит перейти от использования единичных СУИ к СУИ массового производства.

В будущем повысить эффективность обслуживания клиентов предполагается путем широкого использования систем самообслуживания (Self-Service), представляющих собой такие решения управления ИТ, которые позволяют клиенту найти ответы на вопросы, касающиеся ИТ-сервисов, или решить проблемы использования ИТ-услуг без обращения в службу поддержки. Как правило, система самообслуживания представляет собой упорядоченную по категориям базу знаний и автоматизированный поиск, доступ к которой клиенты осуществляют через веб-браузер. Как правило, такие системы персонифицированы под каждого клиента. Использование систем самообслуживания экономит массу времени и средств за счет сокращения затрат на поддержку пользователей.

Возрастающее влияние ITSM

По мнению Forrester ИТ-организации для достижения успеха должны совершенствоваться, перемещаясь в сторону процессов управления ИТ-услугами – ITSM, принимая лучшие практики, описанные в ITIL. Forrester определяет ITSM как [6]: «основанные на процессах практики, целью которых является стабильное предоставление ИТ-услуг с учетом потребностей предприятия с акцентом на преимущества для пользователей. ITSM требует смены парадигмы от предоставления и управления ИТ как стеком отдельных технологий на сосредоточении поставки и управления ИТ-услугами с использованием лучших моделей процессов практик, таких как ITIL, признанных во всем мире лучшей практикой ITSM».

В ходе опроса [36] почти пятисот ведущих специалистов, участников американской секции форума IT Service Management Forum (itSMF), работающих в области ИТИ и операционной деятельности по вопросу перспектив развития ITSM после 2011 г., Forrester обнаружил, что почти все крупные предприятия США имеют некоторую форму подхода к ITSM, основанную на ITIL. Причем основанные на ITIL программы и сертификаты направлены не только на повышение производительности и качества ИТ-услуг, но и на улучшение репутации и престижа бизнес-лидеров, контроля расходов на ИТ и пр.

Аналитики Forrester выбрали 18 наиболее важных процессов [6], представляющих собой

прошлое, настоящее и будущее ITSM (см. табл. 3). Эти процессы основаны на ITILv3 и способствуют переходу ИТ-организации от поставщика ИТ-услуг к роли посредника. Часть этих процессов используется много лет, часть являются относительно новыми, а некоторые ранее использовались, но сейчас происходит их возрождение.

Табл. 3.Основные перспективные процессы управления ИТ

Процесс	Описание процесса	Производители ПО	Стоимость
Управление доступностью и производительностью (Availability and performance management)	Отвечает за измерение и улучшение доступности и производительности ИТ-услуг и связанных с ними элементов конфигурации или ИТ-активов (серверы, сети, БД, приложения). Используется для поиска причин проблем с производительностью приложений или бизнес-услуг. Понимание зависимости работы пользователя от конкретных приложений, БД, ПО промежуточного слоя и компонентов ИТС в комплексе позволяет ИТ-подразделению управлять и автоматизировать доступность и производительность, как это необходимо бизнесу.	AppDynamics, Allen Systems Group (ASG), Aternity, BlueStripe Software, BMC Software, CA Technologies, Compuware, Correlsense, dynaTrace Software, Heroix, HP, IBM Tivoli, INETCO Systems, Knoa Software, Nastel Technologies, NetIQ, OPNET, OptTier, Oracle, Precise, Progress Software, Quest Software, ServicePilot Technologies, SL Corporation, Visual Network Systems.	От $100,000 и возрастает с увеличением масштабов ИТИ и степени охвата компонентов.
Управление мощностью (Capacity management)	Оптимизация мощности ИТ-услуг, бизнес-услуг и ИТИ таким образом, чтобы доступные ресурсы были способны предоставить услуги с согласованным уровнем обслуживания экономически-эффективно и своевременно. Процесс включает планирование, оценку и постоянное управление мощностями. Процесс призван избежать ненужных инвестиций в ИКТ, не отвечающих потребностям бизнеса, и ситуаций, когда производительность труда находится под угрозой из-за нехватки или неэффективного использования существующих ИКТ, поддерживающих ИТ-услуги и/или бизнес-услуги.	ASG-Perfman, BMC Software, CA (Hyperformix) Technologies, CiRBA, Orsyp, Systar, TeamQuest.	От $100,000 и возрастает с ростом масштабов ИТИ.
Управление изменениями (Change management)	Отвечает за управление и мониторинг жизненного цикла и рисков всех изменений (добавление, модификация или удаление ИТ-услуг, конфигурационных образов, процессов, документации и т.д.), гарантируя, что используются нормативные методы и процедуры отслеживания изменений. Процесс необходим для обеспечения качества и непрерывности ИТ-услуг, начинается с запроса на изменение, требующего регистрации изменения, приема и классификации, утверждения и планирования, фактического осуществления изменений, оценки изменений и закрытия запроса на изменение.	ASG, Axios Systems, BMC Software, CA Technologies, EasyVista, Cherwell Software, FrontRange Solutions, helpLine, HP, Hornbill, IBM, iET Solutions, LANDesk Software, ManageEngine, Numara Software, Serena Software, ServiceNow, Symantec, VMware.	Внедрение в 2-5 раз больше стоимости ПО. Обслуживание – 10-20% от стоимости ПО. Поддержка 20-30% от цены ПО.
Управление конфигурациями (Configuration management)	Управление конфигурационными элементами, предоставляющими ИТ-услуги или часть ИТ-услуг, на протяжении жизненного цикла элементов. Процесс используется для поддержания бизнес-услуг и их адаптации в соответствии с потребностями бизнеса, и необходим для поддержания работы ИТ-сервиса и снижения затрат на ИТ-услуги и ИТИ.	ASG, Axios Systems, BMC Software, CA Technologies, Cherwell Software, FrontRange Solutions, EasyVista, helpLine, HP, IBM, iET Solutions, LANDesk, ManageEngine, Numara Software, ServiceNow, Symantec, VMware.	Основные затраты на з/п персонала. CMDB/CMS решения часто бесплатные.
Управление постоянством уровня услуг (Continual service-level management)	Отвечает за ведение переговоров по уровню обслуживания и соблюдение требований SLA, отслеживает процессы и услуги для их подстройки к изменяющимся потребностям бизнеса. Отслеживает согласованность целей ключевых ИТ-сервисов между собой и их соответствие целям бизнеса и пользователей.	ASG, Axios Systems, BMC Software, CA Technologies, Cherwell Software, FrontRange Solutions, EasyVista, helpLine, HP, IBM, iET Solutions, LANDesk, ManageEngine, Numara Software, ServiceNow, Symantec, VMware.	Затраты на ПО небольшие. Основные расходы на анализ данных.
Управление взаимоотношениями с заказчиками (Customer relationship management)	Процесс и мероприятия, отвечающие за налаживание отношений с внутренними и внешними заказчиками ИТ-услуг. Способствует оптимизации клиент-ориентированных процессов: управление заказом сервисов, инцидентами, уровнем обслуживания. Новые технологии, такие как социальные сети, позволяют взаимодействовать с заказчиками для повышения качества предоставления и обслуживания сервисов.	Axios Systems, BMC Software, CA Technologies, Cherwell Software, EasyVista, FrontRange Solutions, helpLine, HP, IBM, iET Solutions, LANDesk, ManageEngine, Numara Software, ServiceNow, Symantec, VMware.	ПО недорогое. Основные расходы на опросы и коммуникации.
Управление событиями (Event management)	Управление событиями на протяжении всего их жизненного цикла – от получения и протоколирования сообщений тревоги до автоматизированного разрешения. Позволяет понять, какие проблемы возникают в ИТИ, для последующего разрешения автоматизированным образом или путем эскалации событий экспертам предметной области.	BMC Software, CA Technologies, IBM Tivoli, HP, Microsoft SCOM, NetIQ, Splunk.	Зависит от количества компонентов ИТИ.
Управление инцидентами (Incident management)	Управление незапланированным прерыванием или снижением качества ИТ-услуг от обнаружения инцидента до восстановления функционирования. Процесс включает все проблемы, которые нарушают или могут нарушить текущие сервисы и, как правило, информирует пользователя или инструментарий управления ИТИ.	Axios Systems, ASG, BMC Software, CA Technologies, Cherwell Software, EasyVista, FrontRange Solutions, helpLine, HP, Hornbill, IBM, ICCM Solutions, iET Solutions, LANDesk, ManageEngine, Numara Software, Marvel Solutions, Serena Software, ServiceNow, Symantec, VMware.	От $100,000 и зависит от размера ИТИ. Провайдеры SaaS берут помесячную оплату при трехлетнем контракте.

Процесс	Описание процесса	Производители ПО	Стоимость
Управление жизненным циклом ИТ-активов (IT asset life-cycle management)	Управление жизненным циклом ИТ-активов от закупки до утилизации, с акцентом на расположении, стоимости и праве собственности. Отслеживание активов вручную или с использованием электронных таблиц неэффективно и чревато ошибками. Автоматизация процесса позволяет получить знания о доступных аппаратных и программных активах для принятия решений, управления мощностью и планирования. При использовании облачных технологий необходимо отслеживать инфраструктурные и программные ИТ-активы в режиме реального времени.	Axios Systems, BMC Software, CA Technologies, Cherwell Software, EasyVista, FrontRange Solutions, helpLine, HP, IBM, iET Solutions, LANDesk, ManageEngine, Numara Software, ServiceNow, Symantec, VMware. А также Aspera Software, BDNA, Certero, Dell KACE, Eracent, Express Metrix, Flexera Software, Kaseya, License Dashboard, License Watch, Novell ZENWorks, Scalable, Snow Software.	От $100,000 и зависит от количества активов. В SaaS – помесячная оплата с контрактом на 1-3 года.
Управление финансами (IT financial management)	Использование традиционного бухучета и основанного на значимости инструментария и методов для обеспечения эффективного бизнес-ориентированного управления ресурсами ИТИ при предоставлении клиент-ориентированных ИТ-услуг.	Три ключевых поставщика: Apptio, ComSci и VMware (DigitalFuel). Есть решения у BMC Software, CA Technologies, HP, IBM, и ServiceNow.	От $100,000 и зависит от степени охвата.
Управление знаниями (Knowledge management)	Отвечает за сбор, анализ, хранение и распространение знаний (данные и информация, подготовленные для целевой аудитории) в рамках ИТ-организации. Процесс используется для распространения знаний на всех уровнях техподдержки. Повышение эффективности за счет снижения необходимости заново открыть знания.	Поставщики ITSM добавляют управления знаниями к наборам решений. Только RightAnswers сосредоточены на этой теме и интеграции с другим ПО.	Определяется з/п соответствующих специалистов.
Управление портфелем услуг (Portfolio management of services)	Описание услуг в терминах значимости для бизнеса, сформулированных с точки зрения заказчика, указывая, какие ИТ-услуги имеются, как они объединены в пакет и какие преимущества предоставляют.	В данный момент, управление портфелем услуг производится за пределами инструментария ITSM.	От $100,000 и зависит от количества услуг.
Управление проблемами (Problem management)	Управление жизненным циклом проблем для предотвращения инцидентов и минимизации последствий инцидентов, которые не могут быть предотвращены. Реактивное управление проблемами заключается в ликвидации последствий инцидента. Проактивное управление позволяет разрешить проблемы, прежде чем они окажут негативное влияние на бизнес-процессы. Процесс используется при анализе и поиске причины инцидентов и деятельности, направленной на предотвращение появления инцидентов.	ASG, Axios Systems, BMC Software, CA Technologies, Cherwell Software, EasyVista, FrontRange Solutions, helpLine, HP, Hornbill, IBM, ICCM, iET Solutions, LANDesk, ManageEngine, Numara Software, Marvel, Serena Software, ServiceNow, Symantec, VMware.	ПО недорогое. Основные затраты на з/п специалистов ИТ-подразделения.
Управления релизами и развертыванием (предоставлением услуг) (Release and deployment (service provisioning) management)	Управление релизами процесса планирования, составление графика и контроля за перемещением релизов (наборов аппаратных средств, ПО, документации или других компонентов, как единого целого) для проверки в реальной среде. Управление развертыванием отвечает за продвижение новых или модифицированных видов оборудования, ПО, документации или процессов в реальные условия. Управление релизами используется для управления процессами, системами и функциями, необходимыми для компоновки, сборки, тестирования и развертывания релизов в производство.	ASG, BMC Software, CA Technologies, HP, IBM Tivoli, Microsoft, Nolio, Opscode, Oracle, Puppet Labs, Serena Software, UrbanCode, XebiaLabs.	Определяется з/п специалистов ИТ-подразделения.
Управления каталогом сервисов (Service catalog management)	Создание и поддержание каталога сервисов с указанием статуса, интерфейсов и порядка предоставления услуг. Используются для предоставления деталей и состояния существующих сервисов и бизнес-процессов, поддерживаемых ИТИ, а также сервисов, находящихся в разработке или уже не используемых.	Axios Systems, BMC Software, CA Technologies, Cherwell Software, EasyVista, FrontRange Solutions, helpLine, HP, IBM, iET Solutions, LANDesk, ManageEngine, Numara Software, Serena Software, ServiceNow, Symantec, VMware. А также PMG. newScale.	Инструментарий являются частью существующих ITSM решений + расходы на ведение.
Метрики сервисов и отчетность (Service metrics and reporting)	Идентификация, сбор и представление информации о предоставлении ИТ или бизнес-услуг, определяющей типы и количество услуг, их вклад, риски и улучшение возможностей для бизнеса. Метрики используются для измерения эффективности процессов и сервисов.	Axios Systems, ASG, BMC Software, CA Technologies, Cherwell Software, EasyVista, FrontRange Solutions, helpLine, HP, Hornbill, IBM, ICCM, iET Solutions, LANDesk, ManageEngine, Numara Software, Marvel Software, ServiceNow, Symantec, VMware.	Инструментарий обычно входит в продукты ITSM.
Управление заказом сервисов (Service request management)	Управление жизненным циклом запросов пользователей для получения информации о сервисах или имеющихся аналогах, советов или доступа к ИТ-услугам. Обычно входит в процессы службы поддержки.	ASG, Axios Systems, BMC Software, CA Technologies, Cherwell Software, EasyVista, FrontRange Solutions, helpLine, HP, Hornbill, IBM, ICCM, iET Solutions, LANDesk, ManageEngine, Numara Software, Marvel Software, ServiceNow, Symantec, VMware.	Инструментарий обычно входит в продукты ITSM + расходы на з/п.
Организация времени и управление ресурсами (Time and resource management)	Распределение времени персонала и ресурсов для задач операционной деятельности, проектов и/или функций. Процесс планирования, назначения и контроля времени и ресурсов для эффективного размещения сотрудников с необходимыми навыками на высокоприоритетное сервисное обслуживание и выполнение работ.	Плохая поддержка поставщиками ITSM решений. Могут быть использованы решения по управлению проектами и портфолио.	Определяется з/п специалистов.

Новыми процессами, зрелость которых отражает степень перехода к клиент-ориентированным технологиям [6], являются: управление постоянством уровня услуг, управление взаимоотношениями с заказчиками, управление портфелем услуг. Многие ИТ-специалисты не понимают принципиальной разницы между управлением инцидентами и проблемами. Инциденты являются проявлением разного рода нестандартных ситуаций, а причиной появления проблем могут быть фундаментальные недостатки ИТИ, которые могут приводить к массовой регистрации инцидентов. Поэтому тщательный поиск причин возникновения инцидентов может помочь в ликвидации дефектов проектирования или интеграции компонентов ИТИ.

Для успешного перехода от роли поставщика ИТ-услуг к роли посредника, предлагающего общие и частные ИТ-услуги, ИТ-организации должны оптимизировать внедрение ITSM процессов (табл. 3) и обеспечить скорейшее достижение процессами стадии постоянного совершенствования [6]. Для этого нужно стратегически правильно расставить приоритеты развертывания и развития процессов ITSM с учетом специфики и состояния ИТ-организации, понимания требований к ИТ со стороны бизнеса, осознания необходимости приведения в соответствие ИТ-услуг с бизнес-целями и потребностями пользователей.

Как видно из табл. 3 ASG, Axios Systems, BMC Software, CA Technologies, Cherwell Software, EasyVista, FrontRange Solutions, helpLine, HP, IBM, iET Solutions, LANDesk, ManageEngine, Numara Software, ServiceNow, Symantec и VMware предоставляют продукты для поддержки практически всех процессов. В основном рынок делят межу собой CA Technologies, IBM, HP, BMC Software, Symantec и EMC [34], из которых мегавендорами Forrester называет BMC Software, CA Technologies, IBM Tivoli, HP Software [24]. В то же время отмечается снижение доли рынка этих поставщиков за счет активности новых производителей, предлагающих инновационные или специализированные решения по управлению ИТ с ориентацией прежде всего на малый и средний бизнес.

Эволюция выполнения ИТ-операций от стадии единичных образцов на уровень серийного производства

Для повышения конкурентоспособности ИТ-организации необходимо наряду с сохранением надлежащего качества обслуживания повышать производительность операционной деятельности. В то же время, постоянное снижение стоимости электронных компонентов позволяет при сохранении затрат на закупку оборудования получить более производительные вычислительные и коммуникационные ресурсы, что позволит предоставлять большее количество ИТ-услуг с лучшим качеством за меньшую стоимость [1]. Это в свою очередь вызовет рост спроса на ИТ-услуги, которые могут быть предоставлены в сжатые сроки с низкой стоимостью, что позволит увеличить общее количество и сложность новых бизнес-сервисов, которые должны обслуживать ИТ. Для поддержания экстенсивного роста количества бизнес-сервисов при традиционном развитии организации ИТ в течение последующих пяти лет ИТ-руководство должно:

– обеспечивать повышение производительности при одновременном снижении удельных затрат. Причем увеличение должно измеряться не процентами, а порядками. Этого можно достичь только путем автоматизации развертывания и управления бизнес-услугами;

– сохранять и улучшать качество обслуживания. Качество предоставления бизнес-услуг оказывает непосредственное влияние на результативность и конкурентоспособность бизнеса. Поэтому очень важно соблюдать баланс между качеством сервисов и затратами на обслуживание [1];

– быстро и эффективно удовлетворять спрос на новые услуги. Технология борьбы за клиентов подразумевает постоянное приспособление бизнес-процессов и бизнес-услуг к спросу и быстрота реагирования на изменение спроса является ключевым конкурентным преимуществом [37]. В свою очередь, это требует от ИТ-подразделения быстро внедрять новые ИТ-услуги с сохранением целостности оптимизированной платформы предоставления бизнес-услуг. Это может быть обеспечено путем использования новых технологий, таких как виртуализация и облачные вычисления.

– обеспечивать постоянное развитие ИТ-организации. Отягощенные обслуживанием и поддержкой большого количества бизнес-услуг ИТ-организации часто не в состоянии качественно поддерживать новые услуги и удовлетворить спрос на гибкость и ожидаемое качество обслуживания. В 2009 году 66% ИТ-директоров были обеспокоены неспособностью ИТ-подразделения адекватно поддерживать развитие бизнеса [1]. Для преодоления этой

проблемы ИТ-специалисты должны уменьшить количество времени, которое тратится на ручную настройку и развертывание ИТ-услуг, путем автоматизации сложных процессов, склонных к ошибкам и переделкам.

Традиционный подход борьбы с проблемами в ИТИ путем привлечения большого количества ИТ-специалистов оказывается впредь не эффективным: увеличение персонала подчиняется закону убывающей отдачи, которая может оказаться контрпродуктивной и быстро натолкнуться на финансовые и организационные ограничения [1].

ИТ должны преодолеть все эти проблемы путем повышения производительности на порядок, как это характерно для серийного производства, в течение ближайших пяти лет. Если использовать терминологию модели индустриализации, то существенное увеличение производительности выполнения ИТ-операций возможно только при переходе от единичного производства к серийному или массовому производству [1]. По словам Ч. Перроу: «Производительность в большей степени зависит от масштаба технологических изменений и экономики, чем от человеческих усилий» [38], поэтому ключевым для повышения ИТ-производительности является переход на модель массового производства путем: оптимизации цепочки предоставления сервисов, включая выбор провайдеров, обеспечивающих лучшее соотношение цена/качество, стандартизации ИТИ, предоставляющей услуги, повсеместной автоматизации процессов с реализацией в соответствующем инструментарии для повышения скорости, надежности и предсказуемости предоставления услуг.

Применение стандартизации и масштабной автоматизации ИТ-операций позволяет одному администратору компаний-поставщиков облачных услуг и крупным интернет-провайдерам, таким как Amazon.com, eBay, Google, Yahoo и др. обслуживать 2000 серверов, в то время, как нормой для обычных ИТ-организаций является 20 серверов на одного администратора [1]. Достижению высокого уровня производительности ИТ-организациям препятствует не только слабая автоматизация управления ИТИ, но и многообразие и объем предоставляемых ИТ-услуг.

За многие годы ИТ-организации накопили большие номенклатуры сложных приложений, которые должны быть сохранены. Кроме того, номенклатура приложений и услуг продолжает увеличиваться и все это происходит на фоне необходимости уменьшения стоимости предоставления бизнес-услуг и повышения производительности ИТ-операций. В настоящее время преодолеть это препятствие можно с помощью автоматизации и облачных вычислений [1]:

– автоматизация может снизить остроту проблемы, вызванной большой номенклатурой ИКТ и услуг. Бизнес-услуги часто собираются из существующих компонентов с помощью ПО промежуточного слоя. В результате чего получаются сложные распределенные приложения, которыми трудно управлять. Но в то же время большая номенклатура позволяет выделить множество схожих рутинных операций и задач, которые могут быть автоматизированы с использованием соответствующего инструментария. Разнообразие инструментария порождает новую проблему – обучение персонала для работы с инструментарием и обеспечение совместной работы инструментов и персонала. Эта проблема также может быть решена путем автоматизации, позволяющей сгладить различия и интегрировать инструментарий, упростив администрирование и снизив требования к численности и квалификации персонала;

– облачные вычисления предоставляют большой потенциал для экономии. Опыт применения облачных технологий говорит о том, что относительно дешевые, стандартные аппаратные средства в сочетании с виртуализацией ресурсов позволяют обеспечить большую гибкость в управлении доступом и распределении вычислительных и телекоммуникационных ресурсов, наряду с прогрессом в области СУ консолидированными ресурсами. При этом с точки зрения повышения производительности безразлично какие используются облака – внешние или внутренние.

Влияние технологии облачных вычислений на развитие ИТ

Постоянно возрастающая сложность ИТ влечет за собой увеличение сложности ИТ-управления, требует повышения квалификации персонала и, несмотря на использование виртуализации и автоматизации некоторых процес-

сов ИТ-управления, приводит к увеличению численности персонала ИТ-подразделения. Выходом из такой ситуации является переход к облачным технологиям, которые не только обеспечивают сокращение капитальных затрат, но и требуют изменений в ИТ-технологиях и вызывают необходимость изменений в организационных моделях. Облачные вычисления не только гарантируют экономию за счет масштаба и автоматизации, но обеспечивают рост производительности и могут покрыть будущие потребности не только большого и среднего, но и массового малого бизнеса. В ближайшие пять лет управление ИТ на основе облачных технологий будет развиваться от стадии накопления опыта работы к использованию зрелых решений [1].

Переход на облачные технологии требует изменений в операционной деятельности ИТ-подразделения. Кроме того, необходимо учитывать множество технологических особенностей при использовании облачных вычислений. В случае внутреннего облака это будет отличающаяся от традиционной работа с приложениями, конфигурациями, мониторингом и управлением производительностью. Внешние облака потребуют дополнительного внимания к вопросам безопасности и учета сетевых задержек, которые могут быть существенными.

Для полной реализации преимуществ облачной модели управления ИТ Forrester рекомендует развивать следующие дополнительные дисциплины [1]:

– обнаружение и управление активами. При принятии облачной модели необходимо знать, где и что находится. Поэтому обнаружение и отслеживание активов и приложений в режиме реального времени становится более важным, чем при традиционном подходе к организации ИТ. Конфигурации могут быть легко изменены и приложения могут легко перемещаться, сохранение контроля над ЦОД требует абсолютной четкости. ИТ-подразделения должны внедрять новейшие решения обнаружения зависимостей, имеющихся в различном инструментарии, таких как управление производительностью приложений или мониторинг бизнес-транзакций;

– мобильность приложений. Должна быть возможность легко перемещать приложения, а для обеспечения автоматической конфигурации сами приложения должны обладать способностью к легкой загрузке и настройке. Для этого необходим автоматизированный процесс, который соберет приложения, зависимости между ними и элементы конфигурации в мобильный блок. Это повлияет на жизненный цикл приложений и процесс обновления версий. ИТ-организации должны полностью контролировать развитие приложений и участвовать в автоматизации процесса управления релизами;

– новая архитектура приложений. Традиционная архитектура приложений непригодна для применения в облаках. Современные приложения используют ПО промежуточного слоя для связи иногда очень разных, но функционально зависимых частей кода. Эта зависимость является препятствием к использованию гибкости, обеспечиваемой облачными вычислениями. Необходимы новые архитектуры приложений, ориентированные на применение во внутренних или внешних облаках [39];

– система финансового управления. Проведение тщательного стоимостного анализа должно помочь ИТ-руководителям выбрать, будет ли новая услуга разработана или приобретена, развернута во внутреннем или внешнем облаке;

– инструментарий планирования мощности. Облачные вычисления расширяют возможности, но ресурсы облаков не бесконечны и не бесплатны. Прогнозирование мощности внутренней или внешней по-прежнему требует понимания размеров капитальных и эксплуатационных затрат. Большой выбор и простота закупки, обеспечиваемые облачными технологиями, делают планирование мощности более сложным.

– инструментарий управления рабочими характеристиками и мощностью. Автоматизация мониторинга и распределения ресурсов. Распределение ресурсов основано на использовании свойства гибкости, характерного для виртуализации. Решение об увеличении или уменьшении ресурсов, выделяемых сервису, является функцией службы управления рабочими характеристиками сервиса. Гармоничное распределение ресурсов происходит после процедуры инициализации. Сегодня это ручная задача, требующая от администратора анализа данных о производительности. В будущем управление распределением ресурсов будет производиться на основе обработки сложных взаимосвязанных событий. Управление производительностью приложений – типичное решение, которое сочетает в себе мониторинг производительности с аналитикой, необходимой для

обеспечения гармоничности распределения ресурсов [1];

– решения для управления ИТ. Абстрагирование от разнообразия и маскирование сложности везде, где это возможно. Отличия ИТ должны быть нивелированы с точки зрения управления ИТ, а СУИ должны предоставлять специфический, простой и единый интерфейс взаимодействия со всеми процессами управления ИТ, независимо от ИТИ и ее многообразия. Это необходимо для маскирования разнообразия ИТИ. Независимо от того, развернут ли сервис на физических или виртуальных серверах, с использованием проприетарного или открытого ПО, либо предоставляется посредством SaaS методами облачных вычислений, все детали должны быть известны, не только для операционной деятельности, но и с точки зрения развития и управления ИТ. Поскольку многие решения ИТ-управления по-прежнему специфичны для каждой из ИКТ, ИТ-руководители должны убедиться, что элементы ИТ-управления, предоставляемые разными поставщиками, могут быть легко интегрированы и работать с общим интерфейсом, позволяющим абстрагироваться от каждой из ИКТ, используемых в корпоративной ИТИ.

В настоящее время ИТИ не предназначены для использования облачных сервисов, потому что организованы из набора технологий: мэйнфреймов, серверов, систем хранения, сетей, ПК, телефонии, и пр. (см. рис 1. и рис. 2). Чтобы воспользоваться преимуществами облачных технологий, Forrester дает следующие рекомендации ИТ-директорам [1]:

– роли персонала должны стать более специализированными. В настоящее время сложность ИТИ вынуждает одного администратора поддерживать несколько систем и услуг. Так, администратор сети решает широкий спектр несвязанных задач от разработки и администрирования до устранения неисправностей. Поддержание работоспособности сложных ИТИ не оставляет времени для работы над развитием ИТС;

– перенимать лучшие практики ITIL. Возможно, эту рекомендацию Forrester можно будет принять после того, как будет наработан опыт работы с облачными вычислениями;

– разработать новые метрики, способствующие постоянному повышению эффективности и производительности;

– добавление новых должностей и рабочих групп в ИТ-организации. Появление новых функций при работе с облачными вычислениями требует, например, введения должности архитектора решений, определяющего структуру предоставления сервисов приложениями, функционирующими во внутренних или внешних облаках, осуществляющего рациональное размещение компонентов ИТИ, производящего сборку сервисов путем интеграции внутренних, внешних и облачных услуг, предоставляемых несколькими провайдерами;

– изменение структуры всей ИТ-организации. При переходе к облачным вычислениям повысится специализация ИТ-отделов в области управления проектами, запросами, данными, безопасностью и услугами;

– переход от операционной деятельности к контролю и директивным указаниям. Чем больше услуг будет предоставляться облачными технологиями, тем меньше традиционных функций будут выполнять сотрудники ИТ-подразделения. Так, потребность в разработке, развертывании и поддержке ИТ будет снижаться, а функции проектирования, компоновки, контроля и оценки качества сервисов, предоставляемых облаками, усиливаться вместе с возрастающей ответственностью за правильную интеграцию разрозненных частей, осуществленную путем объединения всех внутренних и внешних услуг;

– акцент на консалтинг. ИТ-специалисты будут оказывать больше консультационных услуг пользователям по вопросам организации и работы бизнес-сервисов, согласования потребностей бизнеса с возможными ограничениями услуг, эффективного и безопасного пользования услугами, баланса затрат и качества услуг, предоставляемых с помощью облачных вычислений.

Одним из основных факторов, сдерживающих стремительное развитие облачных технологий, является недоверие к облакам с точки зрения стабильности предоставления услуг, а также опасения за безопасность корпоративных данных. Кроме того, многие компании четко не понимают, зачем им нужен переход к облачным технологиям.

Выводы

Проанализированы основные тенденции развития управления ИТ. Определены основные факторы, которые окажут влияние на развитие управления ИТ в ближайшие годы. Проанализированы проблемы управления ИТ и ИТ-инфраструктуры. Предложено в корпоративной ИТ-инфраструктуре выделять четыре иерархических уровня: бизнес-приложений, универсальных сервисов, вычислительных ресурсов и сетевого взаимодействия, каждый из которых при необходимости может быть разделен на подуровни. Показана перспективность и целесообразность использования концепции ПО управления ИТ – ITMS2. В развитие ITMS2 предложено выделить пять общих для всех категорий управления процессов: мониторинга, анализа, управления, оптимизации и планирования.

Список литературы

1. Garbani J.-P. IT Infrastructure And Operations: The Next Five Years / J.-P. Garbani, M. Cecere. – Forrester Research, Inc. – 2011. – May 3. – 20 p.
2. Лепетюк А.Л. Наступает новая эпоха ИТ / А.Л. Лепетюк //Информационные технологии для ИТ-менеджмента. – 2011. – №5. – С. 22–27.
3. Теленик С.Ф. Система управління інформаційно-телекомунікаційною системою корпоративної АСУ / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, Р.Л. Соколовський // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», – 2006. – № 45. – С. 112–126.
4. Теленик С.Ф. Система управления информационной инфраструктурой транспортного предприятия / С.Ф. Теленик, А.И. Ролик, П.Ф. Можаровский, А.В. Волошин // Автомобільний транспорт: зб. наук. праць: Вип. 25. – Харків.: ХНАДУ, 2009. – С. 242–245.
5. Ролик А.И. Система управления корпоративной информационно-телекоммуникационной инфраструктурой на основе агентского подхода / А.И. Ролик, А.В. Волошин, Д.А. Галушко, П.Ф. Можаровский, А.А. Покотило // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2010. – № 52. – С. 39–52.
6. Hubbert E. TechRadar™ For I&O Professionals: IT Service Management Processes, Q1 2012 / E. Hubbert, J.P. Garbani, G. O'Donnell, S.Mann, J. Rakowski. – Forrester Research, Inc. – 2012. – Feb. 7. – 44 p.
7. Lambert L. Topic Overview: IT Management Software / L. Lambert, S. Yates, G. Schreck, E. Hubbert, T. M., J-P. Garbani. – Forrester Research, Inc. – 2007. – Nov. 13. – 11 p.
8. Worthen B. IT Governance: ITIL Power / B. Worthen // CIO. – 2005.– Sept. 1. – 7 p.
9. Александров А. Конкретно о CMDB / А. Александров // Открытые системы.– 2007. – №6. – С. 45–51.
10. Introducing the IBM process reference model for IT. IBM Corporation. – 2008. – 21 p.
11. Самофалов К.Г. Электронные цифровые вычислительные машины / К.Г. Самофалов, В.И. Корнейчук, В.П. Тарасенко // Учебник. – Киев: Высшая школа, 1976. – 480 с.
12. Foster I. The Grid: Blueprint for a New Computing Infrastructure / I. Foster, C. Kesselman. – USA, San Francisco, California: Morgan Kaufmann Publishers, Inc. – 1999. – 677 p.
13. Boutaba R. CyberPlanner: A Comprehensive toolkit for network service Providers / R. Boutaba, J. Xiao, I. Aib // NOMS 2008 – 11th IEEE/IFIP Network Operations and Management Symposium. – 2008.– Vol. 11, No. 1. – pp. 379–386.
14. Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model: ISO/IEC 7498-1:1994. Second editions, 1994. – 68 p.
15. Principles for a telecommunications management network: ITU-T Rec. M.3010. – Geneva. – 2000. – 36 p.
16. Катышев С. Об одной концепции управления распределенным ресурсами // Открытые системы. – 1998. – №3.
17. Брукс П. Метрики для управления ИТ-услугами / П. Брукс; Пер. с англ. – М.: Альпина Бизнес Букс, 2008. – 283 с.
18. Information technology. Service management. Part 1: Specification: ISO/IEC 20000-1:2005. – ISO/IEC, 2005. – 16 p.
19. Information technology. Service management. Part 2: Code of practice: ISO/IEC 20000-1:2005. – ISO/IEC, 2005. – 34 p.
20. COBIT 4.1. Российское издание. IT Governance Institute. – М.: Аудит и контроль информационных систем. – 2008. – 230 с.
21. Хэрри М. 6 Sigma. Концепция идеального менеджмента/ М. Хэрри, Р. Шредер. – М.: Эксмо, 2003. – 464 с.
22. Building Bridges: ITIL and eTOM, Release 1.0: TR143. – TM Forum, 2009, July. – 76 p.

23. Hill P. Combine ITIL and COBIT to Meet Business Challenges / P. Hill, K. Turbitt.– BMC Software, Inc. – 2007, March 15. – 16 p.
24. Garbani J.-P. The IT Management Software Megavendors // J.-P. Garbani, P. O'Neill. – Forrester Research, Inc. – 2009. – Aug. 12. – 14 p.
25. Garbani J.-P. Competitive Analysis: Application Performance Management And Business Transaction Monitoring/ J.-P. Garbani. – Forrester Research, Inc. – 2010. – Sep. 9. – 29 p.
26. Garbani J.-P. Market Projections For 2010: IT Management Software / J.-P. Garbani, T. Mende. – Forrester Research, Inc. – 2010. – Feb. 4. – 6 p.
27. Garbani J.-P. It's Time For IT Management Software 2.0 / J.-P. Garbani, T. Mendel, E. Hubbert, E. Radcliffe. – Forrester Research, Inc. – 2009. – Dec. 8. – 13 p.
28. Mendel T. Holistic View: The IT Management Software Market / T. Mendel, C. Townsend. – Forrester Research, Inc. – 2008. – Jun. 19. – 13 p.
29. Womack J. P. Lean Thinking: Banish Waste and Create Wealth in Your Corporation, Revised and Updated / J. P. Womack, D. T. Jones. – Second Edition. – New York: Simon & Schuster, Inc. – 2003. – 397 p.
30. Mendel T. What Vendor Strategists Need To Know About Lean – The Hot Topic For CIOs Right Now / T. Mendel, A. Peters, E. Radcliffe. – Forrester Research, Inc. – 2009.– Oct. 29.
31. Garbani J.-P. The Key To IT Business Alignment Is In Operations/ J.-P. Garbani, S. Yates, R. Batiancila. – Forrester Research, Inc. – 2007. – Apr. 2. – 14 p.
32. Defining And Measuring IT Efficiency: Maximizing Return On Storage Investments // A Forrester Consulting Thought Leadership Paper Commissioned By NetApp. – Forrester Research, Inc. – 2011. – March. – 21 p.
33. Mendel T. Market Overview: IT Financial Management Software / T. Mendel, P. O'Neill. – Forrester Research, Inc. – 2009. – May 7. – 14 p.
34. Leopoldi R. IT Service Management: The Role of Service Request Management/ R. Leopoldi // RL Information Consulting LLC. – 2007. – June, 1. – 9 p.
35. Репин В.В. Процессный подход к управлению. Моделирование бизнес-процессов / В.В. Репин, В.Г. Елиферов. – М.: РИА «Стандарты и качество», 2008. – 408 с.
36. O'Donnell G. The state of it service management in 2011 / G. O'Donnell, D. Washburn, S. Mann, J. Rakowski, E. Hubbert. – Forrester Research, Inc. – 2011. – Nov. 4.
37. Hoppermann J. The Banking Platform Of The Future / J. Hoppermann, P. Murphy, A. Knoll. – Forrester Research, Inc. – 2009. – Dec. 8. – 18 p.
38. Perrow C. Complex organizations: a critical essay / C. Perrow. – New York: McGraw-Hill. – 1986.– 307 p.
39. Staten J. Deliver Cloud Benefits Inside Your Walls / J. Staten, S. Yates, J. Rymer, F. Gillett, L.E. Nelson. – Forrester Research, Inc. – 2009. – Apr. 13. – 14 p.

УДК 004.056

ВОЛОКИТА А.Н.,
КОХАНЕВИЧ И.В.

БЛОЧНОЕ ШИФРОВАНИЕ В CLOUD COMPUTING С ИСПОЛЬЗОВАНИЕМ ДЕРЕВА КЛЮЧЕЙ

Обеспечение безопасности – одна из основных проблем современных облачных вычислений. Именно отсутствие достаточных гарантий безопасности хранения данных становится основным сдерживающим фактором при переходе в «облако». Статья посвящена механизму шифрования данных в cloud computing. Рассматривается распределенная файловая система, в которой файлы разделяются на отдельные блоки с последующим шифрованием на основе бинарного дерева ключей. Также описан способ построения данного дерева. Показаныдальнейшиеперспективыразвитияпредложенногометода.

Providing secure is an important component of cloud computing. Furthermore, the lack of sufficient data security is a main constraint to move to the “cloud”. The article is devoted to data encryption in cloud computing. The author considers a distributed file system in which files are divided into separate blocks. Then he describes a method of encryption of these blocks based on a binary tree of keys. Moreover the author explains a way of constructing of this tree. The paper considers the prospects for further development of the proposed method.

1. Введение

За последние пять лет стремительно набирает обороты новая тенденция в развитии вычислительных технологий – облачные вычисления (англ. Cloud computing). Компания маркетинговых исследований Forrester Research предполагает [1], что суммарная прибыль всех сегментов облачных вычислений (SaaS, PaaS и IaaS) вырастит с 15 миллиардов долларов США в 2010 году до почти 160 миллиардов в 2020, что будет представлять 27% рост в годовом исчислении.

По имеющимся статистическим данным [2], компании в сфере информационных технологий представляютоблачные вычисления будущегов основном как центры обработки данных, которые предоставляют надежную защиту информации, а также возможность динамического роста в виде опций.

Исходя из данных Morgan Stanley Research [3], первое место среди всего списка проблем облачных вычислений занимает проблема обеспечения безопасности. В рамках данного исследования, отсутствие достаточных гарантий безопасности хранения данных было названо самым большим препятствием при переходе в «облако» (24% респондентов), это вдвое больше, чем следующаяпроблема – неочевидность экономической выгоды (12% респондентов).

2. Постановка задачи

При рассмотрении вопроса безопасности данных в облачных вычислениях,рассмотрим более детально следующие пункты: [4, 5, 6]

1. безопасность хранения данных;
2. безопасность сети;
3. локальность данных;
4. целостность данных;
5. сегрегация данных;
6. безопасность доступа к данным;
7. аутентификация и авторизация;
8. уязвимости виртуализации;
9. резервное копирование.

Безопасность данных.

В традиционной модели, при которой приложение развертывается на собственном оборудовании, данные размещаются в пределах границ организации и подчиняются ее политике контроля доступа к информации. Вотличие от традиционной модели, в облачных вычисленияхкорпоративные данные хранятся за пределами организации. Следовательно, поставщик должен принять дополнительные мерыобеспечения безопасности информации и предотвращения несанкционированного доступа из-за уязвимости в приложении или через сотрудника-злоумышленника [7].

Безопасность сети.

Вся информация, которая передается посредством сети, должна быть надежно защищена. Данное условие выполняется с помощью использования механизмов шифрования сетевого трафика, таких как SSLи TLS.

Локальность данных.

В некоторых случаях пользователю необходимо знать, где физически хранятся его данные. Например, в большинстве стран Евросоюза и Южной Америки информация,являющаяся гос-

ударственной тайной, не может покидать пределы государства. В Украине также передача персональных данных иностранным субъектам отношений, связанных с персональными данными, осуществляется лишь при условии обеспечения надлежащей защиты персональных данных, при наличии соответствующего разрешения и в случаях, установленных законом или международным договором Украины, в порядке, установленном законодательством [8].

Целостность данных.

Проблема целостности данных является основной для любой системы. Она решается довольно просто в отдельно стоящей системе с единственной базой данных. Целостность данных в таком случаи поддерживается через транзакции и/или ограничения БД. При этом, транзакции должны следовать принципамACID (atomicity,consistency, isolationanddurability – атомарность, постоянство, изолированность и надежность).Большинство БД поддерживают ACID, тем самым обеспечивая необходимую целостность данных.

Сегрегация данных.

Одной из основных характеристик облачных вычислений является многопользовательский доступ к данным. При этом возникает ситуация, когда на одном и том же устройстве хранится информация различных пользователей. Данная ситуация предоставляет злоумышленнику возможность, например используя SQL-инъекции, получить доступ к чужим данным [9].

Безопасность доступа к данным.

Компании использующие «облако» для своих бизнес процессов, могут использовать собственную политику безопасности, предоставляя сотрудникам разные права доступа к информации. Исходя из этого, модель облачных вычислений должна быть достаточно гибкой, чтобы обеспечить данную возможность.

Аутентификация и авторизация.

Cloud computing должен иметь удобные механизмы аутентификации и авторизации, предоставляя пользователю возможность легко создавать и удалять отдельные учетные записи сотрудников предприятия. При этом возможно использование собственной политики безопасности.

Уязвимости виртуализации.

Виртуализация является одним из главных компонентов облачных вычислений. Данный механизм предоставляет возможность запускать на одном физическом ресурсе набор изолированных друг от друга виртуальных машин. Проблема заключается в том, что современные виртуальные машины не обеспечивают достаточный уровень приватности из-за наличия уязвимостей [10].

Резервное копирование.

Пользователям cloud computing должны предоставляться гарантии быстрого восстановления информации в случаи ее потери. Данное условие выполняется за счет резервного копирования. При этом поставщики услуг облачных вычислений могут хранить около трех копий пользовательских данных.

Описанные выше пункты безусловно влияют на вопрос безопасности данных в cloud computing, при этом, с нашей точки зрения,проблема хранения данных имеет наибольший приоритет.

3. Распределенная файловая система HDFS

При организации облачных вычислений используется достаточно широкий набор современных технологий. В последнее время весьма большую популярность получил свободный Java-frameworkApacheHadoop и одновременно с ним распределенная файловая система HDFS (HadoopDistributedFileSystem). HDFS – это свободный аналог GFS (GoogleFileSystem). Как и GFS HDFS позволяет приложениям легко масштабироваться до уровня тысяч узлов и петабайт данных. На рис. 1 представлена обобщенная структура HDFS [11].

Как видно из рисунка работу всей системы координирует управляющий узел. Данный узел контролирует доступ к данным, а также управляет областью имен файловой системы. Все данные распределяются по узлам данных в виде отдельных блоков, при этом используется механизм репликации.

Использование HDFS в облачных вычислениях позволяет значительно увеличить степень безопасности хранения данных. Основная идея заключается в следующем: каждый файл разбивается на определенное количество блоков, которые хранятся на физически отдельных узлах, затем данные блоки шифруются с использованием отдельных ключей для каждого из них.Рассмотрим данный метод более детально.

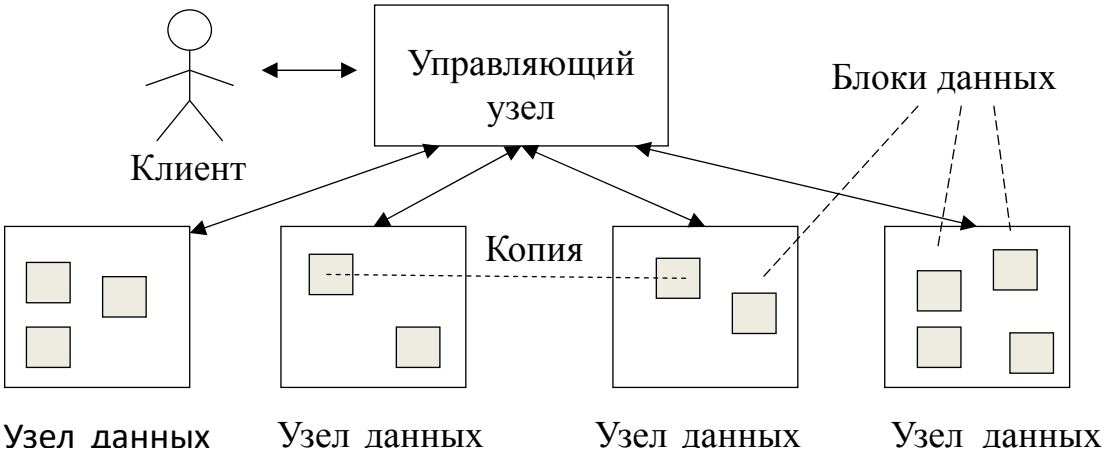


Рис. 1. Обобщенная структура HDFS

4. Модель безопасности хранения данных

Распределенную модель данных представим следующим образом:

$$K_f = f \cdot D_f.$$

f – вектор блоков, из которых состоит файл:

$$f = \{F_1, F_2, \ldots, F_n\}.\ F_i \cap F_j = \varnothing,$$
$$i \neq j; i, j = 1..n.$$

D_f – матрица размером nxL, где L – количество узлов данных. Каждый элемент матрицы может принимать значение 1 или 0, при этом 1 указывает на то, что F_i часть файла f размещена на данном узле, i= 1..n.

K_f – вектор состояния распределенного файла.

Для повышения степени безопасности необходимо использовать механизм шифрования, тогда данная модель будет иметь следующий вид:

$$D'_f = M \ and \ D_f,$$
$$K_f = E(f) \cdot D'_f.$$

M – матрица доступа данного пользователя.

Матрица D'_f – это матрица D_f с учетом прав доступа данного пользователя.

$E(f)$ – вектор зашифрованных блоков файла.

На рис. 3 представлено графическое отображение полученной модели в виде трех уровней защиты.

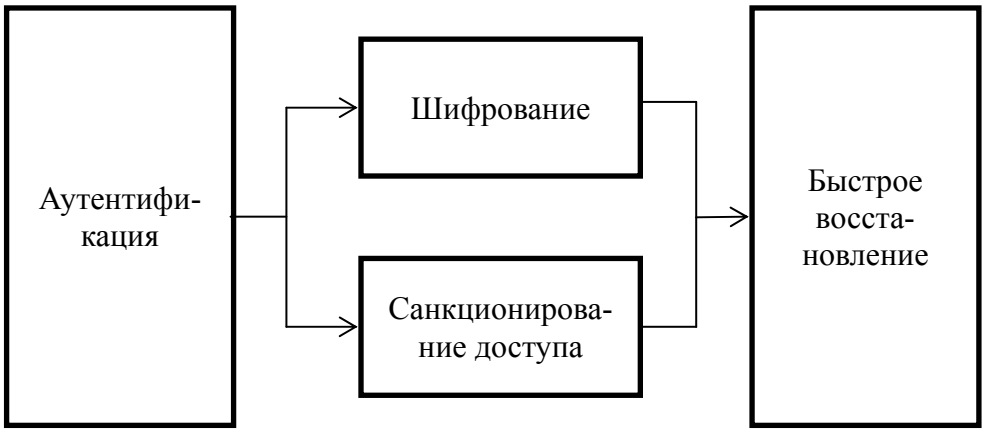


Рис. 2. Модель трехуровневой защиты данных

Первый уровень отвечает за аутентификацию пользователя путем проверки его цифрового сертификата. На втором уровне происходит шифрование данных и определение прав

доступа данного пользователя. На третьем уровне – быстрое восстановление информации в случае ее повреждения или потери.

5. Генерация ключей с использованием бинарного дерева

Как было показано выше, в облачных вычислениях часто используются распределенные файловые системы, в частности HDFS, при этом возможно использовать отдельное симметричное шифрование для каждого блока данных. Однако данный подход имеет существенный недостаток. Поскольку файл может быть разделен на значительное количество блоков, то пользователю также необходимо иметь соответствующее количество ключей. Для решения данной задачи предлагается использовать иерархию ключей.

Для каждого блока $\{F_1, F_2, \ldots, F_n\}$ $(i = 1..n)$ генерируется собственный уникальный ключ, при этом используется следующий принцип: следующий ключ в иерархии может быть получен как комбинация предыдущего ключа с определенной публичной информацией. При этом порождение следующего ключа происходит через хэш-функцию, которая удовлетворяет требование необратимости.

Существует множество способов построения дерева ключей, но для уменьшения количества вычислительных операций целесообразно использовать бинарное дерево. Предположим, что файл разбивается на n блоков $\{D_1, D_2, \ldots, D_n\}$, при этом $2^{p-1} \le n < 2^p$. Следовательно, необходимо построить бинарное дерево с высотой равной p (рис. 3).

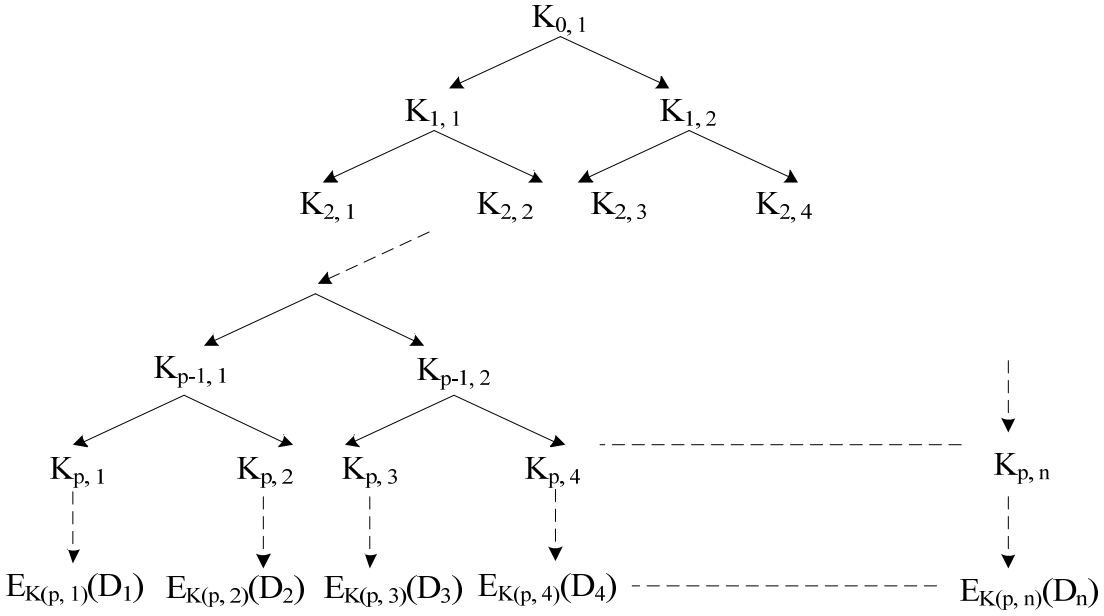


Рис. 3. Бинарное дерево ключей.

Корневой ключ $K_{0,1}$ генерируется пользователем, при этом существует возможность использовать любой алгоритм симметричного шифрования, например AES или IDEA. Первый индекс ключа определяет номер уровня в иерархи, второй – порядковый номер на данном уровне. Также пользователь выбирает порождающую хэш-функцию, которая должна удовлетворять требованию необратимости. Левый и правый производный ключ генерируются следующим образом:

$$K_{i+1,2\cdot j-1} = hash(K_{i,j} \,||\, (2\cdot j\text{-}1) \,||\, K_{i,j})$$

$$K_{i+1,2\cdot j} = hash(K_{i,j} \,||\, (2\cdot j) \,||\, K_{i,j})$$

Аргумент хэш-функции рассчитывается в результате двойной конкатенации ключа-родителя и номера текущего узла.

Процесс сохранения данных происходит следующим образом: пользователь сообщает «облаку» корневой ключ и хэш-функцию, менеджер доступа генерирует дерево, после чего полученные ключи с нижнего яруса иерархии используются в распределенной файловой системе для шифрации данных. Считывание данных происходит аналогичным способом.

6. Анализ вычислительных затрат

В качестве алгоритма хеширования будем использовать MD5, т.е. размер ключа для шифрования блоков будет составлять 128 бит. Сложность дешифрования данного ключа (B_1), при использовании механизма полного перебора, составит $O(2^{127})$ ($B_1 = 2^{127}$). Затраты на вычисление одного ключа –$O(64)$($H_1 = 64$).Для проведения анализа введем понятие эффективности шифрования:

$$E = \frac{B}{H},$$

где B – общая сложность дешифрации файла, а H – суммарная сложность вычисления ключей. В случае, когда весь файл кодируется одним ключом $B = B_1, H = 0$ (поскольку корневой ключ не вычисляется, а предоставляется пользователем).Значение эффективности при этом будет стремиться к бесконечности, по этому, данный случай является вырожденным и далее учитываться не будет.

Предположим, что размер пользовательского файла составляет 1 ГБ. Исходя из того, что размер блоков HDFS64 МБ, получаем 16 блоков ($n = 16$). Учитывая, что $2^{p-1} \le n < 2^p$ высота дерева будет равняться 5 ($p = 5$). Общая сложность дешифрации файла, в этом случаи, составит $B = 16 \cdot B_1$.Суммарная сложность вычисления ключей $H = 30 \cdot H_1$.Как результат получим следующее значение эффективности $E \approx 0.53 \cdot R$.Где $R = \frac{B_1}{H_1}$(данный коэффициент зависит от выбранного алгоритма хеширования).

Общие формулы имеют следующий вид:

$$B = n \cdot B_1;$$

$$H = (2^{p-1} - 2 + n) \cdot H_1.$$

Проведем расчеты, варьируя значениемколичества блоков. Результаты вычислений представлены в табл. 1.

Табл. 1. Значения параметра эффективности

N	***P***	***B/B1***	***H/H1***	***E/R***
2	2	2	2	1.00
3	3	3	5	0.60
4	3	4	6	0.67
5	4	5	11	0.45
6	4	6	12	0.50
7	4	7	13	0.54
8	4	8	14	0.57
9	5	9	23	0.39
10	5	10	24	0.42
11	5	11	25	0.44
12	5	12	26	0.46
13	5	13	27	0.48
14	5	14	28	0.50
15	5	15	29	0.52
16	5	16	30	0.53
17	6	17	47	0.36
18	6	18	48	0.38
19	6	19	49	0.39
20	6	20	50	0.40
21	6	21	51	0.41
22	6	22	52	0.42
23	6	23	53	0.43
24	6	24	54	0.44
25	6	25	55	0.45
26	6	26	56	0.46
27	6	27	57	0.47
28	6	28	58	0.48
28	6	28	59	0.49
30	6	30	60	0.50
31	6	31	61	0.51
32	6	32	62	0.52

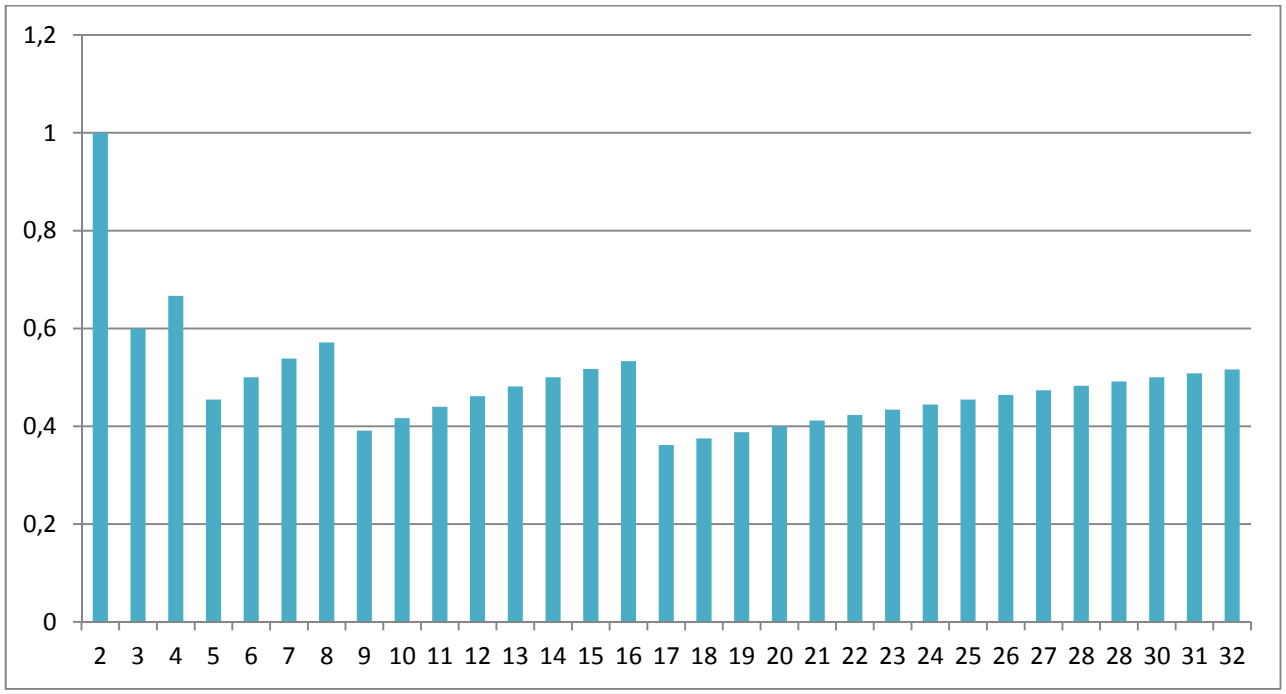


Рис. 4.Гистограмма эффективности шифрования.

Графическое представление результатов изображено на рис. 4. Ось абсцисс – количество блоков. Ось ординат – значение эффективности разделенное на *R*.

Общая сложность дешифрации файла при увеличении количества блоков увеличивается линейно. При этом значение эффективности шифрования меняется в пределах от 0.3*R* до 0.6*R* (при $n > 4$).

Исходя из данных результатов, можно сделать вывод, что при $n > 4$ соотношение общей сложности дешифрации файла к суммарной сложности вычисления ключей варьируется в пределах определенных границ (от 0.3*R* до 0.6*R)*. То есть значение количества блоков должно выбираться лишь с учетом ограничений на задержку процесса считывания или записи, размер блоков, а также размер корпоративных данных, которые требуют дополнительной защиты. В данной ситуации наибольший интерес представляют локальные максимумы, которые стремятся к 0.5*R*. Локальные максимумы достигаются при $n = 2^k$ ($k \in Z$).

7. Заключение

На примере распределенной файловой системы HDFSбыл предложен метод шифрования отдельных блоков файла с использованием бинарного дерева ключей,атакже был описан способ построения данного дерева.

Было показано, что данный метод позволяет увеличить сложность дешифрования файла пропорционально количеству блоков. При этом соотношение общей сложности дешифрования файла к суммарной сложности вычисления ключей меняется в пределах определенных границ. Локальные максимумы достигаются при $n = 2^k$ ($k \in Z$).

В некоторых случаях, например в распределенных файловых системах, размер блоков фиксирован и данный размер достаточно большой (в HDFS64 МБ), поэтому предложенный метод подходит лишь для случаев, когда жизненно важные корпоративные данные также достигают значительных размеров (несколько гигабайт).

В будущем, для уменьшения количества вычислительных операций возможно использование механизма кеширования, что позволит не вычислять целую иерархию ключей при каждой операции считывания или записи. Также возможно рассмотрение других вариантов иерархий ключей.

Список литературы

1. James Staten, Simon Yates, Frank Gillett, WalidSaleh, and Rachel A. Dines. Is Cloud Computing Ready For The Enterprise? //Forrester. – Cambridge, USA.– March 7, 2008.
2. Harry Katzan.Cloud Computing, I-Service, And IT Service Provisioning. //Journal of Service Science. – Savannah State University, USA – 2008 – Vol.5. – P.57-64.
3. Adam Holt, Keith Weiss, CFA1, Katy Huberty, CFA1, Ehud Gelblum. Cloud Computing Takes Off. Market Set to Boom as Migration Accelerates. //Morgan Stanley Research. – May 23, 2011.
4. Weichao Wang, Rodney Owens, Zhiwei Li, Bharat Bhargava. Secure and Efficient Access to Outsourced Data //CCSW'09 – Chicago, USA.– November 13 2009.
5. Jay Heiser, Mark Nicolett. Assessing the Security Risks of Cloud Computing. //Gartner Researche. – 3 June, 2008.
6. Yanpei Chen, Vern Paxson, Randy H. Katz. What's New About Cloud Computing Security? //Electrical Engineering and Computer Sciences, University of California at Berkeley. – January 20, 2010.
7. Ann Cavoukian. Privacy in the clouds. //Identity Journal. – 2008.
8. Закон України «Про захистперсональнихданих», редакція від 01.06.2010.
9. S. Subashini, V.Kavitha. A survey on security issues in service delivery models of cloud computing. // Anna University. – Tirunelveli, India. – 2010.
10. Lisa J. Sotto, Bridget C. Treacy, and Melinda L. McLellan. Privacy and Data Security Risks in Cloud Computing. //Electronic Commerce & Law Report. – February 3, 2010.
11. Dai Yuefa, Wu Bo, GuYaqiang, Zhang Quan, Tang Chaojing. Data Security Model for Cloud Computing. //International Workshop on Information Security and Application. – Qingdao, China.– November 21-22, 2009.

УДК 004.056

ВОЛОКИТА А.Н.,
ВУ ДЫК ТХИНЬ

ИЕРАРХИЧЕСКИЕ АГЕНТЫ БЕЗОПАСНОСТИ В РАСПРЕДЕЛЕННЫХ КОМПЬЮТЕРНЫХ СИСТЕМАХ

В статье исследованы подходы управления безопасностью распределенных компьютерных систем на основе облачных вычислений, а также разработан метод повышения эффективности реагирования на основе иерархических агентов с горизонтальными связями.

In this paper are described the approaches to the implementation of effective security mechanims for the distributed computing systems based on cloud computing, suggested a method for the reaction efficiency increasing based on the hierarchical model of agents.

Введение

В связи со стремительным развитием сетевых технологий в современном мире происходит постоянная интеграция организаций в сеть, что обуславливает множество требований по защите информации от злоумышленников. Для каждого клиента, достижение максимальной степени качества облуживания возможно благодаря индивидуальному подходу по организации и управлению безопасностью его компьютерной системы. Современные средства управления безопасностью основываются на существующих стандартах и спецификациях, что приводит к вопросу, может ли продуктивная защита быть достигнута с помощью повышения эффективности управления безопасностью с использованием современных методов защиты. Одним из важных компонентов системы безопасности являются агенты системы.

Агенты безопасности РКС

Агенты являются одними из основных компонент системы безопасности распределенных компьютерных систем (РКС), которые организованы в иерархию и предназначены для управления безопасностью РКС, в том числе в организации реагирования на вторжения. Каждый агент рассматривается как представитель сети ресурсов на метауровне управления безопасностью ресурсов. Это означает, что агент может рассматриваться как поставщик услуг для выделения средств защиты. Также агенты обмениваются сообщениями внутри своей иерархии, про изменение или выделение ресурсов пользователю, что позволяет распределять ресурсы, внутри сети, равномерно [1].

Каждый агент использует базу данных информации и полномочий (БДИП) для управления безопасностью своего сегмента РКС, а также совместного доступа к служебной информации других агентов иерархии. БДИП это таблицы содержащие информацию об агенте и набор соответствующей сервисной информации о безопасности и возможностях ресурсов в соответствующем сегменте РКС. Агент может поддерживать различные БДИП, соответствующие различным сегментам иерархии РКС:

- сБДИП – используется для записи информации о ресурсах сегмента;
- нБДИП – используется для записи служебной информации, которая полученная от агентов нижнего уровня иерархии;
- вБДИП – для записи служебной информации, которая полученная от агентов верхнего уровня иерархии.

Есть два основных варианта взаимодействия агентов с БДИП – принимать и выдавать данные, каждый из которых периодически происходит самостоятельно или может управляться системой:

- Принятие Данных – агент запрашивает у других агентов служебную информацию, периодически, либо при получении запроса.
- Выдача Данных – агент представляет свою служебную информацию для других агентов в системе, периодически или когда служебная информация изменилась.

Агент использует БДИП в качестве базы знаний, для обнаружения сервисов, которые вызываются приходом запроса. Если агент воспользовавшись БДИП не получает необходимую сервисную информацию, он может подать запрос на его верхний агент или прекратить

процесс. Средство оценки интегрировано в каждый агент и используется для оказания поддержки в процессе передачи данных. Система агента направлена на преодоление разрыва между пользователями облака и ресурсами, позволяя, таким образом, эффективнее планировать заявки на выделения ресурсов с учетом безопасности размещения. Агент может выбрать различные стратегии объявления услуг и предоставления сведений, выбор которых может привести к разным результатам работы.

Структура иерархических агентов безопасности

Структура агента безопасности показана на рис. 1. Каждый слой имеет несколько модулей, которые взаимодействуют друг с другом для выполнения служебных сообщений, или для открытия и запуска передаваемых данных [2,3].

В начальный момент функционирования агент состоит из некоторого начального набора модулей и обладает базовыми знаниями о требуемом уровне защиты, желаемой архитектуре, числе своих ближайших соседей, с которыми он должен будет обмениваться всей необходимой информацией. Сразу же после установки на некотором узле облака агент начинает осуществлять сбор сведений о среде функционирования. На основе собранной модулями анализа среды данных модулем обучения формируются убеждения об условиях функционирования, ближайшем окружении агента, возможных атаках и имеющихся средствах защиты. Это позволяет агенту адаптироваться и подстраивать свою структуру (подключенные модули) под постоянно меняющуюся среду функционирования. Постоянный сбор статистической информации позволяет сформировать убеждения о нормальном режиме работы системы, на которую установлен агент, и на их основе своевременно обнаруживать аномалии в поведении пользователя или программного обеспечения.

Центральным звеном предлагаемой архитектуры является главный модуль системы защиты, который обеспечивает синхронизацию и взаимодействие всех остальных модулей. Он же отвечает за организацию хранения и управления базами знаний, убеждений и оперативных данных.

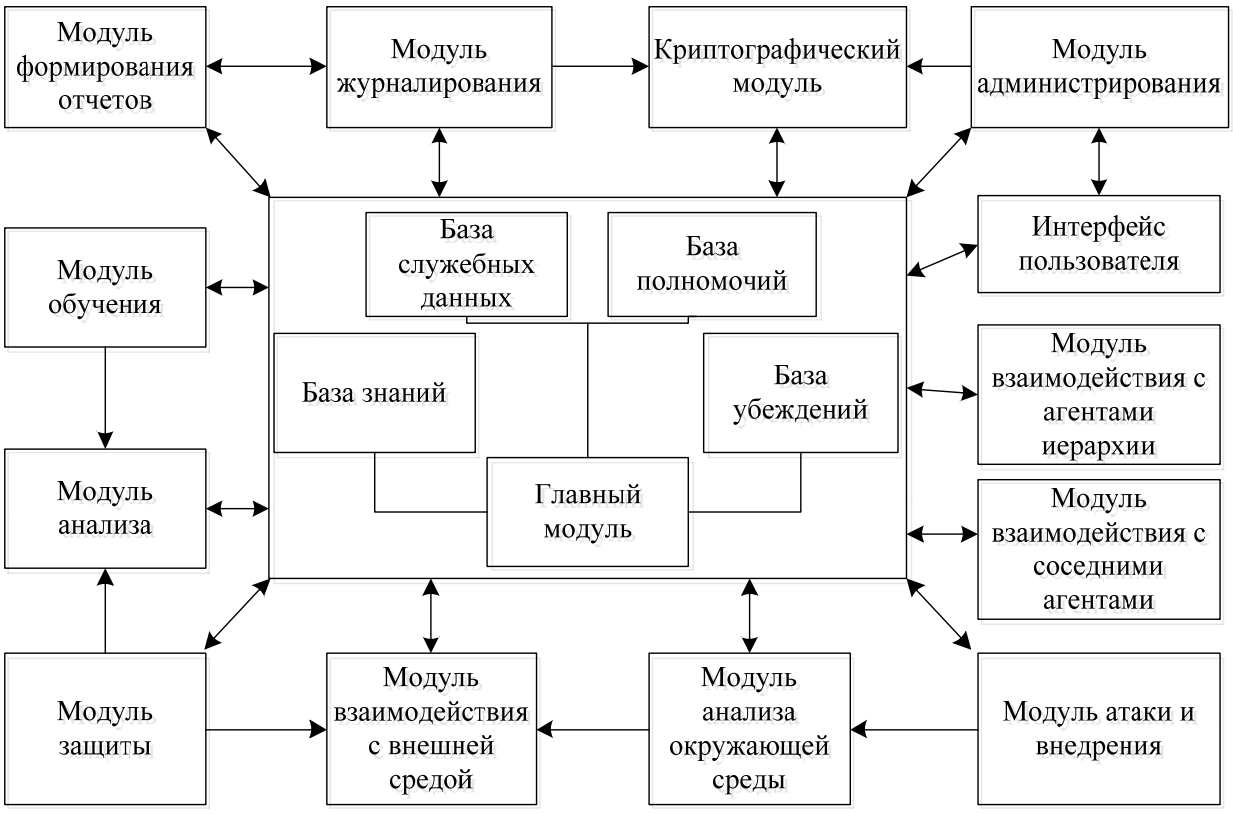


Рис. 1. Структура иерархического агента безопасности РКС

Модуль анализа – осуществляют сбор данных, накопление статистической информации о среде функционирования агента и обнаружение атак.

Модуль обучения – осуществляет формирование убеждений о нормальном функционировании агента и защищаемого элемента сети, ближайшем сетевом окружении агента, состоянии и поведении соседних агентов.

Модуль защиты – отвечают за пассивное и активное противодействие обнаруженным модулями анализа атакам и вредоносным программам, проникшим на защищаемый объект облака, как самостоятельно, так и во взаимодействии с соседними агентами.

Модуль взаимодействия с внешней средой – обеспечивают возможность обмена данными с окружающей средой, мониторинг, отправку и получение необходимой информации.

Модуль анализа окружающей среды – предназначены для сбора информации о текущей обстановке в РКС, сообщениях соседних агентов с целью выявления распределенных атак и накопления статистической информации о нормальном состоянии сети и отдельных ее компонентов.

Модуль атаки и внедрения – предназначены для поиска известных уязвимостей элементов защищаемой сети с целью их своевременного обнаружения и установки необходимых модулей защиты.

Модули взаимодействия с соседними агентами и агентами иерархии – отвечают за организацию взаимодействия отдельных агентов с соседними агентами в рамках единой системы защиты РКС или с агентами верхнего или нижнего уровня соответственно.

Модуль формирования отчетов – обеспечивает формирование отчетов системы безопасности на основе хранящейся в базе данных агента информации. При создании отчетов могут направляться дополнительные запросы соседним агентам для получения более подробной информации и формирования общего для всей защищаемой сети отчета.

Модуль журналирования – отвечает за своевременную запись в базу данных сведений о состоянии агента, подключении новых модулей, об обнаруженных аномалиях и атаках.

Криптографический модуль – предоставляет функции шифрования, проверки сертификатов и цифровой подписи.

Модуль администрирования – предоставляет интерфейсы управления как непосредственно данным агентом, так и остальными агентами сети, за счет отправки общих команд и введения новых модулей и компонентов защиты, пополнения или обновления базы исходных знаний агентов.

Управление агентами и модифицированная модель дискреционного доступа Хартсона

Для управления агентами безопасности РКС используется менеджер БДИП, который контролирует доступ агента к базе данных БДИП со служебной информацией [4]. На рис. 2 показано содержание этой служебной информации.

Рассмотрим ресурс с n каналами связи, где каждый канал P_i имеет свой тип ty_i. С учетом защищенных каналов PS_i каналы связи ресурса могут быть выражены следующим образом:

$$P = \{P_i | i = 1,2, \dots, n\}$$
$$PS = \{PS_i | i = 1,2, \dots, n\}$$
$$ty = \{ty_i | i = 1,2, \dots, n\}.$$

Пусть m количество потоков, которые запущены, или в очереди на передачу. Каждый поток данных A_j имеет два атрибута – время начала передачи ts_j и время завершения te_j. Применение облачного ресурса может быть выражено следующим образом:

$$A = \{A_j | j = 1,2, \dots, m\}$$
$$ts = \{ts_j | j = 1,2, \dots, m\}$$
$$te = \{te_j | j = 1,2, \dots, m\}.$$

MA_j множество каналов связи, MSA_j множество защищенных каналов связи, которые выделено на поток данных A_j:

$$MA = \{MA_j | j = 1,2, \dots, m\}$$
$$MA_j = \{P_j | l = 1,2, \dots, k_j\},$$
$$MSA_j = \{PS_j | l = 1,2, \dots, k_j\},$$

Где k_j число каналов связи, которые выделяются для потоков данных A_j. M и MS – это два 2-д массива, которые описывает взаимосвязи между каналами и потоками с помощью логических значений.

$$M = \{M_{ij} | i = 1,2, \dots, n;\ j = 1,2, \dots, m\}$$
$$M_{ij} = \begin{cases} 1\ if\ P_i \in MA_j \\ 0\ if\ P_i \notin MA_j \end{cases}$$
$$MS = \{MS_{ij} | i = 1,2, \dots, n;\ j = 1,2, \dots, m\}$$
$$MS_{ij} = \begin{cases} 1\ if\ PS_i \in MSA_j \\ 0\ if\ PS_i \notin MSA_j \end{cases}$$

В качестве модели дискреционного доступа будем использовать модифицированное 5-мерное пространство Хартсона [5]. Для этого расширим область безопасности до совокупности шести наборов, включив в модель множество агентов $\{A\}$:

- множество пользователей U;
- множество ресурсов R;
- множество состояний S;
- множество полномочий A;
- множество операций E;
- множество иерархических агентов I;

Тогда область безопасности представляется декартовым произведением:

$$A \times U \times E \times R \times S \times I$$

Пользователи подают запрос на доступ к ресурсам, осуществление которых переводит систему в новое состояние. Запросы на доступ представляются пятимерными кортежами:

$q = (u, e, R', s, i)$,

где $u \in U,\ e \in E,\ s \in S,\ R' \subseteq R,\ I' \in I$.

Запрос удовлетворяется, если оно полностью заключен в область безопасности.

Процесс организации доступа алгоритмически описывается следующим образом:

1. Определить из U те группы пользователей, к которым принадлежит u. Затем выбрать из A те спецификации, которым соответствуют выделенные группы. Этот набор полномочий $F(u)$ определяет привилегию пользователя u.
2. Определить из множества A набор полномочий $P = F(e)$, которые устанавливают e как основную операцию. Полномочия $P = F(e)$ определяет привилегию операции e.
3. Определить из множества A набор полномочий $P = F(R')$, разрешающих доступ к набору ресурсов R'. Полномочия $P = F(R')$ определяет привилегию ресурсов R'.
4. Определить из множества A набор полномочий $P=F(I')$, определяющие полномочия агентов i, отвечающих за безопасность запрашиваемых ресурсов R'. Полномочия $P=F(I')$ определяет привилегию агентов I'.

Полномочия, которые являются общими для всех четырех привилегий, образуют так называемый домен полномочий запроса $D(q)$:

$$D(q) = F(u) \cap F(e) \cap F(R') \cap F(I') .$$

5. Убедиться, что запрашиваемый набор ресурсов R' полностью содержится в домене запроса $D(q)$, т. е. любой r из набора R' хотя бы один раз присутствует среди элементов $D(q)$.

Убедиться, что набор агентов I' полностью содержится в домене запроса $D(q)$, т. е. любой i из набора I' хотя бы один раз присутствует среди элементов $D(q)$.

6. Осуществить разбиение $D(q)$ на эквивалентные классы так, что бы в один класс попадали полномочия, когда они специфицируют один и тот же ресурс r из набора R', а соответственно и один и тот же агент i из набора I'.

В каждом классе произвести операцию логического ИЛИ элементов $D(q)$ с учетом типа операции e. В результате формируется новый набор полномочий на каждую единицу ресурса (и соответственно агента), указанного в $D(q)$ – $F(u, q)$. Набор $F(u, q)$ – фактическая привилегия пользователя u по отношению к запросу q.

7. Вычислить условие фактического доступа, соответствующее запросу q, через операции логического ИЛИ по элементам полномочий $F(u, q)$, запрашиваемым ресурсам r из набора R' и соответствующих агентов i из набора I'. Тем самым получаем набор R'' – набор фактически доступных по запросу ресурсов и соответственно I'' – набор отвечающих за безопасность агентов.
8. Оценить условие фактического доступа и принять решение о доступе:

– разрешить доступ, если R'' и R', и I'' и I' полностью перекрываются;

– отказать в доступе в противном случае.

Агенты иерархии могут быть представлены тремя следующими слоями: коммуникаций, координаций, управления.

Коммуникационный слой каждого агента выполняет функции связи и выступает в качестве интерфейса к внешней среде. Из коммуникационных модулей, агент может получить служебные и открытые сообщения. Он интерпретирует содержимое каждого сообщения и отправляет информацию в соответствующие модули в координационный слой агента. Например, служебное сообщение от другого агента будет направлено в БДИП менеджер в слое координации агентов. Коммуникационный модуль также отвечает за отправку служебных и открытых сообщений другим агентам.

В координационном слое агента есть четыре компонента: менеджер БДИП, средство оценки безопасности, планировщик и сопоставитель. Они работают вместе, для принятия решений о том, как агент должен действовать при получении сообщения с коммуникационного слоя. Например, окончательный ответ на открытое сообщение службы, будет включать передачу данных по локальному ресурсу или отправку запроса на другой агент.

Основные функции слоя управления ресурсами в агенте это: управление потоками данных, распределение и мониторинг ресурсов. Команды передачи данных посылаются от координационного слоя к локальному менеджеру агента, эти команды включают планирование информации для данных и каналов связи. Распределение ресурсов включает в себя оболочки для различных данных.

Экспериментальные исследования

Для проведения экспериментальных исследований была использована распределенная компьютерная система с развернутым средством облачных вычислений в виде файлового хранилища (рис.2). Система состояла из семи компьютеров, на каждом из которых, было установлено программное обеспечение, реализующее облачное хранилище данных, а также один компьютер выделен под роль главного сервера. Каждый файловый сервер оснащен специальными агентами безопасности, задачей которых является определение вторжений на ранних стадиях и быстрое реагирование на них (отправка служебных сообщений на компьютер администратора).

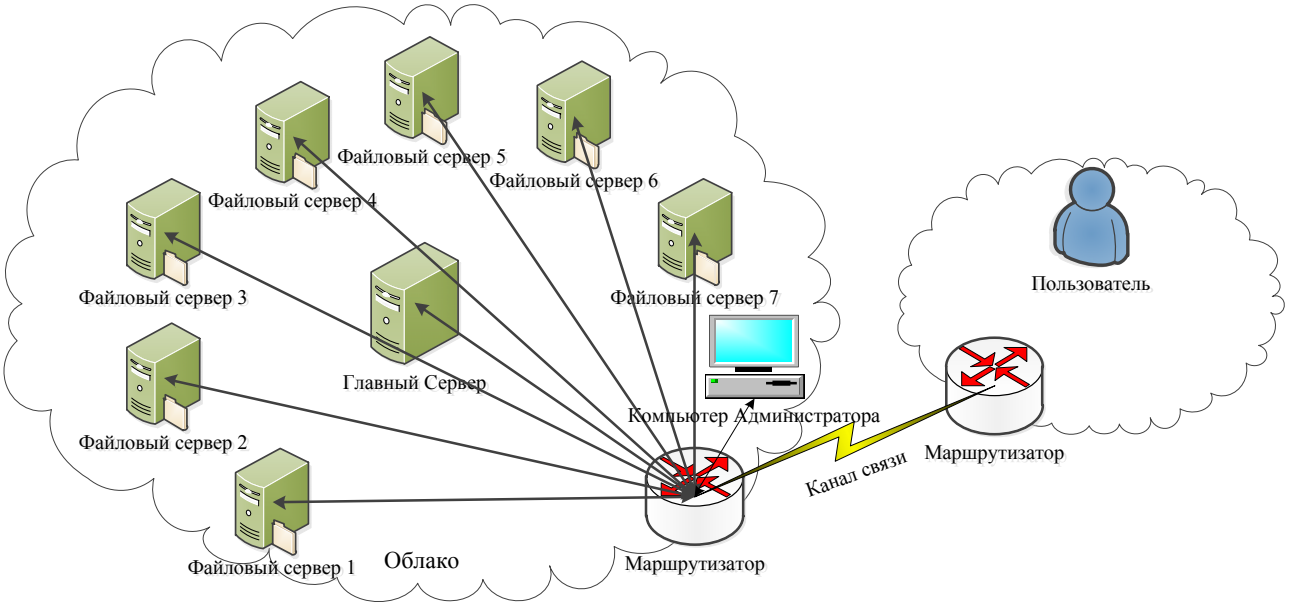


Рис. 2. Схема распределенной компьютерной системы

Следующим этапом эксперимента является внесение части псевдо-вредоносного кода в файлы разных типов (текстовый, графический, аудио, видео) и попытка их передачи на файловые серверы. После того как зараженный файл будет определен, агент безопасности останавливает его передачу и основываясь на собранной информации о сети, определяет самый быстрый путь передачи служебного сообщения с обнаруженной угрозой в блок управления безопасностью распределенной сети (в нашем случае на компьютер администратора сети) для последующей реакции на инцидент безопасности.

Также отслеживается скорость передачи сообщений в блок управления безопасностью для сбора статистики и вычисления средней скорости реагирования на вторжения.

Результаты экспериментальных исследований по оценке эффективности агентов безопасности

Процесс сбора результатов проходит в несколько этапов. Вначале смоделирована классическая модель агентов с вертикальными связями, которая реализует такую последовательность действия во время обнаружения атаки [6]:

1. Агент нижнего уровня обнаруживает угрозу на своем узле и отправляет сообщение об обнаруженной атаке на агент верхнего уровня.

2. Агент верхнего уровня, получив сообщение об угрозе, передает его главному агенту системы (агенту 1-го уровня).

3. Главный агент системы после полученного сообщения об атаке, проводит анализ сообщения.

4. После завершения анализа, всем агентам верхнего уровня системы выдается один из возможных результатов:

- да, обнаружена атака.
- нет, ложная угроза.

5. Агенты верхнего (2-го) уровня соответствующе реагируют и пересылают сообщение об угрозе на агенты нижнего (3-го) уровня.

6. Получив сообщение об угрозе от агентов 2-го уровня, агенты 3-го уровня выполняют соответствующие правила реагирования.

Далее на смоделированной классической системе проведены эксперименты, состоящие в проведении атак на сервера хранилища данных.

При этом собирается статистика по скорости реагирования на возникающие угрозы, а именно – общее время реакции всей системы на обнаруженную угрозу.

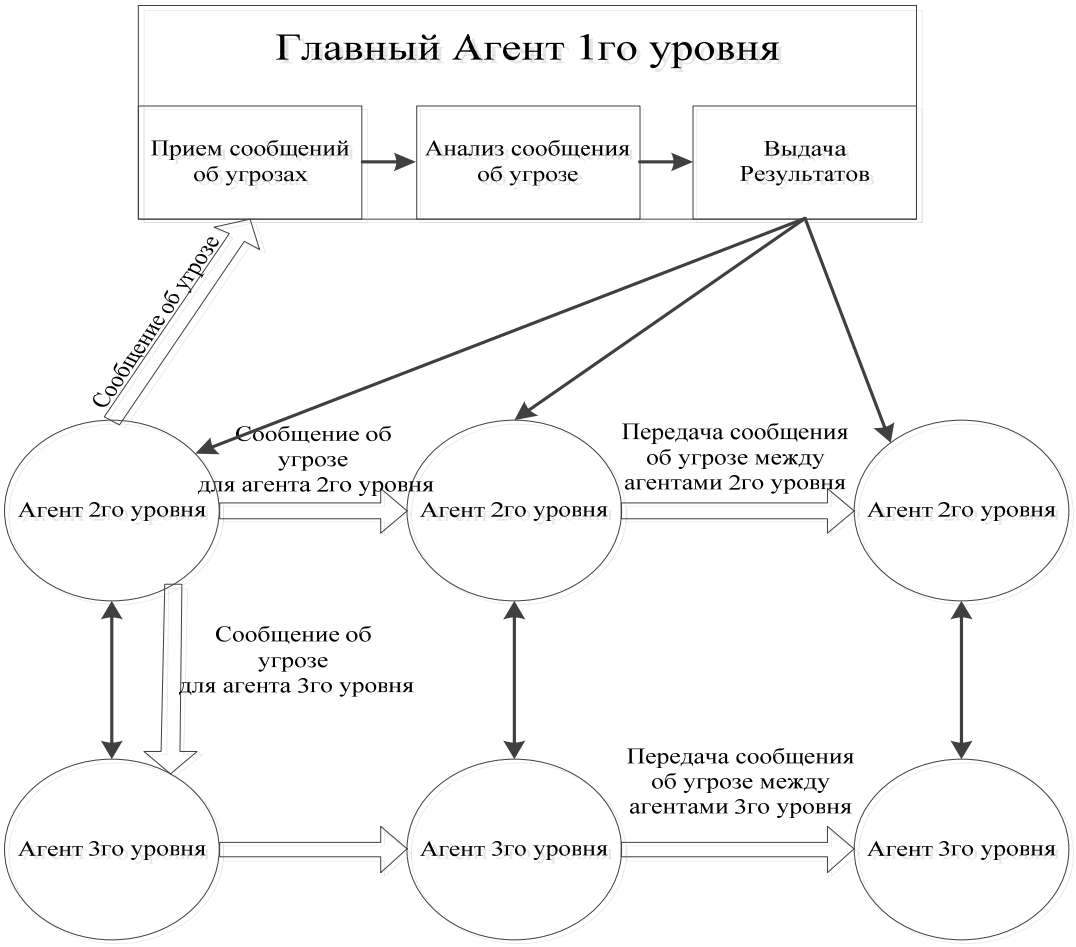


Рис. 3. Иерархическая модель агентов с горизонтальными связями

На рисунке 3 показана предложенная иерархическая модель агентов с горизонтальными связями, которая реализует следующую последовательность действия при обнаружении атаки:

1. Агент нижнего уровня обнаруживает угрозу на своем узле и отправляет сообщение об обнаруженной атаке на агент верхнего уровня, а также на соседние агенты своего уровня.

2. Агент верхнего уровня обнаруживает угрозу на своем узле и отправляет сообщение об обнаруженной атаке на Главный агент 1го уровня, а также на соседние агенты своего уровня.

3. Главный агент 1го уровня проводит анализ полученного сообщения об атаке.

4. Агент 2-го уровня после полученного сообщения об атаке от соседнего агента, выполняет соответствующее правило из базы полномочий, а также передает сообщение своему соседнему агенту 2-го уровня, аналогично агенты 3-го уровня.

5. Главный агент, проведя анализ полученного сообщения об атаке, выдает результаты агентам верхнего уровня:

- да, обнаружена атака.
- нет, ложная угроза.

6. Получив сообщение об угрозе от агентов 2-го уровня, агенты 3-го уровня выполняют соответствующие правила реагирования.

7. Все агенты иерархии, после получения соответствующего приказа о реагировании от главного агента:

- при необходимости изменяют свои правила реагирования;
- отменяют уже принятые действия, на основе сохраненных резервных копий.

Далее на смоделированной иерархической системе агентов с горизонтальными связями проведены эксперименты, аналогичные исследованиям для классической системы.

Рассмотрим фрагменты полученной статистики соотношения размера передаваемых файлов на хранилище данных со скоростью их передачи для агентов с горизонтальными связями и классических агентов (таблица 1).

Как видно из таблицы наблюдается стабильность скорости передачи данных в облачное хранилище, как для классических иерархических агентов, так и для агентов с горизонтальными связями.

Таблица 1.
Соотношение размера файлов и скорости записи в хранилище данных

Размер файла	Скорость при иерархических агентах с горизонтальными связями	Скорость при классических агентах
0,015555	0,323200632	0,323200632
0,135202	2,095765129	2,129567792
0,211943	3,285326761	3,285326761
0,314252	3,264752327	3,934445112
0,534301	4,174226563	4,786957067
1,168256	6,100935829	6,068484043
1,812223	6,653191817	6,298039229
2,395851	6,821277674	7,133226348
3,096903	6,904838667	6,920639213
3,848665	7,078082701	7,297984299
4,269056	7,212802768	7,030354132
4,830636	7,54786875	7,191185928
5,144544	6,987439152	7,30228016
6,894529	7,423305928	7,491437816
7,705764	7,647520484	7,406653697
8,478208	7,262719298	7,359555556
9,65472	7,634362348	7,447422986
10,22208	7,608612805	7,696607556
12,00078	7,498088112	7,611611562
13,1072	7,450523865	7,446189645
15,605589	7,619916504	7,446189645
17,355912	7,690169153	7,746404393
30,360102	7,712938894	7,712938894
44,655521	7,752694618	7,797051176
54,955644	7,806199432	7,650409279
64,183644	7,743926346	7,714380288
66,819222	7,835392229	7,734164571
77,499994	7,800019364	7,725179942
86,663502	7,770849892	9,654611701
89,828166	9,515437505	9,597682534
108,000054	9,506832769	9,669826967
117,143208	7,830093365	7,698362319
120,892616	7,813315375	7,702191761
217,396657	9,622509308	9,691031307

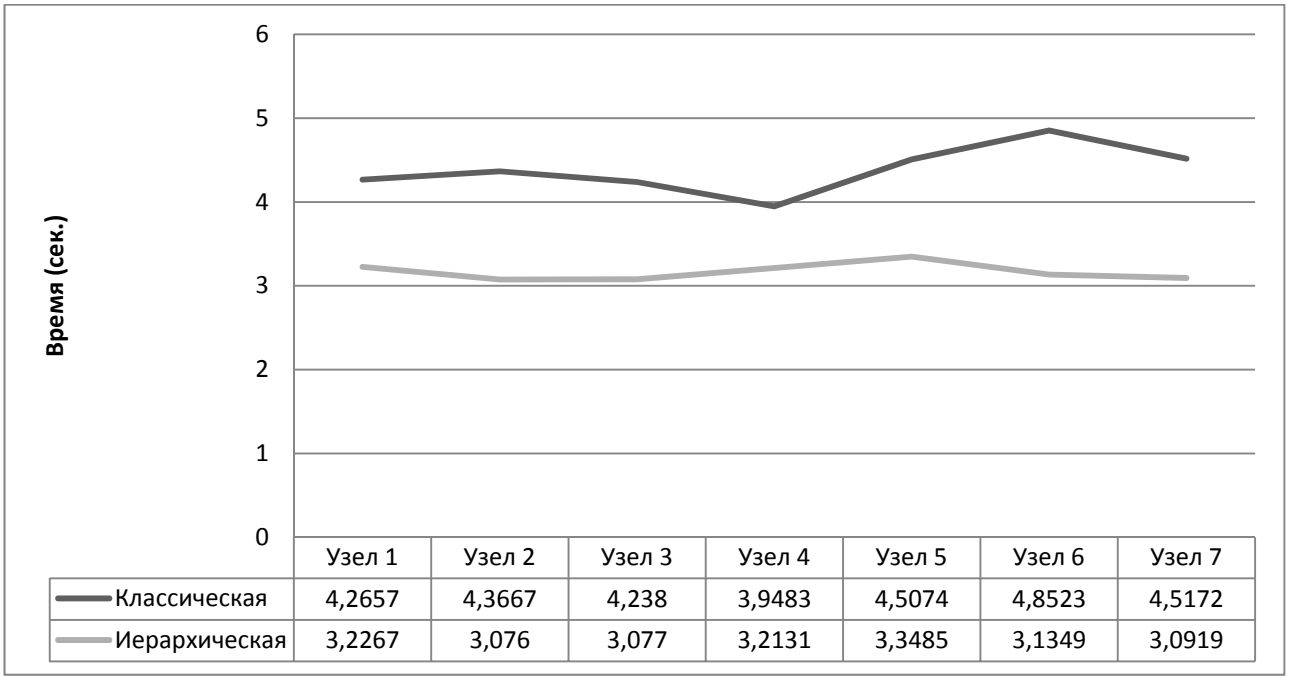
Рассмотрим полученную статистику времени реагирования на угрозу. Всего было проведено 10 экспериментов для каждой из системы, во время которых по сети выполнялась передача данных на сервера хранилища данных.

При каждом эксперименте менялась структура файлов, записываемых в облачное хранилище данных. В файлы добавлялся псевдовредоносный код, на который агенты должны реагировать.

Время реакции системы классических агентов колеблется по результатам проведенных экспериментов в рамках минимального значения 2 секунды и максимального 6 секунд.

Для системы с иерархическими агентами время реакции колеблется от 2,5с до 4,3с., что объясняется наличием упреждающего реагирования на обнаруженную угрозу за счет горизонтальной передачи сообщения соседним агентам.

Как видно из рисунка 4 среднее время реакции каждого узла на потенциальную угрозу модели с классическими агентами составляет от 4 до 5 секунд, в свою очередь система с горизонтальными связями реагирует на угрозы в среднем в рамках от 3 до 3.35 секунд.



	Узел 1	Узел 2	Узел 3	Узел 4	Узел 5	Узел 6	Узел 7
Классическая	4,2657	4,3667	4,238	3,9483	4,5074	4,8523	4,5172
Иерархическая	3,2267	3,076	3,077	3,2131	3,3485	3,1349	3,0919

Рис. 4. Сравнение среднего времени реакции классической и иерархической моделей с горизонтальными связями

Как видно из графиков (рис. 4), разработан-ный метод на основе иерархических агентов с горизонтальными связями позволяет повысить эффективность управления безопасностью компьютерных систем на основе cloud computing за счет снижения скорости реакции системы на вторжение, практически не понизив скорости записи файлов в облачное хранилище данных.

Эксперименты показали, что предложенные средства управления безопасностью на основе иерархических агентов имеют меньшее время реагирования на вторжение, построены таблицы и диаграммы соотношения времени реакции данных и классических агентов

Заключение

В больших распределенных компьютерных системах, на которых построены современные “гиганты” облачных вычислений, очень часто, если не постоянно происходит реконфигурация системы, в связи с постоянным появлением, модификацией и уничтожением виртуальных машин и пользователей. В связи с этим требуется повысить эффективность управления безопасностью. Предложенный метод иерархических агентов с горизонтальными связями позволяет повысит эффективность управления безопасностью распределенной компьютерной системы.

Список литературы

1. Тарасов В. Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. – М.: Эдиториал УРСС, 2002.
2. Tesfatsion L., Judd K. L., Handbook of computational economics: agent-based computational economics. Vol. 2. North-Holland, 2006.
3. Weiming Shen, Distributed Manufacturing Scheduling Using Intelligent Agents, National Research Council Canada, 2002
4. Junwei C., Stephen A. Jarvis, Subhash Saini, Darren J. Kerbyson and Graham R. Nudd Department of Computer Science University of Warwick, Coventry, CV4 7AL, UK ARMS: An agent-based resource management system for grid computing 2002.
5. Гайдамакин Н.А. Разграничение доступа к информации в компьютерных системах. – Екатеринбург: изд-во Урал. Ун-та, 2003. – 328 с.
6. Радченко Г.И., Распределенные вычислительные системы, Челябинск, 2012

УДК 004.6

ЯЦИШИН А.Ю.

ПРОЕКТУВАННЯ МУЛЬТИБАЗОВИХ СХОВИЩ ДАНИХ НА ОСНОВІ ДВОХФАЗНОГО АЛГОРИТМУ

У статті описується підхід до проектування сховищ даних на основі структурованості даних і запитів користувачів. Уводиться поняття мультибазового сховища даних, будується його математична модель та пропонується двохфазовий алгоритм проектування мультибазових сховищ даних із використанням генетичного алгоритму.

The article describes approach to design of data warehouses based on data structuredness and user queries to the data warehouses. The concept of multibase data warehouse is introduced, its mathematical model is built and two-phase building algorithm for multibase data warehouses which uses genetic algorithm is suggested.

Вступ

При проектуванні сховищ даних постає питання розташування даних у сховищі для забезпечення ефективного його функціонування.

У наукових публікаціях були запропоновані різні архітектури сховищ даних і розглянуті питання, пов'язані з джерелами даних. Однак підходу, який би описував проектування сховищ даних з врахуванням інформації про джерела даних і запити, що обробляються цим сховищем, запропоновано не було.

Метою статті є розгляд питання проектування сховищ даних з врахуванням їх структурованості та запитів до сховища.

Огляд існуючих архітектур сховищ даних

«Багатовимірне сховище даних». У книзі «The Data Warehouse Toolkit. Second Edition» Р. Кімболл (R. Kimball) запропонував підхід до побудови сховищ даних, який також відомий як підхід «знизу вгору», оскільки проектування сховища починається з нижнього рівня сховища (вітрин даних) і закінчується його верхнім рівнем (операційними базами даних) [1].

Сховище, побудоване за даним підходом, включає операційні бази даних, перехідну область, вітрини даних, засоби доступу до даних.

У цьому сховищі операційні бази даних зберігають первинну інформацію. Перехідна область слугує для перенесення даних з операційних баз даних до вітрин даних, які містять як детальні, так і агреговані дані.

Запропонована архітектура багатовимірного сховища даних дозволяє ефективно виконувати запити як до агрегованих даних, так і зберігати детальні дані. Використання архітектури шини та понять узгодженості дозволяє отримати незалежні між собою вітрини, які використовують один набір вимірів, що забезпечує можливість паралельної розробки вітрин різними підрозділами, а також цілісність усієї системи в цілому.

«Реляційне сховище даних». Другим основним підходом до побудови сховищ даних є підхід Інмона. Цей підхід відомий як підхід «зверху вниз», оскільки проектування системи починається з верхнього рівня (власне сховища даних) і закінчується нижнім рівнем (вітринами даних) [2, 3]. Сховище, побудоване за таким підходом, було названо «корпоративною інформаційною фабрикою». Крім того, наявність вітрин даних дозволяє отримувати агреговані дані та використовувати переваги систем класу on-line analytic processing (OLAP).

Таке сховище, за Інмоном, містить корпоративні застосування, перехідну область, операційне сховище даних, сховище даних підприємства, вітрини даних відділів, застосування систем прийняття рішень, сховища для дослідження або видобування даних, альтернативні системи зберігання даних.

Незалежні вітрини даних. Певною модифікацією підходу Кімболла є підхід незалежних вітрин даних, описаний у праці [4]. Основною відмінністю цього підходу є незалежність і неузгодженість між собою вітрин даних. Даний підхід не деталізує первинні бази даних, перехідну область і застосування користувачів.

Централізоване сховище даних. Узагальненим варіантом підходу Інмона є підхід до побудови централізованого сховища даних, у якому багатовимірні представлення забезпечуються реляційною базою даних [5]. Відмінністю від підходу Інмона є те, що в сховищі немає

вітрин даних. Цю архітектуру слід вибирати тоді, коли сховище даних є інтегральною частиною стратегічного рішення.

Об'єднане сховище даних. Об'єднане сховище даних [5] передбачає наявність уже розгорнутої системи (бази даних, сховища даних, застосування і т.ін.) та призначене для отримання доступу до даних з існуючих джерел. Дані інтегруються (логічним або фізичним способом), використовуючи спільні ключі, глобальні метадані, запити та інші методи. Використання цієї архітектури рекомендується в існуючій складній інфраструктурі підтримки прийняття рішень, коли фірма не хоче перебудовувати цю інфраструктуру. Основною перевагою цієї архітектури є простота проектування і можливість використання існуючих систем для побудови сховища.

Основним недоліком цієї архітектури є складність аналізу даних у вітринах різної структури та гетерогенність побудованої в результаті системи.

Цей підхід бажано використовувати у випадку, коли ресурси є обмеженими, навики IT-персоналу є низькими, і сховище не є стратегічним рішенням (наприклад, частиною домену)

Об'єднане сховище даних. Своєрідним поєднанням підходів Інмона та Кімболла є так зване об'єднане (federated) сховище даних [6], яке складається з первинних баз даних (реляційні бази даних), вітрин даних (багатовимірні сховища) ,загальної перехідної області, що виконує операції отримання, перетворення та завантаження даних ;власне об'єднаного сховища даних, в яке завантажуються дані з реляційних баз даних і з якого вивантажуються дані для багатовимірних вітрин даних.

Особливою рисою цієї архітектури є те, що в ній загальні дані розподіляються між сховищами даних і вітринами даних,.

Об'єктно-реляційне сховище даних.

У статті "Metadata for Object-Relational Data Warehouse" розглядається об'єктно-реляційне сховище даних, що будується на основі логічної архітектури сховища [7] з використанням об'єктно-орієнтованого підходу та містить шар метаданих [8].

При цьому підході більшість шарів отриманої архітектури, крім носія даних, складається з множини об'єктів різних типів, які виконують основні функції кожного компонента.

У даній архітектурі, потік даних аналогічний потокам у інших архітектурах сховищ даних, де дані збираються з різних систем операційних баз даних, узагальнюються, агрегуються та інтегруються в сховище даних і використовуються тільки для читання даних і підтримки комплексного аналізу [1, 7, 9, 10].

Ця архітектура складається з наступних компонентів.

1. Шар інтерфейсу застосувань. У цьому шарі сховища об'єкти інкапсулюють складні процеси від користувачів сховищ даних. Об'єкти даного компоненту діляться на різні групи, які обслуговують різні служби. Ґрунтуючись на їх функціональних можливостях, кожна служба відповідає на відповідні запити додатків сторонніх розробників, аналізаторів даних та інших користувачів системи сховища даних.

2. Компоненти збору даних, що можна розглядати як інструмент, який будує підсистему даних сховища. Об'єкти збору даних отримують, перетворюють і передають дані з оперативних сховищ даних в об'єктно-реляційну базу даних сховища.

3. Компоненти управління сховищем, які безпосередньо звертаються до даних сховища даних або отримують дані з інших баз даних. Вони надають послуги, які з'єднують шар інтерфейсу застосувань та об'єктно-реляційну базу даних.

4. Метадані, які застосовуються для задавання інформації про дані сховища даних.

5. Носії даних. Представлення даних, в об'єктно-реляційному сховищі даних відрізняються від такого у реляційному середовищі і об'єктно-орієнтованому сховищі даних. Наприклад, в об'єктно-реляційному сховищі даних прості дані можуть бути розміщені в багатовимірних структурах, що схожі на реляційних системи управління базами даних [7, 9, 10]. У випадку складних даних, дані можуть бути змодельовані в ієрархічні структури об'єктів, як це пропонується для об'єктно-орієнтовної СКБД.

Гібридне сховище даних. У роботі [11] було запропоновано концепцію узагальненого гібридного сховища даних.

У гібридному сховищі присутні реляційна та багатовимірна бази даних. Для проектування гібридного сховища поєднуються підходи проектування «зверху вниз» і «знизу вгору», тобто реляційна і багатовимірна бази даних проектуються в комплексі для забезпечення оптимального функціонування сховища даних у цілому.

Крім того, гібридне сховище даних має свою систему керування (СКГСД), за допомогою якої здійснюється робота з сховищем, наприклад, виконання запитів до сховища.

Проектування гібридних сховищ даних здійснюється на основі генетичного алгоритму, який визначає ознаки розміщення областей сховища, матеріалізованості та індексування стовпців таблиць сховища.

Рівні структурованості даних

Однією з істотних характеристик даних є їх структурованість, що може суттєво вплинути на дисковий простір, який займає сховище даних, тому що у випадку неструктурованих і слабко-структурованих даних вони зберігаються розріджено. Це призводить до падіння швидкодії сховища даних у залежності від операцій «введення – виведення».

Структурованість даних традиційно є одним з ключових чинників при ухваленні рішення про відповідний формат представлення даних (наприклад, реляційна бази даних для структурованих і у форматі XML для частково структурованих даних).

Цей вибір, в свою чергу, зазвичай визначає організацію даних (наприклад, з використанням теорії залежностей і нормальних форм для реляційної моделі та ієрархічної структури тегів у випадку XML), що має вирішальне значення при прийнятті рішення про те, як індексувати ці дані (наприклад, індекси на основі збалансованого дерева для реляційної моделі і індекси на основі схеми перелічення у випадку XML).

Структурованість також суттєво впливає на доступ до даних (наприклад, за допомогою SQL запитів для реляційних баз даних і XPath / XQuery для XML).

Таким чином, від структурованості даних залежать усі аспекти керування даними, а продуктивність СКБД вимірюється за даними з очікуваним рівнем структурованості (наприклад, в тесті TPC-H для реляційної моделі і XMark для даних XML).

Дані у джерелах можна класифікувати за рівнем структурованості наступним чином.

Структуровані дані. Це ті дані, які мають чітко задану структуру і точно відповідають цій структурі. Щодо структурованих даних не виникає питань щодо їх розміщення, оскільки вони зберігаються щільно в силу наповненості всіх атрибутів в всіх кортежах даних, а тому їх в розміщують в реляційних базах даних.

Слабко-структуровані дані. Існує декілька визначень слабко-структурованих даних.

За одним із них, *"слабко-структуровані дані є даними, які описують самі себе"* [12]. Відповідно до цього формулювання структури даних зберігаються разом зі своїми даними як метадані за допомогою міток. Ці мітки представляють собою семантику кожного елемента даних. Крім того, дані значення пов'язані один з одним за допомогою вбудованої ієрархії, яка є природнім зв'язком між елементами даних.

За іншим формулюванням, *"слабко-структуровані дані є даними, що не мають схеми"* [13], тобто відсутня фіксована, жорстка схема, якій повинні відповідати дані.

Існують також визначення слабко-структурованих даних, заснованих на нерегулярності даних, яку вони можуть представляти. Наприклад, до слабко-структурованих даних відносяться *"дані, які представлені деякими закономірностями* (*це не зображення або текст*)*, але, можливо, не так добре* (*структуровані*)*, як деякі реляційні дані або дані ODMG"* [14].

В роботі «Ozone: Integrating Structured and Semistructured Data» визначено слабко-структуровані дані як "*дані, які є нерегулярними або які відображають неоднорідність структури та типу, оскільки вона не може відповідати жорсткій, заздалегідь визначеній схемі* " [15].

Слабко-структуровані дані з точки зору структури мають такі ключові характеристики:

- структури цих даних є нерегулярними, неявними та носять частковий характер;
- слабко-структуровані дані є більш гнучкими в порівнянні з сильно-структурованими даними.

Основні характеристики слабко-структурованих даних з точки зору схеми даних можна описати наступним чином:

- використовуються апріорні, а не апостеріорні схеми, та індикативні, а не обмежувальні структури;
- схеми можуть не враховуватися та суттєво змінюватися;
- типи даних елементів є еклектичними та не є точними;
- немає чітких відмінностей між елементом схеми та даних.

Частково-структуровані дані. У роботі «A Performance Analysis of a Hybrid Relational-XML

Approach to Store Partially-structured Data» було дане наступне визначення частково-структурованих даних:

«Частково-структуровані дані – це гібридні дані, частково структуровані і частково слабко-структуровані. Вони містять точки входу з структурованих даних до частково структурованих даних і навпаки" [14].

Згідно цієї роботи частково-структуровані дані розміщуються в документах XML наступним чином.

Частково-структурований документ XML містить у своїй ієрархічній структурі хоча б один високо-структурований і, як мінімум, один слабко-структурований елемент, де кожне піддерево з коренем у високо-структурованому елементі може містити комбінацію високо-структурованих елементів та/або слабко- структурованих елементів, які є вузлами цього піддерева. Кожне піддерево з коренем у слабко-структурованому елементі, складається тільки з слабко-структурованих елементів в якості його вузлів.

Неструктуровані дані. Це такі дані, для яких не визначена жодна структура. Для таких даних постають суттєві проблеми їх структурованого зберігання, оскільки вони не мають метаданих, а тому відсутня будь-яка інформація щодо типів даних і рекомендацій щодо їх оптимального зберігання.

Варто зазначити, що така класифікація поширюється не тільки на дані, а й на джерела даних. У дослідженні [15] XML-документи класифікуються у залежності від ступеня структурованості даних, що містяться в документі:

- високо-структуровані документи, що містять лише високо-структуровані дані, тобто дані, які мають чітку структуру або організацію та відповідають реляційній або об'єктно-орієнтованій моделі даних;
- слабко-структуровані документи містять лише слабко-структуровані дані;
- неструктуровані документи містять лише неструктуровані дані.

Аналіз існуючих досліджень з проектування сховищ даних

Аналіз вище наведених підходів до побудови сховищ даних дозволяє зробити наступні висновки.

По-перше, існуючі архітектури сховищ даних не передбачають розподілу даних по їх структурованості.

По-друге, наявних компонент в описаних архітектурах сховищ даних недостатньо для оптимального збереження даних всіх рівнів структурованості.

Зокрема, розміщення в одній базі даних (з використанням її моделі даних) сильно-структурованих, слабко-структурованих та частково-структурованих даних має наступні недоліки.

При представленні частково структурованих даних з використанням моделі високо-структурованих даних (наприклад, реляційної моделі) отримуємо складну структуру даних із надлишком при збереженні слабко-структурованої частини даних. Будь-які незначні зміни в цій структурі даних можуть суттєво вплинути на схему високо- структурованих даних. Крім того, модель високо-структурованих даних не надає навігаційного підходу для пошуку її даних (так званий перегляд даних), яка є корисною особливістю моделей слабко-структурованих даних.

Якщо ж представити сильно-структуровані дані з використанням моделей слабко-структурованих даних, то ці моделі не враховують структуровану частину даних високо-структурованої частини документа. Це означає, що дані розглядаються як слабко-структуровані, при цьому втрачаються їх переваги як сильно-структурованих – наприклад, типізація цих даних для оптимізації їх зберігання та виконання запитів до них.

Мультибазове сховище даних

Для проектування сховища даних з врахуванням як запитів, так і структурованості даних пропонується взяти за основу концепцію гібридних сховищ даних і розширити сховище даних базою даних XML та прямим використанням файлової системи (див. [11]).

Це розширення дасть змогу розміщувати структуровані дані в реляційній базі даних, слабко-структуровані – в багатовимірній базі даних, частково-структуровані поділяти між реляційною базою даних і базою даних XML, а неструктуровані дані зберігати у вигляді файлів файлової системи. За допомогою системи керування сховища даних буде забезпечений доступ до інформації незалежно від того, де вона розміщена.

У зв'язку з вищевикладеним визначимо мультибазове сховище даних як розширення гібридних сховищ даних наступним чином.

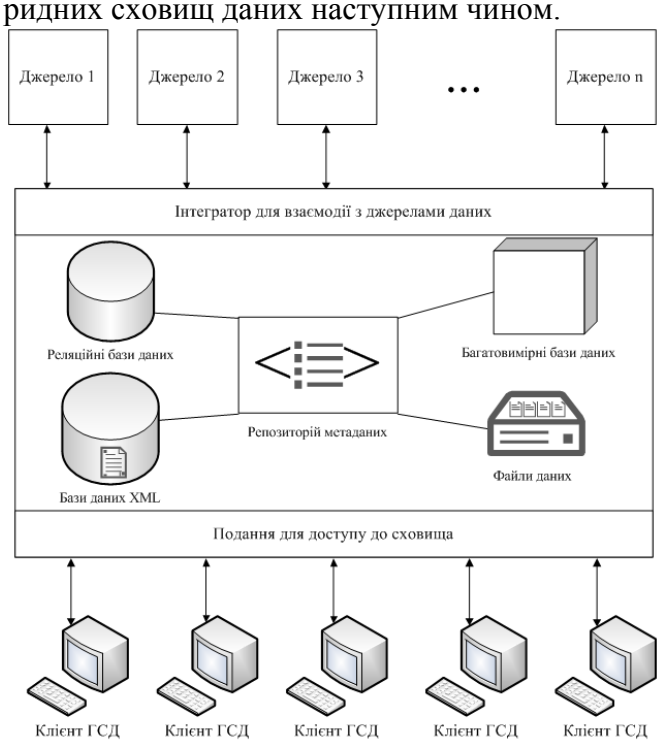


Рисунок 1. Архітектура сховища даних згідно концепції мультибазових сховищ даних

Визначення 1. Мультибазове сховище даних (рис.1) – це сховище даних, яке містить реляційні бази даних, багатовимірні бази даних, бази даних XML, файли файлової системи, репозиторій метаданих, інтегратор джерел даних, подання для доступу до сховища.

Основними характерними властивостями, які відрізняють мультибазові сховища даних від інших сховищ даних, є наступні.

1. Наявність своєї системи керування мультибазовим сховищем даних (СКМСД), за допомогою якої здійснюється робота з сховищем (виконання запитів до сховища).
2. Наявність у сховищі реляційної БД, основними призначенням якої є зберігання структурованих даних і даних, до яких здійснюється частий доступ.
3. Наявність у сховищі багатовимірної БД, яка може містити як атомарні, так і узагальнені дані. Основним призначенням багатовимірної бази даних є зберігання даних, до яких виконуються складні запити.
4. Наявність у сховищі бази даних XML, основним призначенням якої є зберігання слабкоструктурованих даних і слабкоструктурованої частини частково-структурованих даних;
5. Збереження неструктурованих даних у вигляді файлів, що зберігаються безпосередньо у файловій системі.
6. Взаємодія з джерелами даних, що здійснюється за допомогою інтегратора та полягає у відслідковуванні змін даних і метаданих, які відбуваються у джерелах, і застосуванні цих змін відповідно до налаштувань сховища даних.
7. Уніфікований доступ користувачів до сховища даних через подання сховища даних, який дає змогу користувачам звертатися до даних за допомогою єдиного інтерфейсу, незалежно від фізичного та логічного розташування цих даних у сховищі.

При проектуванні мультибазового сховища даних для забезпечення його оптимального функціонування у цілому пропонується поєднати підходи проектування на базі структурованості джерел даних і запитів до сховища даних.

Математична модель мультибазового сховища даних

Спочатку визначимо джерела даних, для яких проектується сховище даних.

У випадку розділених структурованих файлів метадані задаються частково. У таких файлах можуть бути задані лише назви полів даних, що входять у файл, а типи даних не задаються. Розділені структуровані файли визначаються через множину атрибутів даних $SF = \{sf_i\}$.

У випадку структурованих файлів XML метадані задаються за допомогою схеми, яка містить назву тегу, тип його значення та інші метадані. Файли XML визначаються через множину атрибутів тегів $X = \{x_i\}$.

У випадку реляційних баз даних структура даних задається за допомогою відношень. Відношення визначаються як впорядкований набір атрибутів відношення $R = (r_1, r_2, ..., r_1)$, де R – відношення, а r_i – атрибути відношення. Якщо два відношення $R_1 = (r_1, r_2, ..., r_1, r_{l+1}, ..., r_{l+p})$ та $R_2 = (r_l, r_{l+1}, ..., r_{l+p}, ..., r_{l+k})$ мають спільні атрибути $r_1, r_{l+1}, ..., r_{l+p}$, то в одному з відношень цей набір атрибутів є первинним ключем і приймає

унікальні значення, а в іншому він є зовнішнім ключем.

У випадку багатовимірних баз даних структура даних задається набором вимірів $D=\{d_1,d_2,...,d_q\}$ та мір $M=\{m_1,m_2,...,m_u\}$.

Використовуючи ці позначення, представимо мультибазове сховище даних (МСД) наступним чином:

$$MDW = <B,T,A,F,V> ,$$

де B – множина атрибутів даних сховища;

T – множина таблиць даних сховища;

A – множина областей сховища;

F – множина файлів сховища;

V – множина подань для доступу до сховища.

Визначення 2. *Атрибутом даних b сховища* називається іменована сутність, яка задається його назвою, типом значення та доменом (можливим діапазоном значень). Для множини атрибутів даних B сховища виконується:

$$B = \{b_i\} = SF \cup X \cup R \cup D \cup M .$$

Атрибут сховища даних може бути представлений атрибутом відношення реляційної БД, виміром або мірою багатовимірної БД, тегом XML.

Визначення 3. Таблицею t даних сховища називається впорядкована множина наборів значень атрибутів. Ці набори атрибутів називаються кортежами таблиці. Для множини таблиць T даних сховища виконується:

$$T = \{t_i \mid t_i = \{b_j\}\} .$$

Таблиця сховища даних може бути представлена таблицею реляційної БД, набором значень міри для всіх можливих значень вимірів, набором файлів XML, які мають спільну множину тегів.

Визначення 4. *Областю сховища даних a* називається множина зв'язаних таблиць, які мають хоча б один спільний атрибут:

$$A = \{a_i\} \mid a_i = \{t_j\}, j \in \overline{1,w}, \exists b_k \in t_{j_1}, b_k \in t_{j_2})\} .$$

Визначення 5. *Файлом сховища даних f* називається іменована послідовність бітів, яка зберігається у вигляді файлу файлової системи.

Визначення 6. *Поданням сховища даних v* є впорядкована трійка

$$v = <B_v,T_v,S_v>,$$

де $B = \{b_v\}$ – множина атрибутів даних, на яких визначене це подання; $T = \{t_v\}$ – множина таблиць даних, на яких визначене це подання; $S = \{s_v\}$ – множина умов вибору даних.

Задача проектування мультибазового сховища даних

Визначимо задачу проектування мультибазових сховищ даних наступним чином.

Нехай, задані множини атрибутів розділених файлів SF, файлів XML X, відношення у реляційній базі даних R, виміри D і міри M багатовимірної бази даних. Крім того, задане значення граничної частоти доступу до даних F_c.

Необхідно спроектувати МСД, а саме:

- визначити області сховища даних A, таблиці T, атрибути B, файли F;
- знайти значення структурованості джерел даних ST_s та ознак розміщення даних джерел по структурованості SL_s;
- знайти значення ознак розміщення областей по запитах L_a, індексування полів таблиць M_c та матеріалізації I_c, на яких значення цільової функції часу виконання запиту буде мінімальним серед всіх можливих наборів значень цих змінних.

Оскільки МСД є розширенням гібридного, то можна здійснити його проектування на основі запитів згідно підходу, описаного у [11].

Для врахування структурованості даних при проектуванні МСД необхідно визначити процедури розрахунку структурованості даних різних джерел і розподілу даних різного рівня структурованості у сховищі даних.

Розрахунок структурованості ST_s різних джерел даних здійснимо із використанням наступних співвідношень.

Для реляційних баз даних ST_s=1. Це випливає з того, що дані в реляційних базах є структурованими.

Для багатовимірних баз даних у якості структурованості можна взяти зважену суму наповненостей по кожній мірі

$$ST_S = \sum_{\forall M \in \{M\}} CV(\{D_M\}, M) \times WT(M), \qquad (1),$$

де M – міри джерела S, а D_M – всі виміри, на яких визначені елементи для мірі M, $CV(\{D_M\}, M)$ – структурованість для міри M; $WT(M)$ – коефіцієнт зваженості для міри M .

Структурованість CV даних міри M знаходять за формулою:

$$CV(\{D_M\}, M) = \frac{\sum_{\forall D \in \{D_M\}} E_D^M}{\prod_{\forall D \in \{D_M\}} |D|}, \qquad (2),$$

де E_D^M – кількість клітинок, які визначені по виміру D міри M.

Коефіцієнт зваженості $WT(M)$ для міри M знаходять за формулою:

$$WT(M)=\frac{\sum_{\forall D\in\{D_M\}}|D|}{\sum_{\forall M'\in S}\sum_{\forall D\in\{D_{M'}\}}|D|}. \quad (3)$$

Для повністю заповненої бази маємо

$$\sum_{\forall D\in\{D_M\}}E_D^M=\prod_{\forall D\in\{D_M\}}|D|. \quad (4),$$

При умові (4) з формул (1-3) випливає, що ST_S=1, а для пустої бази даних ST_S=0.

Для файлів даних XML можемо скористатися співвідношеннями, визначеними в [16].

Системою типів T є наступна множина

$$\tau=\{\mathrm{T}\mid\exists(s,w3type,T)\in D\} \quad (5)$$

Набором екземплярів I типу T в системі типів D є множина

$$I(T,D)=\{s\mid\exists(s,w3type,T)\in D\}. \quad (6)$$

Набором властивостей P типу T в системі типів D є множина:

$$P(T)=\{p\mid s\in I(T,D),\exists(s,p,o)\in D\}. \quad (7)$$

Кількість входжень властивості p в набір екземплярів $I(T, D)$ (позначимо через $OC(p, I(T, D))$) визначається наступним чином:

$$OC(p,I(T,D))=\\ =\left|\{s\mid s\in I(T,D),\exists(s,p,o)\in D\}\right|. \quad (8)$$

Структурованість для одного типу даних визначається за формулою:

$$CV(T,D)=\frac{\sum_{\forall p\in P(T)}OC(p,I(T,D))}{|P(T)|\times|I(T,D)|}. \quad (9)$$

Структурованість набору даних $CH(\tau,D)$ визначається за формулою:

$$CH(\tau,D)=\sum_{\forall T\in\tau}WT(CV(T,D))\times CV(T,D), \quad (10)$$

де $\mathrm{WT(CV(T,D))}$ – коефіцієнт зваженості для типу даних T у наборі D, який визначається як

$$WT(CV(T,D))=\frac{|P(T)|+|I(T,D)|}{\sum_{\forall T'\in\tau}|P(T')|+|I(T',D)|}. \quad (11)$$

Крім того, якщо у двох наборів даних (які містяться або в двох тегах одного файлу, або в двох різних файлах) є спільні теги (атрибути), то їх розраховуються як один з'єднаний набір, а загальна структурованість всього джерела розраховується наступним чином:

$$ST_S=CH(D,S)=\sum_{\forall D\in S}CH(\tau_D,D)\times\\ \times WT(CH(\tau_D,D)). \quad (12)$$

При цьому коефіцієнт зваженості визначається наступним чином:

$$WT=\frac{\sum_{\forall T\in\tau_D}(|P(T)|+|I(T,D)|)}{\sum_{\forall D'\in S}\sum_{\forall T\in\tau_D}(|P(T)|+|I(T,D)|)}. \quad (13)$$

Для нерозподілених файлів ST_S=0.

Для розподілу даних у сховищі по їх структурованості, виділимо такі класи структурованості даних.

1. Ідеально-структуровані дані. Для таких даних ST = 1, оскільки множина атрибутів кожного елемента набору даних співпадає з множиною всіх атрибутів набору

$$\forall I\in D: A(I)=A(D). \quad (14)$$

Для джерел з ідеально-структурованими даними SC_s.= 0.

2. Сильно-структуровані дані. Для таких даних:

$$\frac{|A(D)|\times|I|-|I|}{|A(D)|\times|I|}<ST<1. \quad (15)$$

При цьому існує хоча б один елемент набору даних, множина атрибутів якого дорівнює множині атрибутів набору:

$$\exists I\in D: A(I)=A(D). \quad (16)$$

Для джерел з ідеально структурованими даними SC_s.= 1.

3. Слабко-структуровані дані. Для таких даних об'єднання множин атрибутів усіх елементів набору даних дорівнює множині всіх атрибутів набору, але при цьому всі множини атрибутів елементів набору включені строго в множину всіх атрибутів набору:

$$\forall I\in D:\bigcup_{I\in D}A(I)=A(D),\\ \forall I\in D: A(I)\subset A(D). \quad (17)$$

Для джерел з слабко-структурованими даними SC_s.= 2.

4. Частково-структуровані дані. Такі дані є по суті змішаним типом між структурованими та слабко-структурованими. Для таких даних, як і для слабко-структурованих, усі множини атрибутів елементів набору строго включені в множину всіх атрибутів набору даних, але існує така підмножина атрибутів, на яких ці дані є ідеально або сильно структурованими:

$$\forall I\in D:\bigcup_{I\in D}A(I)=A(D),\\ \forall I\in D: A(I)\neq A(D),\\ \forall I\in D:\exists\{A\}\subset A(I). \quad (18)$$

Структурованість таких даних обмежена знизу наступним значенням

$$ST > 1/|A(D)|. \quad (19)$$

Для джерел з слабко-структурованими даними SC_s.= 3.

5. Неструктуровані дані. Такі дані представляються одним або декількома елементами без атрибутів:

$$|I| > 0, |A(D)| = 0. \quad (20).$$

Для джерел з слабко-структурованими даними SC_s.= 4.

Двохфазний алгоритм проектування мультибазових сховищ даних

Проектування МСД на базі структурованості даних виконаємо за допомогою двохфазного алгоритму, першою фазою якого є розміщення даних джерел на основі їх структурованості (проектування сховища даних), а другою – перерозподіл даних на основі запитів користувачів (оптимізація сховища даних).

Перша фаза алгоритму.

Як вхідні дані використовуються задані користувачем метадані джерел даних.

Вихідними даними є сховище даних з розміщеними у ньому даними.

1. Користувач визначає джерела даних $S = \{s_i\}, i = \overline{1,n}$.

2. Покласти $i = 1$.

3. Визначити структурованість ST_s джерела s_i, користуючись формулами (1, … ,13).

4. Визначити клас структурованості SC_s джерела s_i згідно формул (14, …, 20).

5. Якщо $SC_s = 0$ або $SC_s = 1$, то дані джерела s_i розмістити в реляційній базі даних (SL_s=0) і перейти до п. 9, інакше перейти до п. 6.

6. Якщо $SC_s = 2$, то дані джерела s_i розмістити в багатовимірній базі даних (SL_s=1) і перейти до п. 9, інакше перейти до п. 7.

7. Якщо $SC_s = 3$, то поділити джерело даних s_i на два джерела $s_{i'}$ та $s_{i''}$, для яких $SC_{s_{i'}} < 1$ (ідеально-структуровані та сильно структуровані дані), $SC_{s_{i'}} = 2$ (слабко-структуровані дані) та перейти до п. 5 для джерела $s_{i'}$ та розмістити дані джерела $s_{i''}$ у базі даних XML п. 6. для джерела $s_{i''}$, інакше перейти до п. 8.

8. Якщо $SC_s = 4$, то дані джерела s_i розмістити у файлах безпосередньо у файловій системі.

9. Покласти $i = i + 1$.

10. Якщо $i \le n$, перейти до п. 3, інакше перейти до п. 11.

11.Кінець першої фази.

Після виконання першої фази алгоритму сховище даних починає свою роботу.

Друга фаза алгоритму.

Як вхідні дані використовуються поточна структура сховища даних та запит до сховища даних.

Вихідними даними є сховище даних, яке оптимізоване відносно популяції запитів, до якої належить запит, що виконується.

1. Перевірити належність останнього виконаного запиту до поточної популяції запитів. Популяцією запитів вважається послідовність запитів однакового типу (точкового або інтервального) до одної і тої самої області сховища даних.

2. Якщо запит належить до нової популяції ,виконати генетичний алгоритм, інакше перейти до п. 3

3. Кінець другої фази.

Генетичний алгоритм мінімізує цільову функцією часу виконання запитів:

$$z = \sum_{t=1}^{n} t_i(L_a, \{I_c\}, \{M_c\} + \sum_{j=1}^{n-1} T_j + (T_a^{'} - \hat{T}_a)L_a), \quad (21)$$

де n – кількість таблиць запиту, а – область, що містить таблицю i, $\{I_c\}$ та $\{M_c\}$ – ознаки індексування та матеріалізації стовпчиків c таблиці i, $F_a = \sum_{i \in a} f_i$ – частота доступу до області a, $F_a^{'} = F_z$ – встановлене граничне значення частоти доступу до даних області а, $\hat{T}_a = 1/F_a$ – середній час виконання запитів, $T_a^{'} = 1/F'_a$ – граничне значення часу виконання запиту до даних області a.

Генетичний алгоритм є стандартним [17] з імовірністю мутації 0,05, імовірністю схрещування 0,8 , селекцією за методом колеса рулетки та одноточковим схрещуванням. Як початкова популяція береться особина, утворена з поточного стану сховища даних, і всі особини, у яких $\forall c\, I_c = 0$ та $\forall c\, M_c = 0$. Умовою завершення для популяції k є $f_k^* - f_{k+1}^* < \varepsilon$, де f_k^* – найменше значення цільової функції (21 серед усіх особин популяції k, ε – задане мале число.

Перша та друга фази поєднуються у загальному алгоритмі роботи МСД, що описується наступним чином.

1. Виконати першу фазу алгоритму проектування МСД для множини джерел S.

2. Поки сховище даних не отримало повідомлення про завершення роботи, виконувати наступне:

2.1. Якщо сховище даних знаходиться у стані «готовність», то:

2.1.1. Приймати запити від користувачів МСД та інтегратора.

2.1.2. Якщо отриманий запит від користувачів МСД, то:

2.1.2.1. Обробити запит, повернувши результат користувачеві.

2.1.2.2. Виконати другу фазу алгоритму проектування МСД. На час перепроектування встановити стан «обслуговування» сховища.

2.1.2.3. Встановити стан сховища «готовність» і повернутися до кроку 2.1.1.

2.1.3. Якщо отриманий запит від інтегратора про додавання або оновлення джерела даних *s*, то:

2.1.3. Виконати першу фазу алгоритму проектування МСД для джерела даних *s*. На час зміни встановити стан «обслуговування» сховища.

2.1.3.2. Встановити стан сховища «готовність» та повернутися до п. 2.1.1.

2.1.4. Якщо отримане повідомлення є запитом завершення роботи, перейти до п. 3.

3. Завершити роботу сховища даних.

Варто підкреслити, що друга фаза алгоритму проектування МСД не стосується баз даних XML і файлів, розміщених на файловій системі, а призначена лише для оптимізації виконання запитів певної популяції. Ця фаза повторюється для різних популяцій запитів,забезпечуючи оптимізацію розташування даних тих областей, до таблиць яких виконувалися запити.

Двохфазний алгоритм проектування МСД є скінченним, тому що у першій фазі алгоритму маємо скінченну кількість джерел даних, а друга фаза, яка є виконанням генетичного алгоритму для певного запиту, є скінченною в силу скінченної кількості змінних та області застосування даного алгоритму.

Збіжність алгоритму випливає зі збіжності стандартного генетичного алгоритму. Згідно теореми Голланда кількість індивідів, що зберігають, мають певний шаблон (сукупність генів, які потрібно зберегти в наступній популяції), зростає для кожної наступної популяції [17], що забезпечує отримання нових особин зі збереженням цього шаблону та кращим значенням ЦФ. У підсумку це дозволяє знайти оптимальне значення ЦФ.

Висновки

Аналіз існуючих робіт, що стосуються архітектур сховищ даних і структурованості даних показав, що при застосуванні існуючих підходів до проектування сховищ даних не можливо забезпечити ефективне розміщення даних різних класів структурованості. Запропоновано поняття мультибазових сховищ даних, для яких побудовано математичну модель сховища даних і сформульовано задачу їх проектування. Описано розрахунок структурованості джерел даних і методику визначення класів структурованості джерел даних.

Для проектування МСД запропоновано двохфазовий алгоритм, першою фазою якого є розміщення даних джерел на основі їх структурованості, другою – перерозподіл даних на основі запитів користувачів.

Використання такого підходу передбачає проектування на базі двох незалежних критеріїв розподілу даних у сховищі, при чому ці критерії застосовуються послідовно – при ініціалізації сховища та у процесі роботи сховища.

У подальших дослідженнях варто здійснити експериментальну оцінку отриманих теоретичних результатів для різних джерел даних, носіїв даних сховища даних і популяцій запитів до сховища даних, а також розглянути питання класифікації запитів до сховища даних і розподілу даних на основі запитів різних класів.

Список літератури

1. Ralph Kimball. The data warehouse toolkit: the complete guide to dimensional modeling [Текст] / Ralph Kimball – Wiley, 2002 – 436 p.
2. Claudia Imhoff.Corporate information factory [Текст] / Claudia Imhoff, Ryan Sousa – John Wiley & Sons, 2001 – 382 p.
3. W. H. Inmon. Corporate Information Factory Components [Електронний ресурс] / W. H. Inmon – Inmon Data Systems – Режим доступу : http://www.inmoncif.com/view/26.

4. Hugh J. Watson. Data Warehouse Architectures: Factors in the Selection Decision and the Success of the Architectures [Електронний ресурс] / Hugh J. Watson, Thilini Ariyachandra 2003 – Режим доступу : http://www.terry.uga.edu/~hwatson/DW_Architecture_Report.pdf.
5. Wayne Eckerson. Four Ways to Build a Data Warehouse [Електронний ресурс] // Wayne Eckerson – TDWI - 2003 – Режим доступу : http://www.tdwi.org/research/display.aspx?ID=6699
6. Douglas Hackney. Architectures and Approaches for Successful Data Warehouses. [Текст] / Douglas Hackney – www.egltd.com/presents/ArchitecturesApproaches.pdf.
7. M. Wu. Research Issues in Data Warehousing. [Текст]. Datenbanksysteme in Büro, Technik und Wissenschaft/ M. Wu, A. P. Buchmann – BTW 1997: p. 61-82.
8. Thanh N. Huynh. Metadata for Object-Relational Data Warehouse. [Текст] Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000) / Thanh N. Huynh, Oscar Mangisengi, A Min Tjoa – 2000.
9. J. M. Firestone. Object-Oriented Data Warehousing [Текст].Executive Information Systems, Inc., white paper No. 5 / J. M. Firestone – 1997.
10. Ken Orr. Data Warehousing Technology. A white paper [Текст] / Ken Orr. The Ken Orr Institute , 1996.
11. Томашевський В.М. Математична модель задачі проектування гібридних сховищ даних з врахуванням структур джерел даних [Текст]. Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. /Томашевський В.М., Яцишин А.Ю. – К.: Век+, – 2011. – № 53. – 211 с.
12. Serge Abiteboul – Querying Semi-Structured Data [Текст]. ICDT '97 Proceedings of the 6th International Conference on Database Theory / Serge Abiteboul – 1997.
13. S. Bergamaschi. Semantic Integration of Semistructured and Structured Data Sources. [Текст]. ACM SIGMOD Record – Volume 28 Issue 1, March 1999 /. S. Bergamaschi, S. Castano, M. Vincini – 1999.
14. Yasser Abdel Kader. A Performance Analysis of a Hybrid Relational-XML Approach to Store Partially-structured Data [Текст]/. Yasser Abdel Kader – University of Sheffield, Department of Computer Science – 2007.
15. Tirthankar Lahiri.– Ozone: Integrating Structured and Semistructured Data [Текст] DBPL '99 Revised Papers from the 7th International Workshop on Database Programming Languages: Research Issues in Structured and Semistructured Database Programming/ Tirthankar Lahiri, Serge Abiteboul, Jennifer Widom – London – 2000.
16. Songyun Duan. Apples and oranges: a comparison of RDF benchmarks and real RDF datasets [Текст]. SIGMOD '11 Proceedings of the 2011 international conference on Management of data / Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, Octavian Udrea – New York – 2011.
17. Пирогов В.В. Исследование применимости генетических алгоритмов в автоматизированном проектировании вычислительных сетей и в задачах размещения : дис. канд. техн. наук: 05.13.12 / Пирогов Владимир Витальевич – Ульяновск, 2001 – 199 с.

УДК 681.327.12.001.362

ТЕЛЕНИК С.Ф.,
ДОРОГИЙ Я.Ю.

ИССЛЕДОВАНИЕ ПАРАМЕТРОВ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ ДЛЯ ЗАДАЧИ РАСПОЗНАВАНИЯ ЧЕЛОВЕКА ЗА ФОТОПОРТРЕТОМ

В статье рассмотрен вопрос влияния параметров сверточной нейронной сети для задачи распознавания человека за фотопортретом. В ходе исследования найден оптимальный набор параметров для решения задачи распознавания человека на базе ORL Faces.

In the article the question of influence the parameters of the convolution neural network for pattern recognition problem in human portraits was considered. The study found the optimal set of parameters to solve the problem of recognizing a person on ORL Faces dataset.

Введение в проблему

В настоящее время обнаружение человеческого лица на изображении или в видеопотоке становится очень важной темой для научных исследований из-за широкого диапазона ее возможных приложений, таких как управление контролем доступа в системах безопасности, человеко-машинное взаимодействие и многих других. Также, обнаружение лица на изображении является первым шагом в процессе решения задачи идентификации личности человека по изображению лица или распознавания эмоционального выражения лица.

Сейчас для решения данной задачи начинают интенсивно применяться искусственные нейронные сети. Искусственные нейронные сети давно и успешно применяются для решения многих задач распознавания. Достоинством использования нейросетей для решения задачи обнаружения/распознавания лица является возможность получения классификатора, хорошо моделирующего сложную функцию распределения изображений лиц. Недостатком же является необходимость в тщательной и кропотливой настройке нейросети для получения удовлетворительного результата классификации.

В последнее время основной топологией нейронной сети используемой для решения задачи локализации лица на изображении является полносвязная нейронная сеть без обратных связей, так называемый многослойный персептрон. Но его применение к решению данной задачи представляет несколько трудностей:

- во-первых, как правило, изображения имеют большую размерность, соответственно вырастает размер нейронной сети (количество нейронов и весов). Большое количество параметров увеличивает вместимость системы и соответственно требует большей обучающей выборки, что увеличивает время и вычислительную сложность процесса обучения;
- во-вторых, недостаток полносвязной архитектуры в том, что топология ввода полностью игнорируется. Входные переменные могут быть представлены в любом порядке, не затрагивая цель обучения. Напротив, изображения имеют строгую 2-мерную местную структуру: переменные (пиксели), которые являются пространственно соседними, чрезвычайно зависимы.

На преодоление этих недостатков направлена архитектура сверточных нейронных сетей.

Статья является продолжением исследований сверточных нейронных сетей, впервые описанных в [1,2].

Анализ существующих решений

В 1981 году нейробиологи Торстен Визел и Девид Хабел исследовали зрительную кору головного мозга кошки и выявили, что существуют так называемые простые клетки, которые особенно сильно реагируют на прямые линии под разными углами и сложные клетки, что реагируют на движение линий в одном направлении.

Позднее Ян Лекун предложил использовать так называемые сверточные нейронные сети, как аналог зрительной коры головного мозга для распознавания изображений [1,2].

Данный тип сетей хорошо зарекомендовал себя для решения проблемы распознавания человека по фотопортрету, но задача выбора оптимальных параметров сети является открытой и плохо описанной в литературе.

Цель работы

Ввиду неочевидности работы сверточных нейронных сетей, а также множества факторов, которые влияют на скорость ее обучения (количество карт признаков, плотность связей между картами признаков, размер окна, площадь перекрытия, начальная инициализация весов и прочие), очень сложно заранее предугадать, какие значения будут оптимальными для конкретной практической задачи.

В данной работе рассмотрен ряд экспериментов, в ходе которых были подобраны оптимальные параметры для решения задачи распознания человеческих лиц с помощью сверточной нейронной сети на базе изображений ORL Faces.

Архитектура классической сверточной нейронной сети

Идея сверточных нейронных сетей заключается в чередовании сверточных слоев (C-layers), субдискретизирующих слоев (S-layers) и наличии полносвязных (F-layers) слоев на выходе.

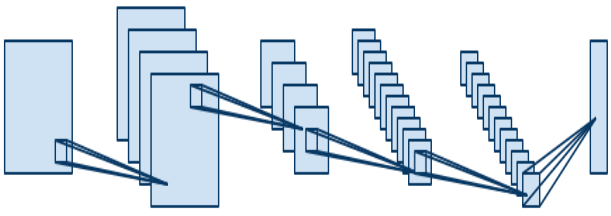


Рис. 1. Структура СНС

Такая архитектура заключает в себе 3 основных парадигмы:

- локальное восприятие;
- разделяемые веса;
- субдискретизация.

Локальное восприятие подразумевает, что на вход одного нейрона подается не всё изображение (или выходы предыдущего слоя), а лишь некоторая его область. Такой подход позволяет сохранять топологию изображения от слоя к слою.

Концепция разделяемых весов предполагает, что для большого количества связей используется небольшой набор весов. Т.е. если у нас имеется на входе изображение 32x32 пикселя, то каждый из нейронов следующего слоя примет на вход только небольшой участок этого изображения размером, к примеру, 5x5, причем каждый из фрагментов будет обработан одним и тем же набором. Важно понимать, что самих наборов весов может быть много, но каждый из них будет применен ко всему изображению. Такие наборы часто называют ядрами (kernels).

Большинство систем распознавания изображений строятся на основе двумерных фильтров. Фильтр представляет собой матрицу коэффициентов, обычно заданную вручную. Эта матрица применяется к изображению с помощью математической операции, называемой сверткой. Суть этой операции в том, что каждый фрагмент изображения умножается на матрицу (ядро) свертки поэлементно и результат суммируется и записывается в аналогичную позицию выходного изображения. Основное свойство таких фильтров заключается в том, что значение их выхода тем больше, чем больше фрагмент изображения похож на сам фильтр. Таким образом, изображение, свернутое с неким ядром даст нам другое изображение, каждый пиксель которого будет означать степень похожести фрагмента изображения на фильтр.

Разберем более подробно процесс распространения сигнала в С-слое.

Каждый фрагмент изображения поэлементно умножается на небольшую матрицу весов (ядро), результат суммируется. Эта сумма является пикселем выходного изображения, которые и формируют карту признаков. Следует сказать, что в идеале не разные фрагменты проходят последовательно через ядро, а параллельно всё изображение проходит через идентичные ядра. Кроме того, количество ядер (наборов весов) определяется разработчиком и зависит от того какое количество признаков необходимо выделить. Еще одна особенность сверточного слоя в том, что он немного уменьшает изображение за счет краевых эффектов.

Суть субдискретизации и S-слоев заключается в уменьшении пространственной размерности

изображения. Т.е. входное изображение грубо (усреднением) уменьшается в заданное количество раз. Чаще всего в 2 раза, хотя может быть и не равномерное изменение, например, в 2 раза по вертикали и в 3 раза по горизонтали.

Использование субдискретизации необходимо для обеспечения инвариантности к масштабу.

Чередование слоев позволяет составлять карты признаков из карт признаков, что на практике означает способность распознавания сложных иерархий признаков.

Обычно после прохождения нескольких слоев карта признаков вырождается в вектор или даже скаляр, но таких карт признаков становится сотни. В таком виде они подаются на один-два слоя полносвязной сети. Выходной слой такой сети может иметь различные функции активации. В простейшем случае это может быть тангенциальная функция, также успешно используются радиальные базисные функции.

Конфигурация пакета PANN для работы со сверточными нейронными сетями

Рассмотрим конфигурацию необходимую для эмуляции сверточных сетей. Как описывалось в [3,4], конфигурация задается в XML-файле:

```
<config>
  <net>
    <layer>5</layer>
    <layer>6</layer>
    <layer>7</layer>
    <density>0.2</density>
    <window_height>5</window_height>
    <window_width>5</window_width>
<window_vert_overlap>3</window_vert_overlap>
<win-
dow_horiz_overlap>3</window_horiz_overlap>
    <threads>8</threads>
    <seed>42</seed>
  </net>
  <weight_randomization>
    <min>-0.1</min>
    <max>+0.1</max>
  </weight_randomization>
  <lms>
    <learning_rate>0.1</learning_rate>
    <annealing_tsc>100</annealing_tsc>
    <epochs>200</epochs>
  </lms>
  <faces>
    <men>5</men>
    <train_percent>60</train_percent>
    <report_frequency>1</report_frequency>
    <stop_error>0.5</stop_error>
  </faces>
</config>
```

Рассмотрим параметры более детально:

- layer – количество карт признаков в паре слоев подвыборки-свертки.
- density – плотность связей между картами признаков слоя свертки предыдущей пары слоев и карт признаков слоя подвыборки следующей пары слоев. 0 – нет связей, 1.0 – полносвязная связь.
- window_height/width – размер окна подвыборки.
- window_horiz/vert_overlap – 3/3 – площадь перекрытия соседних окон.
- threads – количество одновременных потоков.

  seed – база для генератора псевдослучайных чисел. Этим параметром инициализируется генератор псевдо-случайных чисел. Таким образом, при каждом запуске сеть будет иметь одно и тоже начальное состояние. Это избавляет от внесения нежелательных флуктуаций в эксперимент и позволяет анализировать качество обучения при изменении различных параметров.
- weight_randomization – разброс весов.
- LMS – алгоритм обучения по методу градиентного спуска с мерой ошибки по методу наименьших квадратов.
- learning_rate – скорость обучения.
- annealing_tsc – временная константа для обучения с моделированием отжига. Моделирование отжига при изменении веса во время обучения учитывает еще и величину изменения во время предыдущего прогона. Т.е. отжиг как бы ни дает весу резко поменять направление изменения веса в пространстве весов.
- train_percent – размер обучающего множества в процентах.
- report_frequency – частота отклика.

– stop_error – общая ошибка системы, которая считается приемлемой и при ее достижении обучение останавливается.

Эксперименты и результаты

В данном исследовании использовался ранее разработанный пакет PANN [3,4]. Эксперименты проводились на общепринятом наборе лиц для тестирования систем распознавания человека Olivetti Research Lab's (ORL) Face Database. В ней имеются по 10 фотографий 40 различных людей.

Рис. 2. Лица из базы ORL Face

Проведя предварительный анализ сверточных нейронных сетей, а также изучив результаты исследований [2] был выбран ряд параметров:

$$layer \in (3,5,6)$$
$$density \in (0.2,0.5,0.7)$$
$$window \in (3,5,10) \qquad overlap \in (1,3,5)$$
$$randomization \in (0.01,0.1,0.3)$$
$$rate \in (0.01,0.3,0.5)$$

Итого было произведено $3^6 = 729$ обучений нейронной сети. Как было сказано ранее, использовалась база для инициализации генератора псевдослучайных чисел. Это позволяет гарантировать, что полученные результаты зависят только от изменения перечисленных выше параметров.

Первые же результаты экспериментов однозначно показали, что размер окна и перекрытия окон связаны и не имеет смысла разделять эти параметры. Было определено, что оптимальным перекрытием будет 50-60% от размера окна. В первом приближении было решено взять заведомо завышенные параметры нейронной сети:

$$layer = 6$$
$$density = 0.7$$
$$window = 10$$
$$overlap = 5$$
$$randomization \in = 0.3$$
$$rate = 0.5$$

На рис. 3 изображен график значения ошибки от количества эпох обучения.

Рис. 3. Динамика обучения СНС с завышенными параметрами обучения

Видно, что сеть совершенно не поддается обучению. За 200 эпох качество распознавания осталось практически на начальном уровне. Более того, после 100 эпох сеть неожиданно пошла в разнос.

Путем подбора количества слоев удалось достичь лучших результатов (рис. 4):

$$layer = 5$$
$$density = 0.7$$
$$window = 10$$
$$overlap = 5$$
$$randomization \in = 0.3$$
$$rate = 0.5$$

Рис. 4. Динамика обучения СНС с оптимальным количеством слоев

Зафиксировав количество слоев на 5, удалось получить хоть и плохой, но более-менее стабильный результат. Сеть однозначно научилась выделять определенные признаки, но все еще не достигает желаемого качества распознавания за при-

емлемое время. Следующим шагом было определение оптимального размера окна и перекрытия окон:

$$layer = 5$$
$$density = 0.7$$
$$window = 5$$
$$overlap = 3$$
$$randomization \in = 0.3$$
$$rate = 0.5$$

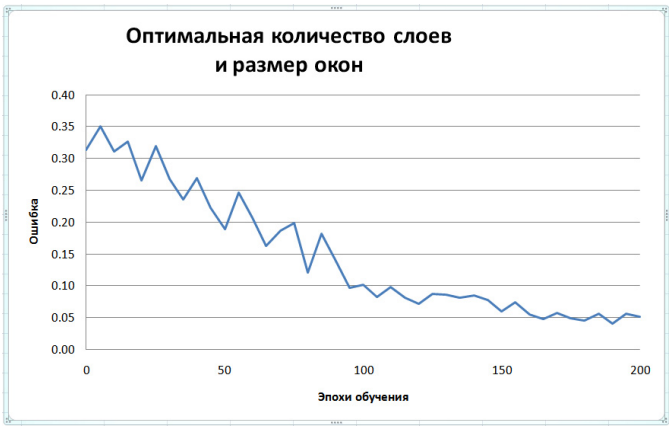


Рис. 5. Динамика обучения СНС с оптимальным количеством слоев и оптимальным размером окна

С такими параметрами нейронная сеть смогла достигнуть неплохих результатов, но все еще заметны скачки (рис. 5). Следующий шаг – оптимизация плотности связей:

$$layer = 5$$
$$density = 0.5$$
$$window = 5$$
$$overlap = 3$$
$$randomization \in = 0.3$$
$$rate = 0.1$$

Рис. 6. Динамика обучения СНС с оптимальным количеством слоев, размером окна и плотностью связей

Теперь сеть ведет себя более предсказуемо, всплески намного уменьшились, но необходимое качество все еще не достигнуто (рис. 6).

Оптимизировав начальную инициализацию весов, удалось уменьшить скорость достижения хорошего распознавания (рис. 7):

$$layer = 5$$
$$density = 0.5$$
$$window = 5$$
$$overlap = 3$$
$$randomization \in = 0.1$$
$$rate = 0.1$$

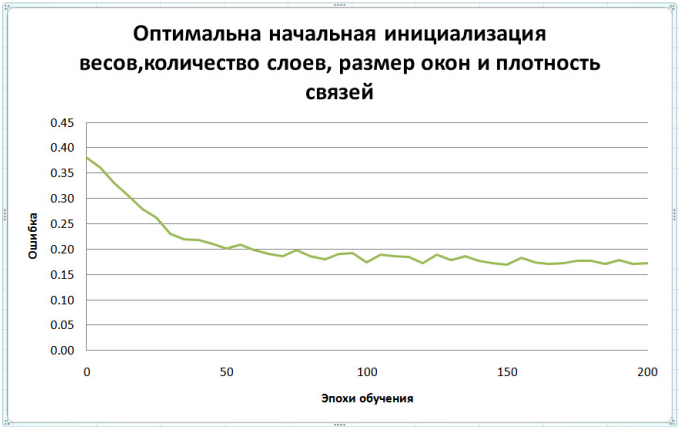


Рис. 7. Динамика обучения СНС с оптимальным количеством слоев, размером окна, плотностью связей и инициализацией весов

Кривая ошибки сети достаточно быстро выходит на прямой участок, что свидетельствует о быстром нахождении оптимальных признаков. Но как видно из рис. 7 скорость обучения недостаточна.

Увеличим скорость обучения:

$$layer = 5$$
$$density = 0.5$$
$$window = 5$$
$$overlap = 3$$
$$randomization \in = 0.1$$
$$rate = 0.3$$

Как видно из рис. 8, сеть менее чем за 50 эпох выходит на допустимое качество, а за 200 эпох обучения ошибка уменьшается до 0.01-0.02.

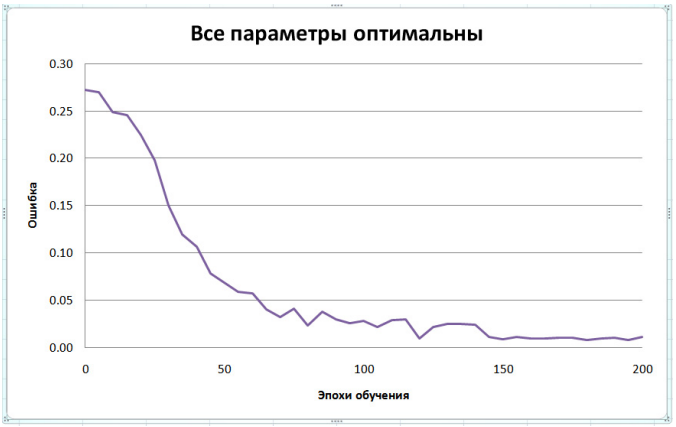


Рис. 8. Динамика обучения СНС с оптимальными параметрами обучения

Выводы

В ходе исследования был проведен ряд экспериментов для определения наиболее оптимальных параметров обучения сверточной нейронной сети.

Анализ результатов показал, что наиболее оптимальными будут такие значения параметров:

$$layer = 5$$
$$density = 0.5$$
$$window = 5$$
$$overlap = 3$$
$$randomization \in = 0.1$$
$$rate = 0.3$$

Количество слоев влияет на качество распознавания образов. Чем больше слоев – тем больше связей и тем дольше проходит обучение, что не приемлемо.

Плотность связей в значительной мере влияет на возможность внесения в нейронную сеть необходимого количества признаков. Слишком слабосвязная сеть не может запомнить необходимые признаки. Слишком сильносвязная – наоборот пытается запомнить слишком много признаков и часто выдает ошибки, останавливаясь в области локального минимума.

Как было сказано ранее, размер окна подвыборки и области перекрытия должны соответствовать друг другу. Слишком большое перекрытие окон вносить много лишней информации в сеть. Для картинок из базы ORL Face (95x95 пикселей), оптимальный размер окна 5x5 пикселей с перекрытием в 3 пикселя со всех сторон.

Начальная инициализация позволяет быстро достигнуть стабильного распознавания.

Большая скорость обучения позволяет ускорить процесс обучения, но также, из-за больших значений сеть может пойти в разнос. Оптимальным значением будет 0.3.

Список литературы

1. LeCun Y. A theoretical framework for backpropagation // Proc. of IEEE. – 1998. – P.21-28.
2. LeCun Y., Bottou L., Bengio Y., Haffne P. Gradient-Based Learning Applied to Document Recognition // Proc. IEEE. – 1998. – P.59-67.
3. Дорогой Я.Ю., Яшин В.Е. Программный комплекс для симуляции многопоточных нейронных сетей // Вісник НТУУ “КПІ”, “Інформатика, управління та обчислювальна техніка”. – 2008. – №49. – С.123-127.
4. Дорогий Я.Ю., Яшин В. Є., Яцук С. В. Застосування багатопотокового симулятора нейронних мереж до задачі розпізнавання облич // Тези 5-ї Міжнародної науково-технічної конференції «Інформаційно-комп’ютерні технології 2010». – Житомир: ЖДТУ, 20-22 травня 2010 року. – С.53-55.

УДК 62-50

ПАВЛОВ А.А.,
КАЛАШНИК В.В.,
ХАЛИМОН А.Ю.

ПОСТРОЕНИЕ ОДНОМЕРНОЙ НЕЛИНЕЙНОЙ РЕГРЕССИИ НА ОСНОВЕ СПЛАЙН-ТЕХНОЛОГИИ И НОРМИРОВАННЫХ ОРТОГОНАЛЬНЫХ ПОЛИНОМОВ ФОРСАЙТА

В статье приводится метод восстановления одномерной нелинейной регрессии на заданном отрезке изменения значений аргумента, с произвольно распределенной аддитивной помехой, верхняя оценка дисперсии которой считается известной. В качестве сплайн-функций выбираются степенные полиномы, коэффициенты которых находятся как с помощью нормированных ортогональных полиномов Форсайта на основе теоретических и экспериментальных исследований, приведенных в [2], так и с помощью специально сконструированных задач линейного программирования. В завершении приводится обоснованный конструктивный критерий, гарантирующий точное решение поставленной задачи.

The article describes method of univariate non-linear regression recovery on given range of argument value variation, with arbitrary distributed additional noise, upper estimate of which is considered to be known. Power polynomials are chosen as spline functions, which coefficients can be found using Forsyth normalized orthogonal polynomial based on theoretical and practical research, given in [2], or using special linear programming models. Also, substantiated constructive criterion, that ensures accurate solution of formulated problem, is given.

Введение

Проблема построения нелинейной регрессии по зашумленным (с произвольным распределением) данным является одной из наиболее практически востребованных задач, которая даже в одномерном случае представляет в настоящее время достаточный теоретический и практический интерес [1,2].

Постановка задачи

Регрессионная модель имеет вид

$$Y(x) = \varphi(x) + \Delta,\ M\Delta = 0, \qquad (1)$$

где $\varphi(x)$ неизвестная (предполагается непрерывная и непрерывно дифференцируемая функция), $\Delta-$ произвольно распределённая случайная величина, верхняя оценка дисперсии которой σ^2 – известна, x – скалярный действительный аргумент функции $\varphi(x)$. Экспериментальные данные представлены в следующем виде: на отрезке [a,b] для значений аргумента x – $a = x_1, x_2, ..., x_n = b$ заданы числа y_i, $i = \overline{1,n}$. $y_i = \varphi(x_i) + \delta_i$, (2) δ_i – реализации случайной величины Δ. Предполагается, что нахождение практически точных значений $\varphi(x)$ $i = \overline{1,n}$ с достаточной для инженерной практики точностью задает функцию $\varphi(x)$ на отрезке [a,b].

Метод решения задачи

Используя сплайн-технологию отрезок [a,b] разбиваем на отрезки B_j, $j = \overline{1,m}$; $B_j = [a_j, b_j]$, $a_1 = a, b_j = a_{j+1}, j = \overline{1,m-1}, b_m = b,$.
$b_j \in \{x_1, ..., x_n\} = I_n$. Каждый отрезок B_j содержит n_j чисел из множества I_n. На B_j отрезке функция $\varphi(x)$ аппроксимируется полиномом степени m_j, $m_j < n_j$.

$$\varphi(x) \approx a_{0j} + a_{1j}x + ... + a_{m_j j}x^{m_j} \quad (3)$$

Для нахождения оценок $a_{ij}, i = \overline{1 \vee 2, m_j}$ используем нормированные ортогональные полиномы Форсайта [2]. Как показано в [2] (гл. 6) для σ^2 не превышающей 10^4 и $a \geq 40$ для гарантировано точной оценки коэффициентов a_{ij}, $i = \overline{2, m_j}$ достаточно взять $n_j \geq 10$ (в предположении что (3) точно описывает $\varphi(x)$ на отрезке B_j). Более того, [2] на примерах показало, что для $n_j = 10$ a_{1j} находится с точностью до второго знака после запятой. Критерием правильности построения отрезков B_j, $j = \overline{1,m}$ является существование натуральных чисел $m_j^1 < m_j$, $j = \overline{1,m}$ для которых оценки $a_{lj}, l = \overline{m_j^1 + 1, m_j}$, $j = \overline{1,m}$ практически являются оценками нулей.

Если это условие не выполняется, необходимо увеличить количество экспериментов на отрезке [a,b].

Таким образом аппроксимирующие полиномы на отрезках B_j, $j=\overline{1,m}$ построены с точностью до значений a_{0j}, либо a_{0j}, a_{1j}, $j=\overline{1,m_j}$.

Потребуем, чтобы в точках $b_j = x_{(j)}$, $j=\overline{1,m-1}$ значения апроксимирующих полиномов и их первых производных совпадали с заданной точностью. Для этого решим следующие задачи линейного программирования.

Модель 1

$$\min \sum_{i=1}^{n} z_i \tag{4}$$

$$-\varepsilon_1 \le \sum_{l=0}^{m_i^1} a_{li} x_{(i)}^l - \sum_{l=0}^{m_{i+1}^1} a_{l(i+1)} x_{(i)}^l \le \varepsilon_1 \tag{5}$$
$$i=\overline{1,m-1}$$

$$-\varepsilon_2 \le \sum_{l=1}^{m_i^1} l a_{li} x_{(i)}^{l-1} - \sum_{l=1}^{m_{i+1}^1} l a_{l(i+1)} x_{(i)}^{l-1} \le \varepsilon_2 \tag{6}$$
$$i=\overline{1,m-1}$$

$$-\phi_0^{-1}\left(\frac{1-\alpha}{2}\right)\frac{\sigma}{\sqrt{n}} \le \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{l=0}^{m_{j_i}^1} a_{lj_i} x_i^l\right) \le \phi_0^{-1}\left(\frac{1-\alpha}{2}\right)\frac{\sigma}{\sqrt{n}} \tag{7}$$

$$-z_i \le y_i - \sum_{l=0}^{m_{j_i}^1} a_{lj_i} x_i^l \le z_i, i=\overline{1,n} \tag{8}$$
$$z_i \ge 0, i=\overline{1,n}$$

где j_i – номер B_j отрезка, j_i равняется j если x_i принадлежит отрезку B_j.

Переменными задачи (4)-(8) являются $a_{0i}, a_{1i}, i=\overline{1,m}, z_i=\overline{1,n}$.

ϕ_0 – функция Лапласа, $\alpha \le 0{,}05$; $\varepsilon_1 \ge 0, \varepsilon_2 \ge 0$ – заданные константы.

Ограничения (7) с вероятностью $1-\alpha$ реализуют проверку статистической гипотезы о том, что полученные по восстановленной регрессии $\hat{\varphi}(x)$ числа $\hat{\delta}_i$ являются реализациями случайной величины Δ, где среднее арифметическое на основании теоремы Муавра-Лапласа имеет нормальное распределение.

Модель 2

Отличается от модели 1 тем, что в ней переменными являются $a_{0i}, i=\overline{1,m}; z_j, j=\overline{1,n}$. Модель 2 используется когда оценка $a_{1i}, i=\overline{1,m}$ с помощью нормированных ортогональных полиномов является достаточно точной.

Модель 3

Отличается от модели 1 тем, что переменными являются $a_{lj}, l=\overline{0,m_j^1}$; $j=\overline{1,m}$; $z_i, i=\overline{1,n}$.

Модель 4

$$\min \sum_{i=1}^{n} z_i \tag{9}$$

$$-\varepsilon_1 \le a_{0i} + a_{1i}x_{(i)} + \sum_{l=2}^{m_i^1}(a_{li}+\hat{a}_{li})x_{(i)}^l - a_{0(i+1)} - a_{1(i+1)}x_{(i)} - \sum_{l=2}^{m_i^1+1}(a_{l(i+1)}+\hat{a}_{l(i+1)})x_{(i)}^l \le \varepsilon_1 \tag{10}$$

$$-\varepsilon_2 \le a_{1i}x_{(i)} + \sum_{l=2}^{m_i^1} l(a_{li}+\hat{a}_{li})x_{(i)}^{l-1} - a_{1(i+1)}x_{(i)} - \sum_{l=2}^{m_i^1} l(a_{l(i+1)}+\hat{a}_{l(i+1)})x_{(i)}^{l-1} \le \varepsilon_2 \tag{11}$$

$$-\Phi^{-1}\left(\frac{1-\alpha}{2}\right)\frac{\sigma}{\sqrt{n}} \le \frac{1}{n}\sum_{i=1}^{n} y_i - a_{0j_i} - a_{1j_i}x_i - \sum_{l=2}^{m_{j_i}^1}(a_{lj_i}+\hat{a}_{lj_i})x_i^l \le \Phi^{-1}\left(\frac{1-\alpha}{2}\right)\frac{\sigma}{\sqrt{n}} \tag{12}$$

$$-z_i \le y_i - a_{0j_i} - a_{1j_i}x_i - \sum_{l=2}^{m_{j_i}^1}(a_{lj_i}+\hat{a}_{lj_i})x_i^l \le z_i, z_i \ge 0, i=\overline{1,n}. \tag{13}$$

$$-\varepsilon_3 \mid a_{lj} \mid \le \hat{a}_{lj} \le \varepsilon_3 \mid a_{lj} \mid, l=2, m_j^1, j=\overline{1,m} \tag{14}$$

Переменными задачи (9)-(14) являются a_{0j}, $a_{1j}, \hat{a}_{lj}, l=2, m_j^1, j=\overline{1,m}$, $z_i, i=\overline{1,n}$; $\varepsilon_1 > 0$, $\varepsilon_2 > 0$, $\varepsilon_3 > 0$ – заданные погрешности.

В результате решения задач линейного программирования (4)-(14) получаем оценки неизвестной линии регрессии $\varphi(x)$, $\hat{\varphi}^l(x), l=\overline{1,4}$. На сегменте $[a,b]$ необходимо обоснованно выбрать наилучшую аппроксимацию. Предлагается сле-

дующая процедура выбора наилучшей аппроксимации из претендентов $\hat{\varphi}^l(x), l=\overline{1,4}$.

По каждой функции $\hat{\varphi}^l(x)$ находим оценки $\hat{\delta}_i^l, i=\overline{1,n}$, реализаций δ_i случайной величины Δ.

Случайным образом, имитирующим независимые испытания над Δ моделируем искусственным образом заданное число экспериментов $y_i^{lj}, x_i,\ i=\overline{1,n},\ \ j=\overline{1,k},\ \ k$ – количество искусственно конструируемых экспериментов $y_i^{lj}=\hat{\varphi}^l(x_i)+\hat{\delta}_i^{lj}, i=\overline{1,n}, j=\overline{1,k}$. Числа $\hat{\delta}_i^{lj}$ каждый раз случайным образом выбираются в каждом j-том ($j=\overline{1,k}$) искусственном эксперименте из множества $\{\hat{\delta}_i^l\}$.

По l-той задаче линейного программирования находим $\hat{\varphi}^{lj}(x), j=\overline{1,k}$, аппроксимирующий $\varphi(x)$ на отрезке $[a,b]$ ($\hat{\varphi}^{lj}(x)$ - j-тая аппроксимация $\varphi(x)$ построенная по j-ому эксперименту с помощью аппроксимации $\hat{\varphi}^l(x)$.

Построим множество чисел $\left\{\left|\hat{\varphi}^l(x_i)-\hat{\varphi}^{lj}(x_i)\right|, i=\overline{1,n}, j=\overline{1,k}\right\}=\gamma_{lji}\left|\hat{\varphi}^l(x_i)\right|$ и пусть γ_l является максимальным из них $\max\gamma_l=\max\limits_{ji}\gamma_{lji}$. Тогда в качестве наилучшей апроксимации выбирается $\hat{\varphi}^l(x)$, на которой достигается $\min\limits_l\gamma_l$

Выводы: Предложенные алгоритмы оценки неизвестной линии регрессии на заданном отрезке $[a,b]$ используют возможности нормированных ортогональных полиномов [2], сплайн-технологию, универсальность моделей линейного программирования. Эффективность метода следует из того, что он включает в себя три контрольные процедуры:

степень каждого сплайн-полинома меньше числа значений аргумента $\varphi(x)$;

каждая задача линейного программирования включает в себя статистический критерий проверки;

статистически является обоснованным, что число $\min\limits_l\gamma_l=\gamma_t$ может быть небольшим только в случае, когда множество $\{\hat{\delta}_i^t\}$ действительно является множеством реализаций случайной величины Δ, а это возможно, когда $\hat{\varphi}^t(x)$ практически точно оценивает $\varphi(x)$ в точках $x_i,\ i=\overline{1,n}$.

Задача имеет такое же решение, если модель представляется в виде

$$Y(t)=\varphi(t)+\Delta(t), \qquad (15)$$

Где t– действительный аргумент, который интерпретируется как время , $\Delta(t)$-стационарный в узком смысле слова случайный процесс, у которого для $\forall n,t_1,...,t_n$, n-мерная функция распределения $F(t_1,...,t_n)=\prod\limits_{j=1}^{n}F(t_j),\ F(t_j)$ – функция распределения случайной величины $\Delta(t_j)$.

Список литературы

1. Норман Р. Дрейпер Прикладной регрессионный анализ // Норман Р. Дрейпер, Гарри Смит. – Москва, Санкт-Петербург, Киев:, 2007. – 911с.
2. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.

УДК 004.4(043.2)

ВИНОГРАДОВ Ю.М.,
АГЄЄНКО Ю.М.

ПРИСКОРЕННЯ ЕКСПОНЕНЦІЮВАННЯ НА ПОЛЯХ ГАЛУА В СИСТЕМАХ ЗАХИСТУ ІНФОРМАЦІЇ

В статті запропоновано новий спосіб прискорення експоненціювання на полях Галуа. В основу способу покладено використання результатів передобчислень при реалізації однієї з двох базових операцій експоненціювання – множення на основу. Детально описано процедури запропонованого способу: формування значень таблиці передобчислень та виконання операцій з ними. Теоретично доведено та практично підтверджено, що розроблений підхід забезпечує прискорення виконання операції експоненціювання на полях Галуа на 30%.

In article the new method of shortcut exponentiation on Galois fields is proposed. The method is based on utilization of precomputations on executing of multiplication by radix – one of two basis exponentiation operations. Procedures of the offered method, such as forming prediction table values and executing operations with them were described in details. It was proved in theory and confirmed by the experimental way that the offered method provides acceleration executing exponentiation operation on Galois fields by thirty per cents.

Вступ

Арифметичні операції, які виконуються на полях Галуа, відіграють важливу роль в сучасних інформаційних технологіях. Зокрема, вони покладені в основу більшості методів виявлення та корекції помилок при передачі й зберіганні даних, широко використовуються при кодовому ущільненні передачі інформації, в системах вимірювання та реєстрації даних.

Особливо важливу роль відіграють обчислення на полях Галуа в сучасних криптографічних механізмах захисту інформації: вони застосовуються в алгоритмі симетричного шифрування Rijndael, який став переможцем всесвітнього конкурсу AES, а також в механізмах асиметричного шифрування і цифрового підпису на основі еліптичних кривих[1]. У криптографії на основі еліптичних кривих базовою операцією є експоненціювання на полях Галуа, яка виконується над числами великої розрядності (512-1024 біт), що значно перевищує довжину слова сучасних процесорів.

Суттєвою складністю реалізації арифметики на полях Галуа є непристосованість до неї архітектури звичайних процесорних засобів, які орієнтовані на двійкову арифметику. Тому існує необхідність в розробці складних програмних засобів, що уповільнює виконання арифметичних операцій на полях Галуа [2].

Аналіз динаміки розвитку прикладних задач, в яких активно використовується арифметика на полях Галуа, показує, що більша їх частина виконується в реальному часі і потребує швидкої реалізації відповідних обчислень. Іншою суттєвою особливістю використання арифметики на полях Галуа на сучасному етапі розвитку інформаційних технологій є збільшення розрядності чисел.

Все це вимагає розробки нових методів організації обчислень на скінченних полях. В першу чергу, це стосується найбільш трудомістких обчислювальних операцій, таких як експоненціювання.

Таким чином, наукова задача прискорення процедури експоненціювання на скінченних полях, що виконується над числами великої розрядності, є актуальною і важливою для сучасного етапу розвитку інформаційних технологій.

Аналіз способів експоненціювання на полях Галуа

Поле Галуа задається утворюючим нерозкладним поліномом $Q(x)$ степені $n+1$, якому співвідноситься $(n+1)$-розрядне двійкове число M. Операція експоненціювання $A|^E$ rem M на такому полі передбачає, що числа A та E являють собою n-розрядні двійкові коди: $A=\{a_0,a_1,...a_{n-1}\}$ і $E=\{e_0,e_1,...,e_{n-1}\}$, $\forall j\in\{0,1,...,n-1\}$, $a_j\in\{0,1\}$, $e_j\in\{0,1\}$, яким відповідають поліноми: $P(A)=a_0 + a_1\cdot x + a_2\cdot x^2 +...+a_{n-1}\cdot x^{n-1}$ та $P(E) = e_0 + e_1\cdot x + e_2\cdot x^2 +...+e_{n-1}\cdot x^{n-1}$.

До теперішнього часу запропоновано ряд способів виконання операції експоненціювання на полях Галуа [2-5]. Їх аналіз показує, що в якості підвищення швидкодії їх автори розглядають можливість розпаралелювання виконан-

ня при апаратній реалізації базової операції експоненціювання – множення на полях Галуа.

Сама процедура модулярного експоненціювання $A|^{E}$ rem M на полях Галуа, як і звичайне модулярне експоненціювання, зводиться до послідовного виконання n циклів, у кожному з яких здійснюється операція піднесення до квадрату отриманого на попередньому циклі результату (R^2) і, додатково, в залежності від поточного біту експоненти E, – операція множення ($R \otimes A$) без переносів. Досліджується модулярне експоненціювання зліва направо, тобто аналіз розрядів експоненти E виконується починаючи зі старших розрядів. Структурно цей спосіб обчислення експоненти $A|^{E}$ rem M на полях Галуа показано на рис. 1.

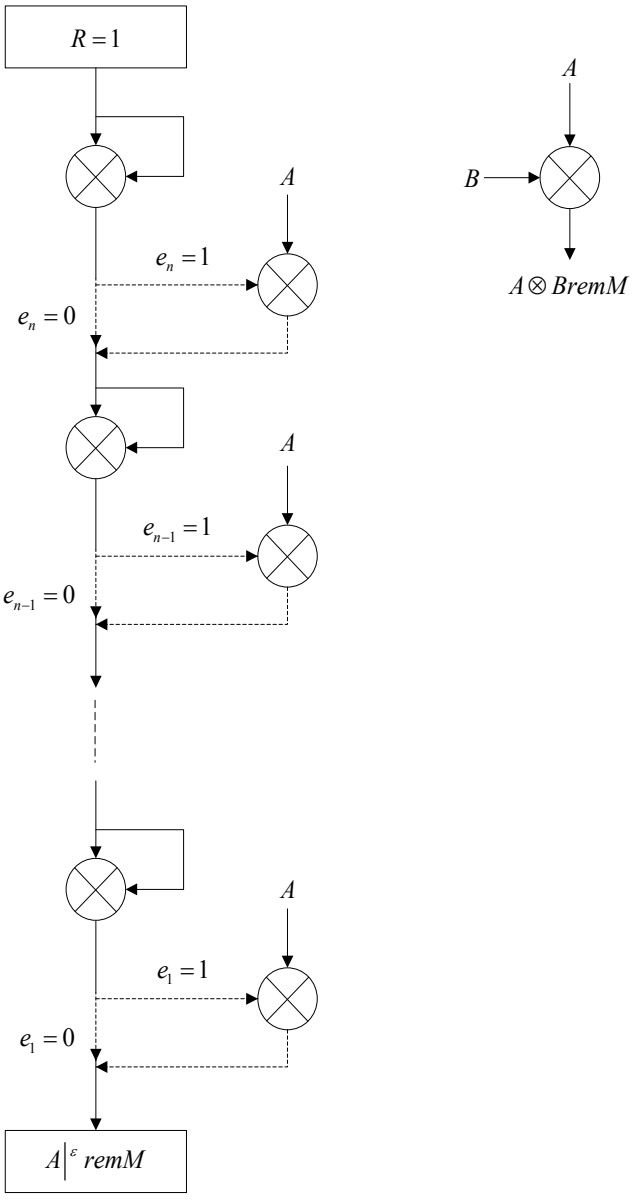


Рис.1. Структура операції експоненціювання

Наприклад, при обчисленні $14|^{10}$ rem 19=6, n=4, A=14_{10}=1110_2; E=10_{10}=1010_2; M=19_{10}=10011_2 динаміка перетворення змінних у відповідності з наведеним на рис.1 алгоритмом представлена в таблиці 1.

Для оцінки швидкодії доцільно провести аналіз виконання процесором базових операцій експоненціювання.

Позначимо за n розрядність числа, а за w – розрядність процесору. На практиці розрядність чисел n значно перевищує розрядність процесору w ($n >> w$), тому обробка виконується w-розрядними фрагментами, кількість яких дорівнює $s = n / w$.

Табл. 1. Цифрова діаграма змінних при виконанні експоненціювання $14|^{10}$ rem 19=6**.**

i	e_i	$R \otimes R$ rem 19	$R \otimes A$ rem 19	R
3	1	1	1110_2	1110_2
2	0	1011_2	-	1011_2
1	1	1001_2	0111_2	0111_2
0	0	0110_2	-	0110_2

Операція множення виконується за n циклів, кожен з яких складається з операції зсуву n-розрядного множеного та, за умови, що поточний розряд множника дорівнює одиниці, – операції логічного додавання.

Для зсуву n-розрядного числа на один розряд необхідно виконати $(s+1)$ процесорних операцій зсуву, причому ця операція виконується в кожному циклі множення, відповідно сумарна кількість процесорних операцій зсуву становить $(s+1) \cdot n$.

Вважаючи, що значення розрядів множника з рівною ймовірністю можуть приймати значення як 0, так і 1, то, в середньому, операція додавання виконується в $n / 2$ циклах. Для логічного додавання n-розрядних чисел слід виконати $(s+1)$ процесорну операцію, так що сумарна кількість операцій при додаванні становить: $(s+1) \cdot (n/2)$.

Отже, загальний час виконання множення без переносів n-розрядних чисел T_{mc} дорівнює:

$$T_{mc} = (s+1) \cdot n \cdot \tau + (s+1) \cdot (n/2) \cdot \tau = 1.5 \cdot n \cdot (s+1) \cdot \tau,$$

де τ − час виконання однієї логічної операції.

Операція множення на полях Галуа передбачає після виконання множення без переносів реалізацію редукції отриманого результату,

тобто приведення його в рамки поля. Редукція виконується шляхом віднаходження залишку від поліноміального ділення $(2\cdot n\text{-}1)$-розрядного результату множення без переносів на $(n+1)$-розрядний код утворюючого поліному M поля. Реалізація редукції передбачає виконання $(n\text{-}1)$ циклів, в кожному з яких відбувається зсув на один розряд $(n+1)$-розрядного коду M та додавання його до поточного залишку в разі, коли його старший розряд дорівнює одиниці. Для зсуву $(n+1)$-розрядного коду M на один розряд потрібно виконати $(s+1)$ процесорних операцій зсуву. Так як ця операція виконується в кожному з $(n\text{-}1)$ циклів редукції, то сумарна кількість процесорних операцій зсуву складає $(s+1)\cdot(n\text{-}1)$. Виходячи з того, що в процесі редукції операція додавання, в середньому, виконується в половині циклів, то середня кількість таких операцій становить $(n-1)/2$. Приймаючи до уваги, що для реалізації цієї операції на w-розрядному процесорі потрібно виконати $(s+1)$ процесорних операцій додавання, середня кількість процесорних операцій для редукції результату множення становить : $(s+1)\cdot(n\text{-}1)/2$. Відповідно, середній час T_R виконання редукції результату множення складає

$$T_R = 1.5\cdot(s+1)\cdot(n\text{-}1)\cdot\tau.$$

Таким чином, загальний час T_{mGF} виконання операції множення на полі Галуа дорівнює:

$$T_{mGF} = T_{mc} + T_R - 3\cdot(n-1)\cdot(s+1)\cdot\tau \approx 3\cdot n\cdot s\cdot\tau \qquad (1)$$

Враховуючи, що значення розрядів експоненти з рівною ймовірністю приймають значення нуля та одиниці, то операція множення на основу виконується, в середньому, в $n/2$ циклах. Відповідно, при звичайному експоненціюванні загальний час T_e експоненціювання становить:

$$T_e = 1.5\cdot n\cdot T_{mGF} \approx 4.5\cdot n^2\cdot s\cdot\tau \qquad (2)$$

Експоненціювання з використанням передобчислень

Проведений аналіз операції експоненціювання показав, що операція множення на постійне число повторюється, в середньому, $n/2$ о раз. Тому доцільно виконати передобчислення, результати яких будуть використовуватися на всіх етапах обчислення і, тим самим, прискорити процес експоненціювання.

Зокрема, пропонується використання результатів передобчислень при виконанні множення проміжного результату на код основи A. Результати передобчислень зберігаються в таблиці, формування якої виконується перед обчисленням $A|^E$ rem M шляхом здійснення редукції зсунутих значень A: T[0]=A, T[1]=$A\cdot 2$ rem M, T[3]=$A\cdot 2^2$ rem M, ..., T[n-1]=$A\cdot 2^{n-1}$ rem M.

Відповідно, для множення n-розрядного проміжного результату $R=\{r_0,r_1,...,r_{n-1}\}$, $\forall j\in\{0,1,...,n\text{-}1\}$ $r_j\in\{0,1\}$, на код A в процесі експоненціювання на полях Галуа пропонується наступний алгоритм:

1. $s=0$; $i=0$.
2. Якщо $r_i=1$, то $s=s\oplus$ T[i].
3. $i=i+1$
4. Якщо $i<n$, повернення на пп.2.

Запропонований спосіб ілюструється в рамках наведеного вище прикладу експоненціювання $14|^{10}$ rem 19, (n=4, $A=14_{10}=1110_2$; $M=19_{10}=10011_2$).

У відповідності із запропонованим способом перед процедурою експоненціювання виконується заповнення таблиці результатів передобчислень. T[0]=A=1110_2; T[1]=$A\cdot 2$ rem M = $11100_2\oplus 10011_2 = 1111_2$; T[2] = $A\cdot 2^2$ rem M = T[1]$\cdot$2 rem M = $11110_2\oplus 10011_2 = 1101_2$; T[3]= $A\cdot 2^3$ rem M = T[2]$\cdot$2 rem M = $11010_2\oplus 10011_2 = 1001_2$.

На третьому кроці експоненцювання для $R=\{1,0,0,1\}$ виконується множення $R\otimes A$ rem 19 = $1001_2\otimes 1110_2$ rem 19 = 126 rem 19 = 7=0111_2 (третій рядок таблиці 1). При виконанні цієї операції з використанням передобчислень у відповідності з запропонованим алгоритмом множення на число A реалізується у вигляді: R=T[0]$\oplus$T[3] = $1110_2\oplus 1001_2 = 0111_2$.

Для формування n-1значень таблиці передобчислень для n-розрядного числа A необхідно виконати n-1 циклів. На кожному з циклів треба робити зсув попередньо отриманого результату і, при необхідності, приведення його в рамки поля, тобто редукцію. Операція зсуву n-розрядного числа на w-розрядному процесорі займає s+1 процесорних операцій. Редукція $(n+1)$-розрядного результату зсуву полягає в додаванні до цього $(n+1)$-розрядного коду утворюючого поліному P за умови, що старший розряд результату зсуву дорівнює одиниці. Оскільки цей розряд з рівною ймовірністю приймає

одиничне і нульове значення, редукція проводиться, в середньому, на половині циклі заповнення таблиці. Приймаючи до уваги, що додавання (n+1)-розрядних чисел потребує s+1 процесорних операцій логічного додавання, то середній час T_T формування таблиці передобчислень становить:

$$T_T = 1.5 \cdot (s+1) \cdot (n-1) \cdot \tau \approx 1.5 \cdot s \cdot n \cdot \tau \qquad (3)$$

При експоненціюванні на полях Галуа з використанням таблиць передобчислень вся процедура експоненціювання залишається такою ж, а змінюється лише множення на основу. Тому доцільно провести аналіз виконання даної операції процесором. Для початку, слід виконувати множення основи на проміжний результат, адже за такої умови множене залишається незмінним і значення таблиць вибираються відповідно до значень розрядів проміжного результату.

При множенні n-розрядної основи на проміжний результат з використанням результатів передобчислень виконується n циклів. В кожному з них виконується операція логічного додавання n-розрядного коду з таблиці. Так як коди таблиці n-розрядні, то їх додавання складається з s процесорних операцій. В результаті логічного додавання розрядність результату не змінюється, а відповідно, редукцію результату проводити не потрібно.

Таким чином, виконання процесором операції множення основи на проміжний результат з використанням таблиць передобчислень займає час T_{TGF}, рівний:

$$T_{TGF} = n \cdot s \cdot \tau. \qquad (4)$$

Порівнюючи (4) та (1) можна зробити висновок, що використання передобчислень дозволяє скоротити час виконання операції множення на полях Галуа не менше, ніж в 3 рази.

Оцінка часу експоненціювання

При експоненціюванні n-розрядного числа A на полях Галуа виконується n кроків, відповідно до розрядів експоненти, на кожному з яких здійснюється піднесення до квадрату, та, за умови, що поточний розряд n-розрядної експоненти E дорівнює одиниці, – множення його на основу A.

Якщо виконувати експоненціювання запропонованим способом, тобто з використанням таблиць передобчислень, то необхідно сформувати таблиці, а потім в n циклах виконується піднесення проміжного результату до квадрату і в $n/2$ циклах – множення на основу. Відповідно, час, який витрачається на ці операції:

$$T_{eT} = n \cdot T_{mGF} + 0.5 \cdot n \cdot T_{TGF} + T_T = 3 \cdot n^2 \cdot s \cdot \tau + \\ + 0.5 \cdot n^2 \cdot s \cdot \tau + 1.5 \cdot s \cdot n \cdot \tau \qquad (5)$$

Час, який витрачається на піднесення проміжного результату до квадрату та множення на основу, пропорційний n^2, а час формування таблиць згідно (3) пропорційний до n. Тому, враховуючи, що розрядність чисел $n > 1000$, часом формування таблиць можна знехтувати.

Таким чином, загальний час експоненціювання на полях Галуа з використанням передобчислень становить:

$$T_{eT} = n \cdot T_{mGF} + 0.5 \cdot n \cdot T_{TGF} \approx 3.5 \cdot n^2 \cdot s \cdot \tau \qquad (6)$$

Співвідношення часу класичної процедури експоненціювання, що визначається (2) та часу експоненціювання за запропонованим способом, що визначається (6) становить:

$$T_e / T_{eT} = 4.5 / 3.5 \approx 1.3$$

Отже, використання при експоненціюванні на полях Галуа результатів передобчислень дозволяє скоротити час виконання цієї операції в рази, тобто приблизно на 30%.

Висновки

В результаті проведених досліджень розроблено новий спосіб прискорення операції експоненціювання на полях Галуа з використанням таблиць передобчислень при виконанні однієї з двох базових операцій – множення на основу.

За рахунок того, що виконання множення на основу прискорюється втричі у порівнянні зі класичною процедурою множення на полях Галуа, розроблений підхід дозволяє скоротити час експоненціювання на 30% порівняно зі звичайною реалізацією експоненціювання.

Список літератури

1. Menezes A.J., Blake I.F., Gao S., Mullin R.C., Vanstone S.A., Yactobian T. Application os Finite Fields.//N.Y. Knuwer Academic Published.-1993.-387p.
2. Стефанская В.А., Мухаммад Мефлех Алиса Абабне, Левчун Д.Ю. К проблеме повышения эффективности аппаратной реализации мультипликативных операций на полях Галуа // Вісник Національного технічного університету України «КПІ». Інформатика, управління та обчислювальна техніка. К., «ВЕК++», -2005.-№43.-с.104-112.
3. Popovici E.M., Fitzpatrick P. Algorithm and Architecture for a Galois Field Multiplicative Arithmetic Processor.// IEEE Transaction on Information theory. Vol. 49.– № 12,-2003.-pp.3303-3307.
4. Wu H., Hasan M.A., Blake I.F., Gao S. Finite field multiplier using redundant representation.// IEEE Trans. Computers, Vol. 51.– № 51, – 2002. – pp.1306-1316.
5. Марковский А.П., Шаршаков А.С. Способ ускоренной реализации экспоненцирования на полях Галуа в системах защиты информации // Збірник наукових праць НАУ. Проблеми інформатизації та управління – 2011, №1(32) – 188с.

УДК 004.2

ЖАБИН В.И.,
ЖАБИНА В.В.,
БЕЗГИНСКИЙ М.А.

ЭФФЕКТИВНОСТЬ РЕАЛИЗАЦИИ ПОТОКОВЫХ ВЫЧИСЛЕНИЙ В СИСТЕМАХ С НЕПОСРЕДСТВЕННЫМИ СВЯЗЯМИ НА ПЛИС

Рассматривается возможность уменьшения необходимого ресурса ПЛИС для реализации потоковых вычислительных систем с непосредственными связями между вычислительными модулями. Показано, что применение вычислительных модулей, обрабатывающих операнды поразрядно с использованием избыточных систем счисления, позволяет совмещать выполнение зависимых по данным операции и сократить число связей между модулями. По сравнению с применением параллельных устройств уменьшается требуемое число функциональных ячеек и выводов ПЛИС при сохранении высокой скорости обработки данных. Показаны примеры погружения в ПЛИС вычислителей полиномов, исследованы их параметры.

Possibility of reduction of the necessary FPGA resource for realization of data flow computing systems with direct connections between computing modules is considered. It is shown that the use of the computing modules, which process operands digit-by-digit with the use of redundant numerical system, enables to perform data flow calculations as well as to reduce the number of connections between computing modules. In comparison with the use of parallel devices the demanded number of FPGA functional cells and pins can be reduced with data processing high speed maintenance. There are shown examples of polynomial calculator immersion into FPGA and investigated its parameters.

Введение

Эффективность параллельных вычислений во многом зависит от реализуемого уровня параллелизма, что определяется зернистостью представления графа задачи. В системах реального времени многие алгоритмы обработки данных имеют мелкозернистую структуру. Например, при решении траекторных задач необходимо обеспечить высокую скорость интерполяции функций различными методами (полиномиальной аппроксимации, разложением в цепную дробь, табличко-алгоритмическими, итерационными и т.д.). Возникает необходимость решения систем алгебраических и дифференциальных уравнений [1]. Наиболее высокая степень распараллеливания может быть достигнута, когда вершинам графа соответствуют отдельные операции. При этом максимально увеличивается число параллельных ветвей, что дает потенциальную возможность использовать большее число параллельно работающих вычислительных модулей (ВМ).

Как известно, скорость обработки данных связана не только с длительностью выполнения операций, но и с затратами времени на обмен информацией между параллельными ветвями. Увеличение степени распараллеливания вычислений сопряжено с ростом интенсивности обмена данными между ВМ. Этот фактор является весьма важным и должен учитываться при выборе архитектуры систем и организации вычислений.

В работе рассматривается возможность уменьшения затрат времени на обмен данными за счет использования потоковых систем с непосредственными связями (ПСНС) между ВМ. В ПСНС выходы одних ВМ подключаются к входам других ВМ в соответствии с графом потока данных (ГПД). В процессе вычислений данные пересылаются от одних ВМ к другим, преобразуясь на каждом шаге в соответствии с операцией, заданной ГПД. При такой организации вычислительного процесса не требуются сложные процедуры пересылки данных между ВМ, что создает предпосылки к уменьшению непроизводительных затрат времени в процессе обработки информации.

При использовании ПЛИС для построения систем, кроме ускорения вычислений, важной задачей является сокращение требуемого ресурса микросхем. Это позволяет улучшить ряд характеристик систем, в том числе, повысить надежность и уменьшить энергопотребление. В работе исследуется возможность решения данной задачи путем сокращения количества связей между ВМ за счет поразрядной передачи данных.

Организация систем с непосредственными связями

Обобщенную функциональную модель систем типа ПСНС можно представить в виде

$$S = \langle DI, DO, M, F, C, T, R, G \rangle,$$

где DI – множество входных данных; DO – множество результатов; M – множество вычислительных модулей; F – система функций преобразования данных; C – характеристика средств коммутации; T – требуемые характеристики системы; R – ограничения, накладываемые на возможности реализации аппаратных средств; G – ограничения, накладываемые на форму представления данных.

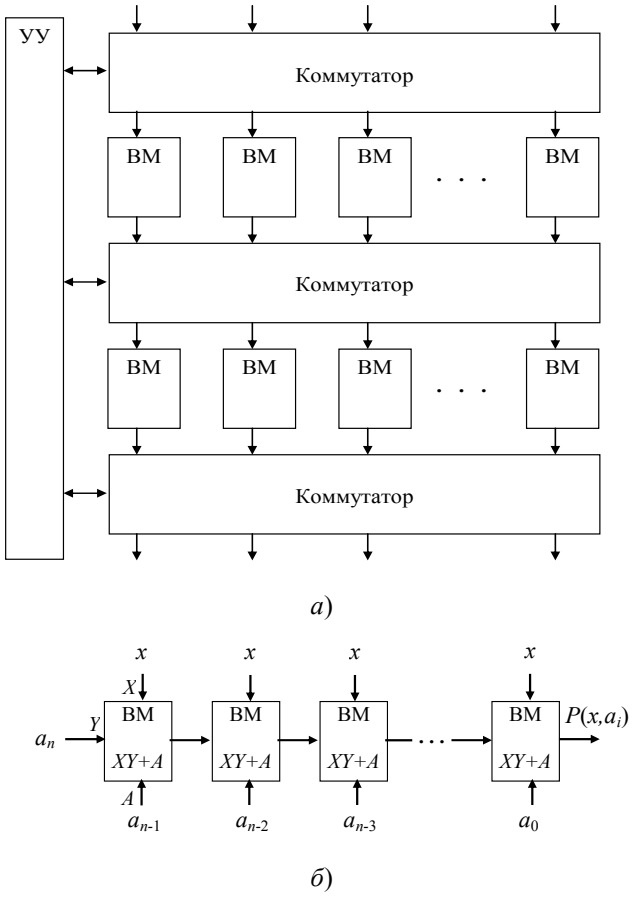


Рис. 1. Системы с непосредственными связями между ВМ: а – с перестраиваемыми связями; б – с жесткими связями (для вычисления полиномов $P(x,a_i)$ по схеме Горнера); УУ – устройство управления

Система функций F описывается в виде алгоритмов выполнения операций в вычислительных модулях. Характеристика C определяет возможности пространственного взаимодействия элементов системы. Требования T к характеристикам системы определяются внешними факторами. Обычно они связаны с целевой функцией системы. Ограничения R в основном определяются требованиями и возможностями элементной базы. Ограничения G в основном связаны с диапазоном представления данных, точностью получения результата и т.д.

В реконфигурируемых ПСНС (рис. 1*а*) необходимое соединение между ВМ обеспечивает коммутационная среда, которая настраивается в соответствии с ГПД. Проблемам реконфигурации систем посвящено много научных работ, в том числе, монографий, например, [2, 3].

Выполнение операций в неавтономном режиме

Использование современной технологии проектирования SoC (System on Chip – система на кристалле) позволяет создавать сложные системы на основе ПЛИС. Однако со стороны элементной базы накладывается ряд ограничений, связанных с числом выводов микросхем, наличием встроенных функциональных узлов и устройств.

При использовании ПЛИС ее ресурсов может не хватить для погружения всей системы. При этом возникает необходимость применения нескольких корпусов микросхем, связанных между собою внешними линиями связи. При параллельной передаче информации между микросхемами возникают проблемы, связанные с возможной нехваткой выводов ПЛИС. Кроме того, снижается надежность системы, так как контактные соединения между микросхемами относятся к ненадежным элементам системы. Возрастает также энергопотребление системы и увеличиваются габаритные размеры.

Учитывая важность указанных проблем, компания Virtual Machine Works разработала технологию под названием VirtualWire (виртуальные соединения), представленную как технология производства больших устройств, для реализации которых приходится использовать несколько микросхем [5]. Поскольку определенное количество внутренних ресурсов не используется в связи с отсутствием для дополнительной аппаратуры выводов микросхем, этот ресурс можно использовать для реализации последовательной передачи разрядов данных с помощью мультиплексирования. Данная технология, хотя и может в определенной степени решить проблему нехватки

выводов микросхем, но создает задержки продвижения потоков данных, что противоречит самой идее потоковой модели вычислений.

Одним из эффективных подходов к решению проблемы сокращения связей между компонентами системы является применение неавтономных методов выполнения операций, основанных на поразрядной передаче информации между ВМ. При таком режиме обработки данных, кроме сокращения числа связей, появляется возможность выполнять зависимые по данным операции в режиме частичного совмещения. При определенных условиях такой режим вычислений создает предпосылки к сокращению времени выполнения зависимых по данным операций.

В ГПД с последовательной и последовательно-параллельной структурой операции, лежащие на одной ветви, распараллелить нельзя, поскольку они зависят по данным (результат предыдущей операции является операндом для следующей). Распараллеливание вычислений на уровне обработки машинных слов в данном случае не представляется возможным.

При любом числе ВМ, работающих с параллельными кодами операндов, время вычислений будет составлять $T = \sum_j t_j$, где j – индекс операции, лежащей на критическом пути в графе алгоритма, а t_j – время выполнения j-й операции. В общем случае T не определяет полное время реализации алгоритма, так как здесь не учитывается время обмена информацией между устройствами, которое существенно зависит от топологии системы.

Таким образом, если в вычислительных системах для выполнения операций используются методы параллельной машинной арифметики, то реализацию последовательности операций нельзя осуществить за время, меньшее суммарного времени выполнения всех операций.

Частичное совмещение во времени выполнения зависимых операций может быть достигнуто за счет применения методов машинной арифметики, позволяющих выполнять операции в неавтономном режиме с использованием избыточных систем счисления с естественным порядком весов. Способы построения таких вычислительных систем известны [4, 6, 7]. В их состав входят квазипараллельные ВМ, позволяющие совмещать выполнение зависимых операций на уровне обработки разрядов слов.

На каждом шаге в ВМ вводится по одному разряду операндов в системе счисления с основанием $k = 2^s$ $(s = 1,2,3...)$ и формируется один разряд результата. При этом разряд промежуточного результата, полученный на i-м шаге в одном ВМ при выполнении j-й операции, может быть использован на $(i+1)$-м шаге в другом ВМ при выполнении $(j+1)$-й операции. При таком режиме вычислений выполнение следующей операции будет начинаться не после завершения выполнения предыдущей операции, а сразу же после получения первого разряда результата этой операции. Режим работы таких ВМ называют неавтономным, так как для выполнения последовательности операций необходимо несколько ВМ, которые совместно выполняют цепочку операций, обмениваясь информацией в процессе работы.

Такие ВМ по структуре ближе к параллельным, а не последовательным устройствам, что определило их название «квазипараллельные». С использованием квазипараллельных ВМ реализуется параллелизм на уровне обработки разрядов операндов.

Разряды результата выдаются со старших разрядов, причем, первый разряд формируются с задержкой на p шагов. Следовательно, число шагов, необходимое для получения n старших разрядов окончательного результата при выполнении цепочки из K операций, составляет

$$N = n - 1 + \sum_j^K \left(p_j + 1\right),$$

где j – индекс операции, лежащей на критическом пути в графе алгоритма, а p_j.- задержка формирования результата при выполнении

j-й операции. Длительность цикла (шага) вычислений в синхронном режиме должна соответствовать условию

$$T_{\text{ц}} \geq \max_j T_{\text{ц}j}$$

где $T_{\text{ц}j}$ – длительность цикла формирования разряда результата при выполнении j-й операции. Тогда время выполнения цепочки операций будет определяться как

$$T = \left[n - 1 + \sum_{j=1}^{K} \left(p_j + 1\right) \right] T_{\text{ц}}.$$

Для сравнения времени выполнения операций в системах с параллельными и квазипараллельными ВМ необходимо определить значения рассмотренных выше параметров t_j,

p_j и $T_{\text{ц}}$ для конкретной структурной организации модулей. Заметим, что эффективным способом ускорения неавтономных вычислений является увеличение величины основания системы счисления. Например, переход от основания k=2 к основанию k=4 сокращает число шагов вычислений примерно в 2 раза, а к основанию k=8 – в 4 раза.

Обоснование метода неавтономных вычислений

Методы неавтономной арифметики в основном рассмотрены для симметричных избыточных систем счисления, которые позволяют реализовать функции, как с положительными, так и с отрицательными частными производными [8, 9]. Однако на практике применяются вычислительные методы, основанные на реализации функций с положительными частными производными. Это имеет место, например, при численном интегрировании, цифровой обработки сигналов, вычислении полиномов. В случае положительных аргументов вычисления могут производиться в смещенных системах счисления, что позволяет упростить квазипараллельные ВМ [7].

Ниже рассматривается реализация на ПЛИС устройств для вычисления полиномов, которые можно рассматривать как фрагменты ПСНС. Рассмотрены варианты построения устройств на квазипараллельных и параллельных ВМ.

Для вычисления полиномов воспользуемся методом Горнера первого порядка. Вычислитель представляет собой цепочку ВМ, каждый из которых выполняет промежуточную операцию $F = XY + A$ (рис. 1б).

Получим алгоритм выполнения операции в неавтономном режиме с использованием смещенной позиционной избыточной системы счисления с естественным порядком весов и основанием k.

Будем считать, что операнды являются дробными n-разрядными числами и вводятся в ВМ со старших разрядов. Результат также формируется со старших разрядов с запаздыванием на p шагов, то есть в ВМ выполняется операция $Z = 2^p(XY + A)$. Операнды X, Y, A можно записать в виде:

$$X = \sum_{i-1}^{n} x_i k^{-i},\ Y = \sum_{i-1}^{n} y_i k^{-i},\ A = \sum_{i-1}^{n} a_i k^{-i}, \quad (1)$$

где $x_i, y_i, a_i \in \{\overline{0,q}\}$ - цифры операндов.

Естественно, что для получения n разрядов функции F необходимо сформировать $m = n + p$ разрядов Z в виде

$$Z = \sum_{i+1}^{m} z_i k^{-i}, \quad (2)$$

где $z_i \in \{\overline{0,q}\}$ - цифры результата.

Коды, содержащие только i старших разрядов, обозначим через Z_i, X_i, Y_i, A_i.

После выполнения m шагов можно получить результат $Z_m = Z$ с погрешностью, не превышающей k^{-m}, если на каждом i-м шаге цифру z_i выбирать таким образом, чтобы выполнялось условие

$$Z_i \le k^{-p}(X_i Y_i + A_i) < Z_i + k^{-i}. \quad (3)$$

Используя методику [7] и формулы (1), (2), (3), можно получить выражение для промежуточной переменной на i-м шаге выполнения операции в виде

$$H_i = kR_{i-1} + k^{-p} X_{i-1} y_i + k^{-p} Y_{i-1} x_i + k^{-p-i} y_i x_i + k^{-p} a_i,$$

на основании которой формируется значение цифры результата z_i и очередной остаток R_i для следующего шага в виде:

$$z_i = \text{ent}\, H_i,\ R_i = \text{rest}\, H_i. \quad (4)$$

Начальными являются значения

$$X_0 = Y_0 = R_0 = 0.$$

Переменная H_i вычисляется за один такт. В отличие от симметричных систем счисления в данном случае не требуется выполнять анализ диапазона изменения значения H_i с помощью логической схемы [8]. Цифра результата формируется автоматически в соответствии с (4) как целая часть H_i. За счет этого уменьшается оборудование ВМ и сокращается время формирования результата.

Реализация вычислителей на ПЛИС

Проведено моделирование устройства для вычисления полинома пятой степени на базе ПЛИС EP3SL340F1760C2 семейства Stratix III фирмы Altera.

Для разработки системы использовалась среда проектирования Quartus II фирмы Altera. Результаты погружения в ПЛИС разработанного ВМ для обработки 64-разрядных операн-

дов (в двоичном эквиваленте) представлены на рис. 2 и в табл. 1.

Описание ВМ выполнено средствами среды Quartus II. В табл. 1 через разделитель показан соответственно используемый для построения вычислителя и общий ресурс ПЛИС определенного вида.

ВМ работают в смещенной системе счисления с основанием $k = 4$, что требует три проводника для передачи одной цифры между модулями. Повышение основания системы счисления позволило в два раза сократить число тактов обработки операндов по сравнению с двоичной системой. В состав ВМ входят блоки для формирования кодов переменных, входящих в правую часть выражения (3) и блок для суммирования этих кодов.

Табл. 1. Используемые ресурсы ПЛИС для одного ВМ

Логические ячейки	Регистры	Выводы
1 214/270 400 (<1%)	235/270 400 (<1%)	15/1120 (1%)

Для вычисления полиномов пятой степени по схеме Горнера первого порядка построено вычислительное устройство на основе пяти ВМ (рис. 2). Вычислитель включает пять блоков умножения с накоплением MULADD64 и четыре блока задержки DELAY3CLK. Наличие блоков задержки связано с тем, что каждый ВМ формирует результат с задержкой. Необходимо чтобы значение X поступало на каждый ВМ одновременно с результатом из предыдущего ВМ. Схема также содержит вход синхросигнала CLK, вход сброса RESET и вход разрешения вычислений ENABLE. Для обеспечения начала работы каждого ВМ в нужный момент сигнал ENABLE также проходит через блоки задержки.

Табл. 2. Используемые ресурсы ПЛИС для схемы вычисления полиномов

Логические ячейки	Регистры	Выводы
6 070/270400 (2%)	1 223/270 400 (<1%)	27/1120 (2,4%)

Результаты погружения разработанного вычислителя полиномов (табл. 2) показывают, что ресурсы ПЛИС используются весьма экономно, как с точки зрения внутренних элементов (2% ячеек и 1% регистров), так и с точки зрения выводов (2,4%). Оставшийся весьма большой ресурс ПЛИС можно использовать для погружения в микросхему других устройств.

Моделирование вычислителя в среде Quartus II показало, что минимальная длительность одного такта работы, в результате которого формируется одна цифра результата, составляет 8 нс. Учитывая, что каждый ВМ вносит задержку начала формирования разрядов результата после поступления на его входы цифр операндов, первый разряд окончательного результата формируется через 120 нс после начала вычислений.

В дальнейшем в каждом такте формируется очередная цифра результата. Таким образом, полный 64-разрядный результат (в двоичном эквиваленте) может быть получен за 368 нс.

Следует заметить, что в системах автоматического управления во многих случаях управляющее воздействие можно начинать формировать при поступлении первого старшего разряда результата, а затем уточнять при поступлении каждого следующего разряда. Отметим также, что при реализации ПСНС на базе заказных СБИС можно получить большую скорость вычислений. Это объясняется возможность оптимизировать схемы узлов системы с учетом особенностей реализуемых алгоритмов.

Для сравнения разных подходов к реализации вычислителя полиномов (с последовательной и параллельной пересылкой данных между ВМ) разработано устройство на основе параллельных ВМ, информация между которыми передается параллельным кодом. Вычислитель также построен по схеме (рис. 1б) и включает пять ВМ, каждый из которых выполняет операцию $Z = (XY + A)$ в автономном режиме.

Использование ресурсов ПЛИС для такого устройства показано в табл. 3, из которой видно, что затраты ресурсов ПЛИС в данном случае больше, чем для рассмотренного ранее устройства.

Табл. 3. Используемые ресурсы ПЛИС для построения параллельного устройства вычисления полиномов

Логические ячейки	Встроенные блоки умножения	Выводы
780/270 400 (<1%)	80/576 (14%)	512/1120 (46%)

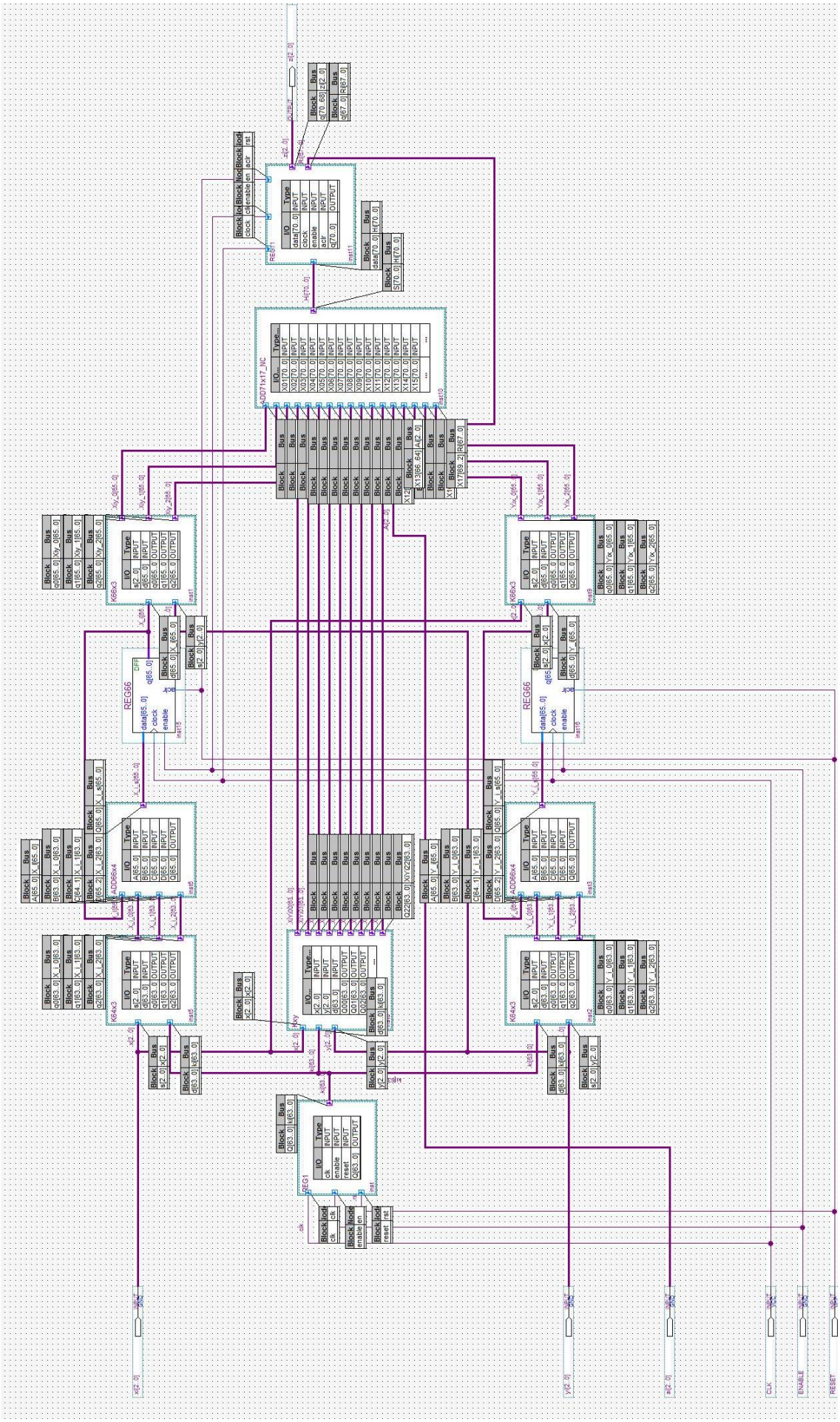


Рис. 2. Схема квазипараллельного ВМ для выполнения одного звена вычислений по схеме Горнера

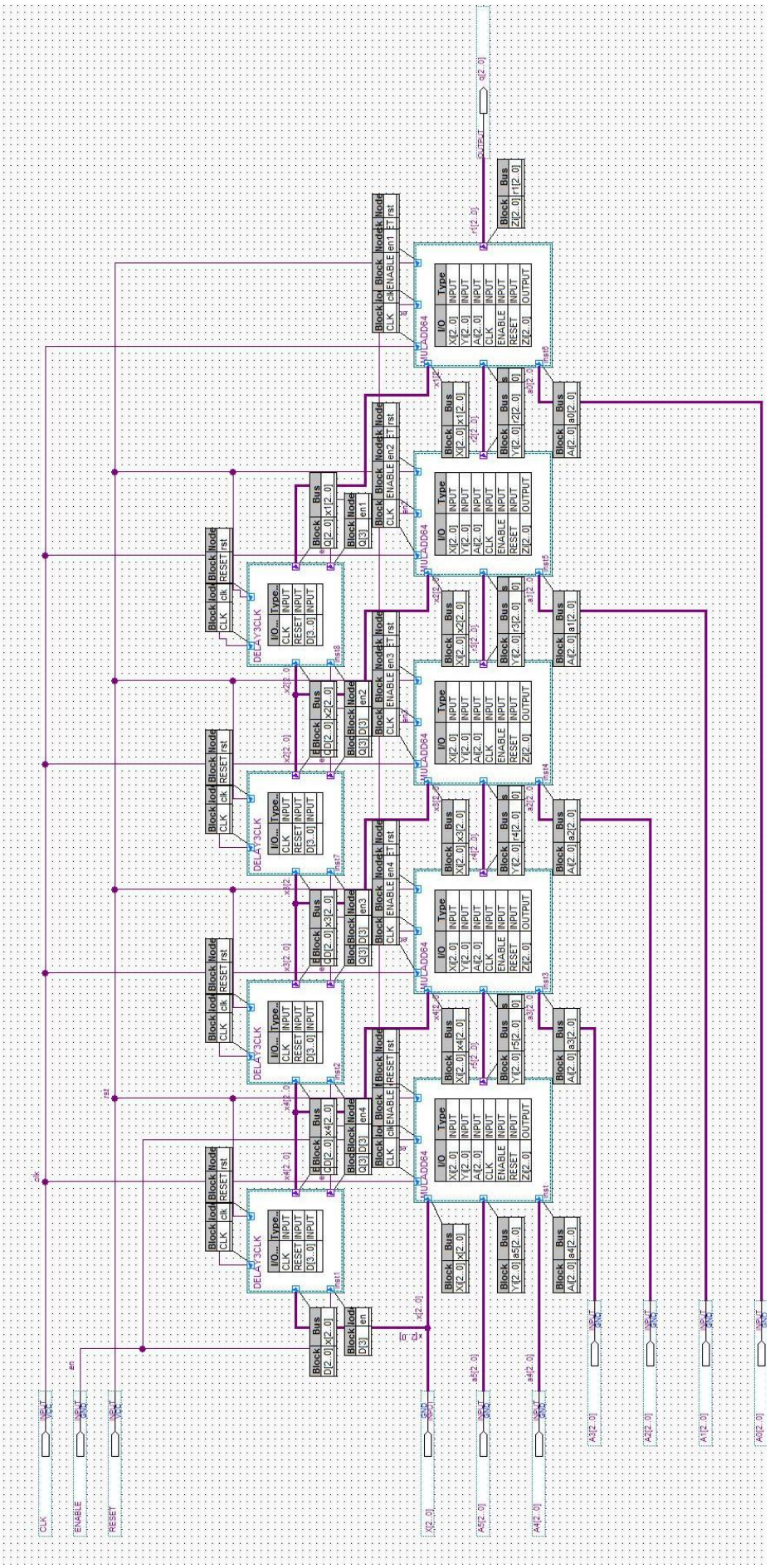


Рис. 3. Схема погружаемого в ПЛИС вычислителя полиномов

Работа данного устройства была промоделирована в среде Quartus II. В результате моделирования определено, что задержка формирования результата в блоке равна 67 нс при одновременном вводе в устройство всех 64-разрядных операндов. Учитывая, что при данной организации вычислений практически половина выводов ПЛИС используется только для вычислителя полиномов, применение такого режима работы устройства вряд ли целесообразно. Более реальным является режим предварительного ввода в определенном порядке 64-разрядных операндов с последующей их обработкой в параллельном устройстве. При таком режиме ввода вряд ли может быть получен существенный выигрыш во времени получения результата по сравнению с устройством на квазипараллельных ВМ. Более точный анализ временных характеристик можно произвести с учетом конкретной реализации режима ввода операндов и вычисления результата.

Заключение

Вычислитель на базе квазипаралельных ВМ с поразрядной передачей данных использует существенно меньше ресурсов ПЛИС. Экономятся как внутренние ресурсы ПЛИС (более чем в 7 раз для рассмотренного примера), так и ее выводы (более чем в 14 раз). Это дает возможность реализовать на той же микросхеме ряд других устройств, относящихся к одной или разным системам.

Построение системы на одной ПЛИС обеспечивает повышение ее надежности, уменьшение энергопотребления и габаритов, а также дает потенциальную возможность повысить частоту тактирования, что, в свою очередь, ускоряет обработку информации.

Таким образом, полученные результаты подтверждают эффективность применения неавтономных методов поразрядной обработки информации со старших разрядов в системах типа ПСНС на базе программируемых и заказных СБИС.

Список литературы

1. Байков В.Д. Решение траекторных задач в микропроцессорных системах ЧПУ / В.Д.Байков, С.Н.Вашкевич. – Л.: Машиностроение, 1986, 105 с.
2. Палагин А.В. Реконфигурируемые вычислительные системы: Основы и приложения / А.В.Палагин, В.Н.Опанасенко. – К.: Просвіта, 2006, 280 с.
3. Каляев И.А. Архитектура семейства реконфигурируемых вычислительных систем на основе ПЛИС / И.А. Каляев, И.И. Левин, Е.А. Семерников // Искусственный интеллект. – 2008. - № 3. – с. 663-674.
4. Жабин В.И. Построение быстродействующих специализированных вычислителей для реализации многоместных выражений / В.И.Жабин, В.И.Корнейчук, В.П.Тарасенко // Автоматика и вычислительная техника. – 1981. - №6. – с. 18-22.
5. Максфилд К. Проектирование на ПЛИС. Архитектура, средства и методы / К.Максфилд. – М.: Издательский дом «Додэка-XXI», 2007, 408 с.
6. Жабин В.И. Выполнение последовательностей зависимых операций в режиме совмещения / В.И.Жабин. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. Наук. Пр. – К.: Век+. – 2007. - №46. – С. 226-233.
7. Дичка И.А. Совмещение зависимых операций на уровне обработки разрядов операндов / И.А.Дичка, В.В.Жабина. // Искусственный интеллект. – 2008. - №3. – С. 649-654.
8. Жабин В.И. Некоторые машинные методы вычисления рациональных функций многих аргументов / В.И. Жабин, В.И.Корнейчук, В.П.Тарасенко // Автоматика и телемеханика. – 1977. - №12. – С. 145-154.
9. Жабин В.И. Методы быстрого неавтономного воспроизведения функций / В.И.Жабин, В.И.Корнейчук, В.П.Тарасенко // Управляющие системы и машины. – 1977. - №3. – С. 96-101.

УДК 004.056

ФЕДОРЕЧКО О.И.,
УВАРОВ Н.В.,
ИСАЧЕНКО Г.В.

К ВОПРОСУ О ПОЛУЧЕНИИ НЕРАЗЛОЖИМЫХ ПОЛИНОМОВ НА ПОЛЯХ ГАЛУА ДЛЯ СИСТЕМ ЗАЩИТЫ ИНФОРМАЦИИ

В статье рассматривается проблема получения неразложимых полиномов на полях Галуа для использования в системах криптографической защиты информации, генерации псевдослучайных чисел и последовательностей, сжатия данных, а также исправления ошибок при их передаче. На основе теоретических и экспериментальных исследований выявлены новые свойства полиномов на конечных полях. Предложен подход к ускорению поиска неразложимых полиномов за счет оптимизации процедуры их селекции и тестирования на основе выявленных свойств.

The article is dedicated to the problem of generating the indecomposable polynomials over Galois fields used in cryptographic information security, generation of pseudo random numbers and sequences, data compression and error-correction during data transmission. New properties of polynomials over finite fields was discovered after theoretical and experimental research. The approach to increase speed of searching indecomposable polynomials is proposed and relies on the optimization of the procedure of their selection and testing using discovered features.

Введение

Теория конечных полей, начатая работами французского математика Э.Галуа в начале 19-века получила широкое практическое применение только в конце следующего, 20-го века с развитием компьютерных и информационных технологий. В настоящее время теоретические положения полей Галуа лежат в основе большинства поточных шифров [1], широко используемых для защиты данных в мобильной связи, компьютерных системах и сетях. Преобразования на полях Галуа используются и в криптографических средствах блокового шифрования, в том числе и алгоритме Rijndael, ставшим победителем всемирного конкурса блоковых шифров AES [2].

Кроме систем защиты информации теоретические положения полей Галуа с 50-х широко используются в компьютерных технологиях для генерации псевдослучайных последовательностей и чисел [3]. Значительная часть широко используемых на практике технологий обнаружения и исправления ошибок передачи данных базируется на математических принципах конечных полей. К их числу относятся циклические коды, коды Рида-Соломона, коды Файра, а также часть взвешенных контрольных сумм [4]. Кроме того, поля Галуа активно используются в системах кодирования результатов измерений и архивировании данных. Новый мощный импульс развитию теории полей Галуа стало их применение в современных системах мобильной связи [5].

Практически во всех перечисленных применениях полей Галуа решается задача выбора образующего полинома $P(x)$, который представляет собой неразложимый полином (неприводимый многочлен) то есть такой, который не может быть представлен в виде произведения двух других полиномов. В алгебре конечных полей образующему полиному поля Галуа можно однозначно сопоставить двоичное число P , которое является простым в этой алгебре, то есть не может быть представлено в виде произведения без переносов других чисел. Задача получения простого в алгебре конечных полей числа P является достаточно сложной и к настоящему времени не существует алгоритмов ее решения, которые бы не предполагали определенного перебора. Для больше части перечисленных выше применений полей Галуа, кроме приложений связанных с защитой информацией, образующий полином $P(x)$ выбирается один раз из специальных таблиц [5].

Для применений полей Галуа, связанных с защитой информации, необходимо решать задачу генерации неразложимых полиномов. С ростом степени полинома (разрядности простого в алгебре конечных полей числа P) сложность задачи его генерации растет экспоненциально [6], поэтому необходимым представляется разработка методов, направленных на

уменьшение вычислительной сложности получения неразложимых полиномов полей Галуа.

Таким образом, задача совершенствования технологии получения неразложимых многочленов, необходимых для использования в качестве образующих полиномов полей Галуа является актуальной для широкого круга практически важных задач современных информационных и компьютерных технологий.

Анализ существующих технологий получения неразложимых многочленов

Задача получения неразложимых многочленов в определенном плане является схожей с задачей получения простых чисел в обычной арифметике [6].

Для обоих задач к настоящему времени не существует метода или алгоритма решения без перебора. Возникновение в 1978 г. криптографии с открытым числом и ее широкое использование стимулировали развитие технология получения простых чисел в обычной арифметике.

К настоящему времени методика получения как простых чисел, так и неразложимых полиномов, в общем виде схожа и состоит в следующем: генерируется случайное число и над ним выполняется последовательность тестов, каждый из которых подтверждает или опровергает гипотезу о том, что число простое. Чем более эффективны тесты и чем большее число их проводится – тем больше вероятность того, что тестируемое число является простым. Даже для чисел, которые успешно прошли все тесты существует низкая вероятность того, что они могут быть разложены на множители. В этой связи для таких чисел принят термин “промышленно простые числа” (ППЧ) [6], чтобы отличить их от истинно простых чисел.

Из сказанного явствует, что эффективность технологий получения ППЧ определяется эффективностью тестов, которая характеризуется:

– вероятностью того, что после прохождения теста число окажется не простым;

– временем выполнения теста.

Оптимизация тестирования состоит в выборе такого набора тестов T_1, T_2,...,T_k, которые позволяют решить одну из двух задач оптимизации тестирования:

1) достижение минимальной вероятности того, что после прохождения тестов T_1, T_2,...,T_k, тестируемое число окажется не простым при заданном ограничении G на суммарное время тестирования $t(T_1)+t(T_2)+...+t(T_k) \le G$, где $t(T_j)$ – время выполнения j-го теста $(j \in \{1,...,k\}$;

2) достижение минимального времени тестирования $\min(t(T_1)+t(T_2)+...+t(T_k))$ для достижения заданной вероятности q того, что после прохождения выбранного набора тестов T_1, T_2,...,T_k, тестируемое число окажется не простым.

Для тестирования простых в арифметическом смысле чисел разработано ряд тестов, наиболее известными из которых являются тест Соловея-Штрассера, тест Лемана, тест Рабина-Миллера [6].

Вполне очевидным, что для повышения эффективности генерации неразложимых полиномов для использования в качестве образующих полей Галуа необходимо разработать эффективные независимые тесты на неразложимость и определить характеристики их эффективности. Для этого, теоретически и экспериментально надо изучить свойства неразложимых полиномов.

Целью исследований является теоретическое и экспериментальное исследование свойств неразложимых полиномов и разработка на этой основе эффективной системы тестов для проверки возможности разложения многочлена на множители.

Исследование свойств неразложимых полиномов

Для создания эффективной системы тестов на неразложимость необходимо с теоретических позиций исследовать специфические свойства, использование которых позволяет сократить время тестирования.

Базовыми операциями над многочленами являются: сложение многочленов, соответствующее логической операции XOR, обозначаемое далее символом $\oplus$, умножение многочленов, соответствующее операции умножения двоичных чисел без переносов, обозначаемое далее символом $\otimes$, операция

Далее в тексте статьи символом операцию «/» будет считаться операцией деления без переносов. Ниже представлено доказательства ряда лемм и теорем, которые могут быть использованы для эффективного тестирования многочленов на предмет их неразложимости.

Лемма 1. Если многочлен $P(a)=a_{n-1}\cdot x^{n-1} + a_{n-2}\cdot x^{n-2} +...+ a_1\cdot x+a_0$ делится без остатка на много-

член $P(b)=b_{m-1}\cdot x^{m-1}+b_{m-2}\cdot x^{m-2}+...+b_1\cdot x+b_0$, то на этот многочлен делится также многочлен $P(c)=a_{n-1}\cdot x^{n+k-1}+a_{n-2}\cdot x^{n+k-2}+...+a_1\cdot x^{k+1}+a_0\cdot x^k$.

Доказательство: Согласно правилам выполнения операции деления многочленов [2] многочлен-делитель $P(b)$ степени m-1, умноженный на x^{n+k-m} суммируется с $P(c)$ с образованием остатка $P_1=P(c)+P(b)\cdot x^{n+k-m}$. Аналогично, при делении многочлена $P(a)$ на $P(b)$ на первом шаге выполняется суммирование $P(a)$ с произведением $P(b)\cdot x^{n-m}$ с образованием остатка $P'_1=P(a)+P(b)\cdot x^{n-m}$. В силу того, что $P(c)=P(a)\cdot x^k$, то $P_1 = P'_1\cdot x^k$. Аналогичные рассуждения полностью справедливы и для вторых остатков: $P_2 = P'_2\cdot x^k$, и для третьих, четвертых и всех остальных остатков, включая последний l-тый: $P_l=P'_l\cdot x^k$. Так как последний остаток P'_l, возникающий в процессе деления многочлена $P(a)$ на многочлен $P(b)$ рван нулю, то равен нулю и остаток P_l от деления многочлена $P(c)$ на многочлен $P(b)$, что и требовалось доказать.

Теорема 2. Если многочлен $P(a)=a_{n-1}\cdot x^{n-1}+a_{n-2}\cdot x^{n-2}+...+a_1\cdot x+a_0$ и многочлен $P(b) = b_{k-1}\cdot x^{k-1} + b_{k-2}\cdot x^{k-2}+...+b_1\cdot x+b_0$ делятся на многочлен $P(c)= c_{m-1}\cdot x^{m-1}+c_{m-2}\cdot x^{m-2}+...+c_1\cdot x+c_0$, то многочлен $P(d) = P(a)\cdot x^{k+q}+ P(b) = a_{n-1}\cdot x^{n+q+k-1}+...+a_1\cdot x^{q+k+1}+a_0\cdot x^{q+k} + b_{k-1}\cdot x^{k-1} + b_{k-2}\cdot x^{k-2}+...+b_1\cdot x+b_0$ также делится без отстатка на полином $P(c)$.

Доказательство: многочлен $P(d)$ можно записать в виде $P(d)=P(a)\cdot x^{k+q}+P(b)$. Выше леммой 1 было доказано, что, если $P(a)$ делится без остатка на $P(c)$, то и $P(a)\cdot x^{k+q}$ также без остатка делится на $P(c)$. По условиям теоремы и $P(b)$ делится без остатка на $P(c)$. Поскольку операция полиномиального деления обладает свойством дистрибутивности [2], то остаток от деления полинома $P(d)$ на полином $P(c)$ равен сумме остатков от деления полиномов $P(a)\cdot x^{k+q}$ и $P(b)$, то есть равен нулю, что и требовалось доказать.

Лемма 2. Если многочлен P_1 является двучленом, то есть может быть представлен в виде $P_l=x^n + x^k$, то он делится без остатка на многочлен $P_3=x+1$.

Доказательство: на основании леммы 1 можно утверждать, что многочлен $P_l=x^n + x^k$ делится без остатка на многочлен P_3, если таким свойством обладает многочлен $P_2=x^{n-k}+1$. В процессе полиномиального деления многочлена P_2 последовательно образуются остатки: $x^{n-k-1}+1$, $x^{n-k-2}+1$, $x^{n-k-3}+1$,..., $x+1$. Поскольку последний из остатков совпадает с P_3, то P_2 делится без остатка на P_3, а, значит, согласно лемме 1 и многочлен P_1 делится без остатка на P_3, что и требовалось доказать.

Теорема 3. Если многочлен P_e содержит четное число членов, то он делится без остатка на многочлен $P_3=x+1$.

Доказательство. Поскольку полином P_e содержит четное число h членов, то он может быть представлен в виде суммы $h/2$ многочленов, каждый из которых содержит в точности два члена. Леммой 2 доказано, что каждый из этих двучленов делится без остатка на многочлен $P_3=x+1$. Согласно свойству дистрибутивности [2] остаток от деления многочлена P_e на P_3 равен сумме остатков от деления двучленов составляющих P_e на P_3, то есть нулю, что и требовалось доказать.

Симметричным по отношению к многочлену $P(a)=a_{n-1}\cdot x^{n-1} + a_{n-2}\cdot x^{n-2} +...+ a_1\cdot x+a_0$ является многочлен $P(\hat{a})=a_0\cdot x^{n-1} + a_1\cdot x^{n-2} +...+ a_{n-2}\cdot x+a_{n-1}$. Симметричными называются двоичные числа a и $\hat{a}$, соответствующие приведенным многочленам. Например, если $a = 1101_2$, то $\hat{a} = 1011_2$.

Теорема 4. Если $a \otimes b = c$, то $\hat{a} \otimes \hat{b} = \hat{c}$. Произведение симметрических двух чисел являет собой симметричное число к произведению этих чисел.

Например, если $a = 1101_2$, $b = 5 = 0101_2$, то $\hat{a} = 1011_2 = 11_{10}$, а $\hat{b} = 1010_2 = 10_2$. Произведение без переносов $a\otimes b = 13\otimes 5 = c = 57_{10} = 0111001_2$, а произведение симметрических $\hat{a} \otimes \hat{b} = 11\otimes 10 = 78_{10} = 1001110 = \hat{c}$.

Доказательство:

Пусть a – n-разрядное двоичное число $a = 2^{n-1}\cdot a_{n-1} + 2^{n-2}\cdot a_{n-2} + ... + 2\cdot a_1 + a_0$, причем для всех $q\in\{0,...,n\text{-}1\}$: $a_q\in\{0,1\}$, соответственно, симметрическое числу a число $\hat{a}$ можно представить в следующем виде: $\hat{a} = 2^{n-1}\cdot a_0 + 2^{n-2}\cdot a_1 + ... + 2\cdot a_{n-2} + a_{n-1}$.

Аналогически, если b – m– разрядное число $b = 2^{m-1}\cdot b_{m-1} + 2^{m-2}\cdot b_{m-2} + ... + 2\cdot b_1 + b_0, g\in\{0,...,m\text{-}1\}$: $b_g\in\{0,1\}$, то симметрическое ему число $\hat{b} = 2^{m-1}\cdot b_0 + 2^{m-2}\cdot b_1 + ... + 2\cdot b_{m-2} + b_{m-1}$. Тогда произведение без переносов этих чисел являет собой число c, разрядностью $l=n+m-1$: $c = 2^{l-1}\cdot c_{l-1} + 2^{l-2}\cdot c_{l-2} + ... + 2\cdot c_1 + c_0$, причем, $h\in\{0,1,...,l\}: c_h\in\{0,1\}$. Соответственно симметричное $\hat{c} = 2^{l-1}\cdot c_0 + 2^{l-2}\cdot c_1 + ... + 2\cdot c_{l-2} + c_{l-1}$.

Если $d = 2^{l-1} \cdot d_{l-1} + 2^{l-2} \cdot d_{l-2} + ... + 2 \cdot d_1 + d_0$ – l-разрядное произведение без переносов $\hat{a} \otimes \hat{b}$, то для того, что бы доказать теорему не обходимо показать, что для каждого $h \in \{0,1,...,l\}$: $c_h = d_{l-h-1}$.

Как биномиальный коэффициент при 2^h , c_u являет собой сумму по модулю два произведений $a_i \cdot b_j$ двоичных разрядов a и b, для которых сумма индексов равняется h: $i+j=h$:

$$c_h = \bigoplus_{i+j=h} a_i \cdot b_j \qquad (1)$$

Выражение (1) может быть преобразовано следующим образом:

$$c_h = \bigoplus_{i+j=h} a_i \cdot b_j = \bigoplus_{i+j=h} \hat{a}_{n-i-1} \cdot \hat{b}_{m-j-1} =$$

$$= \bigoplus_{r+v=m+n-(i+j)-2} \hat{a}_r \cdot \hat{b}_v = \bigoplus_{r+v=l-h-1} \hat{a}_r \cdot \hat{b}_v = d_{l-h-1}$$

Таким образом, доказано, что $c = \hat{d}$.

Следствие:

Если $c \bmod a = b$, то $\hat{c} \bmod \hat{a} = \hat{b}$.

Теорема 5. Если c – непростое, то і $\hat{c}$ – тоже непростое.

Доказательство: Если c – непростое, то его можно представить в виде произведения: $c = a \otimes b$. Тогда, согласно теореме 4 существуют $\hat{a}$ и $\hat{b}$ такие, что $\hat{a} \otimes \hat{b} = \hat{c}$, таким образом $\hat{c}$ можно представить в виде произведения двух чисел. А из этого следует, что $\hat{c}$ непростое.

Теорема 6. Если c – простое, то и $\hat{c}$ – также простое.

Доказательство: Если $\hat{c}$ – непростое, то его можно представить в виде произведения $\hat{c} = a \oplus b$. Тогда, в соответствии с теоремой 4 существуют $\hat{a}$ и $\hat{b}$ такие, что $\hat{a} \otimes \hat{b} = c$, таким образом c также можно представить в виде произведения двух чисел. Это противоречит условию теоремы, согласно которому c – простое. Таким образом, доказано, что если c – простое, то и $\hat{c}$ также простое.

Кроме теоретических, были проведены широкие экспериментальные исследования, направленные на выявления свойств неразложимых многочленов, которые могли бы быть использованы для построения эффективной системы тестов.

Исследовались свойства проверочных многочленов, которые могли бы быть использованы в качестве тестирующих делителей исследуемого многочлена. Проверочные многочлены выбирались только с нечетным количеством членов, и также, обязательным условием было наличие в них члена нулевой степени. В начале эксперимента количество членов проверочных полиномов равнялось трем. При проведении проверок степень проверочных полиномов постепенно повышалась до n-2 (где n– степень тестируемых полиномов). Если были выполнены проверки всех возможных комбинаций для текущего количества членов в проверочных полиномах, то выполнялось увеличение количества членов в проверочных полиномах на два и снова выполнялись проверки с новым количеством членов и с постепенным повышением степени проверочных полиномов. Максимальное количество членов в проверочных полиномах равняется n-2. В случае нахождения полинома результат редукции с тестируемым полиномом, для которого равнялся нулю, тестирование текущего тестируемого полинома прекращалось и констатировался тот факт, что тестируемый полином не принадлежит к множеству неразложимых многочленов. В ходе эксперимента выполнялся подсчет характеристик таких проверочных полиномов, которые указывали на непринадлежность текущего тестируемого полинома к множеству неразложимых, а именно запоминалась их степень и количество членов, поскольку именно эти параметры определяют вычислительную сложность тестирования. Результаты исследований показали, что вероятность отнесения проверяемого многочлена к разряду разложимых при использовании проверочных полиномов зависит от количества их членов.

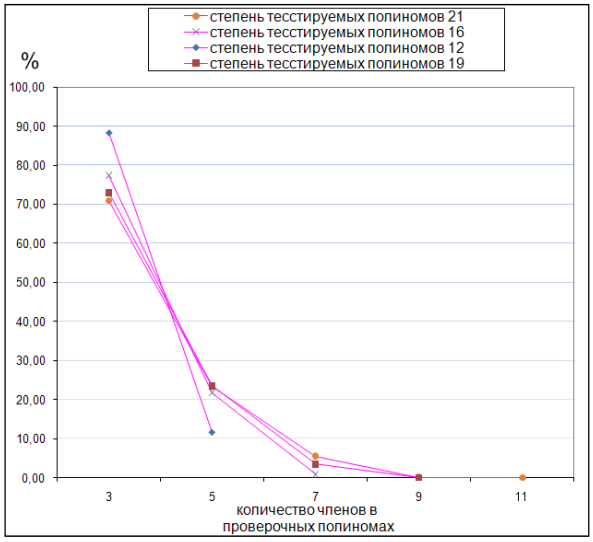


Рис.1. Зависимость количества проверочных полиномов от количества членов в них

На рис.1 показано, что наибольшее количество проверочных полиномов имело три члена, затем их количество постепенно спадает к нулю относительно количества их членов. Поскольку максимальное значение членов в полиноме меньше чем *n*/2, то при проведении эксперимента имела место большая избыточность проверок по количеству членов в проверочных полиномах.

Анализ экспериментальных исследований также показал избыточность проверок по степени проверочных полиномов. На рис.2 показана зависимость количества проверочных полиномов от их степени.

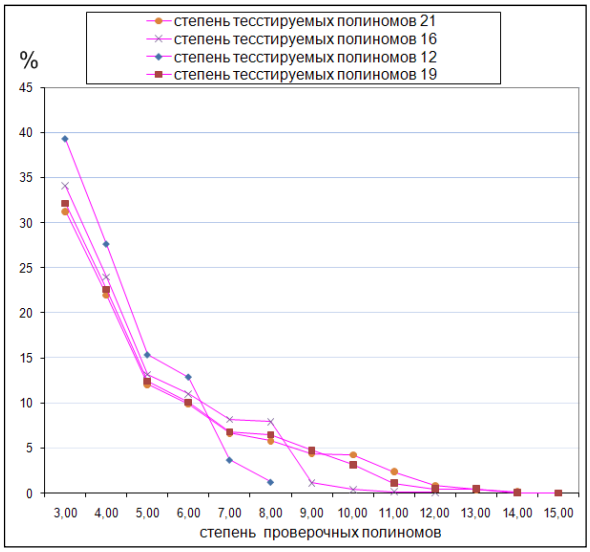


Рис.2. Зависимость количества проверочных полиномов от их степени

Кроме того, в результате проведенных экспериментальных исследований был установлен тот факт, что сложность проверок зависит от количества членов в тестируемом полиноме, а именно количество членов и степень проверочных полиномов возрастает с приближением количества членов в тестируемых полиномах к *n*/2, а на промежутке от *n*/2 до *n* постепенно спадает. Результаты этих исследований представлены на рис. 3 и рис.4.

Из рисунка 3 видно, что при выполнении полного перебора проверочных полиномов значение максимальной степени проверочных полиномов зависит от количества членов в тестируемом полиноме. Это значит что при выборе максимального значения степени проверочных полиномов, надо учитывать количество членов в тестируемом полиноме.

Из рисунка 4 видно, что при выполнении проверок значение максимального количества членов в проверочных полиномах зависят от количества членов в тестируемом полиноме.

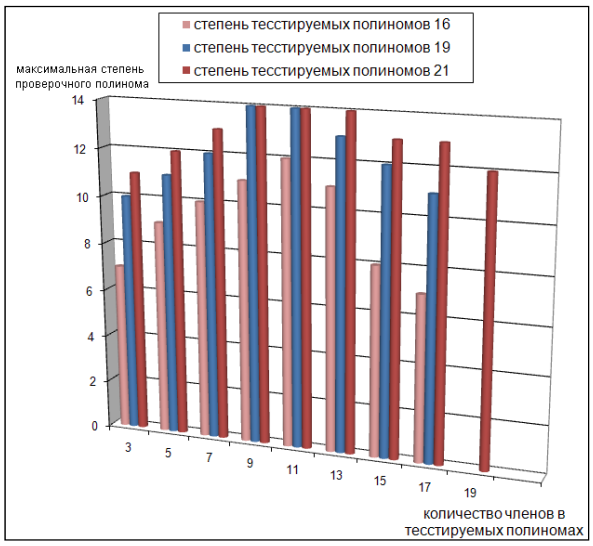


Рис.3. Зависимость максимального значения степени проверочных полиномов от количества членов в тестируемых полиномах

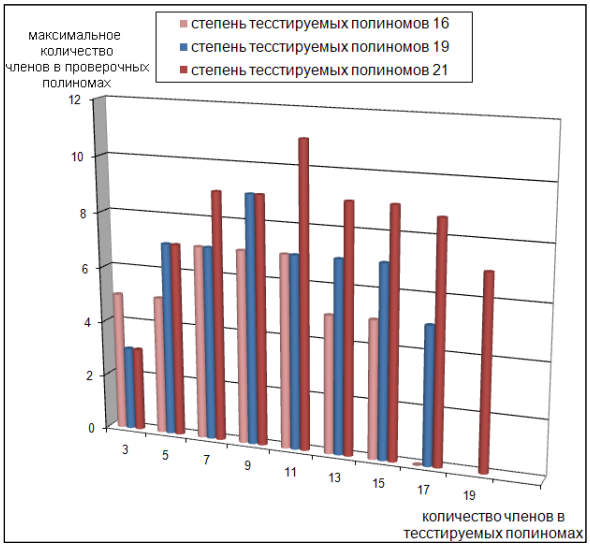


Рис.4. Зависимость максимального количества членов в проверочных полиномах от количества членов в тестируемых полиномах

Из этого следует, что при формировании тестов надо выбирать проверочные полиномы начиная с меньших степеней, причем наиболее эффективными из них являются полиномы, число членов которых близко к половине от их степени.

Организация выбора оптимального плана тестирования

Проведенные теоретические и экспериментальные исследования свойств неразложи-

мых многочленов и проверочных полиномов позволяют сформулировать порядок организации тестирования, позволяющий уменьшить время тестирование и вероятность того, что многочлен, прошедший тесты окажется разложимым.

Критерии к нижнему уровню вероятности ошибки тестирования определяются спецификой использования многочлена. Как, если многочлен используется для построения генератора псевдослучайных последовательностей, то ошибка тестирования будет иметь следствием разделение одного цикла повторения на k локальных, период повторения которых соответственно в k раз меньше.

При использовании формируемого многочлена в криптографических системах цифровой подписи, ошибка теста не является критической – она сказывается только на этапе нахождения мультипликативной инверсии [6]: если многочлен не является простым, то операция нахождения мультипликативной инверсии для небольшого числа чисел не может быть выполнена. Поскольку эта операция осуществляется лицом, подписывающим документ, то необходимо оперативно прервать процесс подписи и выбрать другой многочлен.

При организации процесса тестирования многочлена степени n на предмет его неразложимости рекомендуется выполнять следующее:

1) Выбрать случайное нечетное число, содержащее нечетное число единиц, близкое к $n/4$. Соответствующий этому числу полином использовать в качестве тестируемого.

2) Сформировать список проверочных многочленов, причем их порядок должен соответствовать увеличению их степени, причем степень проверочных полиномов должна не превышать $n/3$ плюс количество членов в тестируемом полиноме. Согласно выше представленным исследованиям количество членов проверочных полиномов не должно превышать $n/4$ плюс количество членов в тестируемом полиноме.

3) С процессе тестирования выполняется деление тестируемого многочлена на проверочный. Каждый тест целесообразно выполнять два раза: по отношению к тестируемому многочлену и по отношению к многочлену, симметричному к тестируемому.

Выводы

В результате проведенных исследований, направвленных на повышение эффективности генерации неразложимых полиномов полей Галуа для систем защиты информации определены стратегии тестирования. Теоретически и экспериментально исследованы специфические свойства неразложимых многочленов, использование которых позволяет существенно сократить время тестирования и повысить его достоверность. На основе изучения свойств проверочных многочленов разработаны рекомендации.

Полученные в работе результаты могут быть использованы для сокращения времени генерации криптографических систем защиты информации с открытым ключом, а также систем поточного шифрования данных.

Список литературы

1. Асосков А.В., Иванов М.А., Мирский А.А., Рузин А.В., Сланин А.В., Тютвин А.Н. Поточные шифры. – М.:Кудиц-образ.-2003.– 334 с.
2. Зенин О.С., Иванов М.А. Стандарт криптографической защиты AES. Конечные поля. – М.:Кудиц-образ.-2002.– 174 с.
3. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. М.: КУДИЦ-ОБРАЗ.– 2003 – 260 с.
4. Марковский А.П., Турченко Ю.А., Саидреза Мехмали, Сакун В.М. Использование взвешенных контрольных сумм для исправления “пачек” ошибок передачи данных // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. К.,"ВЕК++",– 2009.– № 51.– С.90-96.
5. Peterson R.L., Ziemer R.E., Borth D.E. Introduction to Spread Spectrum Communication. Prentice Hall.– 1995.– 695 p.
6. Шнайер Б. Прикладная криптография. Протоколы и алгоритмы. М.:Триумф.-2003.-816 с.

УДК-004.057.4

ПОЛТОРАК В.П.,
ДОРОШЕНКО К.С.,
НИЩА О.В.

РІШЕННЯ ЗАДАЧІ РОЗБИТТЯ МЕРЕЖІ OSPF НА ОБЛАСТІ

В статті представлено результати дослідження мережі, яка обслуговується протоколом маршрутизації OSPF, на предмет виявлення можливостей застосування вибраного алгоритму автоматичної класифікації для розбиття мережі на внутрішні області. Проведено моделювання об’єктів та серію експериментів на застосування результатів обрахунків алгоритму прикладного аналізу даних «Спектр».

The results of research of network that is served by protocol of routing of OSPF are presented in the article, for the purpose the exposure of possibilities of application of the chosen algorithm of automatic classification for parcelling of network on internal areas. The design of objects and series of experiments is conducted on application of results of accounts of algorithm of the applied analysis of data "Spectrum".

Вступ

Най універсальним і гнучким в налаштуванні протоколом динамічної маршрутизації в корпоративних мережах на сьогоднішній день є відкритий протокол вибору першого найкоротшого шляху (Open Shortest Path First Protocol – OSPF). Протокол орієнтовано на роботу у великих мережах (до 65536 маршрутизаторів) із складною топологією. Він базується на алгоритмі стану каналів зв'язку і має високу стійкість до змін топології мережі і швидку збіжність. При виборі маршруту використовується метрика пропускної спроможності складеної мережі (з метою передачі даних по най швидкісним каналам зв'язку). Протокол може підтримувати різні вимоги IP-пакетів до якості обслуговування (пропускна спроможність, затримка і надійність) за допомогою побудови окремої таблиці маршрутизації для кожного з цих показників. На практиці, за великої кількості маршрутизаторів та нестабільних підмереж збіжність протоколу нелінійно змінюється. Зазвичай такі проблеми усуваються регулюванням затримки реакції або розбиттям системи на внутрішні області вручну адміністратором. Таке вирішення проблеми не є уніфікованим, вимагає багато часу та творчого підходу.

Мета

Дослідження мережі, яка обслуговується протоколом маршрутизації OSPF, на предмет виявлення можливостей застосування вибраного алгоритму автоматичної класифікації для розбиття мережі на внутрішні області.

Огляд існуючих рішень

Область OSPF – це частина автономної системи, яка об'єднує підмережі з прилеглими діапазонами адрес. Однією з головних задач визначення областей OSPF є об'єднання маршрутів, які дозволяють здійснювати маршрутизацію лише даної області, не підключаючи магістральну область[1].

Область визначається як сукупність маршрутів, всередині якої маршрутизатори мають інформацію лише про дану область і про маршрут «за замовчуванням» (до магістралі). Це робить OSPF ефективним протоколом маршрутизації, оскільки для кожного маршрутизатора інформація про інші підмережі не є необхідною. Області нумеруються у форматі 0.0.0.x, де x позначає діапазон підмережі. Магістральна область – це високошвидкісна область, до якої всі інші області OSPF приєднані (всі інші області називаються тупиковими). Трафік різного призначення, що сполучає дві області, завжди буде проходити через магістральну область. Магістральна область позначається як 0.0.0.0.

Тупикова область – це область пов'язана з магістральною областю через (ABR)[3]. При проектуванні топології OSPF необхідно визначити тупикову область та приєднати її до магістральної області і лише в крайньому випадку зв'язувати їх між собою. Перевагою такого об’єднання: буде те, що тупикова область має лише один маршрут для всього трафіку який

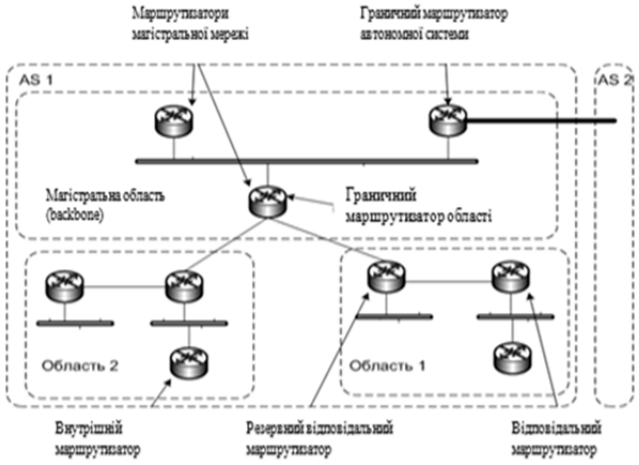


Рис. 1. Автономна система, що розділена на області маршрутизації

виходить за межі області. Малюнок, наведений нижче, демонструє приклад топології OSPF.

На рисунку зображена магістральна (опорна) область (backbone) – область через яку зв'язуються інші маршрутизатори. Маршрутизатор магістральної мережі (Backbone Router – BR) – підключений до магістральної мережі. Граничний маршрутизатор області підключений до декількох областей. Граничний маршрутизатор виділеної області (Autonomous System Boundady Router – ASBR) – підключений до інших автономних систем. Внутрішній маршрутизатор (Internal Router-IR) – це маршрутизатор, у якого всі підключення знаходяться в середині однієї області. Відповідальний маршрутизатор (Designated Router – DR) – збирає і розсилає маршрутну інформацію в виділеній області. Таким чином в OSPF вибудовується ієрархія маршрутизаторів. Резервний відповідальний маршрутизатор (Backup Designated Router-DR) – резервує DR[5].

Так як будь-яка область OSPF – це набір суміжних інтерфейсів (територіальних ліній або каналів локальних мереж), то введення поняття «область» перекриває дві основні задачі в такій мережі – управління інформацією та визначення доменів (областей) маршрутизації.

Розподіл на області призводить до використання двох різних типів маршрутизації OSPF в залежності від того, чи (знаходиться) джерело інформації і одержувач в одній, або в різних областях. У першому випадку має місце внутрішньозонна, у другому – міжзонна маршрутизація[3]. Ця властивість OSPF використовується для обмеження лавинної розсилки оновлень стану всіх каналів. У протоколі маршрутизації з оголошенням стану зв'язків кожний маршрутизатор підтримує базу даних, що описує топологію автономної системи. Цю базу даних називають базою даних стану зв'язків. Кожний взаємодіючий маршрутизатор має ідентичну базу даних. Кожний окремий елемент цієї бази даних є локальним станом певного маршрутизатора (наприклад, використані інтерфейси і досяжні сусіди). Маршрутизатор розподіляє свій локальний стан у автономній системі шляхом лавинної передачі. Саме межі області визначають об'єм лавинної розсилки та кількість обчислень за алгоритмом Дейкстри, що визначають розмірність таблиці маршрутизації на роутері.

Динамічна маршрутизація, яка реалізована в протоколі OSPF полягає в тому, що таблиця маршрутизації постійно відображає точну та актуальну інформацію про топологію мережі. Усі маршрутизатори паралельно виконують однаковий алгоритм. За допомогою бази даних про стан зв'язків кожний маршрутизатор будує область найкоротших шляхів, виступаючи у якості відповідального (DR). Ця область визначає маршрут до кожного адресата автономної системи. Час, який потрібен для того, щоб зміни які відбулися в топології мережі відобразилися в таблицях маршрутизації всіх мережевих маршрутизаторів, називається тривалістю переходу мережі в сталий стан. У цьому контексті перехід мережі в сталий стан є процес наближення до такого стану, в якому інформація у всіх таблицях маршрутизації узгоджена, а самі таблиці залишаються в незмінному вигляді.

Одним з важливих вимог до протоколу маршрутизації є коротка тривалість переходу в сталий стан, оскільки на той час, який маршрутизатор витрачає на обчислення нового оптимального маршруту, нормальний процес маршрутизації може порушитися.

Таким чином, зменшення та узагальнення маршрутної інформації є головною метою введення областей в OSPF.

Експерименти й результати

Для проведення ряду експериментів було обрано програмне забезпеченя Opnet, яке пропонує користувачам графічне середовище для виконання моделювання та аналізу мереж зв'язку. Це п.з. може використовуватися також для налаштування і перевірки протоколів зв'язку, оптимізації та планування мереж.

Для проведення експериментів було обрано однорідну мережу, яка представляє собою окрему автономну область під загальним єдиним керівництвом та маршрутизацією за алгоритмом Дейкстри. В мережі було задіяно 20 маршрутизаторів (Ethernet4_slip8_gtwy), 4 свічі двох видів node_33 and node_28 (Ethernet 16_switch), і node_31 and node_29 (Ethernet 16_bridge), а також 8 моделей, що імітують локальну мережу, кожна з яких підтримує 100 персональних комп'ютерів, тому подана мережа маршрутизації OSPF (рисунок 2) об'єднує 800 користувачів.

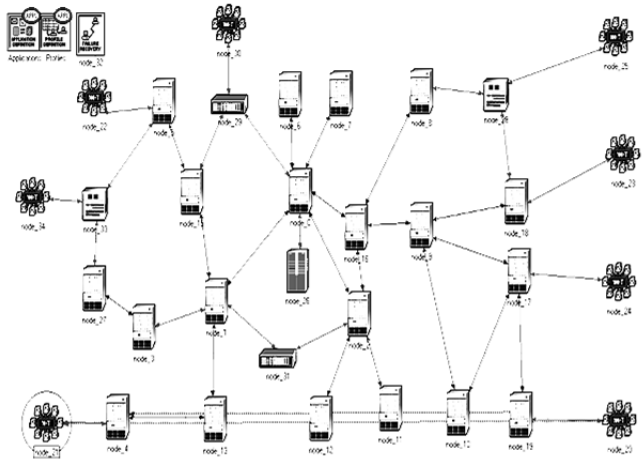


Рис. 2. Загальний вигляд OSPF моделі

У протоколі OSPF для визначення того, який із маршрутів до одержувача є найкоротшим, застосовується вартість, як метрика маршрутизації[6]. Значення вартості присвоюється інтерфейсам (каналам), тому дане значення можна розглянути як «плату» за проходження цього каналу до шляху одержувача. Розрахунок вартості в протоколі OSPF, здійснюється наступним чином: вартість інтерфейсу пропорційна пропускній спроможності цього інтерфейсу. Тобто формула розрахунку вартості набуває такого вигляду:

Вартість = еталонна пропускна здатність / пропускна здатність інтерфейсу.

Еталонна пропускна здатність має за замовчуванням (згідно документу RFC) значення 100 000 000, або 10^8, а пропускна здатність інтерфейсу залежить від типу інтерфейсу.Тому чим вище пропускна здатність (швидкість каналу), тим нижче вартість каналу за протоколом OSPF. Наприклад вартість каналу Ethernet на 10 Мбіт/с дорівнює $10^8/10^7$=10 а вартість лінії T1 складає 10^8/1 544000 = 64.У специфікації OSPF формула обчислення вартості визначена наступним чином: Вартість = 100 000 000 / пропускна здатність (біт/с).

В мережі налаштовано інтерфейси двох типів 40 (Мбіт/с) та 100 (Мбіт/с), таким чином було змодельовано процесс обміну сервісною інформацією, та визначення оптимального маршруту.

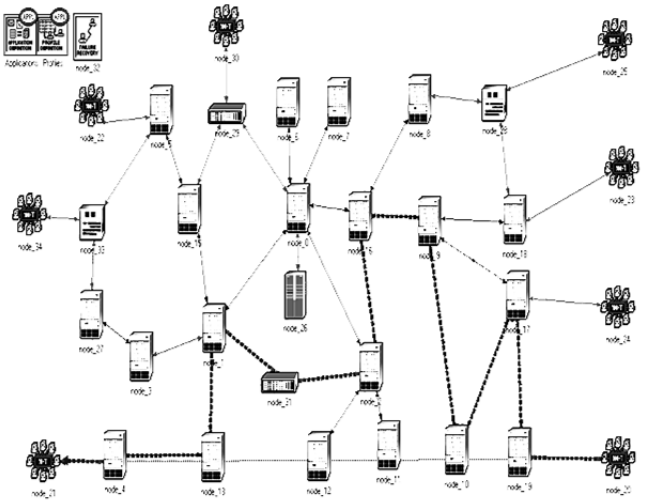


Рис. 3. Розрахований оптимальний маршрут за умови варіації інтерфейсів

На рисунку 3 найкоротший маршрут передачі данних проходить через 8 маршрутизаторів (Ethernet4_slip8_gtwy), 1свіч (Ethernet 16_bridge), від однієї мережі користувачів (100Base T_LAN) до іншої (10BaseT_LAN). Цей маршрут розраховується за пропускною здатністю інтерфейсів, від nod _20 and nod _19 також nod _2 and nod _31, nod _31 and nod _1, nod _4 and nod _21 яка становить 100 (Мбіт/с), а інтерфейси nod _19 and nod _17, nod _17 and nod _9, nod _9 and nod _16, nod _16 and nod _2, nod _1 and nod _13, nod _13 and nod _4 становлять пропускну здатність 40 (Мбіт/с). На наступному рисунку при змінні пропускної здатності між nod_17 and nod_9 (яку зменьшено було в 5 разів) маршрут перенаправлено через nod_10; пропускна здатність між nod_17 and nod_10, nod_10 and nod_9 становила по 40 (Мбіт/с). Оптимальний маршрут позначено штриховою лінією. За даної конфігурації мережі, маємо наступні показники: кількість відправленого трафіку (рисунок 4) та кількість втрачених пакетів під час передачі даних.

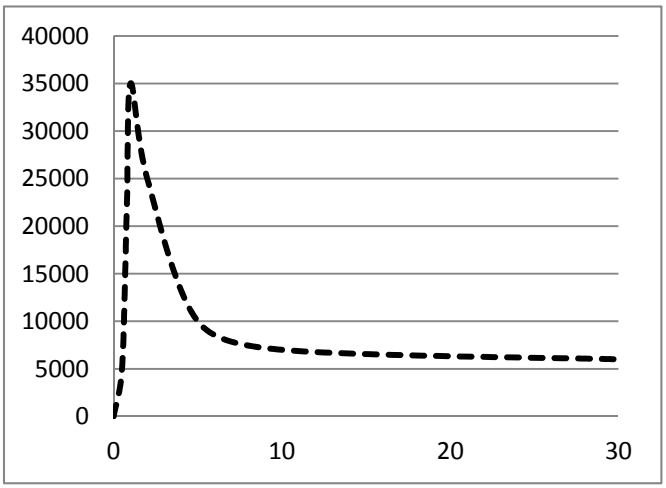


Рис. 4. Time_average (in IP.Traffic Dropped(packets/sec))

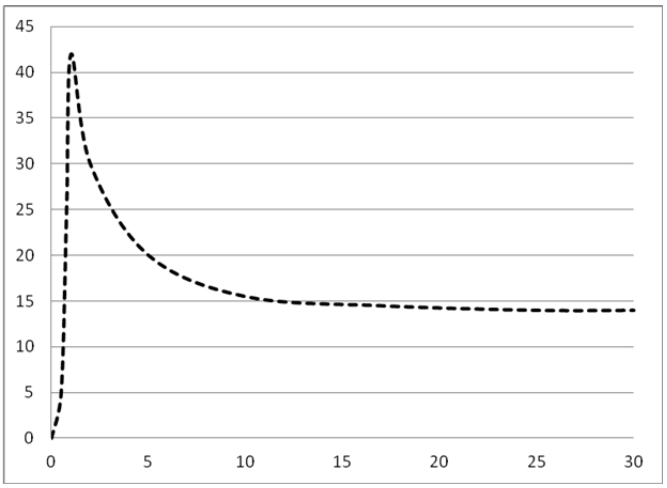


Рис. 5. Time_average (in OSPF.Traffic Sent (bits/sec))

Подальший експеримент полягає у визначенні загального об'єму трафіку при інших рівних умовах у мережі із виділеними внутрішніми областями. Для аналізу можливості виділення внутрішніх областей запропоновано спеціальний алгоритм автоматичної класифікації об'єктів з курсу прикладного аналізу даних, а саме евристичний алгоритм «Спектр», що призначений для агрегування елементів довільної природи, заданих лише матрицею їх зв'язків між собою, і в ідейному плані спирається на гіпотезу компактності. Алгоритм «Спектр» будує послідовність ділянок, кожній з яких належить елемент тільки з одного блоку, про що свідчать великі і близькі один до одного значення на відповідних пологих ділянках послідовності. Коли елементи чергового блоку вичерпуються, алгоритм знаходить найближчий до цього блоку елемент наступного блоку, тому в спектрі зв'язків на кордонах блоків мають місце різкі зменьшення величин.

Ця особливість спектру зв'язків і використовується для виявлення структури елементів, що агрегуються:

-якщо число виділених блоків обумовлено заздалегідь і рівняється К, то для виділених меж цих блоків в спектрі відбираються к-1 найменьших зв'язків: відповідні цим зв'язкам елементи задають у послідовності «В» лівіше кордону (початку) шуканих блоків (зрозуміло, що першу ліву межу задає елемент Х1);

-якщо число блоків заздалегідь не обговорюється і повино бути знайдено в результаті роботи алгоритму, то або задаються абсолютним порогом, зменьшення величини зв'язку нижче якого трактується як перехід до наступного блоку, або відносним порогом, перевищення якого відносним приростом величини зв'язку при переході від і-1 до і-того елементу послідовності «В», у якості нового блоку.

Вхідні дані для роботи алгоритму "Спектр": із структури мережі та вартості інтерфейсів, побудуємо граф зв'язності:

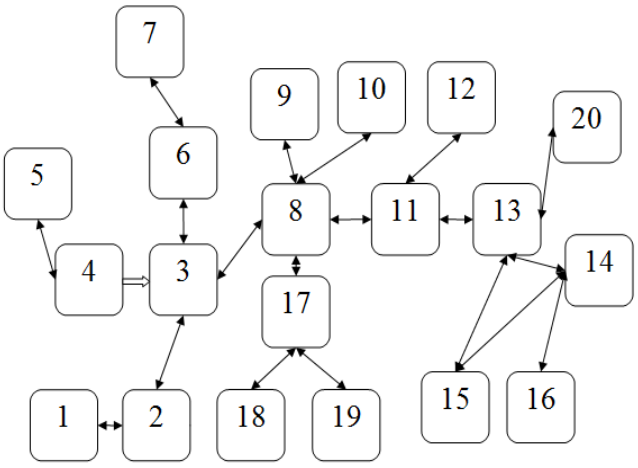


Рис. 6. Вихідні дані (граф, що представляє множину роутерів з протоколом OSPF та налаштованими інтерфейсами двох типів)

де розрахункова вартість двух типів інтерфейсів:

$10^8 \div 4\times10^7 = 2{,}5$ (40 (Мбіт/с))

$10^8 \div 10^8 = 1$ (100 (Мбіт/с))

По вказаним вхідним даним отримаємо матрицю близькості (зв'язків).

Отримаємо спектр зв'язків, що має явно виражений спадаючий характер.

Табл. 1. Матриця зв'язків

Об	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	-	2,5	5	6	8,5	7,5	10	7,5	10	10	10	12,5	12,5	15	15	17,5	10	12,5	12,5	15
2		-	2,5	3,5	6	5	7,5	5	7,5	7,5	7,5	10	10	12,5	12,5	15	12,5	10	10	12,5
3			-	1	3,5	2,5	5	2,5	5	5	5	7,5	7,5	10	10	12,5	5	7,5	7,5	10
4				-	2,5	3,5	6	3,5	6	6	6	8,5	8,5	11	11	13,5	6	8,5	8,5	11
5					-	6	8,5	6	8,5	8,5	8,5	11	11	13,5	13,5	16	8,5	11	11	13,5
6						-	2,5	5	7,5	7,5	7,5	10	10	12,5	12,5	15	7,5	10	10	12,5
7							-	7,5	10	10	10	12,5	12,5	15	15	17,5	10	12,5	12,5	15
8								-	2,5	2,5	2,5	5	5	7,5	7,5	10	2,5	5	5	7,5
9									-	5	5	7,5	7,5	10	10	12,5	5	7,5	7,5	10
10										-	5	7,5	7,5	10	10	12,5	5	7,5	7,5	10
11											-	2,5	2,5	5	5	7,5	5	7,5	7,5	5
12												-	5	7,5	7,5	10	7,50	10	10	7,5
13													-	2,5	2,5	5	7,5	10	10	7,5
14														-	2,5	2,5	10	12,5	12,5	5
15															-	5	10	12,5	12,5	5
16																-	12,5	15	15	7,5
17																	-	2,5	2,5	10
18																		-	5	12,5
19																			-	12,5
20																				-

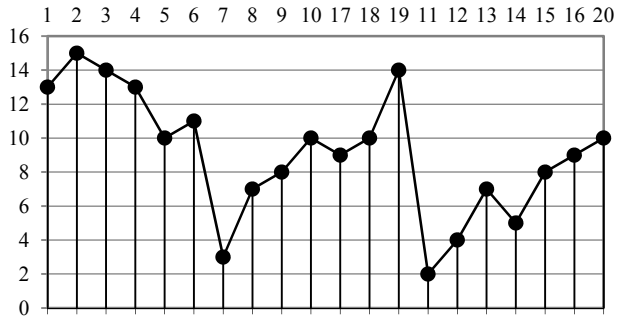


Рис. 7. Спектр близькості елементів по множинам

Програмне забезпечення, що реалізує вказаний алгоритм, взято із курсу "Прикладний аналіз"[2] Малюков М.М.

Результуюча матриця B(I), C(I).

I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
B(I)	5	4	2	6	7	1	2	18	19	17	8	9	10	11	15	16	12	13	14	20
C(I)	-	8	7	6.2	6.1	1.2	1.6	2.7	3.2	3.5	2	2.4	2.9	2.6	3	3.2	3.6	3.6	3.8	3.2

Отримане розбиття пов'язано з тим, що в даному випадку алгоритм ««Спектр»» забезпечує не глобальний, а локальний екстремум якості розбиття елементів вихідної множини. Дійсно, за визначенням алгоритм «Спектр» агрегує елементи з найбільшими зв'язками, так що до кінця роботи алгоритма в числі не вибраних неминуче залишаються тільки слабозвязані елементи. Виражаємо отримані матриці на конфігурації досліджуваної мережі:

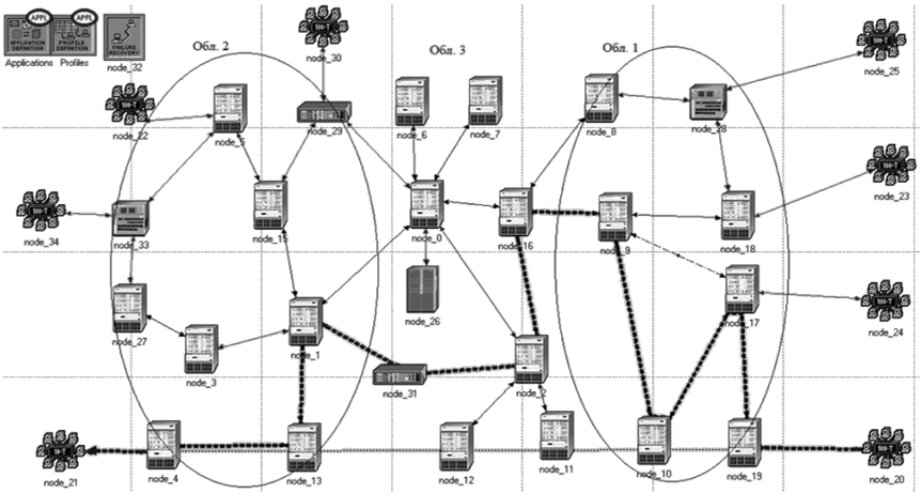


Рис. 8. Виділення внутрішніх областей

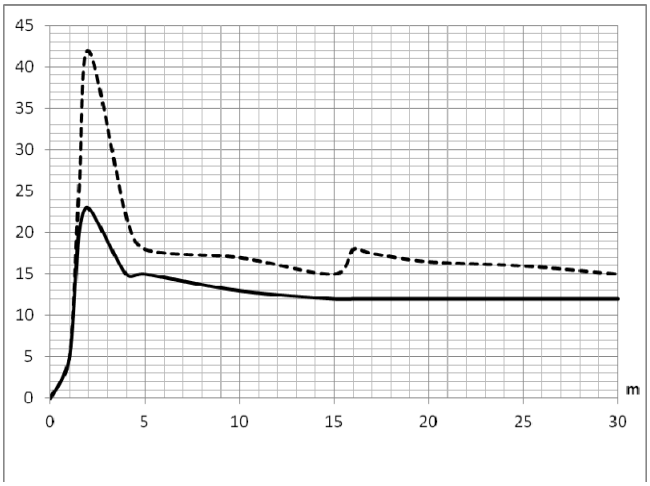


Рис. 9. Рис. Time_average (in IP.Traffic Dropped(packets/sec))

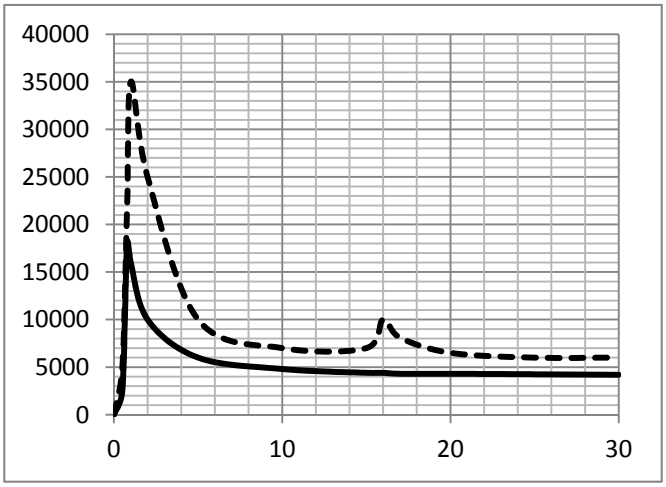


Рис. 10. Time_average (in OSPF.Traffic Sent (bits/sec))

Криві 1, 3 – відображають поведінку системи до початку експерименту. На 15–ій хвилині було спровоковано несправність вузла Nod_18, що призвело до появи додаткової сервісної інформації, тобто загальна кількість пакетів та втрачених пакетів збільшилась. Для мережі з виділеними областями (криві 2, 4 рис. 9-10) загальна кількість пакетів значно меньша так як в цій ситуації ми не враховуємо внутрішній сервісний трафік. В описаній ситуації зовнішні області не реагують на зникнення Nod_18, так як він інкапсульований до виділеної області №3.

Таким чином було опробовано та досліджено застосування алгоритму прикладного аналізу даних для вирішення задачі виділення внутрішніх областей в мережі побудованої на базі протоколу OSPF. Теоретичні положення протоколу підтвердились даними експерименту, що підтвердило можливість використання запропонованого методу автоматичної класифікації. Інформаційна модель мережі та всі аналітичні розрахунки проведено в середовищі Opnet. В подальшому необхідно порівняння методів автоматичної класифікації та їх реалізація в стек протоколу

Список літератури

1. Том М. Томас II Структура и реализация сетей на основе протокола OSPF. Руководство Cisco – OSPF Network Design Solutions. – 2-е изд. – М.: «Вильямс», 2004. – С. 816. – ISBN 1-58705-032-3. С 83-94.
2. Лекции по прикладному анализу данных / Задача классификации и ее постановка / Алгоритм автоматической классификации «СПЕКТР» / Малюков Н.Н., [Електронний ресурс], режим доступу до журналу, http://prand.ru/content/algoritm-avtomaticheskoi-klassifikatsii-spektr-chast-4
3. Протоколы маршрутизации / Семенов Ю.А., [Електронний ресурс], режим доступу до журналу, http://book.itep.ru/4/44/rut_4411.htm .
4. «Маршрутизаторы Cisco. Руководство по конфигурированию».– **2**-*е издание*,– Дэвид Хьюкаби, Стив Мак-Квери, Эндрю Уайтейкер.– 736 стр.,– ISBN 978-5-8459-1755-3, -«ВИЛЬЯМС», 2012. С 100-125.
5. J. Moy. Ascend Communications, Inc. April 1998. OSPF Version 2, [Електронний ресурс], режим доступу до журналу, http://www.ietf.org/rfc/rfc2328.txt.
Полторак В.П. Моделювання процесу маршрутизації в IP мережах з використанням методу динамічного програмування / К.С.Дорошенко, О.С.Квітко, В.П.Полторак // Вісник ЖДТУ / Технічні науки. – 2010. – №2 (53). – С. 154-160: ил.6. – Табл. 2. – Библиогр.

УДК 519.85

ШОВГУН Н.В.

АНАЛІЗ КРЕДИТОСПРОМОЖНОСТІ ПОЗИЧАЛЬНИКА ЗА ДОПОМОГОЮ МЕТОДІВ З НЕЧІТКОЮ ЛОГІКОЮ

В статті зроблено аналіз найбільш широко застосованих методів оцінки кредитоспроможності позичальника. Розглянуті методи оцінки кредитоспроможності на основі нечіткої логіки. Показані результати оцінки кредитоспроможності позичальника за допомогою методів з нечіткою логікою.

This paper is a survey on the most popular methods for assessing the creditworthiness of the borrower. Also, it presented methods for evaluating creditworthiness based on fuzzy logic and obtained results of assessing the creditworthiness of the borrower using methods of fuzzy logic.

1. Вступ

Основним напрямом діяльності комерційного банку є видача кредитів. По деяким оцінкам кредитування дає майже половину прибутку банку.

Разом з тим, кредитування пов'язано з ризиком, зумовленим можливим невиконання своїх обов'язків позичальником. Відомо, що цей ризик є один з найзначніших ризиків банку.

Споживче кредитування фізичних осіб є одним з основних банківських продуктів. При цьому для мінімізації втрат банку потрібен ретельний відбір позичальників та ефективна оцінка їх кредитоспроможності. Для цього в банках існують моделі і методи прийняття рішень по кредитним заявкам.

Насамперед, для прийняття рішення має значення детальна анкета позичальника, у якій містяться багато даних – від матеріального положення позичальника до його особистих якостей. З врахуванням усіх цих даних прийняте рішення повинно мінімізувати ризик та одночасно не повинно мати наслідком необґрунтовану втрату позичальника.

Умови, у яких має бути прийняте рішення, часто характеризується неповнотою даних, їх різнорідністю та недостатністю.

Метою даної статті є дослідження існуючих підходів до визначення кредитоспроможності позичальника, аналіз їх особливостей та вибір методик, які найбільш ефективно дозволяють прийняти рішення у конкретних умовах діяльності комерційного банку. Запропоновано використовувати для аналізу кредитоспроможності фізичних осіб нечіткий контролер Мамдані та нечітку нейрону мережу TSK з механізмом нечіткого логічного висновку Сугено.

2. Аналіз відомих методів оцінки кредитоспроможності позичальника

Найчастіше для оцінки кредито- спроможності позичальника використовують:

– статистичні методи;
– дерева рішень;
– генетичні алгоритми;
– нейронні мережі.

Статистичні методи на основі дискримінантного аналізу використовуються при вирішенні задач класифікації. Так, відома модель Альтмана, що використовується для аналізу кредитоспроможності юридичних осіб, побудована на основі множинного дискримінантного аналізу. Для аналізу кредитоспроможності фізичних осіб використовують лінійну або логістичну регресії [1]. Наприклад, при використанні лінійної регресії, функція, що визначає кредитний рейтинг, апроксимується лінійною функцією щодо компонентів вектора характеристик позичальника, тобто

$$p = a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + \ldots a_N \cdot x_N,$$

де a_0 – вільний член;

a_i при $i = 1,\ldots, N$ – вагові коефіцієнти характеристик позичальника;

x_i – характеристики позичальника.

Всі регресійні методи чутливі до кореляції між характеристиками, тому в моделі не повинно бути сильно корельованих характеристик позичальника.

Статистичні методи спираються на усереднені характеристики вибірки, але при дослідженні реальних складних життєвих феноменів ці характеристики можуть не відповідати дійсності. Оцінка кредитоспроможності позичальника за допомогою цих методів потребує вели-

кої кількості історичних данних щодо кредитів, що не завжди можливо. Також характерною є проблема недостатньої кількості прикладів позичальників, що виявилися не спроможними погасити свою заборгованість. Одне з вирішень цієї проблеми запропоновано в [2]. Суттєвим недоліком статистичних методів вважають вимоги до спеціальної підготовки користувача [3].

Метод дерев рішень відрізняється високою швидкістю обробки даних і навчання при збереженні властивостей систем нечіткого логічного висновку.

При використанні методу дерев рішень для класифікації кредитних заявок застосовується набір правил, що формується при побудові дерева на основі навчальної вибірки. Дерево включає взаємопов'язані початковий (кореневий), проміжні та кінцеві вузли Кожному з вузлів відповідає умова (правило) класифікації об'єктів. Для побудови дерева на кожному внутрішньому вузлі необхідно знайти таку умову, яка б розбивало множину, асоційовану з цим вузлом на підмножину. В якості такої перевірки повинен бути вибрана одна з характеристик. Обрана характеристика повинна розбити множину так, щоб одержані в підсумку підмножини складалися з об'єктів, що належать до одного класу, або були максимально наближеними до цього, тобто кількість об'єктів з інших класів в кожній з цих множин було якомога менше.

Але у метода дерев рішень існують суттєві недоліки. Він не підходять для задач з великим числом можливих розв’язків і умова (правило) може формулюватися тільки в термінах «більше/менше», що заважає застосуванню до задач, де класс визначається більш складним поєднанням змінних [4].

Генетичні алгоритми грунтуються на стохастичному пошуку глобального оптимума цільової функції. Ідея генетичних алгоритмів запозичена в живої природи і полягає в організації еволюційного процесу (за допомогою операцій схрещування, мутації та селекції), кінцевою ціллю якого є оптимальне рішення задачі.

Генетичні алгоритми мають ряд недоліків. Критерій відбору хромосом і сама процедура відбору евристичні і не гарантують знаходження «кращого» рішення. Також необхідно мати в наявності досить великий обсяг вхідних данних для завершення процедури селекції [5].

Під нейронними мережами розуміють обчислювальні структури, які моделюють прості біологічні процеси, що зазвичай асоціюють з процессами в людському мозку. Вони являють собою совокупність елементів (штучних нейронів), пов'язаних між собою синаптичними зв'язками [6].

Недоліком нейронних мереж називають те, що вони являють собою «чорний ящик» та відсутність твердих правил, щодо вибору швидкості навчання мережі для вирішення конкретних задач [6].

Існує багато архітектур нейронних мереж. Так наприклад, в роботі [7] для класифікації клієнтів німецького та австралійського банків використовувалися такі мережі: мережа Кохонена, мережа BackPropagation, радіальнобазисна мережа, каскадна мережа.

Загальними недоліками вищеописаних методів є вимоги до об’єму вхідних даних та жорсткі вимоги до характеристик і критеріїв відбору позичальника. В реальному житті середовище позичальника постійно змінюється, як результат – змінюються його сімейний та фінансовий стан, що вносить невизначеність в інформацію щодо клієнта. Неправильна оцінка позичальника в цих умовах може призвести до збільшення ризику банку чи втрати потенційно надійних клієнтів.

Врахувати таку невизначеність можна за допомогою методів нечіткої логіки. Даний підхід дає можливість працювати як з кількісними, так і з якісними характеристиками.

3. Методи оцінки кредитоспроможності позичальника за допомогою нечіткої логіки

Задачу оцінки кредитоспроможності можна сформулювати таким чином. Кожна кредитна заявка задається вектором $\{X_1, X_2, ..., X_i, ..., X_N\}$, де X_i – певним чином формалізовані дані з анкети позичальника та параметри кредиту. Далі по заданому вектору треба прийняти рішення про надання кредиту, тобто класифікувати позичальника як «надійного», чи класифікувати як «поганого».

В статті [8] використовувався нечіткий контролер з алгоритмом нечіткого логічного висновку Мамдані для аналізу кредитоспроможності юридичних осіб. В загальному вигляді нечіткий логічний висновок має наступні етапи:

1. Визначення множини вхідних змінних:
$$X = \{X_1, X_2, ..., X_i, ..., X_M\};$$
2. Визначення множини вихідних змінних:
$$D = \{D_1, D_2, ..., D_i, ..., D_M\};$$

3. Формування базової терм-множини з відповідними функціями належності кожного терма: $A = \{a_1, a_2, ..., a_i\}$;

4. Формування кінцевої множини нечітких правил, узгоджених щодо використовуваних в них змінних;

5. Знаходження чіткого значення для кожної з вихідних лінгвістичних змінних.

Даний підхід можна використати і для розв'язання задачі оцінки кредитоспроможності фізичних осіб.

Розглянемо алгоритм нечіткого висновку Мамдані.

Нехай базу знань складають два нечітких правила:

$П_1$: якщо x є A_1 і y є B_1, то z є C_1,

$П_2$: якщо x є A_2 і y є B_2, то z є C_2,

де x і y вхідні змінні, z – вихідна змінна, A_1, A_2, B_1, B_2, C_1, C_2 – деякі задані функції належності, при цьому чітке значення z треба визначити на основі даної інформації і чітких значень x, y. Етапами алгоритму є:

1. Введення нечіткості. Знаходимо степінь істиності для передумов кожного правила: $A_1(x_0)$, $A_2(x_0)$, $B_1(x_0)$, $B_2(x_0)$.

2. Логічне виведення. Знаходимо рівні «відсікання» для передумов кожного з правил (з використанням операції мінімума):

$\alpha_1 = A_1(x_0) \cap B_1(y_0)$;

$\alpha_2 = A_2(x_0) \cap B_2(y_0)$.

Далі знаходимо «відсікання» функції належності:

$C_1' = (\alpha_1 \cap C_1(z))$;

$C_2' = (\alpha_2 \cap C_2(z))$.

3. Композиція. Знаходимо об'єднання знайдених відсічених функцій належності з використанням операції максимум, отримуємо підсумкову нечітку підмножину для змінної виходу з функцією належності:

$$\mu_\Sigma = C(z) = C_1'(z) \cup C_2'(z) = (\alpha_1 \cap C_1(z)) \cup (\alpha_2 \cap C_2(z)).$$

4. Зведення до чіткості з використанням, наприклад, центроїдного методу [8].

В алгоритмі нечіткого висновку Сугено, використовується такий набір правил [9]:

$П_1$: якщо x є A_1 і y є B_1, то $z=a_1x+b_1y$,

$П_2$: якщо x є A_2 і y є B_2, то $z=a_2x+b_2y$,

де x і y вхідні змінні, z – вихідна змінна, A_1, A_2, B_1, B_2, C_1, C_2 – деякі задані функції належності, a_1, a_2, b_1, b_2 – деякі числа.

Алгоритм має вигляд:

1. Введення нечіткості як в алгоритмі Мамдані.

2. Нечітке виведення. Знаходимо $\alpha_1 = A_1(x_0) \cap B_1(y_0)$, $\alpha_2 = A_2(x_0) \cap B_2(y_0)$ та індивідуальні виходи правил:

$\dot{z}_1 = a_1 x_0 + b_1 y_0$;

$\dot{z}_2 = a_2 x_0 + b_2 y_0$.

3. Визначення чіткого значення змінної виведення:

$$z_0 = \frac{\alpha_1 \dot{z}_1 + \alpha_2 \dot{z}_2}{\alpha_1 + \alpha_1}$$

Розглянемо нечітку нейронну мережу TSK (Takagi, Sugeno, Kang'a) з механізмом нечіткого логічного висновку Сугено. Правила мережі можна представити у такому вигляді [9]:

$$R_1 : якщо \ \ x_1 \ \epsilon \ A_1^{(1)}; x_2 \ \epsilon A_2^{(1)}, ..., x_n \ \epsilon \ A_n^{(1)},$$

$$то \ y_1 = p_{10} + \sum_{j=1}^{N} p_{1j} x_j;$$

$$R_M : якщо \ \ x_1 \ \epsilon \ A_1^{(M)}; x_2 \ \epsilon \ A_2^{(M)}, ..., x_n \ \epsilon \ A_n^{(M)}, то \ y_M = p_{M0} + \sum_{j=1}^{N} p_{M\ j} x_j,$$

де $A_i^{(k)}$ – значення лінгвістичної змінної x_i для правила R_k з функцією належності

$$\mu_A^{(k)}(x_i) = \frac{1}{1 + \left(\frac{x_i - c_i^{(k)}}{\sigma_i^{(k)}} \right)^{2b_i^{(k)}}}.$$

Композиція результатів має вигляд:

$$y(x) = \frac{\sum_{k=1}^{M} w_k y_k(x)}{\sum_{k=1}^{M} w_k},$$

де $w_k = \mu_A^{(k)}(x)$ – степінь виконання умов правила, $\mu_A^{(k)}(x) = \prod_{j=1}^{N} \left[\frac{1}{1 + \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}}} \right]$.

4. Результати досліджень

Для аналізу кредитоспроможності позичальника за допомогою ННМ TSK та контролера Мамдані використовувалися дані одного з вітчизняних банків по 100 кредитним заявкам за 2010 р.

У якості вхідних змінних використовувався вектор X= {Сума кредиту, Дохід, Термін кредиту, Вік, Термін проживання в квартирі, Освіта}.

Співвідношення навчальної та перевірочної вибірки було вибрано як 70 до 30. Результати досліджень зведені в таблиці 1-2. Результати оцінки кредитоспроможності позичальника за допомогою лінійної та логістичної регресії, кластерного аналізу методом нечітких к-середніх [9] та вище наведених методів представлені в таблиці 3.

Табл. 1. Результати оцінки кредитоспроможності позичальника за допомогою ННМ TSK

Кількість правил	СКО навчальної вибірки	% невірних класифікацій на навчальної вибірці	СКО перевірочної вибірки	% невірних класифікацій на перевірочній вибірці
3	0.0316	2.857	0.0178	6.66
5	0.0149	0	0.0127	3.33
7	0.0194	0	0.027	3.33

Табл. 2. Результати оцінки кредитоспроможності позичальника за допомогою контролера Мамдані

Кількість правил	СКО навчальної вибірки	% невірних класифікацій на навчальної вибірці	СКО перевірочної вибірки	% невірних класифікацій на перевірочній вибірці
3	0.08349	6.66	0.1092	10
5	0.0193	2.857	0.0305	6.66
7	0.0131	1.4286	0.0382	6.66

Табл. 3. Порівняльні результати оцінки кредитоспроможності позичальника різними методами

Метод	СКО навчальної вибірки	% невірних класифікацій на навчальної вибірці	СКО перевірочної вибірки	% невірних класифікацій на перевірочній вибірці
Лінійна регрессія	0.0987	8.57	0.04334	13.33
Кластерний аналіз		8.57		16.66
Логістична регерсія	0.0171	2.857	0.0344	10
ННМ TSK	0.0149	0	0.0127	3.33
НК Мамдані	0.0193	2.857	0.0305	6.66

Як можна бачити з аналізу таблиць, ННМ TSK та контролер Мамдані дають майже однаково задовільні рішення при кількості правил 3-7. Процент невірних класифікацій при застосуванні статистичних методів аналізу тієї самої вибірки надто високий.

5. Висновки

1 в аналізі кредитоспроможності фізично. Основною проблемою ї особи виступає неточність даних, недостатня база знань про минуле клієнтів та необхідність працювати з лінгвістичними характеристиками, які важко піддаються математичній обробці.

2. В специфічних умовах функціонування вітчизняних банків методи з використанням нечіткої логіки, закрема методи на базі нечіткої нейронної мережі TSK та контролера Мамдані дають непогані результати, які дозволяють звести відсоток незадовільних рішень до 3…10%.

Список літератури

1. Руководство по кредитному скорингу / под ред. Элизабет Мэйз;– Минск: Гревцов Паблишер, 2008г. – 458с.
2. Паклин Н.Б., Уланов С.В., Царьков С.В. Построение классификаторов на несбалансированных выборках на примере кредитного скоринга // Искусственный интеллект. – 2010г. – №3 – с. 528-534
3. А.А. Ежов, С.А. Шумкий Нейрокомпьютинг и его применения в экономике и бизнесе. – М.: МИФИ, 1998.– 224 с.
4. Тегеран Т. Программируем колективный разум. – СПб: Символ-Плюс, 2008. – 368 с.
5. В.В. Круглов, М.И. Дли, Р.Ю. Голунов Нечеткая логика и искусственные нейронные сети. – М.: Физматлит, 2000. – 224 с.
6. D. Michie, D.J. Spiegelhalter, C.C. Taylor Machine Learning, Neural and Statistical Classification. 1994
7. Зайченко Ю.П. Оценка кредитних банковских рисков с использованием нечеткой логики// Intelligent Information and Engineering Systems. – 2008. – №13 – с. 190-200
8. Зайченко Ю.П. Нечеткие модели и методы в интеллектуальных системах. – К.: Издательский дом «Слово», 2008. – 334с.

УДК 004.75

КРАВЕЦЬ П.І.,
ШИМКОВИЧ В.М.,
ЗУБЕНКО Г.А.

ТЕХНОЛОГІЯ АПАРАТНО-ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ШТУЧНОГО НЕЙРОНА ТА ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ЗАСОБАМИ FPGA

В даній роботі описана методика програмно-апаратної реалізації штучного нейрону з сигмоїдальною функцією активації засобами FPGA, наведено покроковий алгоритм синтезу штучного нейрону та відповідний займаний ресурс FPGA, розроблена технологія побудови штучних нейронних мереж, зокрема таких як RBF- мережі, динамічні мережі Хопфілда та мережі прямого розповсюдження.

In this paper describes methods of software and hardware implementation of artificial neuron with sigmoid activation function by means of FPGA, provides step by step algorithm of artificial neuron and the corresponding resource occupied FPGA, considered technology of artificial neural networks, in particular such as RBF- network, dynamic network Hopfild and feed-forward neural network.

Розвиток теорії автоматичного управління, як і розвиток будь-якого іншого напрямку науки, характеризується ускладненням розв'язуваних завдань і підвищенням якісних показників необхідних рішень [1]. Традиційні методи управління в основному спираються на теорію лінійних систем, в той час як реальні об'єкти є, за своєю природою, нелінійними. Нейромережеві системи управління являють собою новий високотехнологічний напрямок в теорії управління та відносяться до класу нелінійних динамічних систем [2]. Висока швидкодія за рахунок розпаралелювання вхідної інформації в поєднанні зі здатністю до навчання нейронних мереж робить цю технологію вельми привабливою для створення пристроїв управління в автоматичних системах.

На даний час основним методом реалізації нейромережевих систем управління є програмний, з використанням комп'ютерної техніки чи спеціалізованих контролерів, побудованих на її основі, що значно звужує коло практичної реалізації систем управління із-за значної вартості таких регуляторів і робить їх практично недоцільними та недоступними для використання в простих системах керування, крім того комп'ютерні нейромережеві регулятори мають обмежену швидкодію та потребують значних затрат часу на навчання. Рекурентність і послідовність дій процедури навчання нейромережі при її реалізації на всій множині налагоджуваних параметрів не дозволяє повністю вирішити проблему швидкодії процедури навчання нейромережевих структур в темпі з динамікою об'єкта управління. Єдиною альтернативою цьому є розпаралелення процедури навчання та роботи внутрішніх елементів нейромережевих структур. Такі можливості з'являються при апаратно-програмній реалізації нейромережевих структур побудованих на нейрочіпах чи програмованих логічних інтегрованих структурах (ПЛІС-FPGA) [3].

Із сучасних розробок, виконаних на ПЛІС високої інтеграції, можна відзначити, перш за все, «нейрочіп-8», інструментальну плату XDSP-680 на базі FPGA сімейства Spartan компанії Xilinx з нейромережевою прошивкою, що є спільною розробкою Наукового центру нейрокомп'ютерів (Москва) і «Скан Інжиніринг Телеком» (Воронеж). У розробці цих двох партнерів знаходиться і ряд інших виробів: перспективна розробка «нейрочіп-2000» на базі FPGA Virtex / Virtex-E, а також цілий набір інструментальних плат і на базі ПЛІС різних серій і мезонінних модулів різного призначення, що дозволяють швидко і ефективно створювати обчислювальні системи різного функціонального призначення [4, 5].

Метою даної роботи є розробка технології та методик апаратно-програмної реалізації нейромережевих структур на FPGA для синтезу нейромережевих регуляторів систем управління, що можуть функціонувати в темпі з процесом управління складними динамічними об'єктами.

Однією з проблем при апаратно-програмній реалізації нейромережевих структур є реаліза-

ція штучного нейрона і його активаційних функцій засобами FPGA.

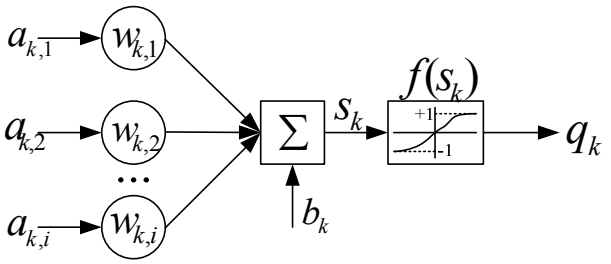


Рис. 1. Схема штучного нейрону

де q_k – вихідний сигнал *к*-го нейрона;

f – активаційна функція нейрона;

$a_{k,i}$ – вхідні сигнали *к*–го нейрона;

$w_{k,i}$ – синаптична вага *к*–го нейрона;

b_k – зміщення *к*–го нейрона.

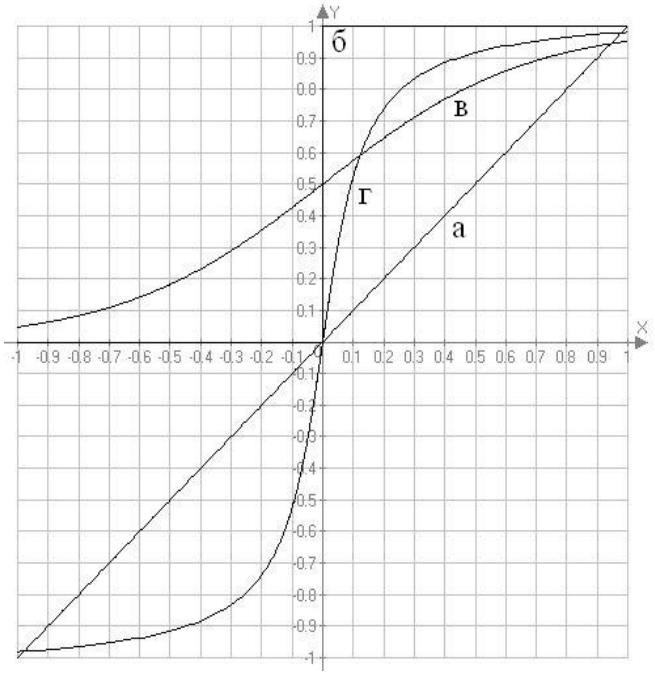


Рис. 2. Функції активації штучних нейронів

В штучних нейронних мережах найбільше поширення (по аналогії з біологічними процесами) отримали функції активації такі як лінійна (рис. 2,а), порогова (рис. 2,б) та сигмоїдальна (рис. 2.в).

Сигмоїдальні функція це найбільш ширше використовуваний тип активаційних функцій. Сигмоїдальні функції є монотонно зростаючими, безперервними і диференційованими.

Під сигмоїдальними функціями розуміється клас функцій, які описуються виразом:

$$f(x,k,b,T,c) = k + \frac{c}{1+be^{Tk}}, \qquad (1)$$

де x,k,b,T,c – параметри $k \in R$; $b \in R, b > 0$; $T,c \in R \setminus \{0\}$.

Якщо прийняти *k*=0, *c*=1, *b*=1, і *T*=-1, то вираз (4) прийме вигляд:

$$f(x,0,1,-1,1) = 0 + \frac{1}{1+1e^{-1x}} = \frac{1}{1+e^{-x}}. \qquad (2)$$

Функція, описана виразом (2) називається «класичний» сигмоїд (рис. 2.в).

Якщо прийняти *k*=1, *c*=-2, *b*=1, и *T*=2, то вираз (2) прийме вигляд:

$$f(x,1,1,2,-2) = 1 + \frac{-2}{1+1e^{2x}} = \frac{1+e^{2x}-2}{1+e^{2x}} = \frac{e^{2x}-1}{e^{2x}+1} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad (3)$$

Функція, описана виразом (3) називається гіперболічний тангенс (рис. 2.г).

Найскладнішою з погляду цифрової реалізації є сигмоїдальні функції активації.

Існуючі підходи цифрової реалізації нелінійних функцій використовують різні методи апроксимації, такі як табличний метод, розкладання в ряд Тейлора, шматково-лінійна апроксимація і т.д. Розглянемо детальніше сигмоїдальну функцію:

1) $f(x) = \dfrac{1}{1+e^{-x}}$

$$f(-x) = 1 - f(x) = 1 - \frac{1}{1+e^{-x}} = 1 - \frac{1}{1+\frac{1}{e^{x}}} =$$

$$= 1 - \frac{e^{x}}{e^{x}+1} = \frac{e^{x}+1}{e^{x}+1} - \frac{e^{x}}{e^{x}+1} = \frac{1}{e^{x}+1}$$

отримаємо:

$$1 - \frac{1}{1+e^{-x}} = \frac{1}{1+e^{x}}, \quad \text{або} \quad 1 - f(x) = f(-x). \qquad (4)$$

Можемо розглядати $f(x)$ тільки для додатних аргументів. Для від'ємних його значення можна знайти по формулі (4).

2) Для додатних аргументів функція приймає значення в діапазоні від 0,5 до 1. Таким чином якщо розрахувати значення функції з точністю до 2-х знаків після коми можливо всього 50 різних значень (0.50;0.51;0.52;…1.00).

Якщо аргумент функції також візьмемо з точністю до 2-х знаків після коми отримаємо таблицю 1:

Табл. 1. Значення сигмоїдальної функції та її аргументу відповідно

x	0	0.01	0.02	0.03	0.04	0.05	…	5.29	5.30
$f(x)$	0.50	0.50	0.50	0.51	0.51	0.51	…	0.99	1.00

Для всіх аргументів $x > 5.3$, $f(x) = 1$. З отриманої таблиці можемо вибрати стовбці в яких значення $f(x)$ змінюється по відношенню до попереднього стовпця та отримати таблицю 2.

Табл. 2. Оптимізована таблиця значень сигмоїдальної функції та її аргументу відповідно

N	0	1	2	3	…	50
x	0	0.03	0.07	0.11	…	5.30
$f(x)$	0.50	0.51	0.52	0.53	…	1.00

Значення $f(x)$ в отриманій таблиці змінюється на 0.01 від стовпця до стовпця, тому:

1. Таких стовпців буде 50, по кількості можливих значень $f(x)$;
2. Значення $f(x)$ для кожного стовпця можна розрахувати по номеру N цього стовпця за формулою:

$$f(x) = 0.5 + 0.01 \cdot N. \qquad (5)$$

Значення функції $f(x)$ таким чином можливо розрахувати не тільки для одного елемента першої таблиці а для групи елементів, для яких $X_N \le X \le X_{N-1}$, де X_N, X_{N-1} – значення x для N, N-1 стовпців другої таблиці. Наприклад x=0.09, 0.09 знаходиться в між значеннями $0.07 \le 0.09 \le 0.11$, 0.07=x_2, 0.11=x_3, тобто N=2, а $f(x)$=0.5+0.02=0.52.

Таким чином на основі масиву з 50 елементів можемо знайти апроксимацію функції $f(x) = \dfrac{1}{1+e^{-x}}$ для будь-якого аргументу з точністю до 2-х знаків після коми.

Алгоритм реалізації штучного нейрону з сигмоїдальною функцією активації на ПЛІС:

Крок 1. Задаємо матрицю синаптичних ваг штучного нейрону W, та вектор констант x відповідно до таблиці 2.

Крок 2. Визначаємо локальні змінні *lin*, *lout*. Для змінної *lout* задаємо початкове значення 100, що у множині дійсних чисел відповідає одиниці.

Крок 3. Змінній *lin* задаємо значення, яке матимемо на виході суматора, що входить до складу штучного нейрона. Це значення можна розрахувати за формулою:

$$lin = \frac{\left(\sum_{i=1}^{n} in_i \cdot W_i\right)}{128}$$

де n – кількість вхідних зв'язків нейрону
in_i – значення на і-тому вході
W_i – вага відповідного входу

Крок 4. Для розрахунку функції активації ввести лічильник. Задати a=0.

Крок 5. Порівняти модуль *lin* з елементами матриці констант (x), x_a, x_{a+1}. Якщо виконується умова $x_a \le |lin| < x_{a+1}$, розрахувати вихідне значення нейрону за формулою (5) *lout*=50+a. Перейти на крок 7. Якщо умова не виконується перейти на наступний крок 6.

Крок 6. Збільшити a на 1. Якщо a=50 перейти на крок 7, інакше – на крок 5. При a=50 вхідне значення сигмоїдальної функції більше ніж 5.3, в цьому випадку її вихід дорівнює 1 (початкове значення).

Крок 7. Якщо *lin*<0, змінити вихідне значення відповідно формулі (4): *lout*=100-*lin*.

Крок 8. Задати значення вихідного сигналу штучного нейрона рівним отриманому значенню локальної змінної.

Побудований штучний нейрон з сигмоїдальною функцією активації на ПЛІС за даною технологією з кроком квантування 0.01 зайняв 765 LUTs (Look Up Table – вентилі логічної матриці). Похибка абсолютна ±0.005. Також було промодельовано штучні нейрони з сигмоїдальною функцією активації на ПЛІС за даною технологією з кроком квантування 0.05 та 0.1 з такими крокоми було використано ресурсу ПЛІС 275 та 179 LUTs відповідно. При зменшенні чи збільшенні кроку квантування значень таблиці 2 буде змінено відповідно і використаний ресурс ПЛІС для одного нейрону.

Далі розглянемо технологію побудови штучних нейронних мереж на ПЛІС.

Однією з головних особливостей нейромереж є паралельна обробка сигналів. Багатошарові нейронні мережі представляють собою однорідну обчислювальну середу. По термінології нейроінформатики це універсальні паралельні обчислювальні структури, призначенні для вирішення самих різних класів задач. Розглянемо

концепцію реалізації нейронних мереж на ПЛІС.

На сьогоднішній день розроблено та досліджено декілька десятків типів штучних нейронних мереж, але основними, принципово різними типами є три типи мереж, що відповідають трьом методам їх навчання, це RBF-мережі, динамічні мережі Хопфілда та мережі прямого розповсюдження.

Основна структура багатошарових нейронних мереж прямого розповсюдження зображена на рис. 3. У повнозв'язних нейронних мережах прямого розповсюдження виходи базових елементів кожного шару сполучені зі всіма входами всіх базових елементів наступного шару.

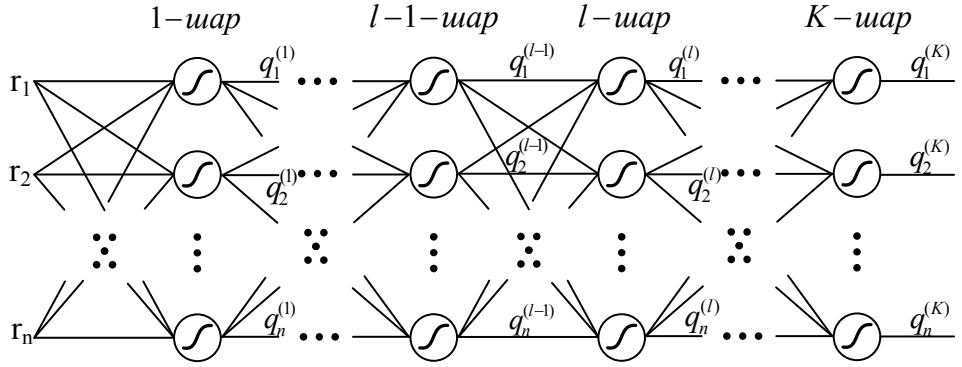


Рис. 3. Структурна схема багатошарової нейронної мережі прямого розповсюдження

У силу теорем Колмогорова-Арнольда та Хехт-Нільсена [7] трьох шарів, вхідний, вихідний та один прихований шар, достатньо для одержання ефективного рішення за допомогою нейронних мереж прямого поширення.

При апаратно-програмній реалізації багатошарових нейронних мереж прямого розповсюдження для всіх нейронів з однаковою нелінійною функцією активації використовується одна і та ж матриця констант, для апроксимації цієї функції. Кожний нейрон представляється на мові VHDL окремим процесом. Мова програмування дозволяє явно вказувати сигнали, які запускають процес. Для запуску нейрона використовується вхідний сигнал цього нейрона. Було промодельовано такі нейромережі, і відповідно займаний ними ресурс FPGA в кількості вентилів логічної матриці LUTs: 1-1-1 – 797, 1-2-1 – 829, 2-2-1 – 1728, 3-2-1 – 2544, 4-2-1 – 3497, 1-3-1 – 941, 1-4-1 – 959, 1-2-2 – 940, 1-2-3 – 970, 1-2-5 – 996. Нейронні мережі представлені трьома числами, де перше кількість нейронів в вхідному шарі, друге кількість нейронів в прихованому шарі і третє кількість нейронів в вихідному шарі. Отримані значення в LUTs можуть дещо змінюватись при зміні констант, які задають синаптичні ваги нейронів.

Мережі зі зворотним розповсюдженням відрізняються від нейромереж прямого розповсюдження наявністю каналів, по яких інформація надходить і у зворотному напрямі від виходів на входи буферного шару. Прийнято виділяти два класи багатошарових нейромереж зі зворотним розповсюдженням: рекурентні та рециркуляційні.

Рекурентні нейромережі також передають сигнали у зворотному напрямі, але по каналах зворотного зв'язку. На даний час розроблено і досліджено ряд модифікацій рекурентних мереж. Найбільше поширення серед рекурентних мереж набули релаксаційні мережі із зворотними зв'язками, названі мережами Хопфілда [2].

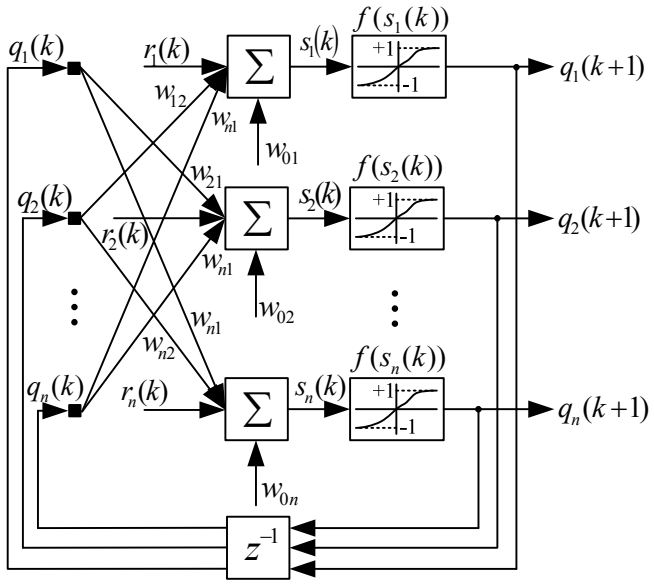


Рис. 4. Структурна схема одношарової мережі Хопфілда

У мережах Хопфілда (рис. 4) відбувається динамічне перетворення в часі вхідного сигналу $\mathbf{r} \in R^n$ в $\mathbf{q} \in R^{N_K}$. Структурна схема динамічної мережі Хопфілда зображена на рисунку 4.

У векторно-матричній формі статика мережі Хопфілда описується співвідношенням:

$$\mathbf{s}(k) = \mathbf{W}\mathbf{q}(k) + \mathbf{r} - \mathbf{w}_0,$$

де компоненти вектора функції s(k) обчислюються як:

$$s_i(k) = \sum_{j=1, j\neq i}^{n} w_{i,j} q_j + r_i - w_{0i}, \quad i = \overline{1,n}.$$

Динаміка мережі Хопфілда забезпечується введенням зворотних зв'язків з виходів $q_j(k)$ на входи базових елементів ($i \neq j$) через елементи затримки $z^{-1} = e^{-\Delta ts}$ (Δt – період дискретизації безперервних функцій $q_j(k)$).

Апаратно-програмна реалізація нейронних мереж Хопфілда на FPGA відрізняється від апаратно-програмної реалізації багатошарових нейронних мереж прямого розповсюдження введенням додаткових зворотних зв'язків і елементів затримки в часі на цих зв'язках, що в свою чергу займає більше простору на FPGA. Було змодельовано нейромережі Хопфілда з одним шаром і трьома нейронами в ньому і двома шарами по три нейрони в кожному використаний ресурс FPGA становив відповідно 2.521 і 4.945 LUTs.

RBF-мережі є також універсальними апроксиматорами і при незначних обмеженнях можуть бути застосовані для апроксимації будь-якої безперервної функції [2, 8, 9]. RBF-мережі по своїй структурі відносяться до двошарових мереж, в яких використовується прихований шар з фіксованим нелінійним перетворенням вектора входу з постійними ваговими коефіцієнтами. Цей шар здійснює статичне відображення вхідних змінних $\mathbf{r} \in R^n$ у нові змінні $\mathbf{q}_1^{(l)} = col(q_1^{(l)}, ..., q_m^{(l)})$. Другий, лінійний вихідний, шар «зважує» ці змінні з вагами, що налагоджуються, $w_i = col(w_{i,1}, w_{i,2}, ..., w_{i,m})$, таким чином, що, наприклад, i-й скалярний вихід RBF-мережі записується у вигляді:

$$q_i^{(2)} = q_i = F(\mathbf{r}) = \sum_{j=1}^{m} w_{i,j} f_i(\mathbf{r}) + w_0 \cdot 1, \quad i = 1,2,...,K$$

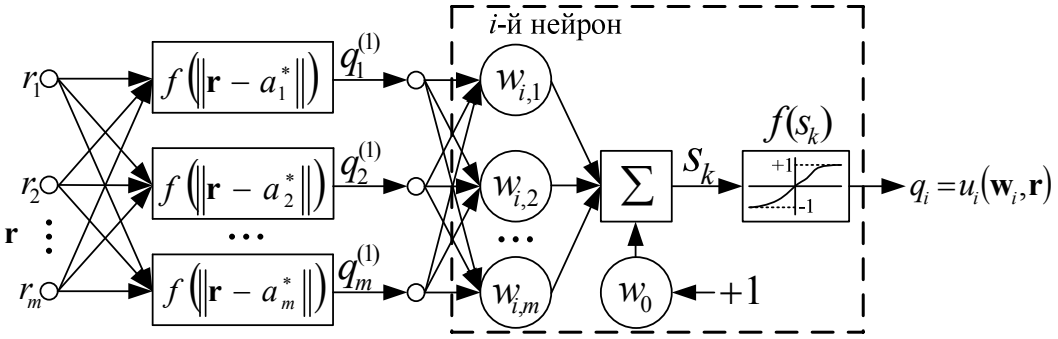


Рис. 5. Структура RBF-мережі

Структура RBF-мережі з вектором входу $\mathbf{r} \in R^n$ і вектором виходу $\mathbf{q}^{(2)} = \mathbf{q} \in R^n$ зображена на рисунку 5.

В якості функцій $f(z)$ на практиці найбільше використовується функція Гауса:

$$f_j(\mathbf{r}) = \exp\left(\sum_{k=1}^{n} 0.5\sigma_{j,k}^{-2} \left\| \mathbf{r} - \mathbf{a}_j^* \right\|^2 \right),$$

де $a_j^* = col(a^*_{j,1}, a^*_{j,2}, \cdots, a^*_{j,n})$ – числовий вектор, а $\sigma_{j,k}$ – « ширина» функції Гауса. Специфіка RBF-мережі полягає у тому, що в робочому режимі в них налагоджуються тільки вагові коефіцієнти лінійного вихідного шару. Помилка апроксимації обчислюється безпосередньо на виході мережі так само, як і в БНМ, але налагодження тільки одного, лінійного по параметрах налагодження шару знімає проблему пошуку глобального мінімуму функціонала навчання (як правило, квадратичного) і сприяє швидкій збіжності процесу навчання мережі.

Проблемою RBF-мереж є вибір числа радіально-базисних функцій, необхідних для апроксимації, і ця обставина стає критичним чинником у використовуванні RBF-мереж в задачах ідентифікації і управління, де необхідна інформація для визначення розмірності RBF-мереж як правило, відсутня. Число необхідних радіально-базисних функцій росте експоненціально із зростанням числа вхідних змінних, тобто із зростанням n. Звідси витікає висновок про застосовність RBF-мереж в тих задачах, де розмірність вхідної множини (вектора $\mathbf{r}$) обмежена і відома (в БНМ таке обмеження відсутнє). Інша проблема застосування RBF-мереж полягає в необхідності попереднього налагодження прихованого шару. Також однією з основних про-

блем при програмно-апаратній реалізації таких нейромережевих структур є реалізація прихованого шару та нелінійних функцій активації $f_i(\mathbf{r})$, зокрема реалізації функції Гауса.

Для реалізації функції Гауса, виведемо її через сигмоїду. Така реалізація функції Гауса дозволить використовувати одну і ту ж таблицю констант для функції активації нейрона вхідного шару RBF-мережі та функції активації нейрона вихідного шару RBF-мережі, для більш оптимального використання ресурсу FPGA.

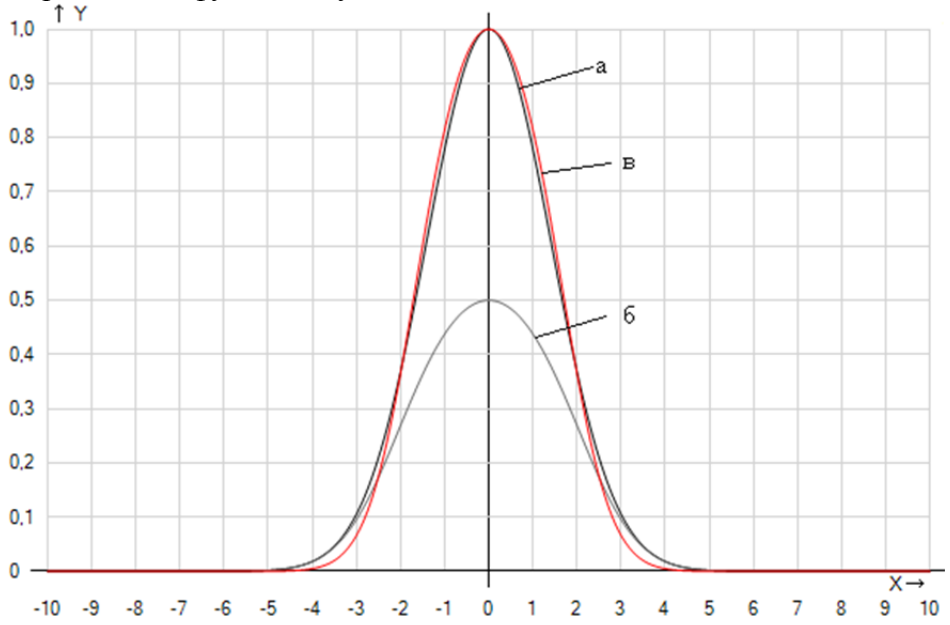


Рис. 6. Функція Гауса

Графік функції Гауса зображено на рисунку 6.а, а описується вона так:

$$y = \exp\left(-\frac{x^2}{\sigma^2}\right),$$

Якщо записати замість експоненти сигмоїду, отримаємо графік зображений на рисунку 6.б:

$$y = \mathrm{sigm}\left(-\frac{x^2}{\sigma^2}\right),$$

Якщо модернізувати цю формулу за допомогою коефіцієнтів при x і σ та помножити її на 2, отримаємо функцію схожу на функцію Гауса (рис. 6.в):

$$y = 2 \cdot \mathrm{sigm}\left(-\frac{192 \cdot x^2}{128 \cdot \sigma^2}\right)$$

Максимальне відхилення від значення функції Гауса, при даній інтерпретації ≈0.0404.

В якості x в дану функцію передається значення $x = \|\mathbf{r} - \mathbf{a}\|$, де $\|\mathbf{r} - \mathbf{a}\|$ – норма вектора. Норму вектора можливо розраховувати різними методами, але найчастіше використовується Евклідова норма:

$$\mathbf{x} = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\sum_{i=1}^{n} (x_i - a_i)^2},$$

тоді

$$x^2 = \sum_{i=1}^{n} (x_i - a_i)^2$$

Таким чином функція активації нейрона вхідного шару RBF-мережі:

$$f(r) = 2 \cdot \mathrm{sigm}\left(-\frac{192}{128} \cdot \frac{\sum_{i=1}^{n} (r_i - a_i)^2}{\sigma^2}\right). \qquad (6)$$

Згідно (6) необхідно виконати операцію ділення на σ^2. На мові VHDL операція ділення добре працює тільки коли дільник рівний степені 2. Для того щоб реалізувати операцію ділення на довільне число σ^2 $\left(y = \frac{x}{\sigma^2}\right)$ можна виконати операцію:

$$\frac{x}{\sigma^2} = \frac{x \cdot sigma}{2^{13}} = \frac{x \cdot sigma}{8192}$$

sigma можна розрахувати за формулою:

$$sigma = \frac{8192}{\sigma^2},$$

тоді

$$\sigma^2 = \frac{8192}{sigma} \qquad (7)$$

Якщо задати константу σ^2 в такому вигляді можна використовувати стандартні операції

мови VHDL. Якщо підставити (7) в (6) отримаємо :

$$f(r)=2\cdot\mathrm{sigm}\left(-\frac{192}{128}\cdot\frac{\sum_{i=1}^{n}(r_i-a_i)^2\cdot sigma}{8192}\right)$$

Таким чином вхідне значення нейрона:

$$lin=\frac{192}{128}\cdot\frac{\sum_{i=1}^{n}(r_i-a_i)^2\cdot sigma}{8192} \quad (8)$$

а функція активації:

$$lout=2\cdot sigm(-lin) \quad (9)$$

З (8) видно що *lin* завжди додатне. Тому сигмо їда у (9) завжди обчислюється для від'ємного числа.

Формула (6) дозволяє обчислити значення для додатного числа. Врахувавши (4) отримаємо аналогічну формулу для від'ємних чисел:

$$f(-x)=1-f(x)=1-(0.5+0.01\cdot N)=0.5-0.01\cdot N$$

Отже згідно (9):

$$lout=2\cdot f(-x)=1-0.02\cdot N \quad (10)$$

Далі розглянемо алгоритм реалізації нейрону прихованого шару з функцією Гауса на ПЛІС.

Крок 1. Задаємо вектор констант *x* відповідно до таблиці 2, *a* і константу *sigma*.

Крок 2. Визначаємо локальні змінні *lin*, *lout*. Для змінної *lout* задаємо початкове значення 0.

Крок 3. Змінній *lin* задаємо значення згідно (8).

Крок 4. Для розрахунку функції активації ввести лічильник. Задати *b*=0.

Крок 5. Порівняти *lin* з елементами вектору констант *x*. Якщо виконується умова $x_b \le lin < x_{b+1}$, розрахувати вихідне значення нейрону за формулою (10) *lout*=100 – 2*b*, перейти на крок 7, інакше перейти на наступний крок.

Крок 6. Збільшити *b* на 1. Якщо *b* =50 перейти на крок 7, інакше – на крок 5. При *b* =50 вхідне значення функції меньше ніж -5.3, в цьому випадку її вихід дорівнює 0 (початкове значення).

Крок 7. Задати значення вихідного сигналу штучного нейрона рівним отриманому значенню локальної змінної.

На FPGA промодельовано RBF-мережу з трьома вхідними нейронами з функцією активації Гауса і одним нейроном з сигмоїдальною функцією в вихідному шарі. Така нейромережа зайняла 3.137 LUTs.

Висновок

В даній роботі описана методика програмно-апаратної реалізації штучного нейрону з сигмоїдальною функцією активації засобами FPGA, наведено покроковий алгоритм синтезу штучного нейрону та відповідний займаний ресурс FPGA. Побудований штучний нейрон з сигмоїдальною функцією активації на ПЛІС за даною технологією з кроком квантування 0.01 зайняв 765 LUTs. Похибка абсолютна ±0.005. Розроблена технологія побудови штучних нейронних мереж, зокрема таких як RBF-мережі, динамічні мережі Хопфілда та мережі прямого розповсюдження. Розробка та моделювання роботи штучного – нейрону та нейроних мереж проходило на програмному забезпеченні Xilinx ISE Design Suite 13.2 та чіпі сімейства Spartan 3 – XC3S200.

Список літератури

1. Егупов Н.Д. Методы робастного, нейро-нечеткого и адаптивного управления. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 744 с.
2. Терехов В.А. Нейросетевые системы Управления: Учеб.пособие для вузов. – М: Высшая школа. 2002. -183с.
3. Соловьев В. Проектирование цифровых систем на основе ПЛИС. – М.: Радио и связь, 2003. – 376с.
4. Гильгурт С.Я. Анализ применения реконфигурируемых вычислителей на базе ПЛИС для реализации нейронных сетей // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Вип. 37. – Київ: 2006. – С. 168-174.
5. Логовский А. Технология ПЛИС и ее применение для создания нейрочипов. // Открытые системы. – 2000. – № 4. – С. 100-102.
6. Barren A. R. Universal approximation bounds for superposition of a sigmoidal function // IEEE Trans. Inform. Theory. 1993. Vol. 39. P. 930 – 945.
7. Саймон Хайкин. Нейронные сети. Полный курс. – М.: Вильямс, 2006. – 1104с.
8. Сигеру Омату. Нейроуправление и его приложения. – М.: ИПРЖР, 2000. – 272с.
9. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. – М.: Горячая линия – Телеком, 2007. – 452 с.

УДК 62-82:658.512.011.56

КИРЮША Б.А.,
ЛАДОГУБЕЦ В.В.

ПОДХОДЫ К УСТРАНЕНИЮ КОНФЛИКТОВ ПРИ СОВМЕСТНОМ АНАЛОГОВО-ЦИФРОВОМ МОДЕЛИРОВАНИИ

Рассмотрены основные конфликты совместного цифро-аналогового моделирования. Описаны известные алгоритмы решения конфликтов временного моделирования. Предложен протокол управления совместным моделированием, учитывающим рассмотренные проблемы и их решения.

The main sources off cosimulation problems are considered. The techniques of cosimulation collisions overcoming are described. The new master-slave protocol for cosimulation problems handling is proposed.

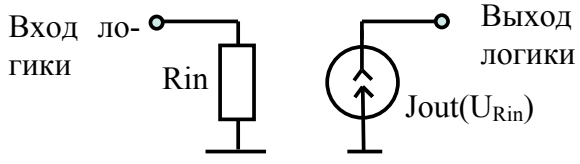
1. Введение

Интерес к средствам совместного аналогово-цифрового моделирования (САЦМ) растет с каждым новым витком интеграции модулей различного типа в рамках одной интегральной схемы. В роли аналоговых блоков могут выступать как электронные, так и более сложные приборы, например приборы электро-механического типа. Реализация совместного моделирования в рамках одной среды и одного алгоритма осуществима только для случаев, когда одна или обе части комплексной модели устройства описываются с рядом ограничений, обусловленных алгоритмом программы временного анализа (ПВМ). Использование отдельных программ моделирования, взаимодействующих через общий интерфейс, расширяет возможности создания моделей сложных процессов, но требуют тщательной проработки алгоритма взаимодействия между пакетами. В данной статье проведен краткий анализ проблем, возникающих при создании интерфейса совместного моделирования на базе которого предложен протокол взаимодействия между программой аналогового моделирования (ПАМ) и программой функционально-логического моделирования (ПФЛМ).

2. Формирование САЦМ

Решение задачи формирования САЦМ, состоит из реализации стандартизированных (в рамках системы) интерфейсов цифро-аналогового/аналогово-цифрового преобразований (ЦАП/АЦП) и синхронизации двух или более алгоритмов временного анализа.

Со стороны программы аналогового моделирования (ПАМ), каждый сигнал интерфейса САЦМ, чаще всего, замещается эквивалентной схемой, изображенной на рис. 1а. Со стороны программы функционально-логического моделирования (ПФЛМ) описания интерфейс описывается в виде, аналогичном изображенному на рис. 1.б[1].



а

```
@ (cross(V(vin))-Vt,1)) LIN=1;
@ (cross(V(vin))-Vt,-1)) LIN=0;
……..
V(vout) <+ transition(voutval, de-
```

б

Рис. 1. Реализация ЦАП/АЦП интерфейсов в ПАМ и ПФЛМ

Организация взаимодействия ПАМ и ПФЛМ реализуется одним из трех способов:

1. Интегрированное ядро косимуляции, при котором модель одного типа реализуется средствами модели другого типа. Например, описание логических устройств переводится в форму эквивалентной схемы.
2. Совмещенная косимуляция, при которой ПАМ и ПФЛМ разделяют общее адресное пространство и управляются общим планировщиком событий.
3. Раздельная косимуляция, при которой используются независимые ПВМ для разных типов моделей, а их взаимодействие

обеспечивается через некоторый общий протокол или третью программу.

Выбор способа ПФЛМ зависит от приоритетов погрешности, простоты и скорости временного моделирования, необходимых для решения конкретных практических задач. Если необходимо обеспечить полный цикл разработки устройства, то раздельное САЦМ обеспечивает наименьшее число переписывания моделей разного уровня абстракции и упрощает переход между соседними этапами разработки (прямое/обратное проектирование) с наименьшими временными затратами.

Численные эксперименты показали, что САЦМ, при использовании в качестве ПАМ пакета ALLTED**[2]**, может быть реализовано любым из перечисленных выше способов. Причина выбора раздельной архитектуры САЦМ состоит в том, что она обеспечивает наибольшую производительность в многопоточном исполнении и упрощает обеспечение совместимости ПВМ, как с другими пакетами, так и с будущими версиями собственных компонентов.

3. Конфликт сильных/слабых сигналов

К сожалению, практическое применение интерфейсов, изображенных на рис. 1, ограничивается только самыми простыми сигналами, когда направление передачи сигнала неизменно со временем и сигнал соответствует либо бинарному, либо действительному типу данных. В цифровой схемотехнике большое значение имеют не только состояния «0» и «1», но и слабый «0», слабая «1», «обрыв», другие уровни сигналов. Значение напряжения и тока сигнала, получаемое от ПАМ, не позволяет точно оценить, в каком из состояний находится сигнал. Перечень данных, необходимых для уточнения оценки включает: сопротивление от сигнала к земле, сопротивление от сигнала к питанию, сопротивление от сигнала к другим источникам опорного напряжения. Теоретически, эту информацию можно извлечь из матрицы описания схемной модели, но, учитывая сложность расчета таких величин на основании промежуточных данных моделирования, это не реализовано ни в одном из исследованных пакетов-аналогов[1,3,4,5,6,7]. Единственное известное решение[8], частично устраняющее данный конфликт, состоит в назначении всем аналоговым сигналам статуса «слабых». Попытка повторить этот прием в обратном направлении приводит к фиксированию состояния аналоговой модели по данному сигналу, что не позволяет реализовать режим сигнала «вход-выход». Для компенсации этого ограничения, в работе [9] аналоговый блок предлагается включать через тестирующие модули, как это изображено на рис. 2.

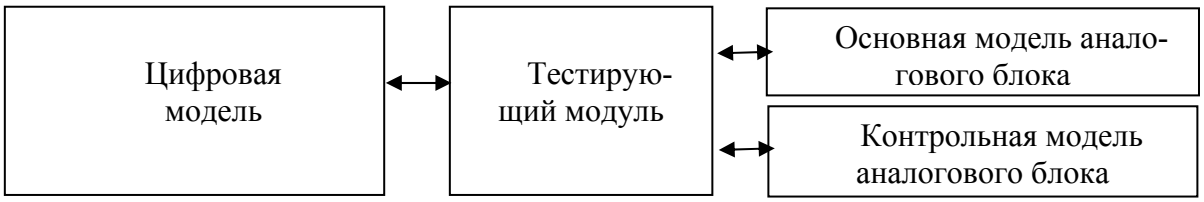


Рис. 2.Схема решения конфликта слабых/сильных сигналов с помощью тестирующего модуля

Тестирующий модуль, в данном случае, это менеджер процессов, контролирующий весь процесс моделирования. Он может порождать и завершать ПВМ, в зависимости от выполнения условий сходимости их результата с результатами других ПВМ. В простейшем случае (рис. 2), он работает по алгоритму:

- При отсутствии конфликтов сигналов, обе аналоговые модели получают и отсылают сигналы одинаково.
- При возникновении конфликта сигналов, как со стороны аналогового и цифрового блоков, так и со стороны нескольких источников аналогового блока, тестирующий модуль посылает на контрольную модель один сигнал, а на основную модель аналогового блока – другой, назначает следующий момент обновления информации и проверяет отсутствие конфликта с одним из блоков.

- В зависимости от приоритетов аналогового и цифрового блоков, а также, сохранения конфликтующего логического уровня сигнала в одном из аналоговых блоков, выбирается наименее конфликтный сигнал.

Такая оценка не дает полной гарантии на точность результата, но позволяет выйти из конфликтной ситуации с наибольшей вероятностью того, что результат – правильный. Тестирующий модуль применим только в случаях, когда аналоговая модель имеет не большое число общих сигналов с цифровой моделью и технические средства позволяют увеличивать количество ПАМ без существенного снижения скорости моделирования. Примеры ситуаций, когда могут возникать конфликты такого рода, приведены на рис. 3.

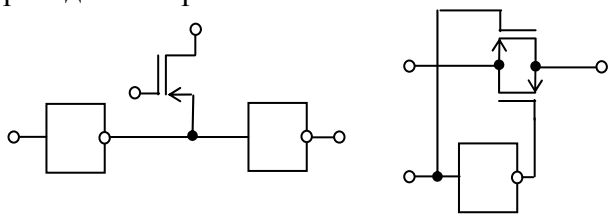


Рис. 3. Примеры задач смешанного моделирования, приводящие к конфликтам между ПАМ и ПФЛМ

4. Конфликт множеств логических уровней

Если в схеме присутствует несколько логических уровней, например для каскадов ввода/вывода и для аналоговых блоков систем на кристалле, количество потенциальных конфликтов растет экспоненциально. Для реализации САЦМ при нескольких логических уровнях алгоритм ЦАП/АЦП стараются организовать так, чтобы как можно больше конфликтов поглощалось единым алгоритмом моделирования. Это реализуется либо путем укрупнения схемотехнических моделей при использовании упрошенных моделей цифровых компонентов, либо понижением уровня абстракции в языках ПФЛМ, таких, как Verilog A/Verilog-AMS**[1]**.

Если скрыть составной сигнал в одном блоке не представляется возможным, некоторые пакеты предлагают механизм программируемых пользователем узлов, с собственным алгоритмом ЦАП/АЦП для сложных сигналов**[4]**. Т.е. задача перекладывается на составителя модели, что требует изучения дополнительного инструментария и особенностей его эксплуатации, в результате чего теряется возможность автоматизированной смены уровней абстракции описания моделей.

5. Конфликт синхронизации ПАМ и ПФЛМ.

Самым большим недостатком раздельной архитектуры САЦМ является наибольшее снижение скорости моделирования, по сравнению с другими архитектурами (интегрированное смешанное моделирование не содержит данной типы проблем в принципе, совмещенная архитектура компенсирует задержки синхронизации быстрым доступом к общему адресному пространству и общим планировщиком событий). Большая часть временных потерь раздельного моделирования приходится на согласование временных расхождений между алгоритмами ПАМ и ПФЛМ. Алгоритмы временного анализа в ПАМ требуют многократных оценок временного шага перед переходом на следующий временной отсчет. В отличие от аналогового алгоритма, в ПФЛМ переход на следующий временной отсчет регулируется только очередью известных функционально-логической модели событий. Таким образом, возникает ситуация когда каждому из алгоритмов требуются данные, еще не известные другому алгоритму.

Для решения проблемы синхронизации может использоваться один из следующих подходов:

- моделирование с минимальным временным шагом;
- моделирование по интерполированным значениям сигналов;
- моделирование с возвратами на предыдущий временной отсчет.

Ни один из перечисленных подходов не гарантирует сходимости вычислений, а его точность зависит от предустановленных параметров и устойчивости исходной модели. Первый подход применим только для задач малой размерности, т.к. требует

максимального объема вычислений, что компенсируется только простотой реализации.

Использование интерполированных значений неизбежно ведет к расхождениям между реальными и интерполированными значениями сигналов, которые могут быть частично скомпенсированы применением первого или третьего подходов в отдельные моменты времени. Технически – второй подход является наиболее сложным для программной реализации.

Третий подход требует существенного вмешательства во внутреннюю работу алгоритмов моделирования ПАМ и ПФЛМ, но в случае успеха, обеспечивает наименьшие потери производительности. Некоторые ПФЛМ, такие как Verilog, предоставляют механизм доступа к внутреннему планировщику событий и данным о состоянии переменных, но сохранение и возврат к предыдущему временному шагу могут быть реализованы только с учетом особенностей реализации этого интерфейса в каждом конкретном ПФЛМ.

Раздельное смешанное моделирование предполагает использование алгоритма, управляющего ПВМ, не зависимо от того какой алгоритм синхронизации будет выбран. При разработке протоколов обмена данными между ПВМ и программой-арбитром необходимо обеспечить выполнение следующих условий:

- Гибкость протоколов должна быть достаточной для смены ПВМ, как минимум, в рамках одного типа моделирования.
- Должен быть точно определен набор данных, хранимых на стороне ПВМ и на стороне арбитра, а также процедура их обновления.
- Должен быть определен минимальный набор служебных сообщений: запрос состояния, пауза, возобновление, аварийное завершение.
- Независимо от того, является арбитр независимой программой или частью одной из ПВМ, должен быть разработан модуль-клиент для всех ПВМ, гарантирующий выполнениевсех перечисленных выше требований.

6. Проблема обнаружения конфликтующих сигналов, нарушающих логику работы устройства

Некоторые конфликты в значениях сигналов не относятся к проблемам алгоритма САЦМ, а являются той ошибкой, которую пытаются выявить разработчики в своем устройстве. Решение конфликтов такого рода в автоматическом режиме не возможно, т.к. только разработчик может точно сказать, какая комбинация сигналов является допустимой, а какая нет. Единственным механизмом, доступным для решения задач такого поиска ошибок в проекте является создание механизма управления параметрами САЦМ пользователем, а также, формирование механизма создания пользовательских типов данных и методов работы с ними на функционально-логическом уровне и пользовательских узлов на схемотехническом. Сложность таких интерфейсов достаточно высока, так как требует не только понимания разработчиком внутренних алгоритмов моделирования для обоих ПВМ. Единственным способом избежать усложнения модели, в данном случае, является вынос «проблемного» участка модели в состав одной из ПВМ(предпочтительно ПФЛМ, т.к. такие программы изначально содержат средства выявления ошибок такого рода).

7. Протокол взаимодействия арбитр-ALLTED

В значительной степени, описанные проблемы могут быть решены с помощью предлагаемого протокола ALLTED-арбитр. Обобщенная схема обмена пакетами между арбитром и ALLTED изображена на рис 4.

На рис. 4 буквой «A» обозначены состояния клиента на стороне ПАМ ALLTED, а буквой «S» – состояния интерфейса к аналоговому пакету моделирования со стороны арбитра. Состав пакетов и правила перехода интерфейса из состояния в состояние изображены на рис. 5.

В зависимости от направления интерфейса (in, out, inout) выбирается схема синхронизации. Считается, что ведущий сигнал должен сохранять свое значение на протяжении заданного временного интервала.

Если условие не выполняется, выполняется запрос на пересчет значений и обновленные данные сохраняются арбитром, вместе с ближайшим прогнозируемым событием изменения сигнала (для ПФЛМ). Оценка принадлежности сигнала к «слабым» выполняется по совокупности факторов: потенциал узла, ток сигнала, ток «нуля», ток «единицы» (берутся минимальные значения токов для равных напряжений). Возможность реализации множественных зависимостей выходного сигнала строится не механизме реализации нелинейных функций пользователя ALLTED. Такая схема обмена данными обеспечивает запаздывание источника сигнала на некоторую фиксированную величину, по отношению к фронтам функционально-логической модели, решая конфликт синхронизации алгоритмов моделирования на уровне арбитра.

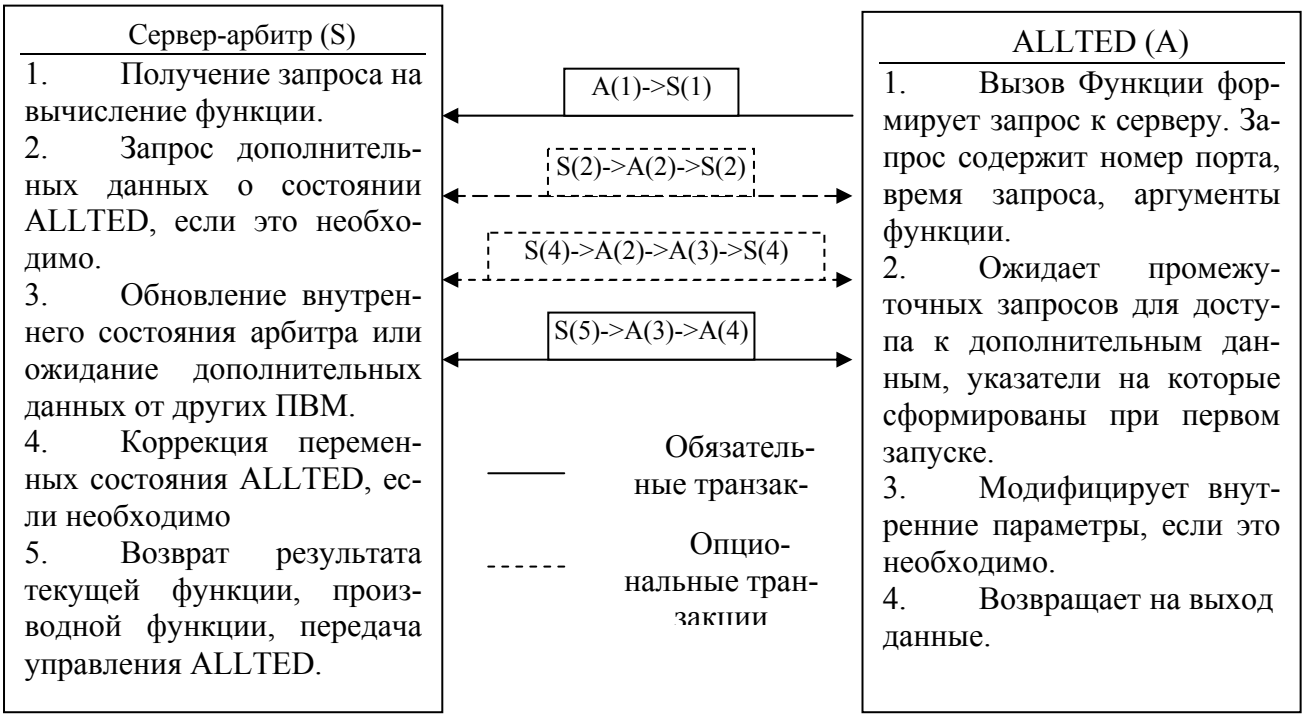


Рис. 4. Обобщенный протокол обмена данными ALLTED-арбитр

Со стороны ПФЛМ реализуется аналогичный протокол, с поправкой на то, что направление передачи сигнала и разделение на «сильные» и «слабые» сигналы выполняется на уровне языка Verilog.

Такая схема не устраняет конфликтов «сильных» сигналов и «гонок», вызываемых изменениями сигналов с нулевой задержкой. При возникновении таких ситуаций предусмотрено критическое завершение процесса моделирования с сообщением об ошибке. Такие ограничения на функционально-логические модели вызваны необходимостью обеспечения разработчика механизмом поиска режимов работы устройства, не имеющих физической реализации.

Заключение

Экспериментальные расчеты показали, что для большинства критических ситуаций, описанных в работе, предложенная схема взаимодействия ПАМ и ПФЛМ позволяет корректно проводить временное моделирование для заданной задержки обновления сигналов. При этом, реализация протокола не требует вмешательства во внутренний алгоритм временного анализа.

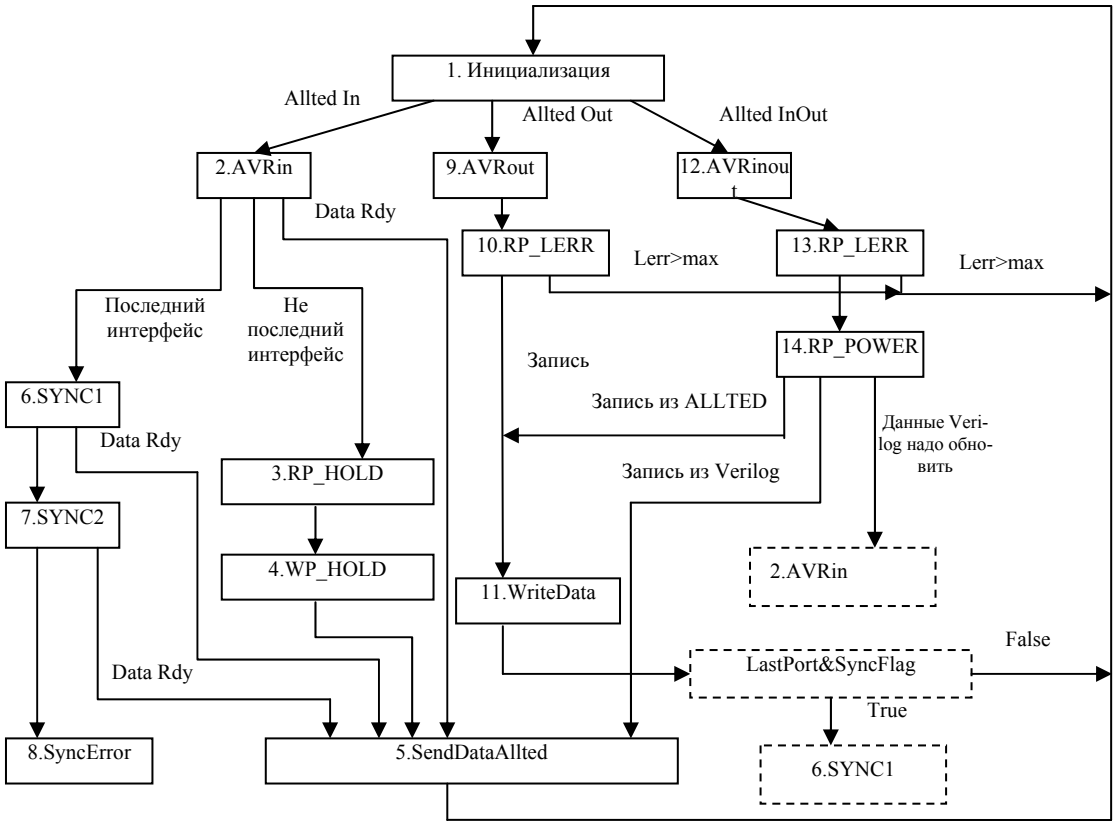


Рис. 5. Событийная модель поведения интерфейса ALLTED-арбитр

Список литературы

1. Описание Cadence AMS. – Режим доступа : http://www.cadence.com/products/pcb/ams_simulator/pages/default.aspx.
2. Безносик О.Ю. Використання пакетів схемотехнічного проектування для моделювання механічних компонентів / Безносик О.Ю., Ладогубець В.В., Фіногенов О.Д., Чкалов О.В. // Системний аналіз та інформаційні технології : 10-а міжнародна науково-технічна конференція «САІТ-2008», Київ : матеріали. – К. : НТУУ «КПІ», 2008. – С. 286.
3. Описание DOLPHIN Smash Vision. – Режим доступа : http://www.dolphin.fr/medal/smash/smash_overview.php.
4. Описание X-Spice ICup. – Режим доступу: http://users.ece.gatech.edu/ ~mrichard/ Xspice/ Xspice_Users_Manual.pdf. – Дата доступа : 30.11.2011.
5. Описание Proteus. – Режим доступу: http://www.labcenter.com/index.cfm. – Дата доступа : 30.11.2011.
6. Описание Symica IC design toolkit. – Режим доступа: http://symica.com. – Дата доступа : 30.11.2011.
7. Описание ADMS. – Режим доступа : http://adms.noovela.com. – Дата доступа : 30.11.2011.
8. Bastian J. Master for Co-Simulation Using FMI / Bastian J. // 8th Modelica Conference, Dresden, Germany, 20-22 March 2011 : proc. – P. 115–120.
9. Youngmin Yi. Fast and Accurate Cosimulation of MPSoC Using Trace-Driven Virtual Synchronization / Youngmin Yi. // Computer-Aided Design of Integrated Circuits and Systems : IEEE Transactions. – Vol. : 26. – Issue : 12,2007. – P. 2186 – 2200.

УДК 62-82:658.512.011.56

ЛЯПИН П.С.,
МЕЛЬНИЧУК Р.М.,
ФИНОГЕНОВ А.Д.

ГРАФИЧЕСКИЕ РЕДАКТОРЫ ДЛЯ СХЕМОТЕХНИЧЕСКОГО ПРОЕКТИРОВАНИЯ

Целью статьи является формирование общих требований для разработки редактора электронных схем в рамках создания средств автоматизированного проектирования с доступом через Internet. На основе анализа существующих решений в данной области предложен набор стандартных функций, необходимых инженеру-разработчику и приоритетных для реализации в веб-редакторе.

The purpose of this paper is the establishment of common requirements for the design editor of electronic circuits in a computer-aided design tools to access the Internet. Analysis capabilities available today decisions in this field will conclude on a set of standard functions necessary development engineer and priorities for implementation in a web editor.

1. Введение

История разработки специализированных графических редакторов насчитывает не одно десятилетие. Параллельно с усовершенствованием аппаратных средств и, в первую очередь, средств визуализации, менялись и подходы к созданию графического интерфейса пользователя и в частности схемных редакторов. Наличие редактора на сегодняшний день стало стандартом де-факто для САПР.

Одним из современных направлений развития САПР является создание веб-средств проектирования (не путать со средствами веб-проектирования), что, соответственно, влечет за собой необходимость в создании соответствующих графических редакторов. При этом абсолютное большинство существующих редакторов предназначено для использования на локальном компьютере пользователя и реализация функций не учитывает специфики веб-приложений, а их графический интерфейс, набор функций, компоновка могут существенно отличаться.

Поэтому выделение типовых функций графических схемных редакторов для дальнейшей реализации в графическом веб-редакторе электронных схем является достаточно актуальной.

2. Классификация и анализ возможностей графических редакторов

Графические редакторы электронных схем в зависимости от режима использования можно разделить на два класса:

– специализированные графические схемные редакторы (ГСР);

– ГСР общего назначения.

Под специализированными ГСР будем понимать схемные редакторы, которые входят в состав программных средств EDA (Electronic Design Automation). Комплексы EDA предназначены для облегчения разработки электронных устройств, создания микросхем, печатных плат [1]. Типичным представителем данного класса графических схемных редакторов является OrCAD [2].

К ГСР общего назначения относятся редакторы, которые помимо графических примитивов и текста позволяют использовать различные символические объекты, в том числе и элементы принципиальных электрических схем, и не направлены на дальнейшую обработку построенной схемы. Данные редакторы имеют более широкие графические возможности по сравнению с редакторами, входящими в состав средств САПР. Чаще всего, использование данных редакторов обусловлено необходимостью разработки документации и требованиями к форматам, используемым в издательствах. Примером таких систем является редактор Microsoft Visio [3].

Для сравнения были выбраны последние версии наиболее распространенных схемных редакторов и пакетов САПР, содержащих их в своем составе (табл. 1).

2.1 Специализированные графические схемные редакторы

Помимо основных возможностей рисования и визуализации, современные специализированные графические схемные редакторы

Табл.1. Графические схемные редакторы

Редактор	Фирма, страна	ОС	Размер дистри-бутива*	Стои-мость*
Altium Designer 9 [4]	Altium Ltd, Австралия	Win	1,63 Гб	0-5000$
AutoTrax EDA 10.01 [5]	Kovac Software, США	Win	23,4 Мб	199$
Board Maker 3 [6]	Tsien Ltd., Великобритания	Win	43,1 Мб	350£
CADSTAR 12.1 [7]	Zuken, Великобритания	Win	171 Мб	1000£
DipTrace 2.1 [8]	Novarm Ltd, Украина	Win, Linux, Mac	46 Мб	585€
EdWinXP 1.80 [9]	Visionics, Швеция	Win	112,2 Мб	100-3030$
Electric VLSI Design System 9.0 [10]	Static Free Software, США	Win, Linux, Mac	18 Мб	0-40$
Electronics Workbench 5.12 [11]	Electronics Workbench, Канада	Win	7 Мб	25-18760$
HiPAC-EDT 1.0 [12]	Infineon Technologies AG, Германия	Win	4.1 Мб	0
Micro-CAP 10 [13]	Spectrum Software, США	Win	700 Мб	4495$
Microsoft Visio 2010 Premium [3]	Microsoft, США	Win	1,5 Гб	554$
NI Multisim 11 [14]	National Instruments, США	Win	400 Мб	2600$
OrCAD 16.3 [2]	Cadence Design Systems Inc., США	Win	1,75 Гб	1729 – 4355$
PlainCAD 2007 [15]	PlainCAD Software, Россия	Win	7.9 Мб	70$
Proteus 7 [16]	Labcenter Electronics Ltd. , Канада	Win	92 Мб	150- 3495£
Pulsonix 6 [17]	Westdev Ltd, Великобритания	Win	930 Мб	600£
SmartDraw VP [18]	SmartDraw LLC, США	Win	22.12 Мб	197$
SPlan 7.0 [19]	ABACOM, Германия	Win	2.31 Мб	39.90€
TINA-TI [20]	Texas Instrumental Inc., США	Win	86.3 Мб	0
TinyCAD 2.80 [21]	Open Source	Win	4,4 Мб	0
Vesys Design 2.0 [22]	Mentor Graphics, США	Win	255 Мб	11000-14000$
Vutrax 13.4 [23]	Computamation Systems Limited, Великобритания	Win, Linux	18 Мб	0 – 4850£

** – в случае отсутствия информации об отдельно поставляемом редакторе в качестве размера дистрибутива и стоимости указываются данные всей САПР (полная версия).*

могут предоставлять дополнительные функции:

– использование библиотеки стандартных компонентов;

– редактирование электрических свойств и прочих атрибутов для компонентов, проводов и контактов;

– иерархическое проектирование;

– отображение списка соединений (англ. “netlist”) и других представлений схемы;

– автоматическое отслеживание и оповещение пользователя о наличии ошибок в схеме;

– автоматическое создание документации;

– возможность обрабатывать входные данные различных форматов и т.д.

При анализе возможностей специализированных редакторов целесообразно выделить две группы:

– программы для решения небольших задач;

– программы для создания сложных систем.

К первой группе относятся Electronics Workbench 5.12, NI Multisim 11, Micro-CAP 10, HiPAC EDT, TinyCad 2.80. Среди этих программ стоит выделить NI Multisim 11.

Multisim 11 разработан американской компанией National Instruments и является улучшенной версией известного продукта Electronics Workbench. Программа может производить моделирование в PSPICE, VHDL, Verilog и позволяет получить соответствующий код описания схемы. Результаты анализа выводятся на виртуальные инструменты, устроенные аналогично реальным (как в Electronics Workbench 5.12), что позволяет пользователям EWB легко адаптироваться к этой программе (рис. 1).

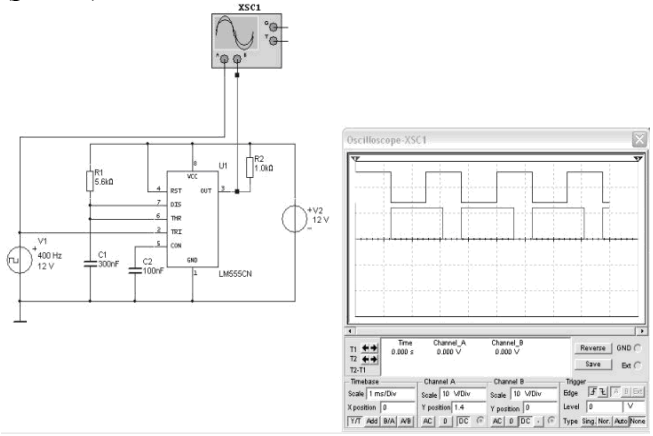


Рис. 1. Моделирование схем в Multisim

Последняя версия Multisim дополнена новыми инструментами и имеет обновленные библиотеки элементов.

Ко второй группе можно отнести OrCad, Proteus, Altium Designer, DipTrace, Pulsonix и CADSTAR 12.1. Особо выделяются здесь OrCad и Altium Designer, так как они предоставляют разработчику широкий выбор инструментов для решения любой задачи в проектировании электрических схем и печатных плат. В то же время, из вышеперечисленных программ они имеют и наибольшую стоимость.

Интересные решения реализованы в графическом редакторе DipTrace 2.1. DipTrace Schematic позволяет работать в нескольких графических режимах: OpenGL, Direct3D, Windows GDI. Также есть две цветовых схемы: с темным фоном, удобная для разработки больших схем, и белая, которая представляет схему так, как она должна выглядеть в документации (рис. 2). По желанию можно создавать свои цветовые схемы. Хотя данная возможность при создании редактора может быть легко добавлена практически во всех средствах разработки, в большинстве рассматриваемых редакторов она отсутствует. Соответственно добавляется ряд «лишних» операций при создании документации.

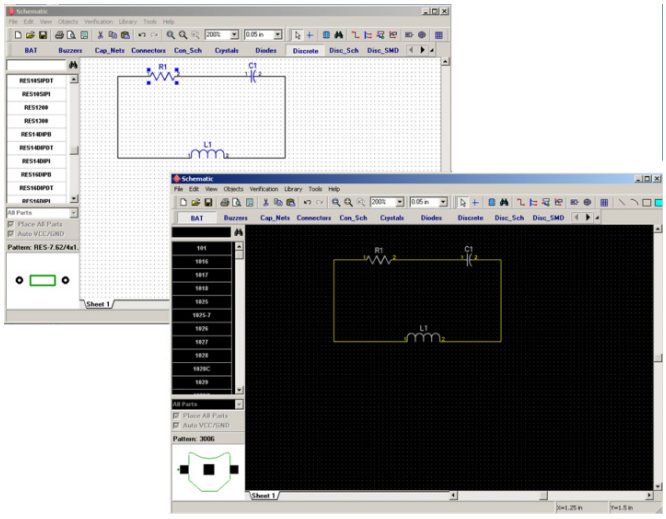


Рис. 2. Цветовые схемы фона рабочей области DipTrace

Также стоит отметить Electric VLSI Design System 9, так как она является единственной из вышеперечисленных программ кроссплатформенной системой автоматизированного проектирования сверхбольших интегральных схем с открытым исходным кодом. При помощи Electric VLSI можно разрабатывать интегральные МОП и биполярные схемы, печатные платы или схемы любого типа. Electric VLSI Design System 9 имеет множество стилей редактирования, включающих планирование, схематику, иллюстрации, архитектурное проектирование. Также, программа может взаимодействовать с различными спецификациями и форматами файлов, такими как VHDL, CIF, GDS II.

В графическом редакторе Electric VLSI отсутствует ряд инструментов, присутствующих в его аналогах: нет режима автоматического построения трасс, отсутствует привязка к сетке и т.д. Однако, данные функции нечасто используются при проектировании больших схем, поэтому это не создает каких-либо трудностей для пользователя.

2.2 Графические схемные редакторы общего назначения

К третьей группе относятся sPlan 7.0, PlainCad 2007, SmartDrawing VP, Microsoft Visio. Практически все редакторы этой группы предоставляют одинаковые возможности и инструменты для рисования электрических схем, хотя и имеют разные дополнительные инструменты для работы с другими видами чертежей (рис. 3).

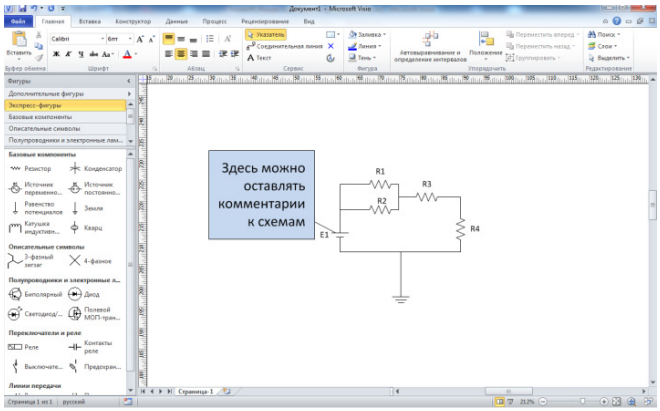


Рис. 3. Построение схемы в Microsoft Visio 2010

3. Функциональные возможности графических редакторов

Функциональные возможности графических редакторов можно разделить на следующие классы: функции редактирования изображения, настройка рабочей области, работа с библиотеками и файловой системой, функции проверок и статистики.

К функциям редактирования изображения относятся:

– редактирование цветов и ширины линий;

– рисование графических примитивов (круги, прямоугольники, многоугольники);

– задание слоя элемента при наложении элементов (z-index) и т.д.

Настройки рабочей области включают функции:

– редактирование фона;

– наличие линейки;

– масштабирование;

– переключение единиц измерения и т.д.

К функциям работы с библиотеками и файлами относятся:

– создание новых элементов и изображений (редактор элементов);

– импортирование изображения из файла;

– сохранение схемы в одном из графических форматов;

– автоматическое резервное копирование (auto backup) и т.д..

Дополнительные функции, которые могут включаться в графические редакторы – это функции различных проверок и статистики:

– проверки на связанность схемы, наличия контакта узлов, висячих узлов и т.д.;

– поиск элемента в рабочей области;

– статистика по количеству узлов схемы, типам элементов и т.д.

Данные функции позволяют составить общее представление о возможностях систем и о целевой аудитории программного обеспечения – от начинающих разработчиков до профессионалов. Также, проверка на наличие вышеперечисленных функций в анализируемых редакторах должна выявить как наиболее, так и наименее востребованные и реализованные функции в графических редакторах электронных схем.

Необходимо отметить также различные способы прокладки соединений между элементами схемы, используемые в графических редакторах: автоматический режим с учетом «обтекания элементов» и ручное соединение по указанным точкам. В автоматическом режиме обычно необходимо явно указывать расположение узлов, т.к. в противном случае соединительные линии не создают пересечений. Кроме того, ряд пакетов включает режим «жесткого соединения», т.е. при перетягивании подключенного элемента за ним тянутся и соединительные линии.

Результаты сравнения (табл. 2) функциональных возможностей различных графических схемных редакторов позволяют выделить три группы функций:

- общеупотребительные, которые реализованы в абсолютном большинстве редакторов (масштабирование, редактор библиотеки элементов, привязка к сетке и т.д.);
- рекомендуемые, которые присутствуют во многих редакторах (линейка, z-index, статистика по схеме и т.д.);
- дополнительные, которые являются особенными или отсутствуют во многих редакторах (создание подсхем, переключение единиц измерения).

Очевидно, что наличие или отсутствие некоторых функций зависит от класса, к которому принадлежит графический редактор: специализированный или общего назначения.

4. Функциональные возможности веб-редактора электронных схем

Реализация большинства рассматриваемых функций в составе веб-редактора не составляет трудностей. Исключением является реализация функций работы с библиотеками и файлами, которые присутствуют во многих редакторах.

Табл. 2. Функциональные возможности графических редакторов

Редакторы	Редактирование цветов и ширины линий	Рисование геометрических примитивов	z-index (задание слоя элемента)	Линейка	Масштабирование	Переключение единиц длины (мм, см, дюймы)	Привязка к сетке / редактирование размера сетки	Редактор библиотеки элементов	Вставка рисунка из файла	Сохранение в графическом формате	Автоматическое резервное копирование (auto-backup)	Создание подсхем	Проверки корректности электрических схем	Поиск элемента на схеме	Статистика по схеме	Буфер обмена
Altium Designer 9	+	+	-	+	+	-	+/+	+	+	+	+	+	+	+	+	+
AutoTraxEDA 10.01	-	+	+	+	+	-	+/+	+	+	+	-	-	+	+	-	+
Board Maker 3	-	+	-	+	+	+	+/+	+	+	+	-	-	+	+	+	+
CADSTAR 12.1	+	+	-	-	+	+	+/+	+	-	-	+	-	+	+	+	+
DipTrace 2.1	+	+	-	+	+	+	+/+	+	+	+	+	+	+	+	-	+
EdWinXP 1.80	+	+	-	+	+	+	+/+	+	+	-	-	+	+	-	+	+
Electric VLSI DS 9.0	+	+	+	-	-	-	+/+	+	-	+	+	+	+	-	+	+
HiPAC-EDT 1.0	-	-	-	-	+	-	+/-	+	-	-	-	-	-	-	-	+
Micro-CAP 10	+	+	+	-	+	-	+/+	+	+	-	-	-	+	+	+	+
Microsoft Visio 2010	+	+	+	+	+	+	+/+	+	+	+	+	+	+	+	+	+
NI Multisim 11	+	-	+	+	+	-	+/+	+	-	-	+	-	+	+	+	+
OrCAD 16.3	+	+	+	+	+	+	+/+	+	-	+	+	-	+	-	+	+
PlainCAD 2007	+	+	+	+	+	+	+/+	-	+	+	+	-	+	-	+	+
Proteus 7	+	+	+	-	+	-	+/-	+	+	+	+	+	+	+	+	+
Pulsonix 6	+	+	-	-	+	-	+/+	+	+	-	-	-	+	+	+	+
SmartDraw VP	+	-	+	+	+	+	+/-	+	+	+	+	-	-	+	-	+
SPlan 7.0	+	+	+	+	+	+	+/+	-	+	+	+	-	-	+	+	+
TINA-TI 9.3	+	-	-	-	+	+	+/-	+	+	-	+	+	+	-	-	+
TinyCAD 2.80	+	+	+	+	+	-	+/+	+	+	+	-	+	+	-	-	+
Vesys Design	-	+	+	-	+	-	+/+	+	-	-	-	-	+	+	+	+
Vutrax 13.4	-	+	+	-	+	-	+/+	+	-	-	-	-	+	-	-	+

Это связано с тем, что по условиям безопасности в современных браузерах ограничена работа с файловой системой пользователя. В данном случае можно предложить два основных механизма реализации требуемого функционала: использование системы загрузки необходимых файлов (например, библиотеки элементов) или использование удаленного хранилища данных.

Недостатками системы загрузки файлов является необходимость предоставления дополнительного программного обеспечения для создания требуемых файлов; наличие функций проверки корректности описания и т.д. В этом случае веб-редактор становится аналогом редакторов, устанавливаемых на локальном компьютере пользователя, и тем самым теряет свои преимущества. Однако такой редактор остается не персонализированным, т.е. данные конкретного пользователя вынесены в отдельные файлы (библиотеки) и не являются частью редактора. В случае работы нескольких человек над одним проектом для корректного отображения результатов другим пользователям достаточно передать файл библиотеки. Механизмы сохранения файлов изображений схемы и/или описания, сформированного на входном языке пакета моделирования (если данная возможность предусмотрена) также будут требовать ручного вмешательства пользователя, а функции резервного сохранения можно реализовать через механизмы, встроенные в веб-браузеры (XPCOM, ActiveX), сторонние дополнения (Adobe Flash), либо современный протокол HTML5 Web Storage.

Удаленное хранилище данных, как возможный механизм реализации функций работы с файлами и библиотеками, может быть реализовано в виде базы данных библиотек/файлов пользователя, регистрация и хранение которых располагается на стороне веб-сервера. В этом случае, взаимодействие пользователя с веб-редактором более привычно для Internet-пользователей и требует наличия лишь браузера.

Отдельного внимания заслуживает необходимость создания механизма авторизации пользователя на сервере и безопасность этой процедуры, что не предполагается в прикладных программах.

5. Заключение

Современные ГСР являются важной составляющей при разработке электронных устройств. На сегодняшний день на рынке представлены программы в различных ценовых категориях и с различной функциональностью, что позволяет разработчику выбрать наиболее подходящее программное обеспечение в зависимости от его требований.

В результате проведенного анализа были определены наиболее распространенные функции, представленные в настоящее время в схемных редакторах различных категорий, а также составлен список интересных (удачных) решений, которые показались нам удобными для реализации в составе веб-редактора электронных схем:

1. Снятие измерений со схемы с помощью виртуальных приборов (Multisim 11);
2. Панель элементов с отдельно вынесенными кнопками категорий (DipTrace);
3. Возможность создавать цветовые схемы рабочей области (DipTrace);
4. Возможность оставлять комментарии к участкам схемы и отдельным элементам (Microsoft Visio);
5. Возможность использования разных режимов построения трасс.

Список литературы

1. Анцифорова Е.С. О необходимости создания унифицированных инструментальных средств для автоматизации проектирования технологической автоматики / Анцифорова Е.С., Раков В.И. // Информационные технологии в науке, образовании и производстве : 4-я Международная научно-техническая конференция «ИТНОП-2010», г. Орел, 22-23 апреля 2010 г : материалы. – В 5-ти т. – Т. 3 – Орел : ОрелГТУ, 2010. – С. 270–282.
2. Cadence OrCAD Solutions [Электронный ресурс]. – Режим доступа : http://www.cadence.com/products/orcad/pages/default.aspx. – Дата доступа : 07.04.2011. – Загл. с экрана.
3. Microsoft Visio 2010 – Office.com [Электронный ресурс]. – Режим доступа : http://office.microsoft.com/en-us/visio/. – Дата доступа : 07.04.2011. – Загл. с экрана.
4. Altium | AD10 [Электронный ресурс]. – Режим доступа : http://products.live.altium.com/. – Дата доступа : 07.04.2011. – Загл. с экрана.

5. AutoTRAX EDA and DEX Schematic Capture & PCB Design Software [Электронный ресурс]. – Режим доступа : http://www.kov.com. – Дата доступа : 07.04.2011. – Загл. с экрана.
6. BoardMaker3 [Электронный ресурс]. – Режим доступа : http://www.tsien.info. – Дата доступа : 07.04.2011. – Загл. с экрана.
7. CADSTAR 12.1 | Zuken [Электронный ресурс]. – Режим доступа : http://www.zuken.com/products/cadstar/cadstar-12-1.aspx. – Дата доступа : 07.04.2011. – Загл. с экрана.
8. DipTrace – Professional Schematic & PCB Design Software [Электронный ресурс]. – Режим доступа : http://www.diptrace.com. – Дата доступа : 07.04.2011. – Загл. с экрана.
9. Visionics – «EDWinXP» The Integrated EDA Software [Электронный ресурс]. – Режим доступа : http://www.visionics.a.se. – Дата доступа : 07.04.2011. – Загл. с экрана.
10. Static Free Software Home Page [Электронный ресурс]. – Режим доступа : http://www.staticfreesoft.com/index.html. – Дата доступа : 07.04.2011. – Загл. с экрана.
11. National Instruments Electronics Workbench Group Announcing NI Multisim 11 and NI Ultiboard 11 [Электронный ресурс]. – Режим доступа : http://www.electronicsworkbench.com/. – Дата доступа : 07.04.2011. – Загл. с экрана.
12. HiPAC EDT [Электронный ресурс]. – Режим доступа : http://www.hipac-edt.com/index.php. – Дата доступа : 07.04.2011. – Загл. с экрана.
13. Spectrum Software – Micro-Cap 10. Analog simulation, mixed mode simulation, and digital simulation software. SPICE and PSpice® compatible circuit simulator. [Электронный ресурс]. – Режим доступа : http://www.spectrum-soft.com/index.shtm. – Дата доступа : 07.04.2011. – Загл. с экрана.
14. NI Multisim and NI Ultiboard New Features – National Instruments [Электронный ресурс]. – Режим доступа : http://www.ni.com/multisim/upgrade.htm. – Дата доступа : 07.04.2011. – Загл. с экрана.
15. Главная – PlainCAD Software – Редактор для построения графических схем [Электронный ресурс]. – Режим доступа : http://www.plaincad.com. – Дата доступа : 07.04.2011. – Загл. с экрана.
16. Labcenter Electronics – Professional PCB Design and Simulation Software [Электронный ресурс]. – Режим доступа : http://www.labcenter.com/index.cfm. – Дата доступа : 07.04.2011. – Загл. с экрана.
17. Pulsonix :: Schematic Capture and PCB Layout [Электронный ресурс]. – Режим доступа : http://www.pulsonix.com. – Дата доступа : 07.04.2011. – Загл. с экрана.
18. SmartDraw – The World's First Visual Processor(tm) [Электронный ресурс]. – Режим доступа : http://www.smartdraw.com/product/. – Дата доступа : 07.04.2011. – Загл. с экрана.
19. sPlan [Электронный ресурс]. – Режим доступа : http://www.abacom-online.de/uk/html/splan.html. – Дата доступа : 07.04.2011. – Загл. с экрана.
20. SPICE-Based Analog Simulation Program – TINA–TI – TI Tool Folder [Электронный ресурс]. – Режим доступа : http://www.ti.com/tool/tina-ti. – Дата доступа : 07.04.2011. – Загл. с экрана.
21. TinyCAD | Download TinyCAD software for free at SourceForge.net [Электронный ресурс]. – Режим доступа : http://sourceforge.net/projects/tinycad/. – Дата доступа : 07.04.2011. – Загл. с экрана.
22. VeSys 2.0 :: Electrical, Cable and Wire Harness Design Engineering Manufacture :: Centricity Consulting [Электронный ресурс]. – Режим доступа : http://www.centricity-consulting.com/index.php/products/vesys-2-0. – Дата доступа : 07.04.2011. – Загл. с экрана.
23. Vutrax Electronics CAD [Электронный ресурс]. – Режим доступа : http://www.vutrax.co.uk. – Дата доступа : 07.04.2011. – Загл. с экрана.

УДК 004.82

ПОСПЕШНЫЙ А.С.

ЭФФЕКТИВНЫЙ ЛОГИЧЕСКИЙ АНАЛИЗ БОЛЬШИХ ОНТОЛОГИЙ ЗА ПОЛИНОМИАЛЬНОЕ ВРЕМЯ

В статье рассматривается проблема вычислительной сложности логического анализа онтологий. Рассмотрено дескриптивные логики $\mathcal{EL}$ и $\mathcal{EL}^{++}$ для которых задача логического анализа имеет полиномиальную сложность. Предложено ряд улучшений для логического анализатора ELK с целью поддержки конкретных доменов ($\mathcal{D}$) и утверждений о экземплярах (ABox).

This paper overviews the problem of computational complexity of reasoning with ontologies. The description logic $\mathcal{EL}$ and $\mathcal{EL}^{++}$ is considered where logical entailment could be accomplished in polynomial time. We propose few modification to new and very promising ELK reasoner to support concrete domains ($\mathcal{D}$) and ABox assertions.

Вступление

Исследования в области представления знаний в значительной степени основаны на методах описания окружающего мира, его объектов и свойств в форме пригодной для использования в интеллектуальных системах. В данном контексте под понятием интеллектуальной системы подразумевается система, способная извлечь неявно указанную информацию из множества явно заданных утверждений.

Методы представления знаний можно условно разделить на две категории. В первую категорию входят методы основанные на формальной логике, а в другую – неформальные методы. В отличии от формальных моделей, в основе которых лежит строгая математическая теория, неформальные методы чаще всего используют когнитивные приёмы для представления знаний в виде специальных структур, например таких как семантические сети или фреймы. При этом, несмотря на наглядность неформальных методов, отсутствие чётких универсальных правил их интерпретации позволяло эффективно применять такие методы лишь для некоторых конкретных задач. Формальные же методы представления знаний изначально развивались на основе механизмов логики предикатов первого порядка, что обусловило их универсальность.

В последствии желание внедрить семантику в интерпретацию семантических сетей и фреймов привело к появлению нового семейства языков представления знаний – дескриптивной логики [1] (ранее известная как терминологическая система, а потом логика концептов). В этом новом формализме были скомбинированы фреймовые структуры с фрагментами логики первого порядка, что позволило создать гибкий и выразительный язык представления знаний с хорошо изученными свойствами. При этом вычислительная сложность языка на прямую зависит от того, какие именно фрагменты логики первого порядка мы выбираем.

На практике одним из самых известных воплощений дескриптивной логики является онтология – формализация некоторой области знаний в виде множества понятий и отношений между ними. Онтологии нашли широкое применение в информатике, в частности в задачах накопления и обработки данных с учётом их семантики, при построении "порталов знаний", моделировании, а также в качестве посредника между пользователем и информационной системой или другими пользователями.

Отдельно стоит упомянуть, что онтологиям выделена центральная роль в планах развития Всемирной паутины. Современные методы автоматической обработки данных, доступных в Интернете, как правило, основаны на частотном и лексическом анализе текстового содержимого, которое, прежде всего, предназначено для восприятия человеком. Применение же метаданных и онтологий позволило бы совершать машинную обработку этих колоссальных объёмов информации, что вызвало б революцию в этой области. Консорциум Всемирной паутины (W3C) утвердил и активно развивает язык описания онтологий OWL, вокруг которого сегодня сконцентрировано значительную часть всех исследований в области представления знаний, онтологического инжиниринга и практического применения дескриптивной логики.

Однако массовому внедрению онтологий и использованию интеллектуальных систем в целом препятствует тот факт, что алгоритмы ло-

гического анализа в таких системах имеют чрезвычайно высокую вычислительную сложность. Начало исследованию вычислительной сложности дескриптивной логики положила статья [2] в которой была затронута эта проблема. В ней рассматривалась оценка вычислительной сложности логического анализа и те факторы, которые на неё влияют. Оказалось, что различные комбинации разрешённых конструкций языка порождали языки описания с совершенно разными вычислительными свойствами. Авторами был исследован один из таких языков – $\mathcal{FL}^-$ для которого алгоритм логического анализа сохранял полиномиальную сложность. Впоследствии был изучен ещё один язык – $\mathcal{EL}$ [3] и $\mathcal{EL}^{++}$ [4], который сохранял разрешимость за полиномиальное время и при этом был практичнее.

Разрабатывая нашу интеллектуальную систему [5] управления ресурсами Грид, в которой центральную роль играют онтологии, мы столкнулись с недопустимо низкой производительностью логического анализа над базой знаний. Перепробовав различные алгоритмы и методы оптимизации мы пришли к выводу, что табличный алгоритм, который лежит в основе почти всех логических анализаторов, не может справиться с онтологиями большого размера так как изначально был рассчитан на языки с большой экспрессивностью и выполняет процесс классификации онтологии путём итеративного построения модели для каждой пары концептов и поиск противоречий в ней. Согласно [1] такая задача имеет сложность $NExpTime$, что в комбинации с очень большим размером онтологии в нашем случае исключает этот подход.

За решением проблемы мы обратились к опыту наших коллег, которые разрабатывали интеллектуальные системы на основе одних из самых больших онтологий на сегодняшний день – *SNOMED CT*, *GALEN* и Gene Ontology. Дабы обеспечить приемлемую скорость классификации и масштабируемость, упомянутые онтологии были построены с применением дескриптивной логики $\mathcal{EL}^{++}$.

Дескриптивная логика $\mathcal{EL}^{++}$

Словарь логики $\mathcal{EL}$ состоит из исчислимо бесконечного множества атомарных ролей N_R, множества атомарных концептов N_C и верхнего концепта $\top$. Сложные концепты могут быть индуктивно построены с помощью выражений конъюнкции ($\mathrm{C} \sqcap D$) и квантора существования $\exists R.C$, где $R \in N_R$.

$\mathcal{EL}$ терминология, так называемый $TBox$ – это конечное множество определений типа $A \equiv C$ и $A \sqsubseteq C$ где никакое имя концепта не определяется более одного раза. Выражение $A \equiv C$ определяет $\mathcal{EL}$-концепт A эквивалентный концепту C, а выражение $A \sqsubseteq C$ выражает вложенность концепта A в концепте C.

Причина, по которой логика $\mathcal{EL}$ не содержит дизъюнкции ($\sqcup$) и квантора всеобщности ($\forall$) заключается в том, что наличие этих операций в языке делает задачу логического анализа $NExpTime$ – полной. Как было показано в [1] и [3] дизъюнкция является одним из основных источников вычислительной сложности и недетерминизма логического анализа, так как требует построения модели для каждого варианта и последующей её проверки. Квантор $\forall$, в свою очередь, может вызывать аналогичный скачок вычислительной сложности в комбинации с экзистенциальной конструкцией $\exists \mathrm{R}.\mathrm{C}$

Логика $\mathcal{EL}^+$ [6] расширяет $\mathcal{EL}$ конструкцией $r \circ s \sqsubseteq t$, где $r, s, t \in N_R$ с помощью которой можно выразить такие важные выражения как иерархии ролей ($r \sqsubseteq s$), транзитивные роли ($r \circ r \sqsubseteq r$) и цепочки ролей. Последние играют особо важную роль в составлении сложных медицинских и био-онтологий.

Наконец, логика $\mathcal{EL}^{++}$ расширяет логику $\mathcal{EL}^+$ номиналами, пустым концептом $\bot$, рефлексивными ролями, а также определением домена и диапазона ролей с некоторыми ограничениями. Наличие концепта $\bot$ среди прочего позволяет сделать утверждение о различии концептов: $C \sqcap D \sqsubseteq \bot$. Полный синтаксис и семантика логики $\mathcal{EL}^{++}$ представлена в Таблице 1. Тут и дальше используется семантика Тарского.

Табл. 1 Синтаксис и семантика языка $\mathcal{EL}^{++}$

Выражение	Семантика
$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
$\{a\}$	$\{a^{\mathcal{I}}\}$
$\mathrm{C} \sqcap \mathrm{D}$	$\mathrm{C}^{\mathcal{I}} \cap \mathrm{D}^{\mathcal{I}}$
$\exists r.\mathrm{C}$	$x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}: (x,y) \in r^{\mathcal{I}} \wedge y \in \mathrm{C}^{\mathcal{I}}$
$\mathrm{C} \sqsubseteq \mathrm{D}$	$\mathrm{C}^{\mathcal{I}} \subseteq \mathrm{D}^{\mathcal{I}}$
$r_1 \circ \ldots \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \ldots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$
$\mathrm{dom}(r) \sqsubseteq \mathrm{C}$	$r^{\mathcal{I}} \subseteq \mathrm{C}^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$\mathrm{ran}(r) \sqsubseteq \mathrm{C}$	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \mathrm{C}^{\mathcal{I}}$
$\mathrm{C}(a)$	$a^{\mathcal{I}} \in \mathrm{C}^{\mathcal{I}}$
$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Консорциум Всемирной паутины (W3C) в рамках новой версии языка описания онтологий Web Ontology Language (OWL) утвердил профиль $OWL\ 2\ EL$ который коррелирует с логикой $\mathcal{EL}^{++}$.

Хорошие вычислительные свойства и достаточная экспрессивность $OWL\ 2\ EL$ были залогом стремительного развития онтологий на его основе. Особенно большую популярность такие онтологии получили в области медицины и генетики.

Вслед за онтологиями появилось множество логических анализаторов способные за приемлемое время производить их классификацию. Среди них особо стоит упомянуть CB, CEL, jCEL, Snoroket и ELK. Исчерпывающий обзор логических анализаторов $\mathcal{EL}^{++}$ представлен в [7].

Упомянутый выше логический анализатор ELK благодаря своей продуманной масштабируемой архитектуре стал в основу нашей работы.

Логический анализатор ELK

ELK является свободно распространяемым логическим анализатором с открытым исходным кодом для $OWL\ EL$ онтологий. По результатам тестирования он способен классифицировать онтологию SNOMED CT, содержащую более 300 000 концептов, менее чем за 5 секунд. Такой результат достигается за счёт оптимизированных алгоритмов логического анализа способных использовать преимущества многоядерных процессоров.

Логический анализ в ELK происходит путём применения к множеству исходных аксиом, содержащихся в исходной онтологии, трансформирующих правил которые порождают новые аксиомы-следствия. Список таких правил приведён в Таблице 2.

Табл. 2 Базовые правила вывода ELK

$$\mathbf{R}_{\sqsubseteq}\frac{\mathbf{C}\sqsubseteq\mathbf{D}}{\mathbf{C}\sqsubseteq\mathbf{E}}:\mathbf{D}\sqsubseteq\mathbf{E}\in\mathcal{O}\qquad \mathbf{R}^{+}_{\sqcap}\frac{\mathbf{C}\sqsubseteq\mathbf{D}_1\ \ \mathbf{C}\sqsubseteq\mathbf{D}_2}{\mathbf{C}\sqsubseteq\mathbf{D}_1\sqcap\mathbf{D}_2}:\mathbf{D}_1\sqcap\mathbf{D}_2\in\mathcal{O}$$

$$\mathbf{R}^{-}_{\sqcap}\frac{\mathbf{C}\sqsubseteq\mathbf{D}_1\sqcap\mathbf{D}_2}{\begin{matrix}\mathbf{C}\sqsubseteq\mathbf{D}_1\\ \mathbf{C}\sqsubseteq\mathbf{D}_2\end{matrix}}\qquad \mathbf{R}^{+}_{\exists}\frac{\mathbf{C}\sqsubseteq\mathbf{D}}{\exists\mathbf{S}.\mathbf{C}\rightarrow\exists\mathbf{S}.\mathbf{D}}:\exists\mathbf{S}.\mathbf{D}\in\mathcal{O}$$

$$\mathbf{R}^{-}_{\exists}\frac{\mathbf{C}\sqsubseteq\exists\mathbf{R}.\mathbf{D}}{\mathbf{D}\sqsubseteq\mathbf{D}}\qquad \mathbf{R}_{\mathcal{H}}\frac{\mathbf{D}\sqsubseteq\exists\mathbf{R}.\mathbf{C}\ \ \exists\mathbf{S}.\mathbf{C}\rightarrow\mathbf{E}}{\mathbf{D}\sqsubseteq\mathbf{E}}:\mathbf{R}\sqsubseteq^{*}_{\mathcal{O}}\mathbf{S}$$

$$\mathbf{R}^{+}_{\top}\frac{\mathbf{C}\sqsubseteq\mathbf{C}}{\mathbf{C}\sqsubseteq\top}\qquad \mathbf{R}_{\mathcal{T}}\frac{\mathbf{D}\sqsubseteq\exists\mathbf{R}.\mathbf{C}\ \ \exists\mathbf{S}.\mathbf{C}\rightarrow\mathbf{E}}{\exists\mathbf{T}.\mathbf{D}\rightarrow\mathbf{E}}:\begin{matrix}\mathbf{R}\sqsubseteq^{*}_{\mathcal{O}}\mathbf{T}\sqsubseteq^{*}_{\mathcal{O}}\mathbf{S}\\ \mathbf{Trans}(\boldsymbol{T})\in\mathcal{O}\end{matrix}$$

В работе [8], которая детально описывает алгоритм работы ELK, было доказано его корректность и полноту.

Для повышения производительности алгоритм был распараллелен таким образом, что бы максимизировать масштабируемость процесса логического анализа.

В статье [9] были представлены правила вывода для работы с номиналами – концептами содержащими всего один элемент. Для их реализации необходимо было ввести понятие достижимости $\rightsquigarrow$ которое интерпретируется следующим образом: $\mathcal{I}\models C\rightsquigarrow D$ если $\mathrm{C}^{\mathcal{I}}\neq\emptyset$ и $D^{\mathcal{I}}\neq\emptyset$

Табл. 3 Правила вывода для номиналов

$$\mathbf{R}^{-}_{\rightsquigarrow}\frac{\mathrm{G}\rightsquigarrow\mathrm{D}}{\mathrm{G}:\mathrm{D}\sqsubseteq\mathrm{D}}\qquad \mathbf{R}^{+}_{\rightsquigarrow}\frac{\mathrm{G}\rightsquigarrow\mathrm{C}\ \ \mathrm{G}:\mathrm{C}\sqsubseteq\exists\mathrm{R}.\mathrm{D}}{\mathrm{G}\rightsquigarrow\mathrm{D}}$$

$$\mathbf{R}_{\{\}}\frac{\mathrm{G}:\mathrm{C}\sqsubseteq\{\mathrm{o}\}\ \ \mathrm{G}:\mathrm{D}\sqsubseteq\{\mathrm{o}\}\ \ \mathrm{G}\rightsquigarrow\mathrm{C}\ \mathrm{G}\rightsquigarrow\mathrm{D}}{\mathrm{G}:\mathrm{C}\sqsubseteq\mathrm{D}}$$

Таким образом, возможности ELK покрывают значительную часть профиля OWL EL. Однако, для практических целей, в том числе и для нашей системы этого было недостаточно. Так ELK не поддерживает логический анализ над типами данных и рассуждения с экземплярами концептов (ABox).

Поддержку типизированных ролей (конкретных доменов) а так же разрешённых выражений с их участием существенно ограничили в профиле языка OWL EL. Единственным разрешённым выражением является равенство (=), так как его присутствие в онтологии почти не влияло на сложность её логического анализа. В то же время, некоторые не очевидные комбинации других операторов могли симулировать поведение дизъюнктивного выражения в онтологии, что является основной причиной значительного увеличения вычислительной сложности. В качестве примера рассмотрим такой набор аксиом: $A\sqsubseteq\exists R.(\leq 5)$, $\exists R.(\leq 2)\sqsubseteq B$ и $\exists R.(\geq 2)\sqsubseteq C$. Не сложно заметить, что таким образом мы неявно выразили дизъюнкцию вида $A\sqsubseteq B\sqcup C$.

За решением этой проблемы мы обратились к работе [10], целью которой было найти такие комбинации выражений с типизированными ролями, при которых сохранялось бы свойство разрешимости логического анализа за полиномиальной время. Авторами статьи были определены все возможные ограничения использования типизированных ролей при построении

концептов для множества натуральных, целых, вещественных и рациональных чисел. На основе представленных доказательств можно сделать вывод, что логика $\mathcal{EL}$ может быть расширена подобными конструкциями, а имеющиеся ограничения не столь существенны и позволяют выражать большинство необходимых выражений.

Для работы с типизированными ролями было предложено внедрить в правила вывода ELK выражения, указанные в Таблице 4, где $A, B \in N_C$, $r \in N_R$, $F \in N_F$ и $F^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \mathcal{D}$, а выражение $r_+ \rightarrow_{\mathcal{D}} \bot$ означает, что из ограничения следует пустое множество, в то время как выражение $r_+ \rightarrow_{\mathcal{D}} r_-$ означает, что ограничение справа от $\sqsubseteq$ удовлетворяет ограничение слева.

Табл. 4 Правила вывода для типизированных ролей

$\dfrac{}{A \sqsubseteq \bot}$	$A \sqsubseteq \exists F.\, r_+ \in \mathcal{O},\ r_+ \rightarrow_{\mathcal{D}} \bot$
$\dfrac{A \sqsubseteq \exists F.\, r_+}{A \sqsubseteq B}$	$\exists F.\, r_- \sqsubseteq B \in \mathcal{O},\ r_+ \rightarrow_{\mathcal{D}} r_-$
$\dfrac{A \sqsubseteq B}{A \sqsubseteq \exists F.\, r_+}$	$B \sqsubseteq \exists F.\, r_+ \in \mathcal{O}$

Следующим значительным недостатком ELK было отсутствие поддержки утверждений об индивидах ($ABox$). Возможность логического анализа с большим количеством индивидов постепенно приобретает всё большее значение для интеллектуальных систем, выдвигая новые требования к их реализации, так как размеры $ABox$ зачастую многократно превышают размеры $TBox$.

Ведётся активная работа по реализации этого функционала и вскоре ELK будет поддерживать работу с экземплярами в рамках профиля OWL EL.

Тем не менее, существует способ симулировать поддержку логического анализа над индивидами в логическом анализаторе без поддержки $ABox$. Для этого необходимо совершить серию трансформаций:

1. Заменить все утверждения о экземплярах вида $C(a)$ на $A^* \sqsubseteq C$, где $A^* -$ новый концепт репрезентирующий a. Для онтологий, составленных с помощью функционального синтаксиса OWL это означает замену аксиом $ClassAssertion(prefix{:}C\ prefix{:}a)$ на аксиомы типа $SubClassOf(\ prefix{:}a\ prefix{:}C\ )$. Для того что бы различать терминологические концепты от концептов-индивидов, имя последних формируется специальным образом либо аннотируется необходимыми метаданными.

2. Заменить все отношения (роли) между индивидами $r(a, b)$ на экзистенциальные конструкции $A^* \sqsubseteq \exists r.B^*$, где A^*и B^* – новые концепты, репрезентирующие индивидов a и b соответственно. Для OWL онтологий это означает замену выражений $ObjectPropertyAssertion$ $(prefix{:}OP\ prefix{:}a1\ prefix{:}a2)$ на выражение типа *SubClassOf (prefix:a1 ObjectSomeValuesFrom(prefix:OP prefix:a2)).*

Представленный подход может быть успешно использован в большинстве случаев после некоторой модификации интерфейса между логическим анализатором и интеллектуальной системой.

Заключение

Использование онтологий в интеллектуальных системах, несмотря на множество преимуществ, имеет один существенный недостаток, который тормозит их внедрение. Задача логического анализа в OWL DL онтологии в худшем случае имеет сложность $NExpTime$, в OWL 2 – $2NExpTime$, Horn $\mathcal{SHIQ} - ExpTime$ и т.д. При достаточно больших базах знаний практическая польза нивелируется чрезвычайно низкой скоростью её обработки.

По этой причины были направлены усилия на поиск логик, в которых бы сохранялась полиномиальная сложность логического анализа, а язык обладал бы достаточной экспрессивностью для практического применения. Семейство логик $\mathcal{EL}$ / $\mathcal{EL}^{++}$ отвечает этим требованиям.

Логический анализатор ELK зарекомендовал себя эффективным и масштабируемым инструментом для работы с онтологиями профиля OWL EL основанного на логике $\mathcal{EL}^{++}$, однако полную поддержку всех конструкций языка ещё предстоит реализовать. В этой работе были рассмотрены некоторые расширения к ELK, которые позволяют совершать логический анализ с типизированными свойствами и множеством экземпляров (ABox). Представленные расширения будут интегрированы в новую версию ELK в скором времени.

Список литературы

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. The Description Logic Handbook: Theory, Implementation, and Applications. 2nd ed. Cambridge University Press, 2007
2. R. J. Brachman, H. J. Levesque. The tractability of subsumption in frame-based description languages. In Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84), pp. 34-37, 1984.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the EL Envelope. In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, pp. 364–369, 2005.
4. F. Baader, S. Brandt, and C. Lutz. Pushing the EL Envelope Further. In Proceedings of the OWLED 2008 Workshop on OWL: Experiences and Directions, 2008.
5. А. Поспешный, С. Стиренко. GRID-DL – семантический информационный сервис ГРИД // Компьютинг, 2011. – Том 10, Выпуск 3. – С. 285 – 294.
6. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is Tractable Reasoning in Extensions of the Description Logic EL Useful in Practice? In Proceedings of the Methods for Modalities Workshop, 2005.
7. K. Dentler, R. Cornet, A. Ten Teije, N. De Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. Semantic Web, vol. 2(2), pp. 71–87, 2011.
8. Y. Kazakov, Markus Krötzsch. Concurrent Classification of EL Ontologies. In Proceedings of the 10th International Semantic Web Conference (ISWC-11), pp. 305-320, 2011.
9. Y. Kazakov, M. Krötzsch, F. Siman. Practical Reasoning with Nominals in the EL Family of Description Logics. Artificial Intelligence, 2011.
10. D. Magka, Y. Kazakov, I. Horrocks. Tractable Extensions of the Description Logic EL with Numerical Datatypes. In Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR 2010), 2010.

УДК 004.77

РОКОВОЙ А.П.

СПОСОБ АДАПТИВНОГО РЕГУЛИРОВАНИЯ ДЖИТТЕР БУФЕРА В VOIP

Предложен способ адаптивного регулирования джиттер буфера в VoIP, учитывающий прогноз будущего состояния сети. Он позволяет поддерживать величину джиттер буфера на минимальном уровне, при котором отсутствует отбрасывание пакетов на приемной стороне, обеспечивая оптимальное соотношение между величиной вносимой задержки джиттер буфером и количеством отбрасываемых пакетов. Экспериментальные результаты показали принципиальную возможность прогнозирования джиттера при передаче голоса в IP сетях.

An adaptive playout buffer to overcome jitter in VoIP is proposed. It based on forecasting of the future network state and allows maintaining the jitter buffer at the minimum level at which packets are not discarded by receiver, providing the optimal ratio between the amount of delay introduced by the jitter buffer and the number of discarded packets. The experimental results showed the fundamental possibility of predicting jitter for voice over IP networks.

1. Введение

Основными факторами, влияющими на качество передачи голоса в IP сетях, являются задержка передачи и процент потери пакетов [1]. На рисунке 1 показаны контуры качества передачи речи (удовлетворенности пользователя) для кодека G.711 с включенной функцией PLC.

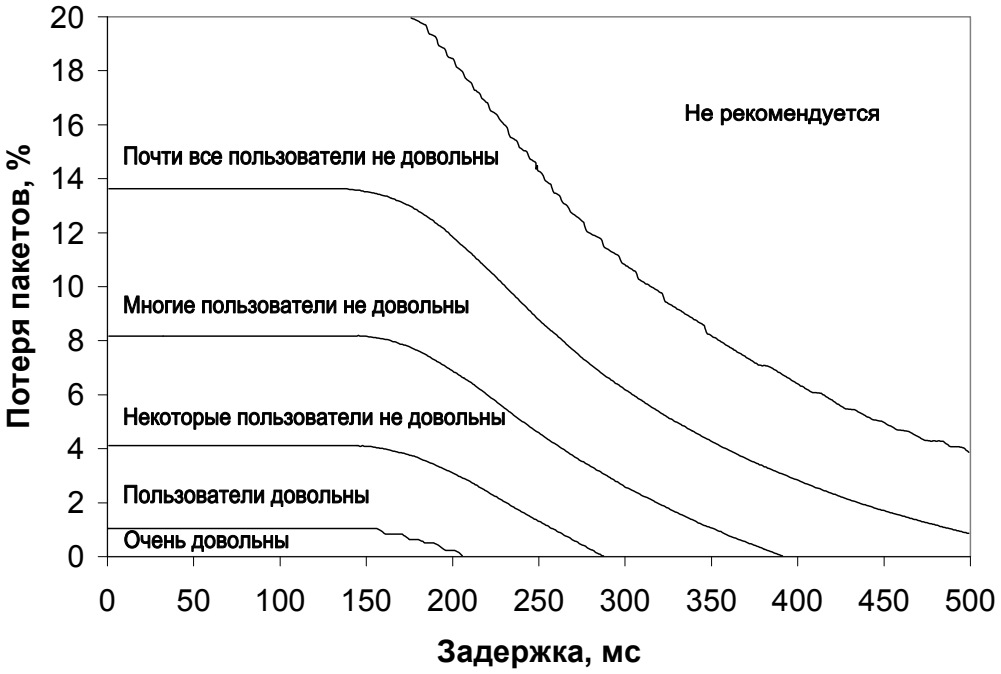


Рис. 1. Контуры качества передачи речи

На рисунке 1 видно, что качество передачи речи уменьшается с увеличением задержки и процента потери пакетов. Необходимо определить составляющие, которые вносят наибольший вклад в задержку и потерю пакетов.

Составляющие задержки при передаче голоса средствами VoIP:

1) аналого-цифровое и цифро-аналоговое преобразование сигнала;

2) кодирование/компрессия и декодирование/декомпрессия;

3) оформление блока данных и распаковка блока данных стеком протоколов TCP/IP;

4) передача данных в сети;

5) джиттер буфер.

Для большинства оконечных устройств, работающих с VoIP, потерями на аналого-цифровое и цифро-аналоговое преобразование, декодирование/декомпрессию и реализацию стека TCP/IP можно пренебречь. Средняя величина задержки, вносимой каждой из оставшихся составляющих VoIP [2, 3], представлена в таблице 1.

Таблица 1

Элемент	*Вносимая задержка*
Оптический кабель	5 мкс/км
Система наземной мобильной связи	80-110 мс
Кодек G.729 для 20 мс блоков	25 мс
Кодек GSM для 20 мс блоков	20 мс
Сетевое оборудование (L3), суммарное время нахождения в очереди и обработка	2-10 мс на узел
Джиттер буфер	20-60 мс

В соответствии с рекомендациями ITU-T G.114 [2] максимальная задержка при передаче голоса в одну сторону не должна превышать 150 мс. Таким образом, в зависимости от параметров сети, до 40% допустимой задержки может составлять время джиттер буфера. Джиттер буфер компенсирует отклонения значений задержки от среднего значения. Прибывающие пакеты на приемной стороне воспроизводятся не сразу, а с определенной задержкой. Чем больше джиттер, тем больше размер джиттер буфера требуется для компенсации изменения задержек, иначе часть пакетов будет отброшена, если они придут позже времени воспроизведения. При максимальном размере джиттер буфера появляется возможность свести количество отбрасываемых пакетов к минимуму, но при этом увеличивается время задержки. При минимальном размере джиттер буфера время задержки уменьшается, но при этом увеличивается количество отбрасываемых пакетов.

Следовательно, размер джиттер буфера должен меняться во времени по алгоритму, учитывающему текущее состояние сети. И чем динамичнее алгоритм, тем выше качество голоса на приемной стороне. Поэтому возникает задача: разработать алгоритм адаптивного регулирования джиттер буфера, работающий с опережением, то есть учитывающий прогноз будущего состояния сети.

Прогнозирование задержки и связанное с ним регулирование целесообразно проводить на конечных узлах, а не на промежуточных. Это связано с высокими требованиями к вычислительным ресурсам предлагаемого способа, а также высокой нагрузкой промежуточных узлов. При этом не требуется модификации программного обеспечения промежуточных узлов, то есть коммутаторов и маршрутизаторов. Этот подход имеет также недостатки: у промежуточных узлов больше средств регулирования трафика: приоритезация, алгоритмы управления очередями и др. Также необходимо иметь в виду, что промежуточные узлы ограничены вычислительными ресурсами и временем обработки потоков данных. Чем более интеллектуальные методы управления трафиком используются, тем больше задержки вносит каждый промежуточный узел, тем хуже качество передачи голосовых данных будет обеспечено.

2. Способы адаптивного регулирования размера джиттер буфера в VoIP

Существующие алгоритмы адаптивного регулирования джиттер буфера можно разделить на 3 класса:

1. Реактивные алгоритмы, которые непрерывно оценивают задержку и джиттер пакетов в сети для расчета граничного времени воспроизведения каждого пакета без учета процента потери пакетов. В работе [4] предложено использовать фиксированные весовые коэффициенты для задержки и джиттера пакетов в сети. Расширение этого алгоритма [5], связанное с использованием методики динамического выбора весового коэффициента задержки в сети.

2. Алгоритмы, использующие функцию распределения вероятностей задержек пакетов. Определяют размер джиттер буфера, применяя как функцию распределения измеренных задержек пакетов, так и желаемый процент потери пакетов. В [6] предполагают использование распределения Парето. Альтернативой использованию известных функций распределения стал алгоритм Concord, описанный в [7]. Этот алгоритм использует заданную гистограмму распределения задержки пакетов в качестве основы для расчета требуемой величины буфера для достижения желаемого процента потери пакетов.

3. Алгоритмы, основанные на качестве воспроизведения, определяют размер буфера путем минимизации функции, учитывающей метрики качества [8]. В [9] используется E-модель, которая является стандартным мето-

дом оценки качества речи, предложенная ITU-T [10].

Большинство из указанных алгоритмов работают по факту, то есть размер буфера изменяется после того, как было обнаружено изменение задержки. В настоящей статье предлагается использовать метод прогнозирования изменения задержки на основе сингулярно-спектрального анализа ряда предыдущих значений задержки. Это позволит поддерживать размер джиттер буфера на минимально необходимом уровне для компенсации флуктуаций времени задержки прибывающих пакетов и уменьшить количество отбрасываемых пакетов.

3. Использования сингулярно-спектрального анализа для прогнозирования задержки пакетов в IP сетях

В [11] было показано, что существует корреляция между самоподобием объема сетевого трафика и RTT.

RTT – Round Trip Time – время обращения, то есть время от момента отправки пакета-запроса в сеть до момента получения пакета-ответа из сети, состоит из времени передачи пакета в прямом и времени передачи пакета в обратном направлениях. Эта характеристика часто используется для оценки времени задержки передачи пакета от одного узла сети к другому. В случае равенства времен передачи пакета в прямом и обратном направлениях задержка получается путем деления RTT на 2.

В большинстве случаев эти времена не совпадают по ряду причин:

1) разные маршруты следования пакетов в прямом и обратном направлениях;

2) разные характеристики каналов в прямом и обратном направлениях;

3) разная загрузка каналов в прямом и обратном направлениях;

4) комбинация перечисленных выше причин.

RTT пакета состоит из суммы задержек, которые вносятся всеми промежуточными узлами и каналами на пути следования пакета. Задержка передачи пакета в Internet состоит из четырех компонент: время обработки пакета в узле, время нахождения пакета в очереди, время распространения сигнала в кабеле, время передачи пакета. При фиксированном размере пакета и одинаковых маршрутах следования изменение задержки передачи пакета может быть связано с переменным временем нахождения пакета в очередях к промежуточным узлам. Таким образом, суммарную задержку передачи пакета $T(t)$ можно выразить формулой:

$$T(t) = T_0 + \sum_{i=1}^{N} Q_i(t) \qquad (1)$$

где: T_0 – постоянная составляющая задержки передачи пакета,

N – количество очередей на маршруте следования,

$Q_i(t)$ – время нахождения пакета в i-ой очереди.

$Q_i(t)$ изменяется пропорционально изменению объема сетевого трафика: при увеличении объема трафика увеличивается время нахождения пакетов в очереди, при уменьшении объема сетевого трафика уменьшается время нахождения пакетов в очереди. В [12] показано, что $Q_i(t)$ ведет себя как процесс с медленно убывающей зависимостью (МУЗ), когда прибытие пакетов к узлу также является процессом с МУЗ, то есть обладает эффектом самоподобия.

Таким образом, можно ожидать, что и задержка передачи пакетов $T(t)$ также обладает эффектом самоподобия. Другими словами временной ряд задержек передачи пакетов имеет свойство долгой памяти, которое дает предпосылки к его прогнозированию с помощью сингулярно-спектрального анализа [13].

4. Прогнозирование джиттера прибывающих пакетов

Предлагаемый способ регулирования джиттер буфера основан на прогнозе значений задержки передачи пакетов. При этом прогноз значений задержки напрямую не учитывается для выбора размера джиттер буфера. Размер джиттер буфера определяется текущим значением джиттера пребывающих пакетов.

Значение джиттера рассчитывается в соответствии с RFC 3550 [14]. Джиттер J_i для i-го пакета определяется по формуле:

$$J_i = J_{i-1} + \frac{|D_{i-1}| - J_{i-1}}{16} \qquad (2)$$

где:

D_i – отклонение от ожидаемого времени прибытия i-го пакета.

Отклонение от ожидаемого времени прибытия i-го пакета D_i определяется по формуле:

$$D_i = (R_i - R_{i-1}) - (S_i - S_{i-1}) \quad (3)$$

где:

R – время прибытия пакета в метках времени RTP,

S – временная метка RTP, взятая из пакета.

Поскольку размер джиттер буфера определяется текущим значением джиттера для последнего прибывшего пакета, то для определения следующего размера джиттер буфера необходимо спрогнозировать значения джиттера, а не задержки для следующего пакета. При этом достаточно спрогнозировать значение джиттера на несколько шагов вперед. Долгосрочный прогноз не требуется по нескольким причинам:

1) чем дальше по временной оси от текущего времени точка прогноза, тем меньше точность прогноза;

2) для вычисления длительных прогнозов требуется больше процессорного времени оконечного узла;

3) можно выполнять коррекцию прогноза с учетом текущего значения джиттера.

Таким образом, размер джиттер буфера для следующего пакета прогнозируется с помощью сингулярно-спектрального анализа ряда предыдущих значений джиттра для прибывших пакетов. Глубина анализа ряда определяется степенью его самоподобия. Спрогнозированное значение используется для задержки следующего пакета на величину, достаточную для компенсации флуктуации времени задержки. При этом обеспечивается минимально возможная задержка, вносимая джиттер буфером, при которой отсутствует отбрасывание пакетов на приемном узле.

5. Применение сингулярно-спектрального анализа для прогнозирования значения джиттера пакетов в VoIP

Для проверки предположения о возможности прогнозирования задержки были проведены опыта в условиях мобильной и стационарной сетей.

В первом случае голосовой трафик проходил через IP сеть, в которой один из сегментов был представлен системой наземной мобильной связи стандарта CDMA2000 с технологией передачи данных EV-DO Rev. A.

Во втором случае голосовой трафик проходил через IP сеть, которая состояла только из стационарных наземных сегментов.

На одном из промежуточных узлов была запущена утилита tcpdump, захватывающая и сохраняющая в файл IP пакеты, проходящие через интерфейсы узла. На оконечных узлах с помощью стандартных средств передачи голоса в IP сети был сгенерирован трафик. Файлы с захваченными голосовыми данными переданы для анализа в приложение Wireshark [15]. В Wireshark выделен временной ряд джиттера для всех пакетов каждого потока. С помощью приложения CaterpillarSSA [16] был выполнен прогноз для временного ряда джиттера на 20 точек. Результаты прогнозирования джиттера для двух временных рядов в мобильной и стационарной сети представлены на рисунке 2 и рисунке 3 соответственно.

На рисунке 2 пунктиром показано значение джиттера для пакетов, захваченных в реальной сети, содержащей один мобильный сегмент. Сплошной ломанной показан прогноз джиттера, рассчитанный, начиная со 155 пакета, на 20 пакетов. В расчете учитывались значения джиттера 100 предыдущих пакетов: с 55 по 154. По графикам видно, что реальные значения и прогноз находятся близко друг к другу. Это позволяет сделать вывод о том, что сингулярно- спектральный анализ может использоваться для прогнозирования значении джиттера пакетов в мобильной IP сети.

На рисунке 3 пунктиром показано значение джиттера для пакетов, захваченных в реальной сети, состоящей только из стационарных сегментов. Сплошной ломанной показан прогноз джиттера, рассчитанный, начиная со 190 пакета, на 20 пакетов. В расчете учитывались значения джиттера 100 предыдущих пакетов: с 90 по 189. По графикам видно, что реальные значения и прогноз находятся близко друг к другу. Это позволяет сделать вывод о том, что сингулярно- спектральный анализ может использоваться для прогнозирования значении джиттера пакетов в стационарной IP сети.

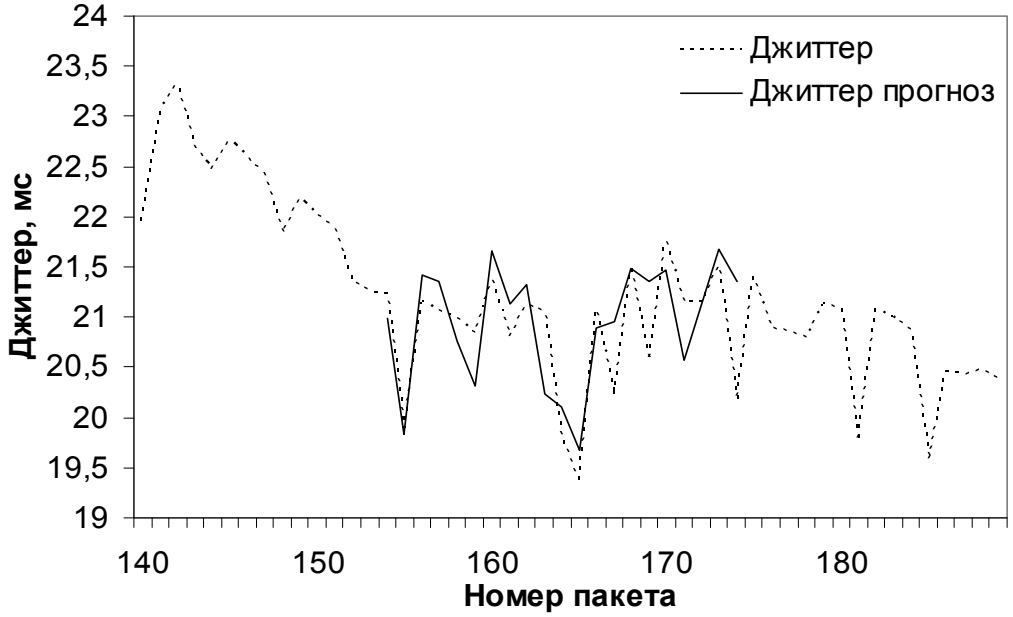


Рис.2. Прогнозирование джиттера в мобильной сети

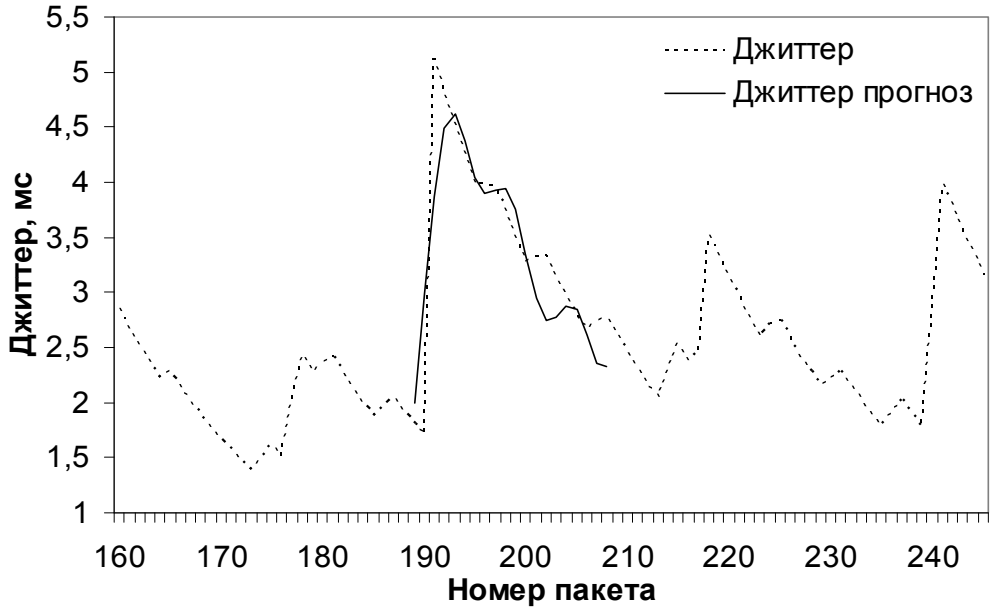


Рис. 3. Прогнозирование джиттера в стационарной сети

6. Заключение

Предложен и обоснован способ адаптивного регулирования джиттер буфера, учитывающий прогноз будущего состояния сети. Он обеспечивает минимальное количество отбрасываемых пакетов при минимальном размере джиттер буфера, а, следовательно, наивысшее качество передачи голоса в IP сети в заданных условиях.

Доказана возможность использования сингулярно-спектрального анализа для прогнозирования задержки и значения джиттера пакетов в IP сетях.

Проведены опыты для стационарной и мобильной сети, подтверждающие практическую возможность прогнозирования джиттера пакетов, а, следовательно и возможность применения предложенного способа в реальных IP сетях.

В качестве дальнейших исследований необходимо провести: выбор начальных параметров для сингулярно-спектрального анализа в зависимости от значений временного ряда задержек приходящих пакетов; зависимость усредненной субъективной оценки (MOS) предложенного способа от степени самоподобия временного ряда задержек.

Список литературы

1. Definition of categories of speech transmission quality, ITU-T Recommendation G.109, Jan 2007.
2. One-Way Transmission Time, ITU-T Recommendation G.114, May 2003.
3. Network performance objectives for IP-based services, ITU-T Recommendation Y.1541, May 2002.
4. R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, Adaptive playout mechanisms for packetized audio applications in wide-area networks, INFOCOM ’94, Networking for Global Communications, 13th Proceedings IEEE, 1994.
5. M. Narbutt, L. Murphy, A new VoIP adaptive playout algorithm, Telecommunications Quality of Services: The Business of Success, QoS 2004, IEEE, 2004.
6. K. Fujimoto, S. Ata, M. Murata, Playout control for streaming applications by statistical delay analysis, Proceedings of IEEE International Conference on Communications (ICC), 2001.
7. C. Sreenan, J.-C. Chen, P. Agrawal, B. Narendran, Delay reduction techniques for playout buffering, IEEE Transactions on Multimedia, 2000.
8. K. Fujimoto, S. Ata, M. Murata, Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications, Global Telecommunications Conference, GLOBECOM ’02, IEEE, vol. 3, 2002.
9. L. Sun, E. Ifeachor, Prediction of perceived conversational speech quality and effects of playout buffer algorithms, IEEE International Conference on Communications (ICC), vol. 1, 2003.
10. The E-Model, A computational model for use in transmission planning, ITU-T Recommendation G.109, Apr 2009.
11. Tatsuya Hagiwara, Hiroshi Majima, and Takahiro Matsuda, Impact of Round Trip Delay Self-Similarity on TCP Performance. IEEE Conference, Computer Communications and Networks, 2001, 166-171.
12. Qiong Li, David L.Mills. On the long-range dependence of packet round-trip delays in Internet. Processings of IEEE ICC'98, v.2. -1998.
13. Соловьев А.Ю. О задаче прогнозирования самоподобных сетевых процессов // II Международная научная конференция «Современные проблемы информатизации в системах моделирования, программирования и телекоммуникациях», 2009.
14. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RFC 3550, RTP: A Transport Protocol for Real-Time Applications, July 2003.
15. http://www.wireshark.org/
16. http://www.gistatgroup.com/cat/programs.html

УДК 004.942

КАЛИНОВСКИЙ Я.А.,
БОЯРИНОВА Ю.Е.

НЕКОТОРЫЕ НЕОБХОДИМЫЕ И ДОСТАТОЧНЫЕ УСЛОВИЯ ИЗОМОРФИЗМА КОММУТАТИВНЫХ ГИПЕРКОМПЛЕКСНЫХ ЧИСЛОВЫХ СИСТЕМ

В работе рассматриваются некоторые необходимые и достаточные условия изоморфизма коммутативных гиперкомплексных числовых систем, которые позволят построить эффективные алгоритмы построения классов изоморфизмов гиперкомплексных числовых систем.

In this paper some necessary and sufficient conditions for the isomorphism of commutative hypercomplex number systems that will build effective algorithms of isomorphism classes of hypercomplex number systems.

Введение

Существует бесконечное множество гиперкомплексных числовых систем (ГЧС), которые отличаются размерностью и законами композиции базисных элементов. Свойства разных ГЧС существенным образом отличаются друг от друга, что влияет на использование этих систем для математического моделирования. Поэтому так важна задача классификации ГЧС [1-3].

Линейное преобразование базиса какой-либо ГЧС приводит к базису другой системы, которая отличается от исходной. Такие системы называются изоморфными. В общем случае не каждые две ГЧС изоморфны.

Отношение изоморфности разбивает всё множество ГЧС одной размерности на некоторое количество классов, которые называются классами изоморфизмов [4,5]. Классификация систем по этому признаку распределяет все системы на классы очень близких, а по сути эквивалентных по свойствам, ГЧС. Такое распределение предполагает последовательный просмотр всевозможных коммутативных ГЧС фиксированной размерности и для каждой просматриваемой ГЧС определить, не принадлежит ли она одному из уже построенных классов изоморфизма, или является первым представителем нового класса изоморфизмов. Для выполнения такой процедуры необходимо последовательно для каждого уже построенного класса изоморфизмов решить вопрос об изоморфизме рассматриваемой ГЧС представителю класса. Так как это занимает много времени, то очень важно найти простые условия изоморфизма двух ГЧС. В данной работе эта проблема решается с привлечением представлений экспоненты.

Изоморфизм ГЧС

При изучении свойств ГЧС часто возникает вопрос изоморфности двух ГЧС. Формально эта задача имеет следующий вид.

Пусть заданы две канонические ГЧС размерности n с базисами $e=(e_1,...,e_n)$ и $f=(f_1,...,f_n)$ соответственно $\Gamma_1(e,n)$ и $\Gamma_2(f,n)$. Их таблицы умножения :

$$e_i \cdot e_j = 0 \cup \gamma_{ij}^k e_k, \quad i,j=1,...,n;\ k \in \{1...n\} \qquad (1)$$

и

$$f_i \cdot f_j = 0 \cup \gamma_{ij}^k f_k, \quad i,j=1,...,n;\ k \in \{1...n\} \qquad (2)$$

Матрицу $L=\left(l_{ij}\right)_{i,j=1,...,n}$, которая связывает базисы этих систем:

$$e_i = \sum_{j=1}^{n} l_{ij} f_j \qquad (3)$$

называют оператором изоморфизма систем $\Gamma_1(e,n)$ и $\Gamma_2(f,n)$.

Тогда условие изоморфизма заданных систем $\Gamma_1 \simeq \Gamma_2$ сводится к существованию нетривиального вещественного решения системы квадратичных уравнений

$$\sum_{j=1}^{n} l_{ij} f_j \cdot \sum_{j=1}^{n} l_{sj} f_j = \sum_{j=1}^{n} l_{kj} f_j \cup 0, \quad i,s=1,...,n;\ k \in \overline{1,...,n} \qquad (4)$$

при условии:

$$\det(L) = \det\left(l_{ij}\right)_{i,j=1,...,n} \neq 0 \qquad (5)$$

Система уравнений (4) – это результат подстановки (3) в таблицу умножения (1). Это система квадратичных уравнений относительно n^2 переменных. Она состоит из $\frac{n^2(n+1)}{2}$ уравнений. Решение таких систем вызывает значи-

тельные затруднения даже при использовании таких мощных систем аналитических вычислений, как Maple, Mathematica и др. При использовании системы Maple успешно решаются системы уравнений для $n=3$. Уже для $n=4$ время решения увеличивается до многих часов, что совершенно недопустимо при решении задачи построения классов изоморфизмов ГЧС, где необходимо решать большое количество систем квадратичных уравнений.

Приведенные резоны определяют актуальность исследований в направлении разработки таких методов установления изоморфизма между ГЧС, которые не требуют решения систем вида (4). Или хотя бы значительно уменьшают количество таких систем.

Метод построения представлений экспоненты от гиперкомплексного переменного с помощью ассоциированной системы линейных дифференциальных уравнений.

Поскольку в работе используется представления экспонент в ГЧС, то будут рассмотрены основные положения метода построения представлений экспоненты от гиперкомплексного переменного с помощью ассоциированной системы линейных дифференциальных уравнений, которые подробно изложены в [5].

Пусть рассматривается ГЧС $\Gamma(e,n)$ размерности n с базисом $e=(e_1,...,e_n)$. В дальнейшем обозначаться будут гиперкомплексные числа большими латинскими буквами:

$$X=\sum_{i=1}^{n}x_ie_i\,;M=\sum_{i=1}^{n}m_ie_i \qquad (6)$$

а вектор-столбцы, составленные из компонентов гиперкомплексных чисел- большими латинскими буквами с чертой:

$$\overline{X}=(x_1,...,x_n)^T\,;\overline{M}=(m_1,...,m_n)^T \qquad (7)$$

Представление экспоненты в системе $\Gamma(e,n)$ от числа $M\in\Gamma(e,n)$, которое будем обозначать $Exp(M)$, есть частное решение гиперкомплексного линейного дифференциального уравнения:

$$\dot{X}=MX \qquad (8)$$

при начальном условии

$$Exp(0)=\varepsilon \qquad (9)$$

где ε - единичный элемент системы $\Gamma(e,n)$.

Для построения решения гиперкомплексного линейного дифференциального уравнения (8) его необходимо представить в векторно-матричной форме. При этом:

$$\dot{\overline{X}}=(\dot{x}_1,...,\dot{x}_n)^T \qquad (10)$$

а вектор-столбец $\overline{MX}$, полученный из гиперкомплексного числа MX, можно представить в виде матричного произведения некоторой матрицы M размерами $n\times n$, элементы которой есть линейные комбинации компонентов гиперкомплексного числа M, на вектор-столбец $\overline{X}$:

$$\overline{MX}=\mathrm{M}\overline{X} \qquad (11)$$

Тогда гиперкомплексное уравнение (8) превратится в систему из n линейных дифференциальных уравнений первого порядка, которая называется ассоциированной системой линейных дифференциальных уравнений:

$$\dot{\overline{X}}=\mathrm{M}\overline{X} \qquad (12)$$

Далее необходимо найти характеристические числа $\lambda_1,...,\lambda_n$ матрицы M, то есть решить характеристическое уравнение

$$\det(\mathrm{M}-\lambda E)=0 \qquad (13)$$

Таким образом, характеристические числа $\lambda_1,...,\lambda_n$ будут зависеть от гиперкомплексного числа M.

После этого нужно построить общее решение, зависящее от n^2 произвольных постоянных, из которых n^2-n линейно зависимы от n свободных переменных. Для получения этих линейных зависимостей необходимо решить соответствующую систему линейных уравнений [1], после чего можно получить общие решения (12), зависящие от n произвольных постоянных - $\overline{X}(t,C_{1,...,}C_n)$. Значения произвольных постоянны х устанавливаются с помощью начального условия (9). Компоненты вектор-столбца решения $\overline{X}$ и будут компонентами экспоненты от гиперкомплексного числа M:

$$Exp(M)=\sum_{i=1}^{n}\overline{x}_ie_i \qquad (14)$$

Метод построения представлений экспоненты от гиперкомплексного переменного с помощью ассоциированной системы линейных дифференциальных уравнений довольно легко формализуется для построения алгори-

тмов и программ в системах символьных вычислений.

Нормальная форма представления экспоненты

В общем случае множество корней $\lambda_1,...,\lambda_n$ характеристического уравнения имеет n корней и может состоять из следующих подмножеств:

1.Подмножество однократных вещественных корней $\lambda_i \in R$; в представлении экспоненты ему соответствует слагаемое вида

$$x_i = \bar{x}_i \cdot e_i = C_i e^{\lambda_i} e_i .$$

2.Подмножество сопряженных пар комплексных корней $\lambda_i, \lambda_{i+1} = \bar{\lambda}_i \in C$.

Обычно при решения систем линейных дифференциальных уравнений для пары комплексно сопряженных корней частное решение берется в виде:

$$x = e^{\mathrm{Re}(\lambda t)}(C_1 \cos(\mathrm{Im}(\lambda)t) + C_2 \sin(\mathrm{Im}(\lambda)t)).$$

В данной работе для записи решения не будет применяться формула Эйлера, так же как и представление вещественной экспоненты через гиперболические функции

$$e^{\varphi} = ch\varphi + sh\varphi ,$$

так как это значительно усложняет структуру формулы представления и затрудняет ее анализ. Вместо этого для пары комплексно-сопряженных корней компоненты представления записываются в виде двух слагаемых:

$$x_i = \bar{x}_i \cdot e_i = C_i e^{\lambda_i} e_i ;$$
$$x_{i+1} = \bar{x}_{i+1} \cdot e_{i+1} = \bar{C}_i e^{\bar{\lambda}_i} e_{i+1} ,$$

но произвольные константы здесь уже не вещественные, а комплексные: $C_i \in C$.

3. Подмножество вещественных кратных корней.

Пусть кратность одного из наборов вещественных кратных корней равна s:

$$\lambda_{i+1} = \lambda_{i+2} = ...\lambda_{i+s}$$

Тогда, как следует из теории линейных дифференциальных уравнений, этой совокупности корней будут соответствовать s компонентов общего решения вида:

$$x_{i+j} = \bar{x}_{i+j} e_{i+j} = (P_0^j + P_1^j + ... + P_s^j) e^{\lambda_{i+j}} e_{i+j} ;\ j = 1,...,s ,$$

где P_k^j - полином k-ой степени от переменных $m_1,...,m_n$. Вид этих полиномов определяется из определяющего уравнения ассоциированной системы линейных дифференциальных уравнений.

4. Подмножество кратных пар комплексно-сопряженных корней.

Пусть кратность одного из наборов кратных пар комплексно-сопряженных корней равна s. Тогда всего в этом наборе будет $2s$ корней:

$$\lambda_{i+1} = \lambda_{i+3} = ... = \lambda_{i+2s-1} \ ;\ \lambda_{i+2} = ... = \lambda_{i+2s} = \bar{\lambda}_{i+1} .$$

Тогда этой совокупности корней соответствует 2·s компонентов общего решения вида:

$$x_{i+j} = \bar{x}_{i+j} e_{i+j} = (P_0^j + P_1^j + ... + P_s^j) e^{\lambda_{i+j}} e_{i+j} ;$$
$$x_{i+j+1} = \bar{x}_{i+j+1} e_{i+j+1} = (\bar{P}_0^j + \bar{P}_1^j + ... + \bar{P}_s^j) e^{\bar{\lambda}_{i+j}} e_{i+j+1} ;\ j = 1,3,...,2s-1 .$$

Здесь уже будут полиномы с комплексными коэффициентами.

Таким образом, представление экспоненты будет включать в себя сумму n слагаемых, каждое из которых – одночлен, у которого в первых двух случаях три сомножителя: вещественная или комплексная произвольная постоянная, экспонента от вещественного или комплексного характеристического корня и базисный элемент. В третьем и четвертом случаях – четыре сомножителя. К трем предыдущим сомножителям добавляется полином $(s-1)$-ой степени с вещественными или комплексными переменными. Такую форму представления экспоненты будем называть нормальной формой представления.

Для примера приведем оба вида представлений для некоторых ГЧС.

1. Система триплексных чисел T.

	e_1	e_2	e_3
e_1	e_1	e_2	e_3
e_2	e_2	$(e_3 - e_1/2)$	$-e_2$
e_3	e_e	$-e_2$	e_1

Корни характеристического уравнения

$$\lambda_1 = m_1 + m_3;\ \lambda_{2,3} = m_1 - m_3 \pm i m_2;$$

Ненормализованное представление:

$$Exp(m_1e_1+m_2e_2+m_3e_3)=\frac{1}{2}(e^{m_1+m_3}+e^{m_1-m_3}\cos m_2)e_1+$$

$$+e^{m_1-m_3}\sin m_2\cdot e_2+\frac{1}{2}(e^{m_1+m_3}-e^{m_1-m_3}\cos m_2)e_3$$

Нормализованное представление:

$$Exp\ (m_1e_1+m_2e_2+m_3e_3)=$$

$$\frac{1}{2}(e_1+e_3)e^{\lambda_1}+\frac{1}{4}(e_1-e_3-2ie_2)e^{\lambda_2}+$$

$$\frac{1}{4}(e_1-e_3+2ie_2)e^{\overline{\lambda}_2}=$$

$$=C_1e^{\lambda_1}+C_2e^{\lambda_2}+\overline{C}_2e^{\overline{\lambda}_2}$$

2. Система вещественно-комплексных чисел $R\oplus C$.

	e_1	e_2	e_3
e_1	e_1	0	0
e_2	0	e_2	e_3
e_3	0	e_3	$-e_1$

Корни характеристического уравнения:

$$\lambda_1=m_1;\ \lambda_{2,3}=m_2\pm im_3;$$

Ненормализованное представление:

$$Exp(m_1e_1+m_2e_2+m_3e_3)=$$
$$=e^{m_1}e_1+e^{m_2}\left(\cos m_3\cdot e_2+\sin m_3\cdot e_3\right)$$

Нормализованное представление:

$$Exp\,(m_1e_1+m_2e_2+m_3e_3)=$$

$$=e_1e^{\lambda_1}+\frac{1}{2}(e_2-ie_3)e^{\lambda_2}+\frac{1}{4}(e_2+ie_3)e^{\overline{\lambda}_2}=$$

$$C_1e^{\lambda_1}+C_2e^{\lambda_2}+\overline{C}_2e^{\overline{\lambda}_2}$$

5. Действие оператора изоморфизма на представление экспоненты

Для изоморфных ГЧС для операций сложения и умножения образ результата выполнения этих операций равен результату выполнения операции над операндами. Поэтому любое выражение с конечным числом гиперкомплексных операций преобразуется этим же линейным преобразованием.

Представление экспоненты через степенной ряд содержит счетное число операций. Однако и в этом случае, как будет показано ниже, изоморфное преобразование представления экспоненты от числа в одной ГЧС приведет к представлению экспоненты от образа этого числа в другой ГЧС.

Действительно, пусть даны две изоморфные ГЧС $\Gamma_1(e,n)\simeq\Gamma_2(f,n)$ и линейное изоморфное преобразование L:

$$\Gamma_1(e,n)\simeq\Gamma_2(f,n) \qquad (15)$$

и

$$L:e_k=\sum_{j=1}^{n}l_{kj}f_j;\ \ k=1,...,n \qquad (16)$$

Посмотрим, во что превратится число $X=\sum_{i=1}^{n}x_ie_i\in\Gamma_1(e,n)$ при переходе к системе $\Gamma_2(f,n)$ с помощью изоморфизма L:

$$\begin{aligned}&X=x_1e_1+x_2e_2+...+x_ne_n\Leftrightarrow\\&x_1(l_{11}f_1+l_{12}f_2+...+l_{1n}f_n)+\\&+x_2(l_{21}f_1+l_{22}f_2+...+l_{2n}f_n)+...\\&...+x_n(l_{n1}f_1+l_{n2}f_2+...+l_{nn}f_n)=\\&=(x_1l_{11}+x_2l_{21}+...+x_nl_{n1})f_1+\\&+(x_1l_{12}+x_2l_{22}+...+x_nl_{2n})f_2+\\&+...+(x_1l_{1n}+x_2l_{2n}+...+x_nl_{nn})f_n=\\&=y_1f_1+...+y_nf_n\in\Gamma_2(f,n)\end{aligned} \qquad (17)$$

где

$$y_i=x_1l_{1i}+x_2l_{2i}+...+x_nl_{ni} \qquad (18)$$

Тогда

$$\overline{Y}=L^T\overline{X} \qquad (19)$$

то есть компоненты гиперкомплексного числа $Y\in\Gamma_2(f,n)$ (вектор-столбец $\overline{Y}$) получаются умножением слева вектор-столбца $\overline{X}$ на транспонированную матрицу оператора изоморфного преобразования L^T.

Значит, если к экспоненте от гиперкомплексного числа X в ГЧС $\Gamma(e,n)$ применить линейное преобразование изоморфизма L, то получится экспонента от гиперкомплексного числа $Y\in\Gamma_2(f,n)$, являющегося образом числа X:

$$Exp(X)=\sum_{k=0}^{\infty}\frac{X^k}{k!}\Leftrightarrow^{L}\sum_{k=0}^{\infty}\frac{Y^k}{k!}=Exp(Y)\in\Gamma_2(f,n) \qquad (20)$$

Таким образом, подвергая изоморфному преобразование экспоненту в одной ГЧС, можно получить экспоненту в изоморфной ГЧС от чисел-образов. То же самое можно сказать и о представлениях экспонент, поскольку их построение по степенному ряду даст единственное представление.

Тогда главный результат состоит в следующем: если есть две изоморфные системы (15) и их изоморфизм (16), то изоморфное

преобразование представления экспоненты в одной из ГЧС есть представление экспоненты в другой ГЧС.

Набор корней характеристического уравнения и изоморфизм ГЧС

Если гиперкомплексная числовая система $\Gamma_1(e,n)$ является прямой суммой k числовых систем Γ_{1i}:

$$\Gamma_1 = \bigoplus_{i=1}^{k} \Gamma_{1i}, \qquad (21)$$

то нормальная форма представления ее экспоненты состоит из нормальных форм экспонент каждой из входящих подсистем. То есть число слагаемых равно числу корней характеристического уравнения, которое равно, в свою очередь, размерности всей ГЧС. Каждое слагаемое определяется, прежде всего, одним из корней характеристического уравнения.

При переходе линейным преобразованием от базиса e системы $\Gamma_1(e,n)$ к изоморфной ей системе $\Gamma_2(f,n)$, корни характеристического уравнения, входящие в слагаемые экспоненты системы $\Gamma_1(e,n)$ будут изменятся. Так как эти корни являются функциями от компонентов числа $\overline{M}$, то они будут меняться по (17), то есть умножением слева вектор–столбца $\overline{M}$ на транспонированную матрицу оператора изоморфного преобразования L^T. Значит, корни характеристического уравнения преобразуются линейно. А это означает, что их тип не меняется: разные вещественные корни переходят в разные вещественные, разные комплексные в разные комплексные, одинаковые корни – в одинаковые, вещественные не могут преобразоваться в комплексные и наоборот. Действительно, уравнение (13) можно представить в виде

$$(\lambda-\lambda_1)(\lambda-\lambda_2)...(\lambda-\lambda_n)=0 \qquad (22)$$

где λ_i, $i=1,...,n$ – корни характеристического уравнения. Поскольку они зависят от компонентов $\overline{M}$, то линейное преобразование их не меняет типа. А это означает, что нормальная форма экспоненты системы $\Gamma_2(f,n)$ имеет такую же структуру, что и экспонента в системе $\Gamma_1(e,n)$.

Если базис системы $\Gamma_1(e,n)$ преобразовать другим линейным преобразованием, то получится система $\Gamma_3(g,n)$, изоморфная $\Gamma_1(e,n)$:

$$\Gamma_3(g,n) \simeq \Gamma_1(e,n)$$

и, ввиду транзитивности отношения изоморфизма, получаем:

$$\Gamma_3(g,n) \simeq \Gamma_2(f,n).$$

Таким образом, преобразуя всевозможными невырожденными линейными преобразованиями какой-либо базис, которому соответствует фиксированный набор корней характеристического уравнения, можно получить весь класс изоморфизмов. То есть данному классу изоморфизмов будет соответствовать один и только один набор корней характеристического уравнения.

К сожалению, обратное утверждение неверно – одному и тому же набору корней могут соответствовать различные неизоморфные ГЧС. Это может произойти в том случае, когда в составе корней характеристического уравнения есть кратные вещественные или (и) комплексные корни кратности, большей 2. Корням такой кратности соответствуют несколько классов изоморфизмов неразложимых ГЧС, а кратности 2 соответствует только один класс изоморфизмов системы дуальных чисел D с таблицей умножения:

D	e_1	e_2
e_1	e_1	e_2
e_2	e_2	0

Кратности 3 соответствуют 2 класса изоморфизмов. Таблицы умножения представителей классов приведены ниже.

Γ_{31}	e_1	e_2	e_3
e_1	e_1	e_2	e_3
e_2	e_2	0	0
e_3	e_3	0	0

Γ_{32}	e_1	e_2	e_3
e_1	e_1	e_2	e_3
e_2	e_2	e_3	0
e_3	e_3	0	0

Кратности 4 будут соответствовать 6 классов изоморфизмов. Таблицы умножения представителей классов приведены ниже.

Γ_{41}	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	$-e_1$	e_4	$-e_3$
e_3	e_3	e_4	0	0
e_4	e_4	$-e_3$	0	0

Γ_{42}	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	0	0	0
e_3	e_3	0	0	0
e_4	e_4	0	0	0

Γ_{43}	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	e_3	0	0
e_3	e_3	0	0	0
e_4	e_4	0	0	0

Γ_{44}	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	e_4	0	0
e_3	e_3	0	e_4	0
e_4	e_4	0	0	0

Γ_{45}	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	e_4	0	0
e_3	e_3	0	$-e_4$	0
e_4	e_4	0	0	0

Γ_{46}	e_1	e_2	e_3	e_4
e_1	e_1	e_2	e_3	e_4
e_2	e_2	e_4	e_4	0
e_3	e_3	e_4	0	0
e_4	e_4	0	0	0

В обоих ГЧС характеристические уравнения будут иметь трехкратные корни $\lambda_{1,2,3} = m_1$. В ГЧС Γ_{41} уравнение имеет двукратную пару комплексно-сопряженных корней:

$$\lambda_{1,2} = m_1 \pm im_2; \lambda_{3,4} = m_1 \pm im_2 .$$

Остальные ГЧС имеют вещественные четырехкратные корни $\lambda_{1,2,3,4} = m_1$. Поэтому, судя по характеристическим корням, всегда можно утверждать, что система Γ_{41} не изоморфна ни одной из остальных систем и обратно. Но об изоморфизме в совокупности систем Γ_{42}, $\Gamma_{43}, \Gamma_{44}, \Gamma_{45}, \Gamma_{46}$ только по характеристическим корням ничего сказать нельзя. Использованием других методов, установлено, что в этой совокупности нет ни одной хотя бы пары изоморфных систем.

Если кратность характеристических корней ГЧС равна ее размерности, то ГЧС неизоморфна никакой прямой сумме ГЧС низших размерностей, ибо в противном случае она бы имела нормальную форму представления экспоненты совершенно другой структуры. Если имеется еще одна ГЧС с такими же свойствами, то судить об их изоморфизме только по виду наборов характеристических корней нельзя. Для решения этого вопроса необходимы дальнейшие исследования.

Выводы

Пусть количественный состав корней характеристического уравнения (13) по всем подмножествам типов: вещественных разных, пар комплексно-сопряженных, кратных вещественных по их кратностям и кратных пар комплексно-сопряженных корней также по их кратностям называется *типом набора корней.* Тогда, можно сформулировать условия изоморфности пар ГЧС одной размерности.

1) Если *тип набора корней* одной ГЧС отличается хотя бы в одном компоненте от *типа набора корней* другой ГЧС, то эти ГЧС являются неизоморфными. Это условие является достаточным, но не необходимым, так как существуют пары ГЧС с одинаковыми *типами наборов корней*, у которых есть кратные корни, но вопрос об изоморфизме которых остается открытым.

2) Если в наборе корней пары ГЧС отсутствуют корни с кратностями выше второй, то необходимым и достаточным условием их изоморфности является совпадение их *типов наборов корней.*

3) Для пары ГЧС в случае совпадения *типов наборов корней*, в которых имеются корни кратности выше второй, необходимым и достаточным условием изоморфности является существование обратимого линейного преобразования, связывающего базисы этих ГЧС.

Список литературы

1. Olariu S. Complex numbers in N dimensions // ELSEVIER Science B.V. 2002. – P.242
2. Petersson H.P. The classification of two-dsmensional nonassociative algebras // RosultMath. – 2000. – Vol. 37. – P. 120–154.
3. Lounesto P. Octonions and triality // Advances in Applied Clifford algebras –2001. –№.2. –191–213p.
4. Khosravi B. The Number of Isomorphism Classes of Finite Groups with the set of Order Components of $C_4(g)$ // Applicable Algebra in Engeeniring, Communication and Computing. – 2005. – Vol. 15. – P. 349–359.
5. Синьков М.В. Конечномерные гиперкомплексные числовые системы. Основы теории. Применения. / М.В. Синьков, Ю.Е. Бояринова, Я.А. Калиновский. – К.: Инфодрук, 2010.– 388с.

Відомості про авторів

Агеєнко Ю.М.	студент кафедри ОТ НТУУ «КПІ»
Амонс О.А.	к.т.н., доцент кафедри АУТС НТУУ «КПІ»
Барановський О.М.	аспірант ФТІ НТУУ "КПІ"
Безгинский М.А.	студент кафедри ОТ НТУУ «КПІ»
Болдак А.О.	к.т.н., докторант, доцент кафедри ВТ НТУУ «КПІ»
Бояринова Ю.Е.	к.т.н , с.н.с., ИПРИ НАНУ
Булах Б.В.	аспірант ІПСА НТУУ "КПІ"
Виноградов Ю.М.	старший викладач кафедри ОТ НТУУ «КПІ»
Волокита А.Н.	к.т.н., докторант кафедри ОТ НТУУ «КПІ»
Ву Дык Тхинь	аспірант кафедри ОТ НТУУ «КПІ»
Грибенко Д.В.	студент кафедри ОТ НТУУ «КПІ»
Дорогий Я.Ю.	асистент кафедри АУТС НТУУ «КПІ»
Дорошенко К.С.	асистент кафедри АУТС НТУУ «КПІ»
Єфремов К.В.	директор Української філії Світового Центру даних НТУУ "КПІ"
Жабин В.И.	д.т.н., професор кафедри ОТ НТУУ "КПІ"
Жабина В.В.	старший викладач каф. програмного забезпечення комп. систем НТУУ "КПІ"
Зиненко А.И.	студент кафедри ОТ НТУУ «КПІ»
Зубенко Г.А.	студент кафедри АУТС НТУУ «КПІ»
Исаченко Г.В.	студент кафедри ОТ НТУУ «КПІ»
Калашник В.В.	студент кафедри АСОІУ НТУУ «КПІ»
Калиновский Я.А.	д.т.н., пр.н.с. ИПРИ НАНУ
Качинський А.Б.	д.т.н., професор кафедри ІБ ФТІ НТУУ "КПІ"
Киричек О.О.	студент кафедри АУТС НТУУ «КПІ»
Кирюша Б.А.	ННК "ІПСА" НТУУ "КПІ"
Коханевич И.В.	студент кафедри ОТ НТУУ «КПІ»
Кравець П.І.	к.т.н., доцент кафедри АУТС НТУУ «КПІ»
Ладогубец В.В.	к.т.н., доцент кафедри САПР НТУУ «КПІ»
Луцкий Г.М.	д.т.н., професор, завідувач кафедри ОТ НТУУ "КПІ"
Ляпин П.С.	ННК "ІПСА" НТУУ "КПІ"
Марковский А.П.	к.т.н., доцент кафедри ОТ НТУУ «КПІ»
Мельников О.В.	н.с. каф. АСОИУ НТУУ "КПІ"
Мельничук Р.М.	НТУУ «КПІ»
Мисюра Е.Б.	к.т.н, с.н.с. каф. АСОІУ НТУУ "КПІ"
Михайлов В.В.	к.т.н, с.н.с. каф. АСОІУ НТУУ "КПІ"
Нища О.В.	НТУУ «КПІ»
Павлов А.А.	д.т.н., професор каф. АСОИУ, декан факультету ІОТ НТУУ "КПІ"
Петренко А.І.	д.т.н., професор, зав. Кафедри САПР ННК ІПСА НТУУ "КПІ"
Погребнюк І.М.	аспірант кафедри ІСТ Національного Транспортного Університету
Полторак В.П.	к.т.н., доцент кафедри АУТС НТУУ «КПІ»
Поспешный А.С.	аспірант кафедри ОТ НТУУ «КПІ»
Роковой А.П.	старший викладач кафедри ОТ НТУУ «КПІ»
Ролик А.И.	к.т.н., доцент кафедри АУТС НТУУ «КПІ»
Рябикіна В.О.	НТУУ «КПІ»
Семенюк Ю.В.	НТУУ «КПІ»
Стиренко С.Г.	к.т.н., доцент кафедри ОТ НТУУ «КПІ»
Теленик С.Ф.	д.т.н., професор, зав. кафедри АУТС НТУУ «КПІ»
Томашевський В.М.	професор кафедри АСОІУ НТУУ "КПІ"

Уваров Н.В.	НТУУ «КПІ»
Уварова Н.В.	НТУУ «КПІ»
Федоречко О.И.	стуент кафедри ОТ НТУУ "КПІ"
Финогенов А.Д.	к.т.н., каф. СП ІПСА НТУУ "КПІ"
Халимон А.Ю.	студент кафедри АСОІУ НТУУ «КПІ»
Хмелюк В.С.	асистент кафедри АУТС НТУУ «КПІ»
Чалий О.І.	студент кафедри АУТС НТУУ «КПІ»
Чекалюк В.В.	аспірант ІПСА НТУУ "КПІ"
Шевченко К.Ю.	магістр кафедри АСОІУ НТУУ «КПІ»
Шимкович В.М.	аспірант кафедри АУТС НТУУ «КПІ»
Шовгун Н.В.	аспірант ІПСА НТУУ "КПІ"
Щербатенко О.В.	к.т.н, с.н.с. каф. АСОИУ НТУУ "КПІ"
Ясінський В.В.	директор Інституту моніторингу якості освіти НТУУ "КПІ", професор
Яцишин А.Ю.	аспірант НТУУ «КПІ»