ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
"Київський політехнічний інститут"

Інформатика, управління та обчислювальна техніка

57

2012

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

ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТА УКРАЇНИ «КПІ»

Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 57

Київ “ВЕК+” 2012

УДК 004

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2012. – № 57. – 171 с.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 4 від 26.11.2012

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 26.11.2012. Формат 60×84 1/16. Гарнітура Times. Папір офсетний № 1.
Наклад 150 прим.
Надруковано в ЗАТ “ВІПОЛ”, 03151 м.Київ, вул. Волинська, 60.

ЗМІСТ

УДК 004.722

КУЛАКОВ Ю.А.,
ГОРОБЕЦ А.Н.

АЛГОРИТМ СОЗДАНИЯ ТОПОЛОГИИ БИНОМИАЛЬНЫЙ ГРАФ В ПОЛНОСВЯЗНОЙ СИСТЕМЕ

В данной работе предложен алгоритм формирования топологии биномиальный граф в полносвязной системе. Параллельно выполняющиеся части алгоритма формируют кольцевую топологию, а затем заполняют в каждом узле соответствующие списки, в соответствии с правилами, которые описывают данную топологию.

In this paper we described an algorithm for the constructing binomial graph topology in a fully connected system. Parallel execution of the algorithm parts form a ring topology, and then fill in lists of links to nodes, in accordance with rules that describe binomial graph topology.

Создание топологии биномиальный граф.

Для систем, которые состоят из набора параллельно выполняющихся процессов, важным параметром является задержка при обмене информацией. Обмен сообщениями между процессами происходит при помощи исполнительной среды. Конфигурация данной среды должна оптимальным образом распределять нагрузку на узлы системы. В качестве топологии, которая будет снижать нагрузку на пересылки, была предложена топология биномиальный граф [2]. Для её построения, в исходном алгоритме [7] в качестве базовой топологии указано дерево процессов. Выбор такой топологии не случаен – такая топология на самом деле очень легка для создания – процессы, решающие сложные задачи, или изначально разработанные как многопоточные создают новые потоки, запускают на выполнение другие задачи. В процессе работы выстраивается иерархия процессов, которая и представляет собой дерево[1].

Подобное решение также вызвано тем, что при инициации параллельной исполнительной среды, для которой будет строиться данная топология, системный загрузчик изначально создает дерево процессов, которое содержит необходимую для данного алгоритма информацию. У каждого процесса в дереве есть идентификатор своего родительского процесса (как указатель или определенное уникальное значение), идентификаторы дочерних процессов.

В топологии кольцо каждый узел знает идентификатор узла, который следует за ним, и идентификатор предыдущего узла. Такая топология равномерно распределяет нагрузку, связанную с передачей сообщений между узлами. Помимо этого, в каждом элементе кольца формируются два массива идентификаторов узлов. Данные узлы расположены на расстоянии степеней двойки от данного узла, в направлении по часовой и против часовой стрелки.

Алгоритм построения биномиального графа из дерева

Для создания выше описанной топологии используется алгоритм, в котором можно выделить два под алгоритма: алгоритм создания кольца процессов из дерева процессов, и алгоритм создания биномиального графа из кольца процессов [3, 6].

Алгоритм построения кольца процессов из дерева процессов представлен в работе [7].

Основной идеей первого алгоритма является создание цепочек из родителей и их первых потомков, и соединение этих цепочек в кольцо.

Создание цепочек происходит с помощью правил 1 и 2 из алгоритма 1. Правило 1 может быть использовано любым процессом, у которого есть хотя бы один процесс-наследник. При выполнении его, первый процесс-наследник записывается как следующий процесс в кольце. Посылая сообщение со своим идентификатором, процесс устанавливает себя как предыдущий элемент кольца для своего первого дочернего процесса при помощи правила 2.

Следует отметить, что в результате получится цепочки, с «листом» дерева на одном конце, и каждый лист дерева будет являться конечной точкой какой-либо цепочки.

Объединение цепочек в кольцо сводиться к поиску для каждого «листа» первого свободного процесса для установки его как следующего в кольце [5]. Правило 3 может быть выполнено для листа, и в результате будет посла-

но сообщение Info в родительский процесс. Правило 4 описывает каким образом поток должен реагировать на получение сообщения Info.

Во втором алгоритме каждый узел регулярно посылает идентификаторы своих соседей друг другу (правило 1). Когда процесс получил идентификатор процесса на расстоянии 2^i, он сохраняет его в соответствующем массиве, и пересылает его на расстояние 2^i в обоих направлениях.

Построение топологии биномиальный граф в среде со случайной конфигурацией

Для реализации алгоритма в системе, в которой не создано дерево процессов программы, необходимо видоизменение связей среды для создания дерева. С учетом того, что конфигурация дерева не имеет кардинального значения для алгоритмов, описанных выше, алгоритм построения топологии дерева сводится к выбору корневого узла и пошагового формирования веток дерева.

Для определения корневого узла будем исходить из того, что алгоритм построения кольца из древовидной структуры выполняется параллельно для каждой ветки, соответственно выбор корневым узлом узла с наибольшей степенью положительно скажется на конфигурации дерева.

Алгоритм создания дерева будет выглядеть следующим образом:

1) Поиск узла с наибольшей степенью и установка его как корневого

2) Установка всех связанных с текущим узлом узлов как дочерних узлов в дереве.

3) Всем дочерним узлам текущий узел указывается как родительский узел в дереве. Все дочерние узлы добавляются в список.

4) Пока в списке есть хоть один узел:

– Первый узел в списке устанавливается как текущий;

– Все узлы, которые непосредственно связаны с текущим, не отмеченные как обработанные и не находятся в списке указываются как дочерние для текущего;

– Всем дочерним узлам текущего узла текущий узел указывается как родительский в дереве. Все дочерние узлы добавляются в список;

– Текущий узел отмечают как обработанный и удаляют из списка.

Основным преимуществом данного алгоритма является отсутствие комплексного преобразования сети. Дерево формируется на базе уже существующих связей.

Алгоритм создания топологии биномиальный граф для полносвязной системы

Алгоритмы, рассмотренные выше, являются избыточными для сильно связных систем. Особенно явно это проявляется в полносвязных системах, так как необходимые для биномиального графа связи уже присутствуют, и создание промежуточных топологий с переконфигурированием системы будет явно излишним. Для подобных систем необходим алгоритм, который выделит необходимые связи из уже имеющихся, и создаст соответствующие списки соседей слева и справа для каждого из узлов. Также алгоритм должен выполняться параллельно на каждом из узлов, что ускорит построение. Топологии биномиальный граф.

В общем случае, для каждого из N узлов необходимо создать два списка – CW и CCW, которые содержат узлы, находящиеся на расстоянии $\{(i+1) \bmod n, (i+2) \bmod n, \dots, (i+2^k) \bmod n \mid 2^k <= n\}$ и $\{(n-i+1) \bmod n, (n-i+2) \bmod n, \dots, (n-i+2^k) \bmod n \mid 2^k <= n\}$ соответственно[4].

Связи узлов на расстоянии 1 формируют кольцевую топологию, конкретная реализация которой не имеет особого значения для формирования остальной части списков.

Каждый конкретный узел может сформировать связь на расстоянии 2^i для любых двух узлов на расстоянии 2^{i-1} слева и справа. Узлы получают сообщение, в котором указан идентификатор узла и расстояние до него, и устанавливают соответствующие значения в своих списках CW и CCW.

Алгоритм построения топологии биномиальный граф в полносвязной системе будет состоять из двух частей. Первая часть алгоритма случайным образом будет формировать кольцевую топологию, выбирая соседние узлы на расстоянии 1. Вторая часть алгоритма будет формировать соответствующие списки CW и CCW для каждого из узлов.

Исходными данными для алгоритма является список идентификаторов доступных узлов из текущего узла. Алгоритм выполняется параллельно на каждом из узлов топологии.

Алгоритм построения топологии биномиальный граф в полносвязной сети.

1) Послать в случайный узел запрос наличия «соседа справа». При получении в ответе своего идентификатора установить узел, в который отправлялся запрос, как текущий «сосед слева» и добавить его идентификатор в список CW.

2) Ожидать запроса наличия «соседа справа». При получении такого запроса, если «сосед справа» не определен, установить узел, который прислал запрос, как текущий «сосед справа», добавить его идентификатор в список CCW и отправить идентификатор текущего «соседа справа» в ответе.

3) Отправить идентификатор текущего «соседа справа» в узел с идентификатором указанным в качестве текущего «соседа слева».

Отправить идентификатор текущего «соседа слева» в узел с идентификатором указанным в качестве текущего «соседа справа».

4) Ожидать получения идентификатора узла от текущего «соседа справа». При получении, если полученный идентификатор не содержится в списке CW и не является текущим «соседом слева», то установить полученный идентификатор как текущий «сосед слева» и добавить его в список CW. Иначе установить сигнал завершения.

Ожидать получения идентификатора узла от текущего «соседа слева». При получении, если полученный идентификатор не содержится в списке CCW и не является текущим «соседом справа», то установить полученный идентификатор как текущий «сосед справа» и добавить его в список CCW. Иначе установить сигнал завершения.

5) Если не установлен сигнал завершения, перейти к пункту 3.

Пункты 1 и 2 являются первой частью алгоритма. Они выполняются параллельно на всех узлах, и случайным образом формируют кольцо узлов. Пункты 3 и 4 выполняются в цикле, формируя списки CW и CCW узлов, до тех пор, пока не будет достигнуто максимальное значение расстояния ({(i+1) mod n, (i+2) mod n, … , (i+2^k) mod n | 2^k <= n} и {(n-i+1) mod n, (n-i+2) mod n, … , (n-i+2^k) mod n | 2^k <= n} для соответствующих списков). Это условие выполнится, когда для каждого из узлов будут достигнуты наиболее удаленные в соответствии со структурой топологии узлы.

Данный алгоритм, в полносвязной системе будет формировать топологию биномиальный граф более оптимальным способом. Это достигается за счет отсутствия промежуточных преобразований топологии, параллельного выполнения частей алгоритма на каждом из узлов.

Пример работы алгоритма

Для создания выше описанной топологии в полносвязной системе на 8 узлов будут произведены следующие действия. Каждый узел будет иметь уникальный идентификатор и список связей со всеми узлами в системе. На рисунке 1 показано исходное состояние системы.

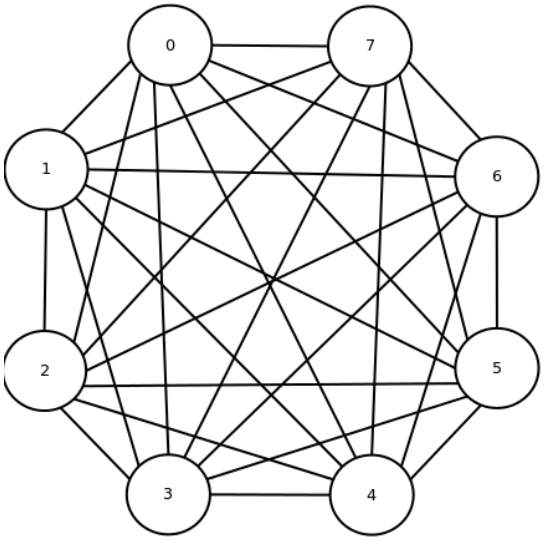


Рис. 1. Исходное состояние системы

На первом этапе проводится формирование кольца узлов случайным образом. Результат выполнения первого этапа представлен на рисунке 2.

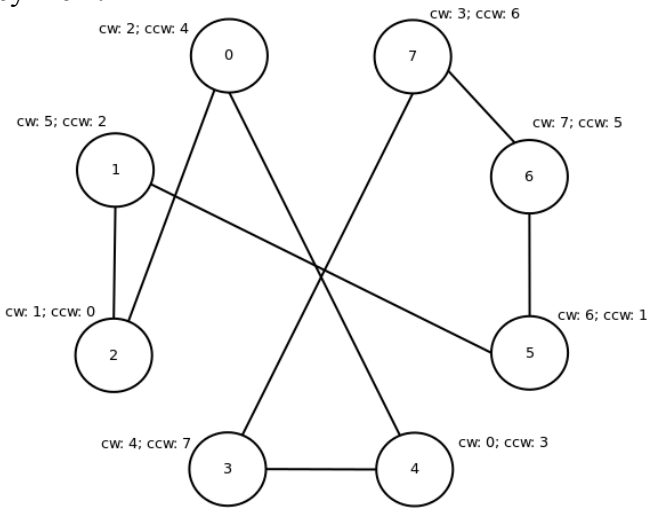


Рис. 2. Результат работы первого этапа алгоритма

Как видно из рисунка, у каждого из узлов начали формироваться списки соседних узлов

по часовой стрелке (CW) и против часовой стрелки (CCW).

На следующем этапе продолжается формирование данных списков. На первом шаге в список добавляются узлы, которые находятся на расстоянии 2. Как описывалось выше, это происходит в результате отправки текущего узла из списка соседних узлов слева в узел, который является текущим в списке соседних узлов справа, и наоборот. Результат первого шага второй части алгоритма представлен на рисунке 3.

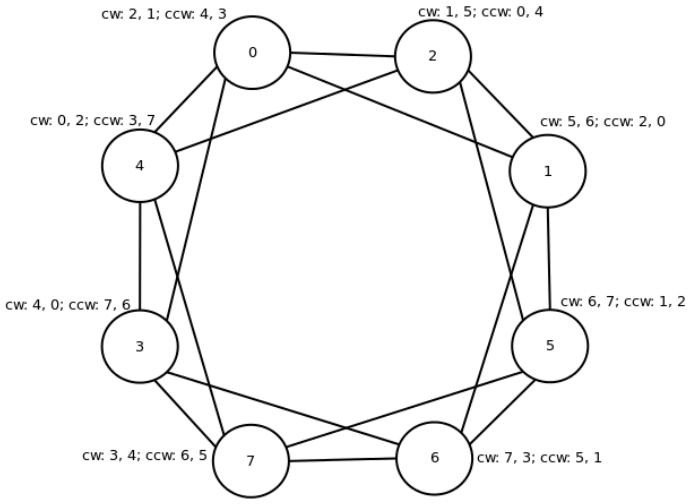


Рис. 3. Результат первого шага второй части алгоритма

Условие завершения работы второго этапа алгоритма не выполняется, следовательно, производится следующий шаг второй части алгоритма. Результат его работы представлен на рисунке 4.

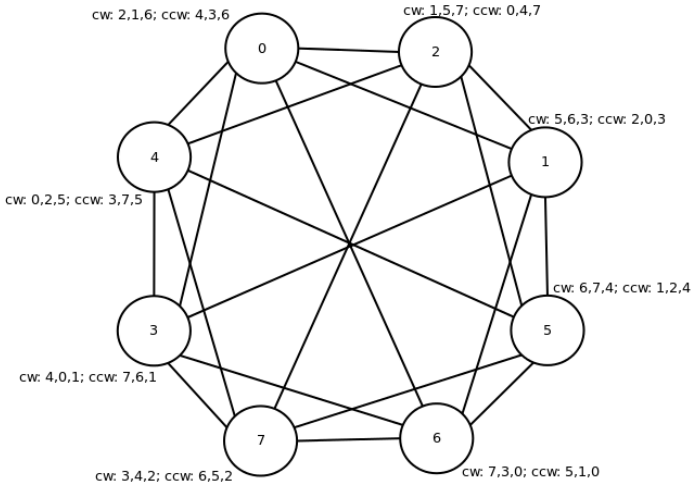


Рис. 4. Результат работы второго шага второй части алгоритма

Как видно из рисунка 4, текущий узел из списка CW является также текущим узлом в списке CCW, что является условием окончания работы второй части алгоритма.

В результате работы алгоритма в каждом из узлов было сформировано два списка узлов, в которых содержатся узлы, находящиеся на расстоянии 2^k, что собственно формирует список связей для топологии биномиальный граф.

Список литературы

1. Sutter, H. Software and concurrency revolution / H. Sutter, J. Laurus // Queue. – New York: ACM, 2005. – Vol. 3, № 7. – P. 54-62. – ISSN 1542-7730.
2. T. Angskun, G. Bosilca, and J. Dongarra. Binomial graph: A scalable and fault-tolerant logical network topology. – Heidelberg: Parallel and Distributed Processing and Applications, ISPA 2007 – Vol. 4742/2007 of Lecture Notes in Computer Science – P. 471–482. Springer Berlin /.
3. Lemarinier P. Constructing Resiliant Communication Infrastructure for Runtime Environments / Pierre Lemarinier George Bosilca, Camille Coti, Thomas Herault, and Jack Dongarra (University of Tennessee Knoxville, Universite Paris Sud, INRIA) – 2009 P. 4-5.
4. Bacon, D.F. Compiler transformations for high performance computing / D.F. Bacon, S.L. Graham, O.J. Sharp // ACM Computing Surveys. – New York: ACM, 1994. – Vol. 26, № 4. – P. 345-420. – ISSN 0360-0300.
5. Hendren, L. J. Parallelizing programs with recursive data structures / L. J. Hendren, A. Nicolau // IEEE Transactions on Parallel and Distributed Systems. – Piscataway, New Jersey, USA: IEEE Press, 1990. – Vol. 1, № 1. – P. 35-47. – ISSN 1045-9219.
6. Herault T. A model for large scale self-stabilization. / Thomas Herault, Pierre Lemarinier, Olivier Peres,. Laurence Pilard, Joffroy Beauquier // IEEE International on Parallel and Distributed Processing Symposium, march 2007 – P. 1–10, – IPDPS 2007.
7. Bosilca, G. Constructing Resiliant Communication Infrastructure for Runtime Environments / George Bosilca, Camille Coti, Thomas Herault, Pierre Lemarinier, Jack Dongarra // University of Tennessee Knoxville University of Tennessee Knoxville, Universite Paris Sud, INRIA

УДК 004.89:681.518

РОЛИК А.И.,
КОНОНЕНКО Ю.А.

МЕТОД ОЦЕНКИ СОСТОЯНИЯ ЭЛЕМЕНТОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Предложен метод оценки состояния элементов информационно-телекоммуникационных систем с использованием нейронных сетей. Приведена сравнительная характеристика результатов оценки состояния с помощью нейронных сетей различных типов. Разработан метод оценки состояния элементов с помощью машины опорных векторов.

The method of components state appraisal of Information and telecommunication systems' with neural networks application is proposed. The comparative description of the state appraisal results with neural networks of various types is given. The method of components state appraisal with the assistance of support vector machine is designed.

Введение

Бизнес рассматривает информационные технологии (ИТ) в качестве средства повышения своей производительности и улучшения конкурентоспособности. Эффективность и надежность предоставления ИТ-услуг бизнес-пользователям зависит от надежного и качественного функционирования информационно-телекоммуникационной системы (ИТС). ИТС, как совокупность информационной системы и телекоммуникационной сети, является информационно-инфраструктурной основой выполнения бизнеса. Для автоматизации управления ИТС и поддержки принятия решений администраторами информационно-коммуникационных технологий разрабатываются и внедряются системы управления ИТС (СУИ) [1]. Для эффективного управления ИТС необходимо, чтобы СУИ получала и обрабатывала информацию о состоянии всех составляющих ИТС. Поэтому данная статья, посвященная разработке метода оценки состояния элементов и подсистем ИТС с использованием нейронных сетей, является актуальной.

Постановка проблемы

Для поддержания значений параметров, характеризующих надежность и эффективность функционирования ИТС на заданном соответствующим регламентом уровне, администраторы должны максимально быстро обнаруживать и устранять неисправности в ИТС, а также своевременно осуществлять мероприятия по поддержанию параметров производительности ИТС на заданном уровне. Для автоматизации выполнения этих функций в СУИ должна непрерывно поступать информация о состоянии элементов мониторинга и управления (ЭМУ), а также результаты анализа тенденций изменения этих состояний, необходимые для проактивного управления ИТС. Поскольку для оценки состояния ЭМУ необходимо выявлять связи и оценивать значения разных по типу быстроизменяющихся параметров, учитывать зависимости между состояниями элементов различных иерархических уровней ИТС, быстро адаптироваться к изменению структуры связей параметров и вкладу параметров в общую оценку состояния ЭМУ, целесообразным видится применение нейронных сетей при оценке состояния ЭМУ. Поэтому возникает необходимость определения типа нейронной сети, которая покажет наилучшие результаты при решении задач автоматической оценки состояния ЭМУ.

Анализ публикаций

В [1] предлагается метод кодирования состояний ЭМУ, а оценка работоспособности ЭМУ осуществляется с помощью тестовых проверок, когда для обнаружения неисправного элемента производится последовательность тестов, причем каждая последующая проверка выбирается с учетом результатов предыдущей проверки. Для успешной реализации этого метода необходимо иметь обширную базу знаний, кроме того, требуется значительное время для поиска неисправного ЭМУ.

В [2] предложено оценивать работу элементов и подсистем ИТС по интегральному показателю качества функционирования. В [3] предложен метод сведения метрик оценки качества

функционирования ЭМУ, позволяющий своевременно реагировать на изменение состояния ЭМУ. Достоинством этих подходов является их гибкость. Однако, при увеличении количества параметров значительно увеличивается объем вычислений.

В [4] рассмотрен агентский подход к анализу и оценке состояния ЭМУ. Сравниваются два способа получения значений о параметрах, характеризующих состояние элементов: непосредственное обращение сервера СУИ и опрос элементов ИТС агентами с передачей обобщенных результатов серверу СУИ.

Целью данной работы является разработка метода оценки состояния элементов ИТС с использованием нейронных сетей и определения типа сети, наиболее подходящего для решения задач оценки состояния.

Суть предлагаемого метода оценки состояния элементов ИТС

Анализ известных подходов к определению состояния элементов ИТС [1–4] позволяет сделать вывод о том, что задачу оценки и прогнозирования состояния ЭМУ можно свести к задаче создания технологии, в которой математические модели элементов ИТС играют роль супервизоров для искусственных нейронных сетей (НС).

Для обучения НС формируют множество $R=\{S,W,Y\}$, где $S=\{s_n\}$ – множество значений параметров ЭМУ, $n=\overline{1,N}$, N – количество параметров ЭМУ, влияющих на его состояние; $W=\{w_n\}$, $n=\overline{1,N}$ – множество стартовых весовых коэффициентов НС; $Y=\{y_k\}$, $k=\overline{1,K}$, – множество контрольных состояний ЭМУ.

В процессе обучения НС на ее входы поступают входные сигналы, соответствующие элементам s_n, $n=\overline{1,N}$. Взвешенные весовыми коэффициентами соединения w_n, $n=\overline{1,N}$, входные сигналы суммируются, проходят через передаточную функцию, генерируют результат y, поступающий на выход НС, где он сравнивается с контрольным значением из множества Y. По результатам отклонения y от $y_k, k=\overline{1,K}$ формируется значение ошибки E, на минимизацию которой и направлено обучение НС с супервизором. Обучающая выборка должна быть сформирована таким образом, чтобы примеры охватывали всю плоскость исследований, а ошибка E стремилась к минимуму в каждом эксперименте.

Первым и одним из наиболее сложных этапов разработки системы оценки состояния ЭМУ с использованием НС является формирование обучающей выборки. На этом этапе из накопленных данных о работе ЭМУ формируется обучающая выборка так, чтобы максимально полно и равномерно были представлены все возможные состояния ЭМУ. От качества обучающей выборки зависит качество обучения. На сегодняшний день отсутствуют эффективные формальные критерии оценки качества обучающей выборки, на основании которых возможна разработка методик, ориентированных на оптимизацию этой выборки. Отсутствуют формальные, обоснованные с точки зрения особенности задачи и средств ее решения, методики повышения качества обучающей выборки. Поэтому было принято решение использовать нормированные значения параметров ЭМУ в закодированном виде, оценка которых произведена администратором предварительно.

При формировании обучающей выборки необходимо определить критерии подбора значений параметров ЭМУ. В равной степени должно быть уделено внимание всем возможным состояниям ЭМУ, а также должна прослеживаться зависимость состояния ЭМУ от значений параметров. Каждый ЭМУ имеет множество P_i параметров $p_{n,i}$, $n=\overline{1,N_i}$, $i=\overline{1,I}$, где N_i – количество параметров *i*-го ЭМУ, I – количество ЭМУ, оказывающих влияние на качество функционирования *i*-го ЭМУ. Множество P_i формируется на основании анализа функциональных и физических параметров, а также состояния других ЭМУ, влияющих на состояние оцениваемого *i*-го ЭМУ [2]. Значение каждого из параметров множества P_i, $i=\overline{1,I}$ нормируется и приводится к отрезку [0,1], причем ЭМУ, функционирующему в соответствии с заданным регламентом, соответствуют значения параметров, равные 1. После этого значение параметра кодируется и подается на вход НС.

Количество рабочих входов НС должно соответствовать количеству N_i параметров *i*-го ЭМУ, оценка состояния которого осуществля-

ется. Для этого при оценке состояния *i*-го ЭМУ активизируется количество входов НС, равное N_i.

Оценка состояния ЭМУ заканчивается появлением на выходе НС закодированного значения состояния $\overline{Y}$:

$$\overline{Y} = f\left(s_{n,i}, n = \overline{1,N}, i = \overline{1,I}\right),$$

где $s_{n,i}$, $n = \overline{1,N}, i = \overline{1,I}$ – значение параметра $p_{n,i}$.

Для кодирования состояния ЭМУ целесообразно использовать метод, предложенный в [1]. Суть кодирования заключается в следующем.

Для каждого *i*-го ЭМУ, $i = \overline{1,I}$ значения параметров $s_{n,i}$, $n = \overline{1,N}$ приводятся к численному виду, преобразовываются так, чтобы максимальному значению параметра $p_{n,i}$ $-\max s_{n,i}$ соответствовал максимальный положительный вклад в определение состояния *i*-го ЭМУ, и нормируются относительно $\max s_{n,i}$. При этом значения $s_{n,i}$ будут лежать в отрезке [0,1], т. е. $0 \le s_{n,i} \le 1$, для всех $n = \overline{1,N}$, $i = \overline{1,I}$.

Интервал изменения $s_{n,i}$, $n = \overline{1,N}$ разбивается на $M_{n,i}$ непересекающихся диапазонов $d_{n,i}^{(m_n)}$, $m_n = \overline{1,M_{n.i}}$ с использованием $L_{n,i} = (M_{n,i} - 1)$ пороговых значений $P_{n,i}^{(l_n)}$, $l_n = \overline{1,L_{n.i}}$. Нумерация диапазонов начинается от значения $\max s_{n,i}$, равного 1.

Принадлежность значения $s_{n,i}$ к диапазону $d_{n,i}^{(m_n)}$, $s_{n,i} \Rightarrow d_{n,i}^{(m_n)}$, определяется следующим образом. Если выполняется условие:

$$\frac{1}{(L_{n,i}+1)}(L_{n,i}+1-d_{n,i}^{(m_n)}) \le s_{n,i} \le$$

$$\le \frac{1}{(L_{n,i}+1)}(L_{n,i}+2-d_{n,i}^{(m_n)}),$$

состояние $s_{n,i}$ кодируется символом $A_{b_{m_n}}^{(M_{n,i})}$ так, что значение $b_{m_n}, m_n = \overline{1,M_{n.i}}$ соответствует номеру диапазона $d_{n,i}^{(m_n)}$, в котором находится значение $s_{n,i}$.

Символ $A_{b_{m_n}}^{(M_{n,i})}$ принадлежит алфавиту $A = \left\{A_{b_{m_n}}^{(M_{n,i})}, m_n = \overline{1,M_{n,i}}, n = \overline{1,N}\right\}$.

Кодовая комбинация для обозначения состояния S_i *i*-го ЭМУ, $i = \overline{1,I}$ будет выглядеть следующим образом:

$$A_{b_{m_1}}^{(M_{1,i})} A_{b_{m_2}}^{(M_{2,i})} \ldots A_{b_{m_N}}^{(M_{N,i})}.$$

Пример кодирования состояния ЭМУ для случая пятипорогового ($L_{2,i} = 5$) ранжирования состояний приведен в табл. 1.

Кодовая комбинация для обозначения состояния S_i *i*-го ЭМУ, $i = \overline{1,I}$ будет выглядеть, например, следующим образом: $A_3^{(5)} A_4^{(5)} A_4^{(5)} A_3^{(5)} A_4^{(5)}$.

Комбинация поступает на входы обученной нейронной сети. На ее выходе состояние элемента характеризуется одним символом алфавита $A = \left\{A_{b1}^{(M_{1,i})}, m_1 = \overline{1,M_{1,i}}\right\}$. Значения порогов устанавливаются администраторами и могут меняться в процессе работы. НС выдает код состояния, в котором находится ЭМУ в текущий момент времени. Схема метода поясняется рис. 1.

Табл. 1. Пример кодирования состояния ЭМУ при пятипороговом ранжировании

Код состояния *i*-го ЭМУ при *n*=5	Показатель производительности *i*-го элемента	Лингвистические переменные описания состояния элемента
$A_4^{(5)}$	$P_{3,i}^{(l_3)} \le s_{1,i} \le 1$	«Отлично»
$A_3^{(5)}$	$P_{2,i}^{(l_2)} \le s_{1,i} \le P_{3,i}^{(l_3)}$	«Хорошо»
$A_2^{(5)}$	$P_{1,i}^{(l_1)} \le s_{1,i} \le P_{2,i}^{(l_2)}$	«Удовлетворительно»
$A_1^{(5)}$	$P_{0,i}^{(l_0)} \le s_{1,i} \le P_{1,i}^{(l_1)}$	«Неудовлетворительно»
$A_0^{(5)}$	$0 \le s_{1,i} \le P_{0,i}^{(l_0)}$	«Критично»

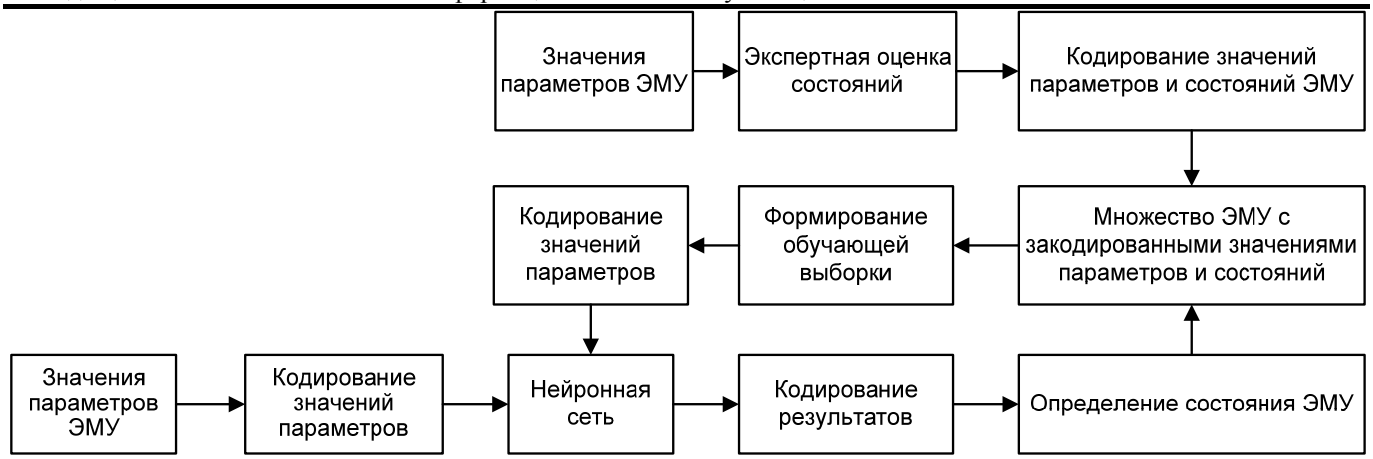


Рис. 1. Схема метода оценки состояния элементов ИТС

Результаты исследований

Обучение НС заканчивается построением гиперплоскостей, разделяющих область возможных состояний ЭМУ на непересекающиеся зоны, каждая из которых соответствует отдельному состоянию, определяемому, например, по табл. 1. Для оценки состояния элементов ИТС использовались нейронные сети трех типов: сеть Хопфилда (СХ), радиально базисная нейронная сеть (РБС) и машина опорных векторов (МОВ). Для каждой НС проводилось обучение с супервизором, при этом учитывались индивидуальные особенностей работы НС [5–8].

В качестве примера для проверки предлагаемого метода оценки состояния элементов ИТС с помощью нейронных сетей выбран ЭМУ типа «сервер», состояние которого оценивается по значениям таких параметров: 1) время ожидания доступа к носителям; 2) загрузка процессора/оперативной памяти; 3) свободная оперативная память: виртуальная память и память подкачки; 4) загруженность сетевых интерфейсов; 5) температура физических компонентов. По первым четырем параметрам оценивается текущая работоспособность сервера, а пятый позволяет выявить возможную причину неисправностей. Для первого исследования метода использовалась выборка по элементам ИТС типа «сервер» с параметрами 1)–4) (см. табл. 2), а для второго – с параметрами 1)–5) (см. табл. 3). Количество проводимых испытаний 100 и 1000.

Табл. 2. Количество ошибок, допущенных СХ, РБС и МОВ при четырех параметрах

Тип НС	Состояние, в котором находится ЭМУ тестовой выборки											
	$A_0^{(5)}$		$A_1^{(5)}$		$A_2^{(5)}$		$A_3^{(5)}$		$A_4^{(5)}$		*Случайное*	
	100	1000	100	1000	100	1000	100	1000	100	1000	100	1000
СХ	2	17	4	32	3	34	3	39	1	9	3	38
РБС	0	3	1	3	0	12	1	9	0	2	1	7
МОВ	0	0	0	2	1	2	0	2	0	0	0	2

Табл. 3. Количество ошибок, допущенных СХ, РБС и МОВ при пяти параметрах

Тип НС	Состояние, в котором находится ЭМУ тестовой выборки											
	$A_0^{(5)}$		$A_1^{(5)}$		$A_2^{(5)}$		$A_3^{(5)}$		$A_4^{(5)}$		*Случайное*	
	100	1000	100	1000	100	1000	100	1000	100	1000	100	1000
СХ	3	23	5	52	3	42	5	37	2	11	4	36
РБС	0	4	1	3	0	12	1	8	0	5	1	14
МОВ	0	1	0	4	1	3	0	4	0	0	0	3

Как видно из табл. 2 и 3, СХ показала худшие результаты распознавания состояния. При определении состояния ЭМУ, не являющемся предельным во множестве состояний: «критическое» – «плохое» – «удовлетворительное» – «хорошее» – «отличное», погрешность доходит до 5%.

РБС показала результаты на порядок лучше СХ. Это обусловлено тем, что на выходе НС выполняется операция математического округ-

ления чисел до ближайшего целого. Максимальная погрешность до операции округления составляла 0,4. Постоянная погрешность в 85% случаев находилась в диапазоне до 0,1, что является хорошим результатом.

МОВ в данной задаче представляет собой линейный пороговый классификатор, обучающийся по прецедентам, который для множества P_i, $i=\overline{1,I}$ параметров *i*-го ЭМУ, принимающих значения из множества S_i значений параметров, и множества контрольных значений $Y=\{y_k\}$ строит алгоритм $\alpha_i : s_i \to y_k$, $i=\overline{1,I}$, аппроксимирующий целевую зависимость на всем пространстве изменения значений S_i, $i=\overline{1,I}$ [9]. Точность построения гиперплоскости зависит от объема и качества обучающей выборки. Вероятность погрешности не превышает 0,01–0,02%.

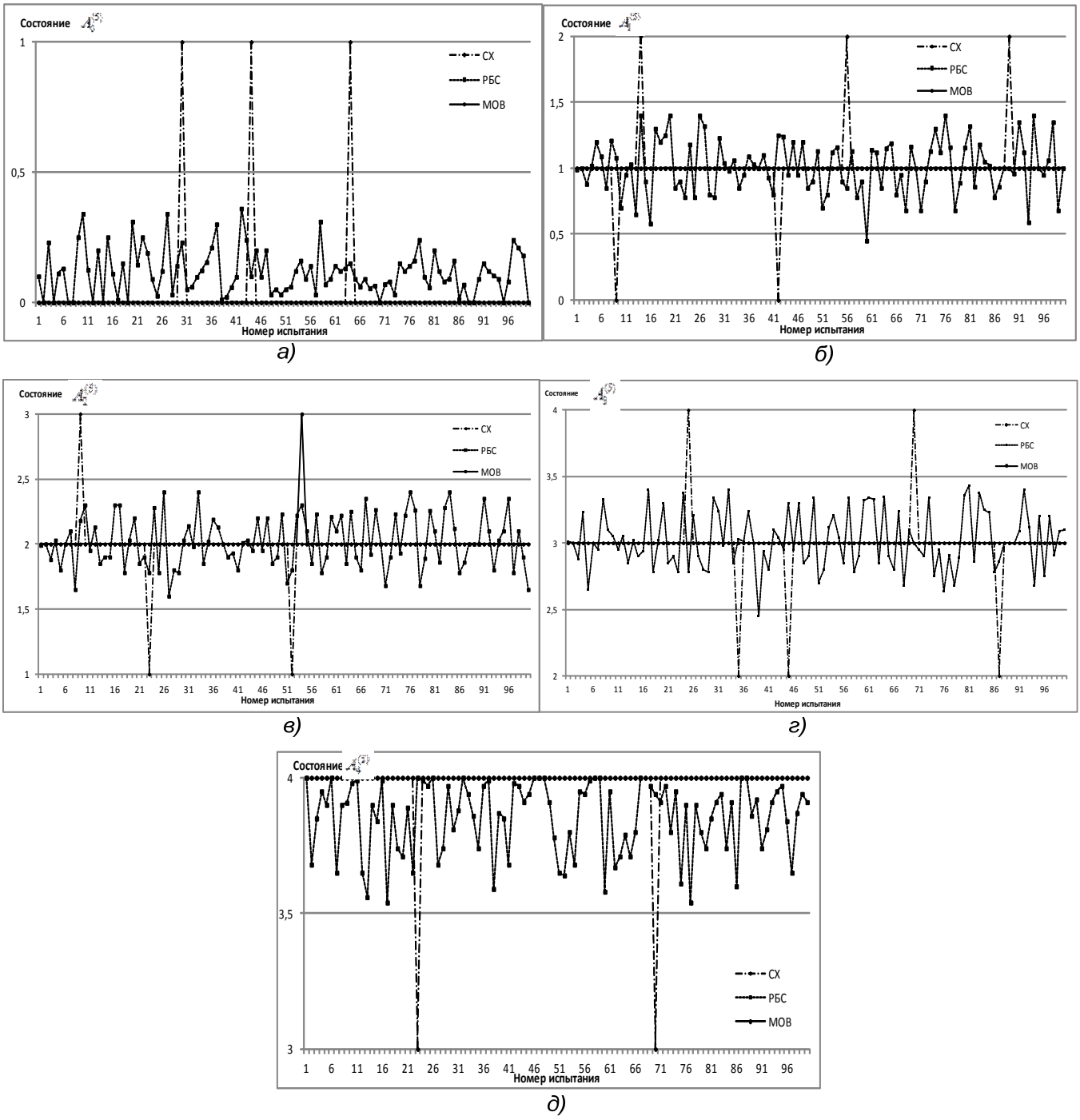


Рис. 2. Результаты тестирования СХ, РБС и МОВ при n=5 для: а) ЭМУ, находящихся в состоянии $A_0^{(5)}$; б) ЭМУ – в состоянии $A_1^{(5)}$; в) ЭМУ – в состоянии $A_2^{(5)}$; г) ЭМУ – в состоянии $A_3^{(5)}$; д) ЭМУ – в состоянии $A_4^{(5)}$

Из графиков (см. рис. 2) видно, что СХ допускает 2–5% ошибок при оценке состояния элементов ИТС, находящихся в приграничной области состояний.

Особенности работы РБС подразумевают написание подпрограммы, которая будет осуществлять операцию математического округления на выходе НС, поскольку в РБС выход является аналоговым. РБС допускает в среднем 1% ошибок.

Все допущенные нейронными сетями ошибки были связаны с недостаточной точностью построения разделяющей гиперплоскости. Как видно из результатов практических исследований, наиболее точно гиперплоскости определяются при помощи МОВ.

По показателям простоты и скорости обучения, быстродействия сети и точности результатов целесообразно использование машины опорных векторов в качестве основной аналитической составляющей подсистемы оценки состояния элементов ИТС.

При использовании МОВ для определения состояния ЭМУ решается задача определения принадлежности ЭМУ к одному из двух классов. В этом случае необходимо реализовать $M_{n,i}$ параллельных МОВ – линейных пороговых классификаторов, обучающихся по прецедентам. Каждая из параллельных МОВ строит собственную разделяющую гиперплоскость по уравнению $< W, S >= W_0$, где $W_0 = (w_{00}, w_{01}, ..., w_{0m})$, $m = \overline{1, M_{n,i}}$, – скалярный порог между классами. Главной задачей обучения является построение оптимальной разделяющей гиперплоскости с максимизацией ее ширины.

МОВ соотносит объект с определенным классом по следующей схеме [9]:

$$Y_k = \text{sign}(\sum_{i=1}^{N} w_i s_i - w_0) = \text{sign}(< W, S > - W_0).$$

Структура нейронной сети для оценки состояния ЭМУ с использованием МОВ приведена на рис. 3.

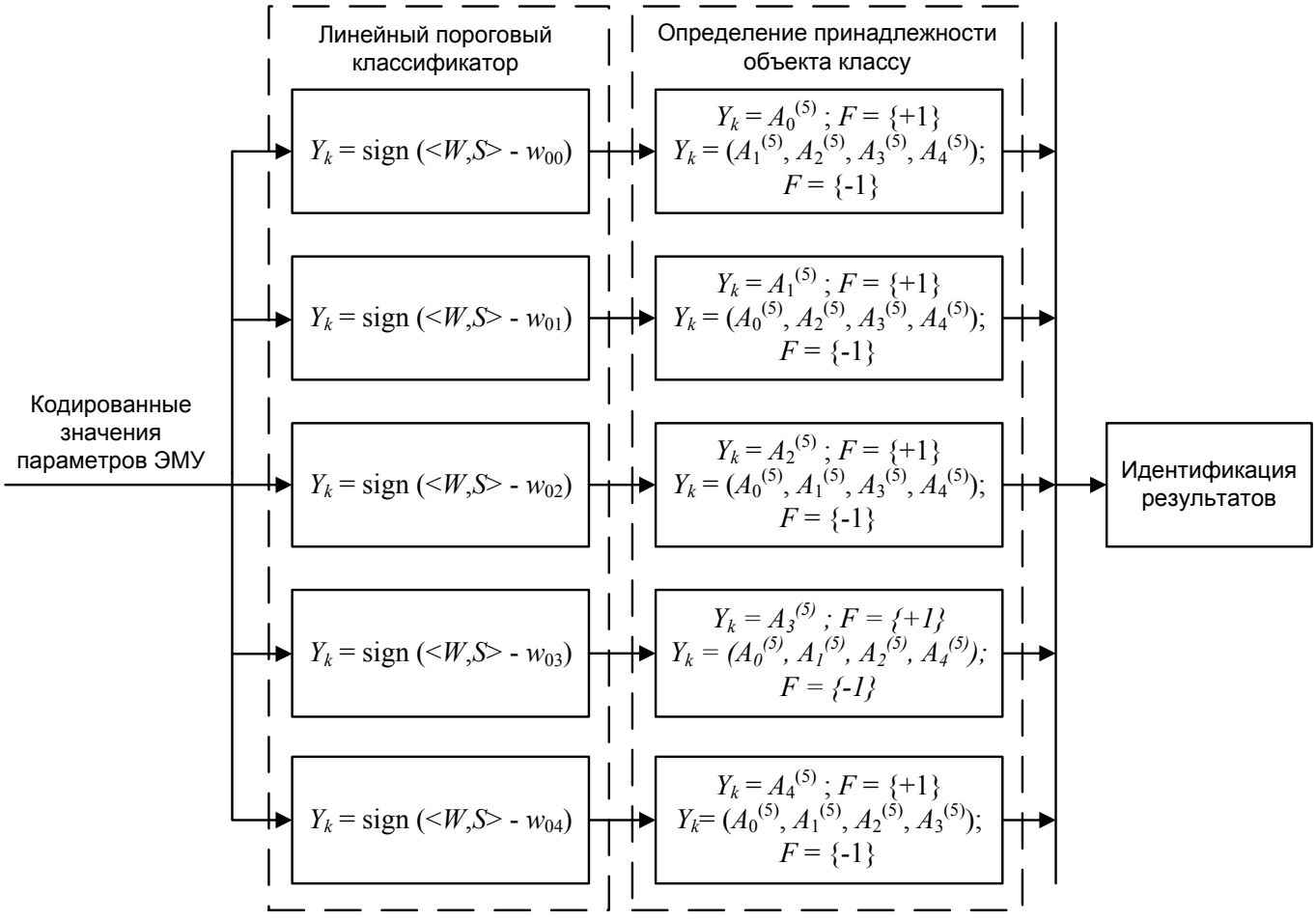


Рис. 3. Структура нейронной сети на основе машины опорных векторов

Нейронная сеть, построенная по методу опорных векторов и показавшая лучшие результаты при экспериментальных исследованиях, является мощным и перспективным направ-

лением обработки данных при решении задачи классификации состояний элементов ИТС. Это позволяет рекомендовать ее для применения в системах управления ИТС.

Выводы. Предложено для оценки состояния элементов ИТС использовать аппарат нейронных сетей. Приведена система кодирования состояний элементов ИТС при формировании обучающей и тестовой выборок. Для решения задачи определения состояний ЭМУ ИТС протестированы сеть Хопфилда, радиально-базисная нейронная сеть и нейронная сеть, построенная по методу опорных векторов. Приведена сравнительная характеристика результатов оценки состояния с помощью этих сетей. Разработан метод оценки состояния элементов ИТС с помощью машины опорных векторов. Алгоритм определения состояния элементов ИТС применен для оценки состояния элементов ИТС в СУИ SmartBase ITSControl, разрабатываемой в НТУУ «КПИ».

Список литературы

1. Теленик С.Ф. Методы диагностики компонентов информационно-телекоммуникационных систем/ С.Ф. Теленик, А.И. Ролик, Ю.С. Тимофеева // «Наукові вісті» Ін-ту менеджменту і економіки «Галицька академія». – Івано-Франківськ, 2009. – № 1 (15). – С. 49–58.
2. Ролик А.И., Глушко Е.В. Анализ качества функционирования элементов информационно-телекоммуникационных систем// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка.– К.: – 2008. – № 48. – С. 113–120.
3. Ролік О.І. Метод зведення метрик якості функціонування компонентів ІТ-інфраструктури за допомогою апарату непараметричної статистики / О.І. Ролік, П.Ф. Можаровский, В.М. Вовк, Д.С. Захаров // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2011. – № 53. – С. 160–169.
4. Ролік О.І. Застосування агентського підходу до управління інформаційно-телекомунікаційною системою АСУ спеціального призначення / О.І. Ролік, П.Ф. Можаровський, О.О. Покотило // Пріоритетні напрямки розвитку телекомунікаційних систем та мереж спеціального призначення: V наук.-практ. семін., 22 жовт. 2009 р.: доповіді та тези доповідей: – К.: ВІТІ НТУУ «КПІ», 2009. – С. 228–229.
5. Boser B.E. A training algorithm for optimal margin classifiers/ B.E. Boser, I.M. Guyon, V.N. Vapnic // ACM. – 1992. – С. 1–7.
6. Круглов В.В. Искусственные нейронные сети. Теория и практика. 2-е изд. // В.В. Круглов, В.В. Борисов // ГЛ Телеком. – 2002. – С. 63–65.
7. Хайкин С. Нейронные сети. Полный курс. Второе издание // С. Хайкин. – Вильямс. – 2006. – С. 107, 371–373, 396–398, 419–425.
8. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы // Д. Рутковская, М. Пилиньский, Л. Рутковский . – ГЛ Телеком. – 2006. – С. 37–43.
9. Воронцов К.В. Лекции по методу опорных векторов // К.В. Воронцов // 2007. – С. 2–17.

УДК 519.854.2

ПАВЛОВ А.А.,
МИСЮРА Е.Б.,
СПЕРКАЧ М.О.

ИССЛЕДОВАНИЕ СВОЙСТВ ЗАДАЧИ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ ВЫПОЛНЕНИЯ ЗАДАНИЙ С ОБЩИМ ДИРЕКТИВНЫМ СРОКОМ ПАРАЛЛЕЛЬНЫМИ ПРИБОРАМИ ПО РАЗНЫМ КРИТЕРИЯМ ОПТИМАЛЬНОСТИ

Рассмотрены свойства задачи календарного планирования выполнения заданий параллельными приборами равной производительности по трем критериям оптимальности с общим директивным сроком. Показана взаимосвязь между всеми критериями оптимальности. Приведен алгоритм решения задачи нахождения минимума максимального момента завершения выполнения заданий, позволяющего получить расписание, в котором все задания выполняются без запаздывания.

The properties of the calendar scheduling task of the assignments' fulfillment are considered using parallel machines of equal performance on three criteria of optimality with common prescriptive timeline. The relationship between all the criteria of optimality is shown. An algorithm for solving the problem of finding the minimum of the maximum completion time of the assignments' fulfillment is given. It provides a schedule with all tasks carried out without delay.

Введение

Рассмотрены свойства задачи календарного планирования выполнения заданий параллельными приборами равной производительности по трем критериям оптимальности с общим директивным сроком. Показана взаимосвязь между всеми критериями оптимальности. Приведен алгоритм решения задачи нахождения минимума максимального момента завершения выполнения заданий, позволяющего получить расписание, в котором все задания выполняются без запаздывания.

Общая постановка задачи

Рассмотрим свойства и проведем сравнительный анализ следующей задачи календарного планирования по разным критериям оптимальности.

Задано множество заданий J, число приборов m, для каждого задания $j \in J$ известна длительность выполнения p_j. Все задания имеют общий директивный срок d. Предполагается, что все задания множества J поступают одновременно, процесс обслуживания каждого задания можно начать в любой момент времени, он будет протекать без прерываний до завершения обслуживания задания. Все приборы работают без прерываний.

Необходимо построить расписание σ выполнения заданий $j \in J$ на m приборах одинаковой производительности, по следующим критериям оптимальности:

1) минимизация суммарного запаздывания выполнения заданий:

$$F(\sigma) = \sum_{j \in J} \max\left[0; C_j(\sigma) - d\right],$$

где $C_j(\sigma)$ – момент завершения выполнения задания j в расписании σ;

2) найти максимальный момент запуска заданий на выполнение r, позволяющий получить допустимое расписание (расписание, в котором все задания выполняются без запаздывания);

3) построение допустимого расписания с минимальным суммарным опережением времени завершения выполнения заданий, назначенных на каждый прибор, относительно максимального момента завершения заданий $C_{\max}$:

$$E_\Sigma = \sum_{k=1}^{m} E_k = \sum_{k=1}^{m} \left(C_{\max} - C_k\right) \to \min,$$

где C_k – момент завершения выполнения заданий на приборе k,

$C_{\max}$– максимальный из всех C_k, $k = \overline{1,m}$.

В [1] приведены два ПДС-алгоритма решения задачи по критерию 1, которые применяются в зависимости от параметров исходных данных. ПДС-алгоритмы решения задачи включают полиномиальную составляющую и приближенный алгоритм и строятся только на направленных перестановках. Полиномиальная составляющая алгоритма задается детерминированной процедурой последовательного выполнения направленных перестановок, общее количество которых ограничено полиномом от числа заданий и количества приборов. Полином

задается исследователем и зависит от допустимого времени вычисления задачи и мощности вычислительного комплекса.

На основании теоретических свойств задачи в [1] определены условия, выполнение любого из которых на допустимом расписании гарантирует его оптимальность. В результате решения задачи получаем либо строго оптимальное решение полиномиальной составляющей алгоритма (если в процессе вычислений удовлетворялось одно из условий оптимальности допустимого расписания), либо приближенное с верхней оценкой отклонения от оптимального.

Утверждение 1. Решение задачи по критерию 2 эквивалентно решению следующей задачи: для произвольного момента запуска приборов получить расписание, у которого максимальное время завершения выполнения работ $C_{\max}$ является минимальным.

Доказательство. Пусть существует расписание с произвольным моментом запуска приборов r. Пусть T_k – сумма длительностей заданий на приборе k :

$$T_k = \sum_{j=1}^{n_k} p_{kj} ,$$

где n_k – количество работ, выполняемых на приборе k;

p_{kj} – длительность выполнения j-го задания на приборе k.

Обозначим сумму длительностей на самом загруженном приборе $T_{\max} = \max_k T_k, k = \overline{1,m}$. Оптимальным по критерию минимизации $C_{\max}$ будет расписание, удовлетворяющее условию: $T_{\max} \to \min$. Обозначим сумму длительностей на самом загруженном приборе в оптимальном расписании через $T^*_{\max}$, а максимальный момент завершения заданий в оптимальном расписании через $C^*_{\max} = r + T^*_{\max}$, где r – момент запуска приборов. Определим максимальный момент запуска приборов $r = d - T^*_{\max}$. Поскольку $d - T^*_{\max}$ может быть отрицательным числом, то ноль выбирается так, чтобы $d - T^*_{\max}$ было неотрицательным. Максимальный момент запуска приборов в этом случае равен $r = d - T^*_{\max} \geq 0$. Это расписание допустимо и является оптимальным по критерию 2 по построению. Действительно, для любого $r > d - T^*_{\max}$ не существует допустимого расписания.

Следствие. На равномерном расписании достигается абсолютный оптимум функционала по критерию 2.

Теорема 1. Оптимальные решения задач по критериям 2 и 3 совпадают.

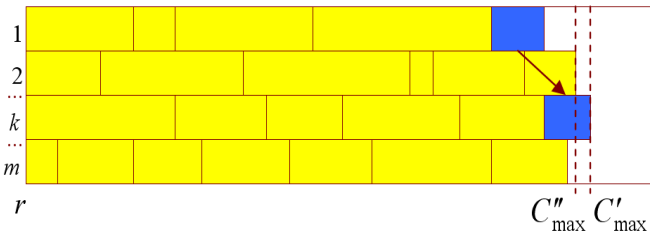


Рис. 1. Гистограмма выполнения заданий

Доказательство. Из двух допустимых расписаний то, у которого максимальный момент окончания выполнения заданий $C_{\max}$ больше ($C'_{\max} > C''_{\max}$, рис. 1: последнее задание перенесено с прибора 1 на прибор k), имеет большее суммарное опережение относительно $C_{\max}$. Действительно, для допустимого расписания E_Σ определяется площадью прямоугольника, ограниченного моментом запуска и $C_{\max}$, минус суммарная площадь загрузки приборов (прямоугольники с высотой 1). Так как суммарное время работы приборов для обоих расписаний одинаково, то чем больше $C_{\max}$, тем больше E_Σ. В оптимальном по критерию 2 расписании, в соответствии с утверждением 1, $C_{\max}$ минимально и, следовательно, E_Σ минимально. Итак, допустимое расписание с максимальным моментом запуска заданий и допустимое расписание с минимальным E_Σ совпадают.

Таким образом, для нахождения минимального $C_{\max}$ предлагается следующий приближенный алгоритм.

1) Определяем $d = \sum_{j=1}^{n} p_j \Big/ m$. Решаем для этого директивного срока задачу по критерию 1 при моменте запуска r = 0 (получаем расписание σ_0).

2) Задаем некоторый шаг Δ. Получаем расписания по критерию 1 для директивных сроков $d \pm \Delta$, $d \pm 2\Delta$, $d \pm k\Delta$. расписания $\sigma_1, \sigma_2, \ldots, \sigma_{2k}$), где $k\Delta$ существенно меньше d.

3) Выбираем из расписаний $\sigma_0, \sigma_1, \sigma_2, \ldots \sigma_{2k}$ то расписание, для которого $C_{\max}$ является наименьшим. Если в этом расписании $C_{\max}$ мало отличается от $C_{\min}$ (минимального значения момента освобождения прибора), то расписание близко к равномерному, и статистически значимо полу-

чено оптимальное расписание для задачи по критерию 2. Если разница $C_{max} - C_{min}$ существенна, то очевидным образом простыми перестановками мы приходим к новому расписанию, в котором увеличивается суммарное запаздывание, но при этом уменьшается максимальная загрузка приборов C_{max}. Новому расписанию, в соответствии с показанным выше, соответствует меньшее, чем у предыдущего, суммарное опережение относительно C_{max}.

4) Момент запуска приборов $r = d - T_{max}$.

Примечание: решение $2k$ задач для $d \pm \Delta$, $d \pm 2\Delta$, … вызвано тем, что функционал минимизации C_{max} и минимальное суммарное запаздывание (критерий 1) хотя и коррелированы, но не совпадают. Теоретически минимальному суммарному запаздыванию может соответствовать очень большая максимальная загрузка приборов. В этом случае решением $2k$ задач находим директивный срок, при котором значению функционала по критерию 1 соответствует минимальный T_{max}. В противном случае, путем перестановок уменьшаем T_{max} и получаем расписание, в котором разность $C_{max} - C_{min}$ минимальна.

Вычислительные эксперименты

Предложенный алгоритм нахождения минимального C_{max} основан на алгоритме решения задачи по критерию 1, и его трудоемкость определяется трудоемкостью алгоритма решения задачи по критерию 1. Для задачи по критерию 1 в [1] были проведены испытания на задачах с размерностью до 40 000 заданий с числом приборов до 30-ти. Вычислительные эксперименты показали, что с увеличением размерности числа заданий увеличивается статистическая вероятность реализации полиномиальной составляющей алгоритма. Были предложены два ПДС-алгоритма: А1 с трудоемкостью $O(n^2m)$ и А2 с трудоемкостью $O(mn \log n)$. Алгоритм А2 построен на основе анализа эффективности алгоритма А1 и выделения наиболее эффективных перестановок с точки зрения реализации условий оптимальности. Среднее время решения задачи полиномиальной вычислительной схемой А1 не превысило 2,4 с для размерности n до 40 000 заданий и m до 30 приборов (исследования проводились на процессоре Pentium Core 2 Duo с тактовой частотой 3,0 ГГц). Среднее время решения задач той же размерности полиномиальной вычислительной схемой А2 не превысило 35 мс. Средняя частота получения оптимального решения для алгоритма А1 составила 89,2 %, для алгоритма А2 – 74,1 %, причем с возрастанием числа заданий увеличивается статистическая вероятность реализации полиномиальной составляющей алгоритмов. Среднее отклонение от оптимального решения для алгоритма А1 составляло величину 0,000283, для алгоритма А2 – 0,00016. При введении дополнительных типов перестановок алгоритм А2 эффективнее алгоритма А1. При этом оптимальное значение функционала достигается полиномиальной составляющей алгоритма в 92 % случаев.

Выводы

Показано, что:

– решение задачи по критерию 2 эквивалентно решению задачи минимизации C_{max} для произвольного момента запуска приборов;

– на равномерном расписании достигается абсолютный оптимум функционала по критерию 2;

– решения задач по критериям 2 и 3 совпадают.

Приведен алгоритм решения задачи нахождения минимального C_{max}, основанный на алгоритме решения задачи по критерию 1. Вычислительные эксперименты показали высокую эффективность алгоритма.

Список литературы

1. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.

УДК 683.513

ПАВЛОВ А.В.

МНОГОЭТАПНЫЙ АЛГОРИТМ МГУА С ОРТОГОНАЛИЗАЦИЕЙ ПЕРЕМЕННЫХ И ЕГО РЕКУРРЕНТНЫЙ МЕТОД РАССЧЁТА КОЭФФИЦИЕНТОВ И КРИТЕРИЯ СЕЛЕКЦИИ МОДЕЛЕЙ

Найдены условия, при которых обобщённый релаксационный итерационный алгоритм моделирования находит точное решение и показано, что при выполнении этих условий он принадлежит классу переборных алгоритмов МГУА с направленным перебором. Предложен многоэтапный алгоритм с ортогонализацией переменных и рекуррентный метод оценивания параметров и расчёта критерия селекции. Сложность расчёта одной модели на каждом этапе алгоритма является постоянной, в отличие квадратичной в известном алгоритме за счет применения ортогонализации переменных и рекуррентных вычислений.

Conditions under which generalized relaxational iterative algorithm finds exact solution have been found; belonging the algorithm to class of multistage GMDH algorithms has been proved in this case. Multistage algorithm with features orthogonalization and recurrent method for models' parameters and selection criterion estimation has been suggested. Model calculation complexity at each algorithm's stage is constant in contrast to quadratic one for well-known algorithm due to features orthogonalization and recurrent computations.

1. Введение

Прикладные задачи принятия решений часто решаются на основе регрессионных моделей (далее просто моделей) объектов и процессов, построенных по эксперименталь-ным данным. Такими задачами, например, являются прогнозирование, экстраполирова-ние, классификация. Одним из эффективных методов построения моделей является метод группового учета аргументов (МГУА). Различают итерационные и переборные алгоритмы МГУА.

Переборные алгоритмы в зависимости от генератора структур могут находить модель глобального минимума внешнего критерия, либо нет – как в случае направленного (ограниченного) поиска модели способом вложенных структур. Поэтому в настоящее время переборные алгоритмы делятся на комбинаторные, реализующие полный пере-бор структур моделей и, те которые выполняют направленный перебор, называе-мые далее многоэтапными. Если комбинатор-ные алгоритмы, которые всегда находят модель, соответствующую глобальному оптимуму выбранного критерия, имеют «проклятие размерности» по количеству переменных, то многоэтапные и итерацион-ные – могут обрабатывать значительно большее количество аргументов (до 1000), но находят при этом решение, соответствующее локальному минимуму внешнего критерия. Отличие многоэтапных алгоритмов от итерационных заключатся в том, что многоэтапные находят структуру и параметры модели (точное решение) за конечное число этапов.

Среди итерационных алгоритмов выделяется класс релаксационных итерационных алгоритмов, основной особенностью которого является способ усложнения структуры модели от итерации к итерации: ошибка между вектором выхода модели (решением) текущей итерации и исходным вектором выходной переменной уменьшается за счёт явного добавления в модель исходных аргументов на каждой итерации.

Метод усложнения структуры модели, в случае добавления в неё на каждом этапе (или итерации) одного отсутствующего из множества исходных аргументов назван мето-дом вложенных структур. Между моделями, полученными релаксационным и много-этапным и алгоритмами, использующие метод вложенных структур, при условии их (структур) совпадения, имеется различие в величинах оценок параметров. Итеративно рассчитываемые оценки в итерационных алгоритмах, в общем случае не совпадают с оценками для той же структуры в многоэтапных алгоритмах, которые вычис-ляются по методу наименьших квадратов (МНК). Следовательно, в отличие от многоэтапных алгоритмов, итерационные не дают точного решения.

Поэтому актуальной задачей является поиск условий, при которых релаксационные итерационные алгоритмы также находят точное решение. Работа посвящена поиску получения точных решений для обобщённого ре-

лаксационного итерационного алгоритма (ОРИА), идентичных многоэтапному алгоритму. Дадим общую для обоих алгоритмов МГУА постановку задачи.

2. Постановка задачи МГУА

Пусть задана матрица данных $\mathbf{X}$, $\dim \mathbf{X} = n_W \times m$, где m – число признаков (аргументов) x_i, $i = \overline{1,m}$, n_W – число наблюдений, и выходной вектор $\mathbf{y}$, $\dim \mathbf{y} = n_W \times 1$. Необходимо найти оптималь-ную по минимуму заданного критерия CR модель $\hat{\mathbf{y}}_{\mathbf{d}_k^*} = \mathbf{f}(\mathbf{Z}_{\mathbf{d}_k^*}, \hat{\Theta}_{\mathbf{d}_k^*})$:

$$\mathbf{f}(\mathbf{Z}_{\mathbf{d}_k^*}, \hat{\Theta}_{\mathbf{d}_k^*}) = \arg\min_{\mathbf{d}_k \in D} CR(\mathbf{y}, \mathbf{f}(\mathbf{Z}_{\mathbf{d}_k}, \hat{\Theta}_{\mathbf{d}_k})), \qquad (1)$$

где оценки параметров $\Theta_{\mathbf{d}_k}$, $\forall\ \mathbf{d}_\mathrm{k} \in D$ являются решением задачи:

$$\hat{\Theta}_{\mathbf{d}_k} = \arg\min_{\Theta_{\mathbf{d}_k} \in \Re^{s_k}} QR(\mathbf{y}, \mathbf{f}(\mathbf{Z}_{\mathbf{d}_k}, \Theta_{\mathbf{d}_k})), \qquad (2)$$

$\mathbf{d}_k$ – бинарный вектор, который определяет подмножество аргументов модели; D – множество всех возможных бинарных векторов размерности l; $\Theta_{\mathbf{d}_k}$ – вектор размерности s_k, составленный из ненулевых компонентов вектора $diag(\mathbf{d}_k)\cdot\Theta$; $\mathbf{Z}_{\mathbf{d}_k}$ – матрица, полученная из вектор-столбцов $\mathbf{z}_j$ матрицы $\mathbf{Z}$, где элементы вектора $\mathbf{d}_{k,j} = 1$, а остальные компоненты вектора $\mathbf{d}_k$ – нулевые; $\mathbf{Z}$ – матрица, образованная из вектор-столбцов, соответствующих переменным в полиномиальной модели вида:

$$f(x_1,...,x_m,\Theta) = \theta_0 + \sum_{i=1}^{m}\theta_i x_i + \sum_{i=1}^{m}\sum_{j=i}^{m}\theta_{i,j}x_i x_j + \\ + \sum_{i=1}^{m}\sum_{j=i}^{m}\sum_{k=i+j}^{m}\theta_{i,j,k}x_i x_j x_k + ...,$$

где θ_i, $\theta_{i,j}$, $\theta_{i,j,k}$,… являются коэффициентами при этих переменных; p – заданная максимальная степень полинома; $\dim \mathbf{Z} = n_w \times l$, $l = C_{m+p}^{p} - 1$. Пусть $QR(\cdot)$ – критерий качества решения задачи оценивания параметров, $CR(\cdot)$ – критерий качества решения задачи определения оптимальной модели.

В качестве критерия $CR(\cdot)$ в МГУА используют «внешние» критерии, основанные на разбиении выборки $\mathbf{W} = (\mathbf{X}\vdots\mathbf{y})$, например, критерий регулярности:

$$AR_B \overset{\Delta}{=} AR_{B/A} = \| \mathbf{y}_B - \mathbf{Z}_{B,\mathbf{d}_k}\hat{\Theta}_{A,\mathbf{d}_k} \|^2,$$

где запись $AR_{B/A}$ указывает на то, что на выборке A при заданной структуре находятся оценки вектора параметров $\Theta_{A,\mathbf{d}_k}$, а на B рассчитывается ошибка модели как значение критерия. Выборка A называется обучающей, а B – проверочной, $W = A \cup B$, $A \cap B = \varnothing$. Задача (2) решается по МНК. $QR(\cdot)$ – критерий остаточной суммы наименьших квадратов (residual sum square (RSS)).

Перейдём к анализу ОРИА описанному в [1].

3. Анализ обобщённого релаксационного алгоритма МГУА

Алгоритм строит модель итерационным способом, уменьшая невязку между решением r-й итерации $\hat{\mathbf{y}}_{A,r}$ и выходом $\mathbf{y}_A$, при добавлении одного из вектор-столбцов $\mathbf{x}_{A,r+1}$ матрицы $\mathbf{X}_A$. Задача (2), решаемая в данном алгоритме на (r+1)-й итерации, в матричной форме выглядит следующим образом:

$$\mathbf{y}_A = \omega_{A,r+1}^{r+1}\mathbf{x}_{A,r+1} + \omega_{A,r+1}\hat{\mathbf{y}}_{A,r}, \qquad (3)$$

где $\omega_{A,r+1}^{r+1}$, $\omega_{A,r+1}$ – неизвестные параметры, оценки которых определяются по МНК; $\mathbf{x}_{A,r+1}$ – вектор-столбец матрицы $\mathbf{X}_A$, добавляемы в модель на (r+1)-й итерации; $\hat{\mathbf{y}}_{A,r}$ – решение, полученное на предыдущей итерации.

Матрица нормальных уравнений для (3) имеет следующий вид:

$$\begin{pmatrix} \hat{\mathbf{y}}_{A,r}^T\hat{\mathbf{y}}_{A,r} & \hat{\mathbf{y}}_{A,r}^T\mathbf{x}_{A,r+1} \\ \mathbf{x}_{A,r+1}^T\hat{\mathbf{y}}_{A,r} & \mathbf{x}_{A,r+1}^T\mathbf{x}_{A,r+1} \end{pmatrix} \begin{pmatrix} \omega_{A,r+1}^{r+1} \\ \omega_{A,r+1} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{y}}_{A,r}^T\mathbf{y}_A \\ \mathbf{x}_{A,r+1}^T\mathbf{y}_A \end{pmatrix}.$$

Оценки коэффициентов $\hat{\omega}_{A,r+1}^{r+1}$, $\hat{\omega}_{A,r+1}$ рассчитываются по формулам:

$$\hat{\omega}_{A,r+1} = \frac{c_{A,r}(\mathbf{x}_{A,r+1}^T\mathbf{x}_{A,r+1}) - b_{A,r}(\mathbf{x}_{A,r+1}^T\mathbf{y}_A)}{a_{A,r}(\mathbf{x}_{A,r+1}^T\mathbf{x}_{A,r+1}) - b_{A,r}^2}, \qquad (4)$$

$$\hat{\omega}_{A,r+1}^{r+1} = \frac{a_{A,r}(\mathbf{x}_{A,r+1}^T\mathbf{y}_A) - b_{A,r}c_{A,r}}{a_{A,r}(\mathbf{x}_{A,r+1}^T\mathbf{x}_{A,r+1}) - b_{A,r}^2}, \qquad (5)$$

где $a_{A,r} = \hat{\mathbf{y}}_{A,r}^T\hat{\mathbf{y}}_{A,r}$, $b_{A,r} = \hat{\mathbf{y}}_{A,r}^T\mathbf{x}_{A,r+1}$, $c_{A,r} = \hat{\mathbf{y}}_{A,r}^T\mathbf{y}_A$ рассчитаны на r-й итерации.

Решение (r+1)-й итерации:

$$\hat{\mathbf{y}}_{A,r+1} = \hat{\omega}_{A,r+1}^{r+1}\mathbf{x}_{A,r+1} + \hat{\omega}_{A,r+1}\hat{\mathbf{y}}_{A,r}.$$

Критерий RSS_A рассчитывается по

$$RSS_A = \mathbf{y}_A^T\mathbf{y}_A - 2c_{A,r+1} + a_{A,r+1}, \qquad (6)$$

а величины $a_{A,r+1}$ и $c_{A,r+1}$ рекуррентно:

$$a_{A,r+1} = a_{A,r}(\hat{\omega}_{A,r+1})^2 + 2\hat{\omega}_{A,r+1}\hat{\omega}^{r+1}_{A,r+1}b_{A,r} + \\ + (\hat{\omega}^{r+1}_{A,r+1})^2 \mathbf{x}^T_{A,r+1}\mathbf{x}_{A,r+1}, \quad (7)$$

$$c_{A,r+1} = c_{A,r}\hat{\omega}_{A,r+1} + \mathbf{x}^T_{A,r+1}\mathbf{y}_A\hat{\omega}^{r+1}_{A,r+1}, \quad (8)$$

$$b_{A,r} = \hat{\mathbf{y}}^T_{A,r}\mathbf{x}_{A,r+1} = \mathbf{x}_{A,r+1}\mathbf{G}_{A,r}(\hat{\Omega}^r_{A,r})^T = \\ = \sum_{j=1}^{r}(\mathbf{x}^T_{A,r+1}\mathbf{g}_{A,j})\cdot\hat{\omega}^j_{A,r},$$

где $\mathbf{G}_{A,r}$ – матрица, состоящая из вектор-столбцов матрицы $\mathbf{X}_A$, вошедших в модель на первых r итерациях; $\hat{\Omega}^r_{A,r}$ – вектор коэффициентов, элементы которого соответствуют вектор-столбцам матрицы $\mathbf{G}_{A,r}$. Верхний индекс $\hat{\Omega}^r_{A,r}$ указывает размерность вектора, а нижний – номер итерации.

После расчёта оценок коэффициентов $\hat{\omega}^{r+1}_{A,r+1}$, $\hat{\omega}_{A,r+1}$ корректируются коэффициенты предыдущих r итераций при вектор-столбцах матрицы $\mathbf{G}_{A,r}$:

$$\hat{\Omega}^r_{A,r+1} = \hat{\Omega}^r_{A,r}\hat{\omega}_{A,r+1}.$$

Критерий регулярности в ОРИА рассчитывается по формуле:

$$AR_{B,r+1} = \mathbf{y}^T_B\mathbf{y}_B - 2c_{B,r+1} + a_{B,r+1}, \quad (9)$$

где величины $c_{B,r+1}$ и $a_{B,r+1}$ рассчитываются по формулам (7), (8) на выборке B.

Скорость сходимости ОРИА существенно зависит от степени ортогональности вектор-столбцов матрицы $\mathbf{X}_A$, а именно: чем более ортогональной является система, тем быстрей скорость сходимости алгоритма [2] . Докажем, что: максимальная скорость сходимости ОРИА к точному решению достигается при ортогональной системе вектор-столбцов матрицы $\mathbf{X}_A$; решение, к которому сходится алгоритм совпадает с истинной моделью (точным решением).

4. Доказательство сходимости алгоритма к точному решению

Утверждение 1. Минимальное количество итераций ОРИА, необходимое для внутренней сходимости алгоритма к точному решению

$$\mathbf{y}_A = \breve{f}(\mathbf{X}_A, \breve{\Theta}) = \breve{\theta}_0 + \sum_{i=1}^{m}\mathbf{x}_{A,i}\breve{\theta}_i$$

по критерию RSS_A равно m и это решение достигается при условии, что вектор-столбцы матрицы $\mathbf{X}_A$ являются ортогональной системой.

Доказательство. Пусть $\mathbf{y}_A = \mathbf{X}_A\Theta_A$, $m = 3$ и вектор-столбцы матрицы $\mathbf{X}_A$ взаимно ортогональны (рис. 1).

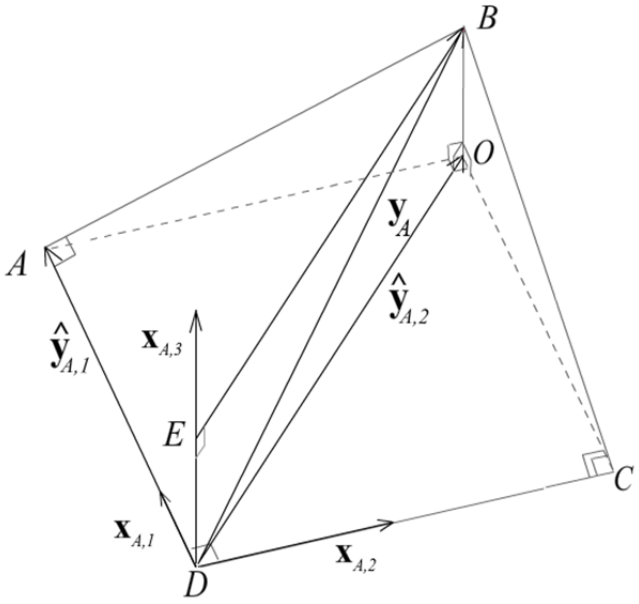


Рис. 1. Процесс построения модели алгоритмом при взаимно ортогональных вектор-столбцах матрицы $\mathbf{X}_A$, $m = 3$

В данном случае скалярное произведение $\hat{\mathbf{y}}^T_{A,2}\mathbf{x}_{A,3} = 0$, поскольку вектор $\hat{\mathbf{y}}_{A,2}$ лежит в плоскости векторов $\mathbf{x}_{A,1}$ и $\mathbf{x}_{A,2}$. При этом невязка $|\mathbf{y}_A - \mathbf{G}_{A,3}\hat{\Omega}^3_{A,3}| = |\mathbf{y}_A - \mathbf{X}_{A,3}\hat{\Omega}_{A,3}| = 0$. Рассмотрим общий случай, когда $\mathbf{y}_A = \mathbf{X}_A\Theta_A$. Пусть алгоритм выполнил r итераций, тогда на $(r+1)$-й итерации $\hat{\mathbf{y}}^T_{A,r}\mathbf{x}_{A,r+1} = 0$ и невязка $|\mathbf{y}_A - \mathbf{G}_{A,r+1}\hat{\Omega}^{r+1}_{A,r+1}| = |\mathbf{y}_A - \mathbf{X}_{A,r+1}\hat{\Omega}_{A,r+1}|$ является минимальным расстоянием между вектором выхода $\mathbf{y}_A$ и вектором решения $\hat{\mathbf{y}}_{A,r+1} = \sum_{i=1}^{r+1}\hat{\theta}_{A,i}\mathbf{x}_{A,i}$, которое может быть получено в плоскости, определяемой первыми $r+1$ вектор-столбцами матрицы $\mathbf{X}_A$. Если продолжить описанный процесс до $r = m$, алгоритм сойдётся к вектору $\mathbf{y}_A$. Причём полученные оценки коэффициентов $\hat{\theta}_{A,i}$, $i = \overline{1,m}$ совпадают с оценками МНК: $\hat{\Theta}_A = \left(\mathbf{X}^T_A\mathbf{X}_A\right)^{-1}\mathbf{X}^T_A\mathbf{y}_A$.

Опишем метод оценивания коэффициентов и расчёта критерия селекции модели для ОРИА при условии взаимно-ортогональных вектор-столбцов входной матрицы $\mathbf{X}^T = (\mathbf{X}^T_A : \mathbf{X}^T_B)$.

5. Рекуррентный метод оценивания параметров и расчёта критериев селекции

Пусть системы вектор-столбцов матриц $\mathbf{X}_A$ и $\mathbf{X}_B$ является ортогональными, тогда $\hat{\mathbf{y}}_{A,r}^T \mathbf{x}_{A,r+1} = 0$, следовательно $b_{A,r} = 0$. Поскольку треугольник ΔDBO – прямоугольный (рис. 1):

$$\mathbf{y}_A^T \mathbf{y}_A = \hat{\mathbf{y}}_{A,r}^T \hat{\mathbf{y}}_{A,r} + (\mathbf{y}_A - \hat{\mathbf{y}}_{A,r})^T (\mathbf{y}_A - \hat{\mathbf{y}}_{A,r}), \qquad (10)$$

Из выполнения равенства (10) следует, что $\hat{\mathbf{y}}_{A,r}^T \hat{\mathbf{y}}_{A,r} = \hat{\mathbf{y}}_{A,r}^T \mathbf{y}_A$, т.е. $a_{A,r} = c_{A,r}$. Тогда формулы (4) и (5) примут вид:

$$\hat{\omega}_{A,r+1} = 1, , \qquad (11)$$

$$\hat{\omega}_{A,r+1}^{r+1} = \frac{(\mathbf{x}_{A,r+1}^T \mathbf{y}_A)}{(\mathbf{x}_{A,r+1}^T \mathbf{x}_{A,r+1})}., \qquad (12)$$

С учётом $b_{A,r} = 0$ и (11) формулы (7) и (8) примут вид:

$$a_{A,r+1} = a_{A,r} + (\hat{\omega}_{A,r+1}^{r+1})^2 \mathbf{x}_{A,r+1}^T \mathbf{x}_{A,r+1}, \qquad (13)$$

$$c_{A,r+1} = c_{A,r} + \mathbf{x}_{A,r+1}^T \mathbf{y}_A \hat{\omega}_{A,r+1}^{r+1}, \qquad (14)$$

С учётом $a_{A,r} = c_{A,r}$ формула (6) примет вид:

$$RSS_A = \mathbf{y}_A^T \mathbf{y}_A - a_{A,r+1}.$$

Аналогичным образом формулы для расчёта $a_{B,r+1}$, $c_{B,r+1}$ примут вид:

$$a_{B,r+1} = a_{B,r} + (\hat{\omega}_{A,r+1}^{r+1})^2 \mathbf{x}_{B,r+1}^T \mathbf{x}_{B,r+1}, \qquad (15)$$

$$c_{B,r+1} = c_{B,r} + \mathbf{x}_{B,r+1}^T \mathbf{y}_B \hat{\omega}_{A,r+1}^{r+1}, \qquad (16)$$

Критерий регулярности рассчитывается по той же формуле (9).

Запишем метод оценивания параметров и расчёта критерия селекции подсчитаем его вычислительную сложность – количество операций умножения и деления. Обозначим это число через *OpNum*. Метод разобьем на две стадии: 1) $r = 1$; 2) $r > 1$, r – номер этапа.

Стадия 1. $r = 1$.

1.1. Рассчитаем коэффициент и критерий селекции (критерий регулярности):

$$\hat{\omega}_{A,1}^1 = \frac{\mathbf{x}_{A,1}^T \mathbf{y}_A}{\mathbf{x}_{A,1}^T \mathbf{x}_{A,1}},$$

$$AR_{B,1} = \mathbf{y}_B^T \mathbf{y}_B - 2\hat{\omega}_{A,1}^1 \cdot (\mathbf{x}_{B,1}^T \mathbf{y}_B) + (\hat{\omega}_{A,1}^1)^2 (\mathbf{x}_{B,1}^T \mathbf{x}_{B,1}) \cdot$$

1.2. Вычисляем начальные значения для величин $a_{A,1}$, $c_{A,1}$, $a_{B,1}$, $c_{B,1}$ по формулам (*OpNum* = 11):

$$a_{A,1} = (\hat{\mathbf{y}}_{A,1}^T \hat{\mathbf{y}}_{A,1}) = (\mathbf{x}_{A,1}^T \mathbf{x}_{A,1})(\hat{\omega}_{A,1}^1)^2,$$

$$c_{A,1} = (\hat{\mathbf{y}}_{A,1}^T, \mathbf{y}_A) = (\mathbf{y}_A^T \mathbf{x}_{A,1}) \hat{\omega}_{A,1}^1,$$

$$a_{B,1} = (\hat{\mathbf{y}}_{B,1}^T \hat{\mathbf{y}}_{B,1}) = (\mathbf{x}_{B,1}^T \mathbf{x}_{B,1})(\hat{\omega}_{A,1}^1)^2,$$

$$c_{B,1} = (\hat{\mathbf{y}}_{B,1}^T, \mathbf{y}_B) = (\mathbf{y}_B^T \mathbf{x}_{B,1}) \hat{\omega}_{A,1}^1 \cdot$$

Стадия 2. $r = \overline{2,m}$. Запишем алгоритм для (r+1)-го этапа.

2.1 Вычисляем оценку коэффициента $\hat{\omega}_{A,r+1}^{r+1}$ по формуле (12). *OpNum* = 1.

2.2 Вычисляем величины $a_{A,r+1}$, $c_{A,r+1}$, $a_{B,r+1}$, $c_{B,r+1}$ по формулам (13)-(16). *OpNum* = 6.

2.3 Рассчитываем критерий регулярности по формуле (9). *OpNum* = 1.

Общее количество операций для построения модели от m аргументов равно $8m + 3$. При этом сложность расчёта модели на текущем этапе является постоянной, и равна *восьми* элементарным операциям. Как видим вычислительная сложность метода не зависит от количества наблюдений n_W и использует лишь $\mathbf{x}_{A,i}^T \mathbf{x}_{A,i}$, $\mathbf{x}_{B,i}^T \mathbf{x}_{B,i}$, $\mathbf{x}_{A,i}^T \mathbf{y}_A$, $\mathbf{x}_{B,i}^T \mathbf{y}_B$, $i = \overline{1,m}$.

Таким образом, ОРИА, принадлежащий классу итерационных алгоритмов, при использовании предложенного рекуррентного метода построения модели имеет такое же решение как и многоэтапный алгоритм МГУА. Поэтому назовём такой алгоритм Многоэтапным Алгоритмом с Ортогонализацией переменных (МАО).

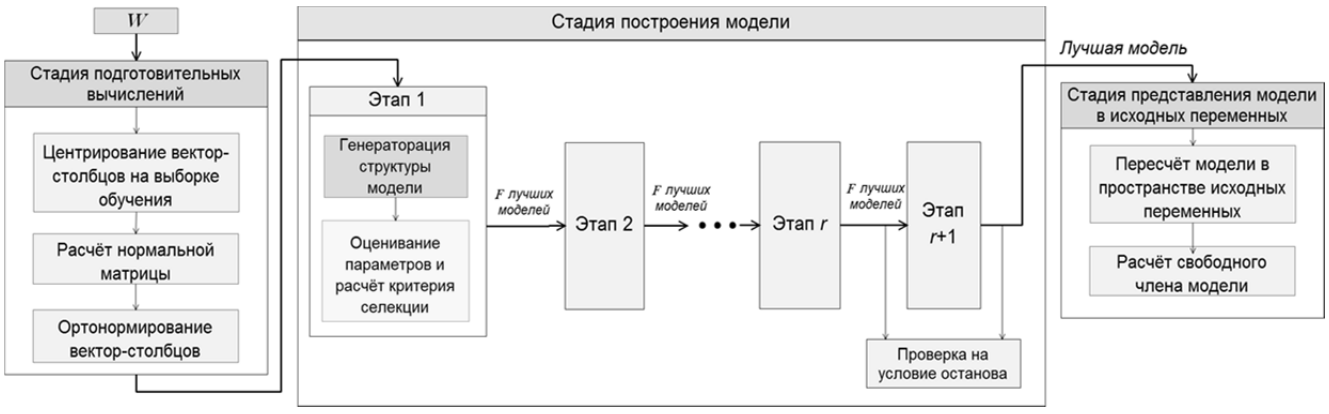


Рис. 2. Процесс построения модели в МАО

6. Многоэтапный алгоритм МГУА с ортогонализацией переменных

Процесс построения функции $f(\mathbf{X},\bar{\Theta})$ в МАО, такой же, как и в многоэтапном алгоритме MULTI [3]. Общая схема работы алгоритма представлена на рис. 2. Как видим, алгоритм состоит из трёх стадий:

1. Стадия подготовительных вычислений, на которой осуществляется:

1.1 Центрирование вектор-столбцов матрицы $\mathbf{W} = (\mathbf{X}\vdots\mathbf{y})$ на обучающей выборке A для ухода от необходимости оценивания свободного члена моделей по формулам:

$$\tilde{\mathbf{x}}_{ji} = \mathbf{x}_{ji} - \bar{\mathbf{x}}_{A,i},\ \bar{\mathbf{x}}_{A,i} = \frac{1}{n_A}\sum_{j=1}^{n_A}\mathbf{x}_{ji},\ i=\overline{1,m},\ j=\overline{1,n_W}$$

$$\tilde{\mathbf{y}}_j = \mathbf{y}_j - \bar{y}_A,\ j=\overline{1,n_W},\ \bar{y}_A = \frac{1}{n_A}\sum_{j=1}^{n_A}\mathbf{y}_j.$$

1.2 Ортогонализация вектор-столбцов матриц $\mathbf{X}_A$ и $\mathbf{X}_B$ для использования рекуррентного метода построения модели, предложенного в предыдущем разделе.

2. Стадия построения модели в ортогонализированном пространстве перемен-ных. Процесс построения модели следующий. На каждом этапе в текущую модель

$$\hat{\mathbf{y}}_s(\mathbf{x}_1,...,\mathbf{x}_s,\hat{\theta}_1,...,\hat{\theta}_s) = \sum_{i=1}^{s}\mathbf{x}_i\hat{\theta}_i$$

добавляется два параметра: аргумент $\mathbf{x}_{s+1}$, не содержащийся среди множества аргументов текущей модели и оценка коэффициента $\hat{\theta}_{s+1}$ при $\mathbf{x}_{s+1}$:

$$\hat{\mathbf{y}}_{s+1}(\mathbf{x}_1,...,\mathbf{x}_s,\mathbf{x}_{s+1},\hat{\theta}_1,...,\hat{\theta}_s) = \sum_{i=1}^{s+1}\mathbf{x}_i\hat{\theta}_i,$$

где s – номер этапа. Оценка коэффициента $\hat{\theta}_{s+1}$ рассчитывается по формуле (12), а критерий селекции модели по формуле (9). На первом этапе строится множество моделей, зависящих от одного аргумента, на втором – от двух, и т.д. до модели, содержащей все аргументы. На каждом этапе осуществляется выбор F лучших моделей по критерию селекции, которые переходят на следующий этап.

3. Стадия представления модели через исходные признаки, где осуществляется переход в исходное пространство и расчёт свободного члена модели как:

$$\hat{\theta}_{A,0} = \bar{y}_A - \sum_{i=1}^{m}\hat{\theta}_{A,i}\bar{\mathbf{x}}_{A,i}.$$

Правилом останова алгоритма может быть, например, условие: CR_s и CR_{s+1} – значения критериев селекции лучших из F моделей s-го и (s+1)-го этапа соответственно.

7. Ортогонализация и переход к исходным переменным

Для того чтобы вычислительная сложность метода построения моделей не зависела от количества наблюдений n_W, он должен использовать элементы матриц $\mathbf{W}_A^T\mathbf{W}_A$, $\mathbf{W}_B^T\mathbf{W}_B$, $\mathbf{W}_A = (\mathbf{X}_A\vdots\mathbf{y}_A)$, $\mathbf{W}_B = (\mathbf{X}_B\vdots\mathbf{y}_B)$. Входными данными для стадии построения модели должны быть элементы $\mathbf{z}_{A,i}^T\mathbf{z}_{A,i}$, $\mathbf{z}_{A,i}^T\mathbf{y}_A$, $\mathbf{z}_{B,i}^T\mathbf{z}_{B,i}$, $\mathbf{z}_{B,i}^T\mathbf{y}_B$, $i=\overline{1,m}$ ортогональных матриц $\mathbf{Z}_A$ и $\mathbf{Z}_B$.

Метод ортогонализации. В его основе лежит ортогонализация Грама-Шмидта [4]. Опишем метод пошагово, получая элементы $\mathbf{z}_{A,i}^T\mathbf{z}_{A,i}$, $\mathbf{z}_{A,i}^T\mathbf{y}_A$, $i=\overline{1,m}$ на выборке A, и подсчитаем его вычислительную сложность. Для выборки B он аналогичен.

1) $\mathbf{z}_{A,1} = \mathbf{x}_{A,1}$, $\mathbf{z}_{A,1}^T\mathbf{z}_{A,1} = \mathbf{x}_{A,1}^T\mathbf{x}_{A,1}$, $\mathbf{z}_{A,1}^T\mathbf{y}_A = \mathbf{x}_{A,1}^T\mathbf{y}_A$;

2) $\mathbf{z}_{A,2} = \mathbf{x}_{A,2} - \gamma_{A,1,2}\mathbf{z}_{A,1}$, $\gamma_{A,1,2} = \dfrac{\mathbf{z}_{A,1}^T\mathbf{x}_{A,2}}{\mathbf{z}_{A,1}^T\mathbf{z}_{A,1}}$;

$$\mathbf{z}_{A,2}^T\mathbf{z}_{A,2} = \mathbf{x}_{A,2}^T\mathbf{x}_{A,2} - 2\gamma_{A,1,2}\mathbf{z}_{A,1}^T\mathbf{x}_{A,2} + \gamma_{A,1,2}^2\mathbf{z}_{A,1}^T\mathbf{z}_{A,1}.$$

$$\mathbf{z}_{A,2}^T\mathbf{y}_A = \mathbf{x}_{A,2}^T\mathbf{y}_A - \gamma_{A,1,2}\mathbf{z}_{A,1}^T\mathbf{y}_A,$$

3) $\mathbf{z}_{A,3} = \mathbf{x}_{A,3} - \gamma_{A,2,3}\cdot\mathbf{z}_{A,2} - \gamma_{A,1,3}\mathbf{z}_{A,1}$,

$$\gamma_{A,1,3} = \frac{\mathbf{z}_{A,1}^T\mathbf{x}_{A,3}}{\mathbf{z}_{A,1}^T\mathbf{z}_{A,1}},$$

$$\gamma_{A,2,3} = \frac{\mathbf{z}_{A,2}^T\mathbf{x}_{A,3}}{\mathbf{z}_{A,2}^T\mathbf{z}_{A,2}} = \frac{\mathbf{x}_{A,3}^T\mathbf{x}_{A,2} - \gamma_{A,1,2}\mathbf{x}_{A,3}^T\mathbf{x}_{A,1}}{\mathbf{z}_{A,2}^T\mathbf{z}_{A,2}},$$

тогда с учётом $\mathbf{z}_{A,i}^T\mathbf{z}_{A,j} = 0$, $i \neq j$:

$$\mathbf{z}_{A,3}^T\mathbf{z}_{A,3} = \mathbf{x}_{A,3}^T\mathbf{x}_{A,3} + \gamma_{A,2,3}^2\cdot\mathbf{z}_{A,2}^T\mathbf{z}_{A,2} +$$
$$+ \gamma_{A,1,3}^2\mathbf{z}_{A,1}^T\mathbf{z}_{A,1} - 2(\gamma_{A,2,3}\mathbf{x}_{A,3}^T\mathbf{z}_{A,2} + \gamma_{A,1,3}\mathbf{x}_{A,3}^T\mathbf{z}_{A,1}),$$
$$\mathbf{z}_{A,3}^T\mathbf{y}_A = \mathbf{x}_{A,3}^T\mathbf{y}_A - \gamma_{A,2,3}\cdot\mathbf{z}_{A,2}^T\mathbf{y}_A - \gamma_{A,1,3}\mathbf{z}_{A,1}^T\mathbf{y}_A.$$

Для вектора $\mathbf{z}_{A,k}$:

$$\mathbf{z}_{A,k} = \mathbf{x}_{A,k} - \sum_{i=1}^{k-1}\gamma_{A,i,k}\mathbf{z}_{A,i},\ \gamma_{A,1,k} = \frac{\mathbf{z}_{A,1}^T\mathbf{x}_{A,k}}{\mathbf{z}_{A,1}^T\mathbf{z}_{A,1}},$$

$$\gamma_{A,2,k} = \frac{\mathbf{z}_{A,2}^T\mathbf{x}_{A,k}}{\mathbf{z}_{A,2}^T\mathbf{z}_{A,2}} = \frac{\mathbf{x}_{A,k}^T\mathbf{x}_{A,2} - \gamma_{A,1,2}\mathbf{x}_{A,k}^T\mathbf{z}_{A,1}}{\mathbf{z}_{A,2}^T\mathbf{z}_{A,2}},\ \ldots$$

$$\gamma_{A,k-1,k} = \frac{\mathbf{z}_{A,k-1}^T\mathbf{x}_{A,k}}{\mathbf{z}_{A,k-1}^T\mathbf{z}_{A,k-1}} = \frac{\mathbf{x}_{A,k}^T\mathbf{x}_{A,k-1} - \sum_{i=1}^{k-2}\gamma_{A,i,k-1}\mathbf{x}_{A,k}^T\mathbf{z}_{A,i}}{\mathbf{z}_{A,k-1}^T\mathbf{z}_{A,k-1}}.$$

Как видим при вычислении коэффициентов $\gamma_{A,i,k}$, $i=\overline{1,k-1}$ можно использовать величины $\mathbf{z}_{A,i}^T\mathbf{z}_{A,i}$, $i=\overline{1,k-1}$, $\gamma_{A,i,j}$, $i=\overline{1,k-2}$, $j=\overline{1,k-1}$, полученные при расчете векторов $\mathbf{z}_{A,i}$, $i=\overline{1,k-1}$, а также величины $\mathbf{x}_{A,k}^T\mathbf{z}_{A,i}$, $i=\overline{1,k-2}$.

При вычислении

$$\mathbf{z}_{A,k}^T\mathbf{z}_{A,k}=\mathbf{x}_{A,k}^T\mathbf{x}_{A,k}+\sum_{i=1}^{k-1}\gamma_{A,i,k}^2\mathbf{z}_{A,i}^T\mathbf{z}_{A,i}-2\sum_{i=1}^{k-1}\gamma_{A,i,k}\mathbf{x}_{A,k}^T\mathbf{z}_{A,i}$$

используются числители $\mathbf{x}_{A,k}^T\mathbf{z}_{A,i}$, $i=\overline{1,k-1}$ формул расчёта коэффициентов $\gamma_{A,i,k}$, $i=\overline{1,k-1}$, а так же $\mathbf{z}_{A,i}^T\mathbf{z}_{A,i}$, $i=\overline{1,k-1}$.

В общем случае

$$\mathbf{z}_{A,k}^T\mathbf{y}_A=\mathbf{x}_{A,k}^T\mathbf{y}_A-\sum_{i=1}^{k-1}\gamma_{A,i,k}\mathbf{z}_{A,i}^T\mathbf{y}_A$$

Если в памяти хранить $\gamma_{A,i,j}$, $i=\overline{1,m-1}$, $j=\overline{1,m}$, $\mathbf{z}_{A,i}^T\mathbf{z}_{A,i}$, $i=\overline{1,m}$, $\mathbf{z}_{A,i}^T\mathbf{y}_A$, $i=\overline{1,m}$, $\mathbf{x}_{A,k}^T\mathbf{z}_{A,i}$, $i=\overline{1,k-1}$, и те же самые величины на выборке B, то $OpNum = 1/3\cdot m^3+4m^2-7/3\cdot m$.

Если же в памяти хранить те же величины, что и в первом случае, кроме $\mathbf{x}_{A,k}^T\mathbf{z}_{A,i}$, $\mathbf{x}_{B,k}^T\mathbf{z}_{B,i}$ $i=\overline{1,k-1}$, то $OpNum = 2/3\cdot m^3 + 5m^2 - 5/3\cdot m$.

Метод перехода в исходное пространство. Пусть в ортогонализированном пространстве была найдена модель: $\hat{\mathbf{y}}_z=\sum_{i=1}^{m}\hat{\varphi}_{A,i}\mathbf{z}_{A,i}$. Модель в исходных переменных имеет вид: $\hat{\mathbf{y}}_x=\sum_{i=1}^{m}\hat{\theta}_{A,i}\mathbf{x}_{A,i}$. Необходимо получить коэффициенты $\phi_{A,i,j}$ при $\mathbf{x}_{A,i}$ в $\mathbf{z}_{A,j}$, $i,j=\overline{1,m}$, и тогда $\hat{\theta}_{A,i}$, рассчитать по формуле:

$$\hat{\theta}_{A,i}=\sum_{j=1}^{m}\hat{\varphi}_{A,j}\phi_{A,i,j},\ i=\overline{1,m} \qquad (17)$$

Опишем метод пошагово для каждого $\mathbf{z}_{A,i}$, $i=\overline{1,m}$.

1) $\mathbf{z}_{A,1}=\mathbf{x}_{A,1}$;

2) $\mathbf{z}_{A,2}=\mathbf{x}_{A,2}-\gamma_{A,1,2}\mathbf{z}_{A,1}$, $\phi_{A,1,2}=-\gamma_{A,1,2}$;

3) $\mathbf{z}_{A,3}=\mathbf{x}_{A,3}-\gamma_{A,2,3}\cdot\mathbf{z}_{A,2}-\gamma_{A,1,3}\mathbf{z}_{A,1}=$

$=(\gamma_{A,2,3}\gamma_{A,1,2}-\gamma_{A,1,3})\mathbf{x}_{A,1}-\gamma_{A,1,3}\mathbf{x}_{A,2}+\mathbf{x}_{A,3}$

$\phi_{A,1,3}=-\gamma_{A,2,3}\phi_{A,1,2}-\gamma_{A,1,3}\phi_{A,2,2}$, $\phi_{A,2,3}=-\gamma_{A,2,3}$;

Для вектора $\mathbf{z}_{A,k}$:

$$\phi_{A,1,k}=-(\gamma_{A,1,k}+\sum_{i=2}^{k-1}\gamma_{A,i,k}\phi_{A,1,i}),$$

$$\phi_{A,2,k}=-(\gamma_{A,2,k}+\sum_{i=3}^{k-1}\gamma_{A,i,k}\phi_{A,1,i}),\ldots$$

$\phi_{A,k-1,k}=-\gamma_{A,k-1,k}$.

Как видим, для вычисления коэффициентов $\phi_{A,i,k}$, $i=\overline{1,k-1}$ используются величины $\phi_{A,i,k-1}$, $i=\overline{1,k-1}$, полученные при расчёте $\mathbf{z}_{A,k-1}$. При расчёте формулы (17) $\phi_{A,i,i}=1$, $i=\overline{1,m}$. В данном алгоритме в памяти необходимо хранить коэффициенты $\phi_{A,i,j}$ $i=\overline{1,m-1}$, $j=\overline{1,m}$, $OpNum = m^3/6-0.5m^2+m/3$.

Таблица 1. Сравнение вычислительной сложности алгоритмов COMBI* и МАО

Стадии	Составляющие стадий	Количество операций *OpNum*	
		МАО	COMBI*
1. Подготовительные вычисления	Центрирование	$m+1$	0
	Расчёт матриц $\mathbf{W}_A^T\mathbf{W}_A$, $\mathbf{W}_B^T\mathbf{W}_B$	$n_W(0.5m^2+1.5m+1)$	$n_W(0.5m^2+2.5m+3)$
	Ортогонализация	$1/3\cdot m^3+4m^2-7/3\cdot m$	0
2. Построение m моделей с количеством аргументов от 1 до m		$8m+3$	$\frac{4}{3}m^3+\frac{25}{3}m^2+22m+\frac{45}{6}$
3. Представление модели в исходных переменных	Переход в исходное пространство переменных	$\frac{1}{6}m^3-\frac{1}{2}m^2+\frac{1}{3}m$	0
	Расчёт свободного члена модели	m	0
Общая вычислительная сложность		$n_W(0.5m^2+1.5m+1)+$ $+\frac{5}{6}m^3+\frac{7}{2}m^2+8m+4$	$n_W(0.5m+2.5m+3)+$ $+\frac{4}{3}m^3+\frac{25}{3}m^2+22m+\frac{45}{6}$
Сложность построения одной модели на (s+1)-м этапе		8	$4s^2+15s+16$

Наиболее быстрым алгоритмом МГУА, позволяющим получить точные решения для m моделей, содержащих от 1 до m аргументов, является модифицированная версия комбинаторного алгоритма МГУА COMBI* [5]. Алгоритм COMBI* использует метод вложенных структур для формирования структуры модели и метод окаймления матрицы нормальных уравнений (рекуррент-ный способ получения оценок МНК) – для оценивания параметров Θ. Этот алгоритма также как и МАО для расчёта моделей использует элементы матриц $\mathbf{W}_A^T\mathbf{W}_A$, $\mathbf{W}_B^T\mathbf{W}_B$.

Вычислительная сложность каждой стадии МАО и COMBI*, а также общее количество операций алгоритмов приведены в таблице 1. Предполагается, что при расчетах МАО в памяти хранятся величины $\mathbf{x}_{A,k}^T\mathbf{z}_{A,i}$, $\mathbf{x}_{B,k}^T\mathbf{z}_{B,i}$ $i=\overline{1,k-1}$ и свобода выбора МАО $F = 1$. При расчёте матриц $\mathbf{W}_A^T\mathbf{W}_A$, $\mathbf{W}_B^T\mathbf{W}_B$ алгоритма COMBI* в них добавлен столбец единиц, отвечающий за свободный член. На второй стадии в COMBI* для каждой из m моделей рассчитывается свободный член, в отличие от МАО, где он рассчитан только для полной модели.

8. Выводы

Теоретически доказано, что при ортогональной системе вектор-столбцов матрицы входов ОРИА позволяет получить точное решение идентичное многоэтапному алгортму.

Предложен многоэтапный алгоритм с ортогонализацией входных переменных.

Для МАО разработан рекуррентный метод оценивания параметров и расчёта критерия селекции моделей. Сложность расчёта одной модели метода является постоянной, в отличие от квадратичной в известной модификации комбинаторного алгоритма COMBI*. Это позволяет решать задачи моделирования количество переменных, в которых на два порядка больше.

Вычислительная сложность МАО при решении системы линейных уравнений размерности $m \times m$ меньше, по сравнению с алгоритмом COMBI* при идентичности их решений, несмотря на то, что в МАО дополнительно осуществляется ортогонализа-ция.

Список литературы

1. Павлов А.В. Обобщённый релаксационный итерационный алгоритм МГУА // Індуктивне моделювання складних систем. Збірник наук. праць. – К.: МННЦІТС, 2011. – С. 95-108.
2. Павлов А.В. Методика экспериментальных исследований сходимости итерационных алгоритмов метода группового учёта аргументов / А.В. Павлов, В.А.Павлов // Вісник НТУУ „КПІ”. Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: «Век+», – 2011. – № 54. – С. 3-8.
3. Ивахненко А. Г. Помехоустойчивость моделирования / Ивахненко А. Г., Степашко В. С. – К.: Наукова думка, 1985. – 214 с.
4. Хорн Р. Матричный анализ / Хорн Р., Джонсон Ч. – М.: Мир, 1989. – 655 с.
5. Степашко В.С. О последовательном оценивании параметров регрессионных моделей / Степашко В.С., Ефименко С.Н. // Кибернетика и системный анализ. – 2005. – № 4. – С. 184–187.

УДК 681.324

СИМОНЕНКО В.П.,
ЩЕРБИНА О.В.

ВЫБОР АРХИТЕКТУРЫ СИСТЕМЫ МОНИТОРИНГА РЕСУРСОВ В ГЛОБАЛЬНЫХ GRID СИСТЕМАХ

В статье выполнен анализ и сравнительная оценка применения централизованной, децентрализованной системы сбора информации о характеристиках вычислительных ресурсов распределенных неоднородных Grid систем по различным показателям.

Analysis and comparative appraisal of the use of centralized and decentralized systems for characteristics of computational resources in distributed heterogeneous Grid systems gathering is performed in this article.

Введение

Мониторинг ресурсов – одна из основных задач планирования в GRID системах.[1-4]

Для сбора информации о вычислительной среде можно использовать несколько стратегий мониторинга для выбора наилучшей в зависимости от конкретных характеристик GRID системы. Правильно сделанный выбор позволяет наиболее оптимальным образом распределять ресурсы GRID системы, а значит, и повысить эффективность работы GRID [5-7].

Для анализа и выбора системы мониторинга можно рассматривать централизованную, децентрализованную и иерархическую структуры. Централизованная система мониторинга сбора и обработки информации – это наиболее простое решение этой задачи. Однако при ее реализации резко повышается объем служебной информации, особенно на этапе сбора. Кроме этого, ввиду того, что *Grid* –система является динамически меняющимся объектом, то возникает проблема непротиворечивости данных в самом объекте (вычислительном узле) и сведениях о нем в центральном узле, что также увеличивает объем служебной информации. Децентрализованная система позволяет решить задачу непротиворечивости, так как информацию о ресурсе клиент получает непосредственно в нем, но такой системе присуща большая загрузка каналов связи, так как для каждого клиента необходимо сканировать все ресурсы *Grid* –системы для поиска нужного. Для выбора структуры системы мониторинга, следует учитывать, что глобальная *Grid* – система имеет ярко выраженную иерархическую структуру [7]. Используя свойства иерархической структуры можно предложить многоуровневую систему мониторинга, обеспечивающую достоинства централизованной и децентрализованной систем.

В пределах данной статьи будет рассмотрено 2 основных используемых типа систем мониторинга ресурсов: централизованный и иерархический (древовидный).

Централизованная система

Централизованная проста тем, что включает в себя один сервер мониторинга ресурсов, который занимается опросом и контролем всех серверов вычислений, так же хранит информацию о них и выделяет нужные ресурсы под конкретные задания клиентов.

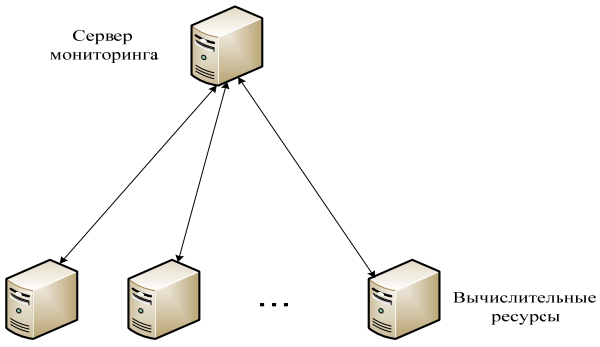


Рис. 1 структурная схема мониторинга централизованной системы

Время поиска ресурсов в такой системе при малом количестве ресурсов близкое к константному, и сводится, по сути, к выборке из базы данных нужных узлов, используя критерии клиента.

Достоинства системы:

- нужен всего один сервер мониторинга
- близкое к константному время поиска при малом количестве ресурсов

Недостатки системы:

- сложность в назначении клиенту такого ресурса, который бы имел с клиентом наибольшую скорость передачи данных, а так же наименьшую латентность;
- большие затраты на мониторинг ресурсов и выбор наиболее подходящих;
- отсутствие надежных линий связи между сервером вычислений и сервером мониторинга, что может приводить к потерям связи и утрате некоторых вычислительных ресурсов;
- при достижении критического количества ресурсов $N_{кр}$ сервер мониторинга перестанет справляться с задачей мониторинга – опросом всех своих подконтрольных серверов вычислений. Это потребует наращивание производительности сервера и повышений качества и количество линий связи от сервера мониторинга к серверам вычислений.

Отсюда видно, что использование централизованной системы мониторинга ресурсов в GRID-системах целесообразно только при не большом количестве ресурсов в этой системе.

Иерархическая система

Иерархическая система состоит множества серверов мониторинга, соединенных между собой в виде дерева, причем, сервера вычислений подключены к нижним серверам мониторинга в структуре дерева. Клиенты, как и сервера вычислений, тоже подключаются к самым нижним серверам мониторинга в иерархии. При подключении к системе, клиент посылает вышестоящему в иерархии серверу мониторинга запрос о выделении нужных

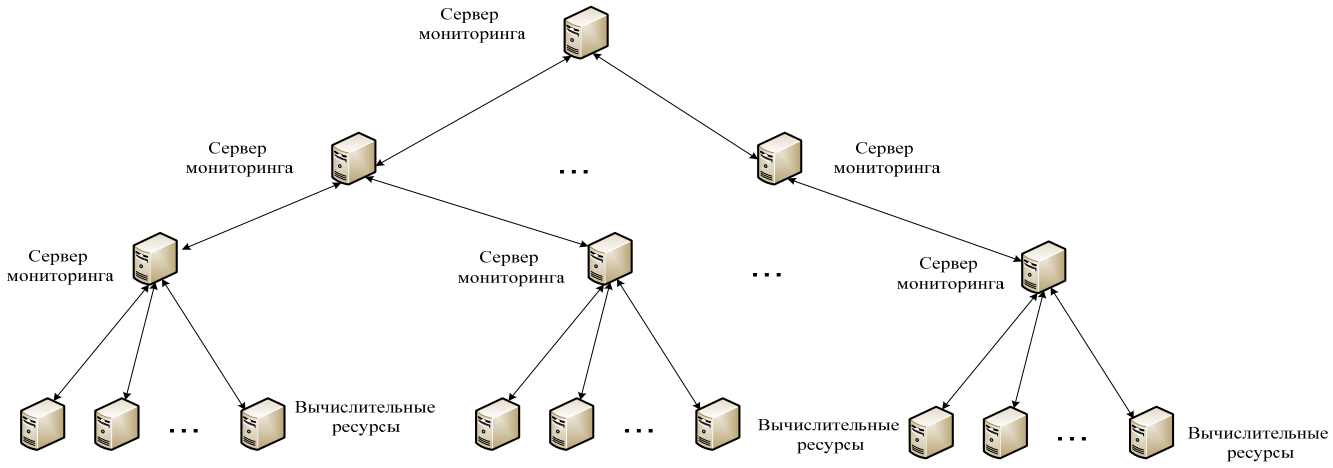


Рис 2. структурная схема мониторинга иерархической системы

ему ресурсов. Сервер мониторинга проверяет наличие серверов вычислений, подходящих к требованиям клиента и если находит таковые, то возвращает клиенту список ресурсов. Если же свободных ресурсов, походящих клиенту нет, то сервер мониторинга, либо сам подключается к серверу, стоящему выше по иерархии и передаёт запрос клиента, либо даёт клиенту адрес этого сервера. Сервер, стоящий выше по иерархии производит такой же поиск, только в своей базе данных, в которой находятся все сервера вычислений нижестоящих серверов мониторинга. Причем, сервер мониторинга и принадлежащие ему ресурсы, от которого пришел запрос, не будет участвовать в поиске. Если этот сервер мониторинга так же не смог найти подходящих ресурсов, то процедура поиска перейдёт к серверу выше по иерархии и так далее. Если корневой сервер так же не смог выделить ресурсы, то клиент остановит поиск и продолжит его после определённой задержки.

Достоинства системы:

- по возможности, клиенту будет предоставлен ресурс наименее отдалённый от него, и соответственно имеющий лучшее соединение(учтена локальная близость ресурса);
- хорошая масштабируемость такой системы, которую можно достичь, вводя новые уровни иерархической структуры;
- при потери связи между некоторым сервером мониторинга и сервером стоящим выше по иерархии возможно временное разделение GRID системы на две части.

Недостатки системы:

- большое количество серверов мониторинга, расположенных по всему миру (можно исключить этот недостаток введя функции мониторинга в перечень задач, решаемых серверами доменов);
- время поиска будет зависеть от успешности нахождения ресурсов, необходимых клиенту и находящихся от него на малом расстоянии.

Сравнительный анализ применения различных систем мониторинга ресурсов

В качестве примера рассмотрим глобальную GRID систему, вычислительные ресурсы которой распределены по всей планете. Будем считать, что:

- количество континентов – 5
- количество стран на континенте – 30 (поскольку нас интересует такая система именно в Европе)
- в каждой стране – 3 больших города;
- в каждом городе – 10 исследовательских института;
- в каждом институте – 5 лабораторий, в которых расположены сервера вычислений.

Зададимся следующими свободными полосами пропускания сетей передачи данных:

- межконтинентальный уровень – 1 Мбит/с;
- уровень континента – 5 Мбит/с;
- уровень страны – 30 Мбит/с;
- уровень города – 50 Мбит/с;
- уровень исследовательского института – 100 Мбит/с;
- уровень лаборатории – 1 Гбит/с.

Объём задания и результата к нему составляет 100 Мбайт; объём описания такого задания – 50 Кбайт.

Время на поиск ресурса в централизованной системе = 50 Кбайт / 1 Мбит/с + 1 сек (локальный поиск) = 1.5 сек

Рассчитаем время на передачу задания и результата к нему при использовании централизованной системы мониторинга:

- вероятность того, что выделенный вычислительный ресурс окажется на одном континенте с клиентом составляет 1/5;
- вероятность нахождения ресурса в одной стране с клиентом – 1/150;
- вероятность нахождения ресурса в одном городе с клиентом – 1/450;
- вероятность нахождения ресурса в одном институте – 1/4500;
- вероятность нахождения ресурса в одной лаборатории – 1/22500.

Исходя из расчетов видно, что централизованная система в основном будет выделять ресурса на отличном от клиента континенте, и вероятность выделения ресурса даже в одном и том же городе очень низка.

Среднее время на передачу задания и его результата = 100 * 8 / (4/5 * 1 (в случае нахождения на разных континентах) + 29/150 * 5 (в разных странах) + 2/450 * 30 (в разных городах) + 9/4500 * 50 (в разных институтах) + 4/22500 * 100 (в разных лабораториях) + 1/22500 * 1000 (в одной лаборатории)) = 800 / (0.8 + 0.96 + 0.13 + 0.1 + 0.02 + 0.04) = 390 сек.

Рассмотрим несколько вариантов GRID систем для расчета времени на поиск ресурса и времени на передачу задания и результата:

1. GRID, в котором требуемый ресурс равномерно распределён по земному шару в количестве 5000 экземпляров, то есть по 1000 на каждом континенте, по 33 в каждой стране, 10 в каждом городе и по 1 в каждом институте. То есть в среднем требуемый ресурс будет найден в одном и том же исследовательском институте, что и клиент;
2. искомых ресурсов – 500 экземпляров и один из них находится в одном и том же городе, что и клиент;
3. искомых ресурсов – 50 экземпляров и один из них находится в одной и той же стране, что и клиент;
4. искомых ресурсов – 5 экземпляров и один из них находится на одном и том же континенте, что и клиент.

Рассчитаем время на поиск ресурса и время на передачу задания и результата для каждого рассмотренного варианта:

1. Время на поиск ресурса = 50 Кб / 100 Мбит/с (доставка описания на сервер мониторинга в институте)+1 сек (локальный поиск) = 0.004+1 = 1 с

Время на передачу задания и его результата = 100 Мб / 100 Мбит/с = 8 сек

2. Время на поиск ресурса = 50 Кб / 100 Мбит/с (доставка описания на сервер мониторинга в институте) + 1 сек (локальный поиск) + 50 Кб / 50 Мбит/с + 1 сек (локальный поиск) = 0.004 + 1 + 0.007 + 1 = 2.01 сек

Время на передачу задания и его результата = 100 Мб / 50 Мбит/с = 16 сек

3. Время на поиск ресурса = 50 Кб / 100 Мбит/с (доставка описания на сервер мониторинга в институте) + 1 сек (локальный поиск) + 50 Кб / 50 Мбит/с + 1 сек (локальный поиск) + 50 Кб / 30 Мбит/с + 1 сек (локальный поиск) = 0.004 + 1 +0.007 + 1 + 0.013 + 1= 3.01 сек

Время на передачу задания и его результата = 100 Мб / 30 Мбит/с = 27 сек

4. Время на поиск ресурса = 50 Кб / 100 Мбит/с (доставка описания на сервер мониторинга в институте) + 1 сек (локальный поиск) + 50 Кб / 50 Мбит/с + 1 сек (локальный поиск) + 50 Кб / 30 Мбит/с + 1 сек (локальный поиск) + 50 Кб / 5Мбит/с + 1 сек (локальный поиск)= 0.004 + 1 + 0.007 + 1 + 0.013 + 1 + 0.08 + 1= 4.1 сек.

Время на передачу задания и его результата = 100 Мб / 5 Мбит/с = 160 сек

Представим полученные результаты таблично и графически:

Таблица 1. Затраты на поиск ресурса и передачу задания и результата в иерархической системе

Количество требуемых ресурсов в системе	5000	500	50	5
Время на поиск ресурса	1	2.01	3.01	4.1
Время на передачу задания и результата	8	16	27	160

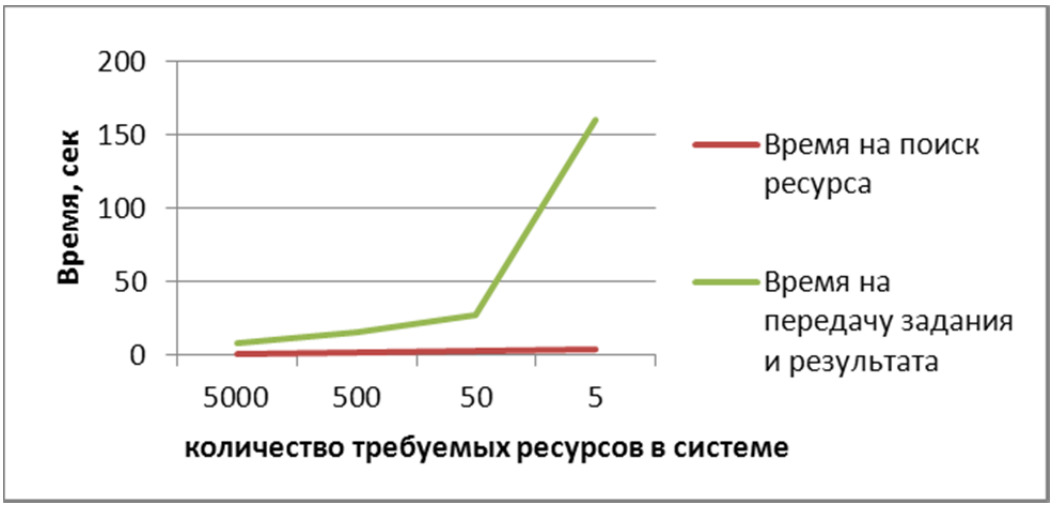


Рис. 3. Затраты на поиск ресурса и передачу задания и результата в иерархической системе

Выводы

Иерархическая система мониторинга ресурсов показывает лучшие результаты, чем централизованная практически на всех размерностях GRID системы.

С уменьшением размерности GRID системы, эффективность использования иерархической системы снижается, и затраты на передачу задания и его результата приближаются к параметрам централизованной системы. Поэтому использование иерархической системы мониторинга целесообразно только в тех GRID системах, где с большой вероятностью требуемый ресурс сможет быть найден в пределах одного города, где находится клиент.

Время на поиск ресурса в иерархической системе мониторинга увеличивается с уменьшением количества ресурсов, которые подключены к GRID системе, что является недостатком такой системы. Время на поиск ресурса в централизованной системе остаётся практически константным и сводится, по сути, к выборке из базы данных.

В связи с тем, что в централизованной системе мониторинга опросом и хранением данных о ресурсах занимается одна вычислительная машина, производительность её необходимо увеличивать с увеличением количества ресурсов, подключенных к GRID системе. В противном случае такой мониторинг не сможет эффективно обслуживать клиентов, а будет практически все своё время

заниматься сбором и обновлением информации о ресурсах, которые он обслуживает.

Список литературы

1. Распределенные вычисления, GRID-технологии или кластеры? [электронный ресурс]: Черняк Л.: Журнал «Открытые системы». – 2004 – вып. 4. – Режим доступу до журналу: http://www.osp.ru/cw/2004/72923/ _- Назва с екрану.
2. Платформа для коммерческихGRID [электронный ресурс]: Ривкин М. Журнал «Открытые системы», 2004 вып. 12. – Режим доступу до журналу: http://www.osp.ru/os/2003/12/183700/ - Назва с екрану.
3. Эксперименты с фрагментами сетей GRID [электронный ресурс]: Шевель А., Корхов В., Журнал «Открытые системы», 2001 вып. 5-6. – Режим доступу до журналу: http://www.osp.ru/cw/2001/05-06/034/http://www.osp.ru/cw/2004/12/01/ – Назва с екрану.
4. Главные вехи в истории метавычислений [электронный ресурс]: Климов А. , Журнал «Компьютерра», 2001 вып. 25. – Режим доступу до журналу: http://www.computerra.ru/offline/2001/402/10913 - Назва с екрану.
5. Вычислительная инфраструктура будущего [электронный ресурс]: Корягин Д.А., Коваленко В.Н. / ИПМ им. М.В.Келдыша РАН. – Режим доступу: http://citforum.ru/hardware/articles/futurvich.shtml – Назва с екрану.
6. Управление заданиями в распределенной вычислительной среде [электронный ресурс]: Коваленко В., Коваленко Е., Корягин Д., Любимский Э., Хухлаев Е.// Журнал «Открытые системы», 2001 вып. 5-6 – Режим доступу до журналу: http://www.osp.ru/os/2001/05-06/180168/ – Назва с екрану.
7. Internet как среда для вычислений [электронный ресурс] /Москалюк А // Журнал «Компьютерное Обозрение», 2002 вып. 21. – Режим доступу до журналу: http://ko.com.ua/internet_kak_sreda_dlya_ vychislenij_10012 – Назва с екрану.

УДК 004.75

СВІРІН П.В.

МЕТОДИКА ВДОСКОНАЛЕННЯ БРОКЕРА ДЛЯ NORDUGRID ARC

Для забезпечення вимог користувачів з продуктивності та ефективності виконання завдань грід – система повинна реалізувати ефективний алгоритм розподілу завдань між доступними на даний час обчислювальними ресурсами. Основна мета такого балансування навантаження в грід-системі – скоротити загальний час виконання завдання користувача і забезпечить ефективність використання обчислювальних ресурсів.

Мета даної роботи – розробка методики створення розподіленого брокера для ARC 2.0 із застосуванням сучасних можливостей платформи. Застосовуючи цю методику користувач може розробити власний брокер, який буде враховувати специфіку середовища чи певної категорії задач, для ефективного розподілу завдань серед обчислювальних ресурсів і інтегрувати його в платформу ARC.

In order to provide users with performance and task execution effectivity GRID has to implement an effective brokering algorithm. The main goal of such load balancing in GRID is to decrease the overall execution time and make utilization of the computing resources effective.

The purpose of the work is to develop the methodology on distributed broker implementation for Nordugrid ARC 2.0 using its modern features. Using this methodology users can implement his own brokers that will take into account specifics of environment or certain task category in order to distribute tasks among the available resources in an effective way and integrate this broker into ARC platform.

1. Вступ

Незважаючи на те, що алгоритми балансування навантаження обчислювальних ресурсів у грід системі вже досить давно досліджуються і існує багато вже, як готових алгоритмічних рішень, так і програмних реалізацій, інтенсивний розвиток грід – технологій, вдосконалення проміжного програмного забезпечення робить проблему балансування навантаження постійно актуальною і інтерес до дослідження в цій області не зменшується. Основна мета такого балансування навантаження в грід-системі – скоротити загальний час виконання завдання користувача і забезпечить ефективність використання обчислювальних ресурсів.

Грід–інфраструктура України побудована з використанням програмного забезпечення проміжного рівня (middleware) ARC (Advanced Resource Connector), що також відоме під назвою проекту NorduGrid [3].

В ARC, як с версії 0.8 так і в нової версії ARC 2.0 використовується як основний принцип максимально повна децентралізацію, тому на кожному робочому місці користувача Грід-мережі встановлюється персональний брокер, функція якого – вибір найкращого ресурсу для виконання завдання користувача, яке необхідно виконати в Грід-мережі.

В УНГ на даний момент використовується випадковий вибір ресурса, який не враховує поточного стану існуючих ресурсів. Для більш оптимального розподілу навантаження серед ресурсів необхідно розробити власні брокери, які будуть враховувать як поточний стан ресурсів, так і політику за якою відбувається балансування навантаження.

Слід відмітити, що до комплекту Nordugrid ARC входять лише найпростіші політики, отже, запропоновані методики можуть використовуватись не тільки в УНГ, а й для інших сегментів і віртуальних, організацій зі специфічними та загальними типами завдань,

2. Постановка задачі для УНГ

Основними задачами, для яких необхідний Грід, є:

- велика кількість задач з невеликими вимогами щодо ресурси. Такі задачі виконуються протягом короткого часу;
- велика кількість задач з великими вимогами щодо ресурсів і які виконуються протягом довгого часу.

Прикладами таких задач є:

- обробка даних експерименту ALICE. Звичайно така задача вимагає 1 процесор, дані передаються на ресурс прямо при обчисленні, максимальний час виконання – 24 години. Кількість таких задач може доходити до сотень тисяч;
- обчислення задач молекулярної динаміки.

Цей клас задач вимагає велику кількість процесорів, подає на обчислення невелику кількість даних, максимальний час розрахунку таких задач – місяці. Кількість таких задач може доходити до тисяч.

Таким чином, використання єдиної стратегії для розподілу різних класів задач не є ефективним. Варіантом рішення цієї проблеми стали спеціалізовані грід системи, такі як AliEN Grid, WeNMR. Однак, кількість класів задач є дуже великою і неможливо розробити систему для кожного класу задач.

Особливосстями украинской Грід інфраструктури є:

- 38 кластеров, малих за своїми обчислювальними потужностями [13];
- в наявності є тільки 2 ресурси з великою обчислювальною потужністю;
- всі ресурси працюють під керуванням ARC;
- різноманітна тематика розрахунків: молекулярна динамика, фізика, хімія, астрономія і т.і., велика кількість віртуальних організацій.

Особливостями брокеру ресурсів в Nordugrid ARC є:

- наявність лише спрощених політик розподілу задач
- орієнтація роботи системи на обробку результатів експеримента ATLAS, в якому переважають короткі задачі з невеликими об'ємами даних. Спеціально для цього експеримента було розроблено брокер, що робить висновок щодо цільового ресурса враховуючи об'єм необхідних даних в кеші обчислювального ресурса і, таким чином, зменшуючи час передачі даних.

Таким чином, в українському сегменті немає брокерів, що підходять для оптимального розподілу задач всіх класів.

В реальності задачі, що вимагають 10-30 процесорів направляються на виконання на кластер інститута кібернетики і очікують в черзі протягом днів. Туди ж можуть потрапляти більш короткі задачі і теж простоювать в черзі.

Отже, мета оптимального брокера для УНГ:

- направлення коротких задач – на більш слабкі ресурси
- направлення довгих задач – на більш потужні.

3. Методика побудови брокера ресурсів

3.1. Загальні положення

Загальна послідовність кроків при подачі завдання полягає в:

1. Опиті інформаційної системи;
2. Відборі кандидатів, характеристики яких відповідають вимогам завдання;
3. Отримання додаткової інформації від додаткових джерел інформації про ресурси;
4. Ранжування списку кандидатів згідно алгоритму брокера:
5. Подача завдання на ресурс.

Для реалізації алгоритму власного брокера необхідно реалізувати кроки 2-4.

Для побудови ефективного брокера необхідно:

- визначити сферу завдань і вимоги середовища, для яких брокер може ефективно робити розподіл між ресурсами;
- забезпечити отримання інформації від інформаційної системи;
- визначити критерії відкидання кандидатури ресурса;
- визначити набір характеристик, за якими буде проводитись оцінка завантаженості обчислювального кластеру;
- розробити порядок кроків для ранжування списку кластерів та відсіювання тих кластерів, що не підходять під вимоги завдання, для якого ведеться пошук кластеру призначення.
- розробити процедуру визначення черговості подачі завдань при необхідності розподілити декілька завдань одночасно.

Існують два методи ранжування ресурсів:

- послідовне оцінювання і відкидання ресурсів за кожним критерієм
- розрахунок інтегральної характеристики для кожного ресурса і подальше ранжування за цією характеристику.

Другий підхід вже розглядався в декількох статтях, зокрема [2], в якій автори будували інтегральну оцінку ресурса за його питомою продуктивністю, об'ємом пам'яті, кількістю процесорів, завантаження процесорів протягом попередньої хвилини та коефіцієнтом мережевого завантаження для ресурса.

Платформа Nordugrid ARC 2.0 надає можливість створення веб-сервісів для реалізації служб спеціального призначення. Отже, можливо реалізувати сервіс, що буде повертати розширену інформацію про ресурс. Ще одним джерелом розширеної інформації можуть бути

системи моніторингу, такі як Nagios та ін. В статті [1] було проведено огляд варіанту побудови брокера для Українського сегмента грід [4]. Розглянутий варіант брокера не вимагав встановлення сервісів, що знаходяться поза межами платформи nordugrid ARC.

З архітектурної точки зору брокери можна поділити на:

- постійні – модулі скомпільовані в програмі
- змінні – такі модулі можливо змінювати без перекомпіляції повного коду клієнтського ПЗ. Наприклад, можливо періодично звантажувати оновлення для брокера в рамках деякого проекту. Код алгоритму в даному випадку написано скриптовою мовою Lua. Під час запуску програми подачі завдання даному скрипту передається список ресурсів-кандидантів, на виході скрипт повертає ранжований список ресурсів.

З алгоритмічної точки зору брокери можна поділити на:

- такі, що обчислюють загальної оцінки для ресурса. При цьому можуть враховуватись як статичні і динамічні характеристики ресурса, так і статичні вимоги завдання до ресурсів, що дозволяє проводити балансування навантаження серед ресурсів згідно певної політики;
- такі, що використовують табличний метод – ранжування ресурсів проводиться згідно з вже накопиченими відомостями щодо довжини виконання певних видів завдань, що існують у проекті.

Основні джерела інформації про ресурси Nordugrid ARC:

- AREX – сервіс, що контролює виконання завдань. Повертає список статичних та динамічних характеристик, що описуються схемою GLUE 2;
- Infosys – інформаційна система Nordugrid ARC для версій 0.8 і нижче. Побудована на основі OpenLDAP і вміщує в собі список зареєстрованих обчислювальних ресурсів та їх поточний стан для статичних та динамічних характеристик;
- ISIS – являє собою реєстр сервісів, з яких можна отримати інформацію про ресурси.

Додаткові можливі джерела:

- додаткові сервіси платформи Nordugrid Arc.
- системи моніторингу Network Weather Service, Ganglia та інші;
- інші джерела.

Наприклад, з поширеною в ГРІД-середовищах системи Використання даних з Ganglia:

- кількість працюючих вузлів ресурса на поточний момент
- мережевий стан ресурса
- завантаження кластера протягом останнього часу

3.2. Характеристики для ранжування ресурсів

3.2.1. Статичні характеристики

Статичні характеристики ресурса отримуються шляхом прямого опитування сервіса AREX або інформаційної системи Infosys. Характеристики, отримані таким шляхом використовуються на першому кроці брокера для відкидання ресурсів, які гарантовано не відповідають вимогам завдання.

Статичні характеристики відображають характеристики обчислювальних ресурсів, що не змінюються з часом. Більшість таких характеристики знаходяться в описі ресурса в інформаційній системі. Прикладами статичних характеристик можуть бути:

- кількість процесорів на ресурсі
- кількість вузлів на ресурсі;
- об’єм пам’яті обчислювального
- продуктивність обчислювального ресурса

3.2.2. Динамічні характеристики

Динамічні характеристики є такими характеристиками, які змінюються з часом і брокер веде ранжування обчислювальних ресурсів, враховуючи значення таких характеристик на момент подачі завдання. Базові динамічні характеристики ресурса (такі як довжина черги) можливо отримати з інформаційної системи і розширити їх список використанням сервісів Nordugrid ARC.

Для отримання розширеної інформації щодо статичних характеристик ресурса та динамічних характеристик можливо використовувати додаткові джерела інформації. Частина динамічної інформації щодо стану ресурса зберігається в інформаційній системі, але недоліком такого підходу є те, що на момент вибору ресурса вона може бути застарілою.

Прикладом використання додаткових сервісів серед брокерів Nordugrid ARC є брокер Data, який опитує сервіс, що повертає об'єм даних, що необхідні для виконання завдання, в кеші ресурса. Недоліком даного підходу є те, що в разі відсутності такого сервісу на ресурсі-кандидаті цей ресурс відкидається. Отже, важливою задачею є розробка механізму компенсації відсутності запису про ресурс в додаткових джерелах інформації і не відкидати їх.

Приклади динамічних характеристик:

- кількість задач в черзі на обчислювальний ресурс;
- завантаженість ресурса;
- доступний дисковий об'єм.

3.2.3. Характеристики, специфічні для задачі

У випадку, в якому необхідно розподілювати серед обчислювальних ресурсів завдання, що пов'язані між собою певним чином, брокер може використовувати додаткові характеристики, що є специфічними для даної групи завдань [10][11] Наприклад, в екпериментах ATLAS [5] та ALICE [6] важливим для швидкості виконання завдань є наявність вже оброблених даних у кеші обчислювального ресурса. Розширені параметри завдання, що відсутні в мовах опису xRSL та JSDL [8][9], можливо вказати в централізованій базі даних завдання (наприклад, в ATLAS це prodDB, що базується на СУБД Oracle [5]). Процес виконання може контролюватись сервісом супервізора, напр. Eowyn [7] чи GANGA[12].

При наявності в рамках певного проекту набору однотипних завдань можливо також використовувати базу даних відповідності типу завдання та конфігурації ресурса до необхідного часу для виконання завдання.

3.3 Специфіка реалізації брокера в Nordugrid ARC 2.0.

В Nordugrid ARC 2.0 модуль брокера реалізується в якості плагіна, що завантажується програмою arcsub. Кожен такий плагін реалізовує специфічний для брокера метод відкидання ресурса, якщо його характеристики не задовольняють завданню, а також метод ранжування ресурсів, який визначає більш підходящим один з двох ресурсів, що подаються на вхід. Даний метод використовується в глобальній процедурі сортування ресурсів.

4. Приклади реалізації алгоритма брокера ресурсів

4.1. Побудова інтегральної характеристики із застосуванням виключно інформації з веб-сервісу платформи ARC.

В статті [14] описаний алгоритм побудови інтегральної оцінки ресурса із застосуванням додаткових сервісів платформи ARC. Використовується сервіс, що повертає повну інформацію щодо черг на ресурсі. При відсутності необхідної інформації щодо черг застосовується методика компенсація відсутності цієї інформації.

Розраховуються загальні довжина черги для кожного ресурса

$$N_i = \sum_{j=1}^{n}(L_j + L_{ja}) \qquad (1)$$

, де N_i – загальна довжина черги на і-му ресурсі, n – кількість черг, що існують на LRMS і-го ресурса, L_i- довжина j-ї черги на даному ресурсі, L_{ja} – довжина черги на рівні Nordugrid ARC, тобто кількість задач, що лише очікують на подачу на рівень LRMS.

Для кожного ресурса генерується коефіцієнт, що розраховується наступним чином:

$$M = rand(0, \frac{1}{2^S}) \qquad (2)$$

де $S=N_i$, якщо можливо отримати повну інформацію про черги на ресурсі з додаткового сервіса, або $S=N_i+1$, якщо такого сервіса на ресурсі немає.

Проводиться ранжування ресурсів за спаданням коефіцієнту M.

4.2. Побудова інтегальної характеристики із застосуванням інформації з Ganglia та веб-сервісу платформи ARC.

Коефіцієнт розраховується схожим чином, але використовуються додаткові дані з різних інформаційних джерел.

Приклад формули розрахунку коефіцієнта:

$$K = K_{nodes\_up}K_{cache}K_q K_{network\_load}K_{mem} \qquad (3)$$

де K_{nodes_up} – кількість активних вузлів на обчислювальному ресурсі, K_{cache} – коефіцієнт наявності необхідних для виконання завдання файлів в кеші даного обчислювального ресурса, K_q – коефіцієнт заповненості черг обчислювального ресурса, $K_{network\_load}$ – коефіцієнт завантаженості мережі на обчислювальному ресурсі, K_{mem} – коефіцієнт завантаженості пам'яті даного ресурса. Коефіцієнти K_{nodes_up},

$K_{network\_load}$, K_{mem} отримуються з системи моніторінгу Ganglia, коефіцієнти K_{cache}, K_q отримуються опитуванням веб-сервіса для платформи ARC. При відсутності відомостей в джерелах інформації щодо даного ресурса проводиться компенсація, подібна до описаної вище.

Описані методи з компенсацією недостатньої кількості додаткових інформаційних ресурсів дозволяють брати участь у ранжуванні ресурсам, на яких не встановлені додаткові інформаційні сервіси, або вони не зареєстровані в системах моніторинга. Така ситуація може виникнути в разі експериментального використання того чи іншого джерела, коли не ще немає чіткого рішення щого його застосування.

4.3. Ранжування згідно табличних даних щодо довжини виконання завдання.

Даний метод підходить для віртуальних організацій, які проводять обробку даних в рамках одного проекту, в якому завдання можна розділити на класи і обчислити залежність середньої довжини виконання завдання від об'єму даних, які необхідно обробити та типу системи на цільовому обчислювальному ресурсі. Якщо завдання неможливо віднести до будь-якого з відомих типів, то йому призначається середня довжина виконання зі всіх зареєстрованих типів завдань.

Прикладом розподілу завдань на класи може виступать певний формат опису завдання, що затверджено для проекту.

В даному варіанті застосовуються наступні типи інформаційних ресурсів:

1) інформаційна система Nordugrid ARC, що показує кількість і типи завдань, які ще не надійшли на PBS;
2) сервіс з прикладу 1, що показує кількість завдань на інших чергах обчислювального ресурса;
3) база даних, в якій зберігаються типи завдань та середні довжини їх виконання.

Розраховується оціночна довжина виконання завдань на ресурсі кандидаті:

$$L = \sum_{i=1}^{n} l_i \qquad (4)$$

де l_i – довжина виконання і-го завдання на даному ресурсі.

Ресурси ранжуються за зростанням оціночної довжини виконання завдань. Таким чином, на завдання надійде на ресурс з мінімальною довжиною виконання завдань, які надійшли раніше.

Також можливо відслідковувати фази виконання завдань, що дозволить більш точно розраховувать оціночну довжину виконання завдань.

До переваг даного метода можна віднести можливість визначення часу початку виконання завдання, що подається на ресурс. До недоліків – необхідність збору і обробки статистики виконання завдань.

4.4. Змінний брокер, що реалізується окремим скриптом.

Скрипт брокера періодично оновлюється з центрального сервера. Брокер ARC передає через інтерфейс Lua чи Python для мови C++ список ресурсів кандидатів, який ранжується всередині скрипта і повертається назад для подачі завдання. Таким чином можливо реалізувати будь-який алгоритм брокера і оновлювати його залежно від вимог віртуальної організації, в якій він використовується, без перекомпіляції клієнтського інструментарію.

Даний підхід відноситься до категорії архітектурних і вимагає лише інтерфейсів для таких скриптових мов на стороні користувача Грід та системи оновлення скриптів брокера.

Заключення

В статті проведено огляд принципів побудови брокерів для Nordugrid ARC, а також огляд можливих джерел інформації для ранжування списку ресурсів брокером. Розглянуто методику формування такої інтегральної оцінки за допомоги інформаційних джерел з комплекту Nordugrid ARC, а також комбінацію інформації з цих джерел і з даними зі сторонніх джерел, які надають додаткову інформацію про обчислювальні ресурси. Представлено варіанти реалізації алгоритму розподілу навантаження.

Отримані результати дозволяють проводити вибір джерел, що роблять можливим адекватних оцінити поточний стан ресурса та вибирати найкращий алгоритм брокера або його архітектури згідно умов обчислювального середовища чи віртуальної організації. Досліджено метод ранжування обчислювальних ресурсів, що мають неповну інформацію про себе в додаткових джерелах.

Використовуючи запропоновані методики можливо підібрати максимально відповідний

метод ефективного розподілу завдання між обчислювальними ресурсами згідно вимог до продуктивності обчислень і завантаження ресурсів. Також за даними методикою також можливо будувати інші варіанти реалізації із застосуванням інших додаткових джерел інформації.

Список літератури

1. А. Петренко Алгоритм оцінки завантаженості ГРІД-сайту / Петренко А.І., Свістунов С.Я. Свірін П.В // Системный анализ и информационные технологии : 13-я международная научно-техническая конференция «САИТ-2011», 23-28 мая 2011, Киев, Украина : материалы. – К. : УНК "ИПСА" НТУУ "КПИ", 2011. – С. 388.
2. Y. Chao-Tung. A Grid Resource Broker with Network Bandwidth-Aware Job Scheduling for Computational Grids / Y. Chao-Tung, C. Sung-Yi, C. Tsui-Ting. // Advances in Grid and Pervasive Computing. – 2007 – Vol. 4459. – pp.1 – 12.
3. Веб-сайт Nordugrid ARC. [Електронний ресурс] – Режим доступу: http://www.nordugrid.org
4. А.Загородний. Украинский академический грид: Українсько-македонський науковий збірник, Випуск 4 / А.Загородний, Г. Зиновьев, Е. Мартынов, С. Свистунов. – Київ 2009, Вид.Національна бібліотека України імені В.І.Вернадського. – с.140-150.
5. A. Read. Complete Distributed Computing Environment for a HEP Experiment: Experience with ARC-Connected Infrastructure for ATLAS. [Електронний ресурс] / A. Read, A. Taga, F. Ould-Saada, K. Pajchel, B. H. Samset, D. Cameron. – Режим доступа: http://www.nordugrid.org/documents/chep07-atlas.pdf
6. B. Beckles. Report on Swegrid/NorduGrid and the relationship between Swegrid and EGEE. [Електронний ресурс] – Режим доступа: https://www.jiscmail.ac.uk/cgi-bin/ webadmin?A3=ind0405&L=ETF&E=BASE64&P=361112&B=---559023410-1144747756-1085753675%3D%3A6504&T=APPLICATION%2Fpdf;%20name=%22Swegrid_report.pdf%22&N=Swegrid_report.pdf
7. J. Kennedy. ATLAS Production System. [Електронний ресурс] – Режим доступа: http://www.etp.physik.uni-muenchen.de/dokumente/talks/jkennedy_dpg07.pdf
8. Extended Resource Specification Language. [Електронний ресурс] – Режим доступа: http://www.nordugrid.org/documents/xrsl.pdf
9. O. Smirnova. NorduGrid's Extended Globus RSL (xRSL) vs JSDL, a comparison. [Електронний ресурс] / O. Smirnova, M. Niinimaki. – Режим доступа: http://www.nordugrid.org/documents/jsdl-vs-xrsl-revised2.pdf
10. J. C. Werner. Grid computing in High Energy Physics using LCG: the BaBar experience. [Електронний ресурс] – Режим доступа: http://www.gridpp.ac.uk/papers/ahm06_werner.pdf
11. L. Boyanov. On the employment of LCG GRID middleware. [Електронний ресурс] / L. Boyanov, P. Nenkova. – Режим доступа: http://ecet.ecs.ru.acad.bg/cst05/Docs/cp/SII/II.11.pdf
12. F. Brochu. Ganga: a tool for computational-task management and easy access to Grid resources [Електронний ресурс] / F. Brochu, U. Egede, J. Elmsheuser et al.. – Режим доступа: http://ganga.web.cern.ch/ganga/documents/pdf/ganga_cpc09.pdf
13. Grid Monitor. [Електронний ресурс] – Режим доступа: http://gridmon.bitp.kiev.ua/
14. А. Петренко. Гібридний Алгоритм брокера для Nordugrid ARC 2.0. / Петренко А.І, Свистунов С.Я., Свірін П.В. // HPC UA 2012: друга міжнародна конференція «Високопродуктивні обчислення», 8-10 жовтня, 2012, Київ, Україна 8-10 жовтня : матеріали. – К. : Національна академія наук України, 2012. – с. 275.

УДК 004.75

ГЕМБА Н.В.

ИССЛЕДОВАНИЕ КАСКАДНЫХ NEO-FUZZY НЕЙРОННЫХ СЕТЕЙ ДЛЯ ФОНДОВЫХ РЫНКОВ

Статья посвящена применению аппарата neo-fuzzy нейронных сетей к решению задачи прогнозирования показателей фондового рынка. Приведена структура исследуемой сети и выполнен сравнительный анализ результатов, полученных для различных финансовых инструментов. В качестве входов сети используются значения курсов акций и индексы.

The article focuses on the application apparatus of neo fuzzy neural networks to solve the problem of forecasting the stock market indices. The structure of the studied network and performed a comparative analysis of the results obtained for different financial instruments. As network inputs uses the values of stock prices and indices.

Введение

Вопросы анализа и прогнозирования колебаний курса акций представляют существенный интерес, как с финансовой, так и с социальной точек зрения, так как мировой фондовый рынок обладает высоким уровнем неопределенности, что приводит к неустранимым рискам при принятии инвестиционных решений. Традиционные методы анализа рисков фондового рынка, основанные на методах теории вероятности, к сожалению, недостаточно валидны, поскольку объекты выборки из генеральной совокупности не обладают свойством статистической однородности, а случайные процессы не имеют постоянных параметров. Использование нейронных сетей для анализа финансовой информации может являться альтернативой (или дополнением) для традиционных методов исследования.

Первые общие исследования метода neo-fuzzy нейросетей проводились в работах [1,2] без привязки к предметной области.

Постановка задачи

В работе решались следующие задачи:

- первичный анализ исходных данных и формирование обучающей выборки;
- выбор количества слоев нейросети
- проведение обучения (настройка коэффициентов) нейронной сети;
- прогнозирование и оценка погрешности.
- сравнительный анализ полученных данных для различных бумаг и рынков.
- решался вопрос о перспективности дальнейшего использования механизма neo-fuzzy нейросетей для анализа фондового рынка.

Целью работы является адаптация метода neo-fuzzy нейросетей для анализа и прогнозирования фондового рынка.

Первичный анализ исходных данных

В качестве данных для анализа были выбраны котировки акции компаний BP, Oracle на NYSE и индекс PTC. В качестве входящих данных использовались изменения цен закрытия. На рисунке 1 представлен исходный тренд и тренд изменения цен закрытия для котировок акций BP.

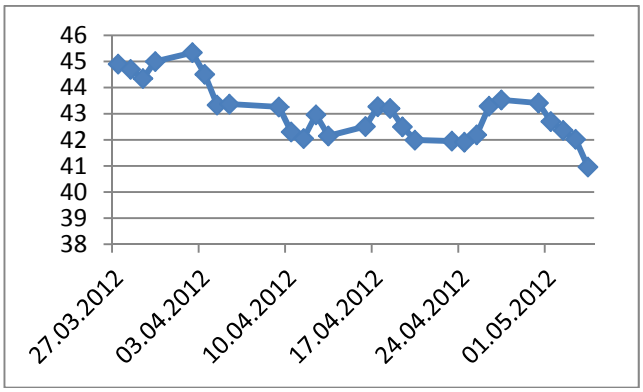


Рис.1(а). Изменение цен закрытия

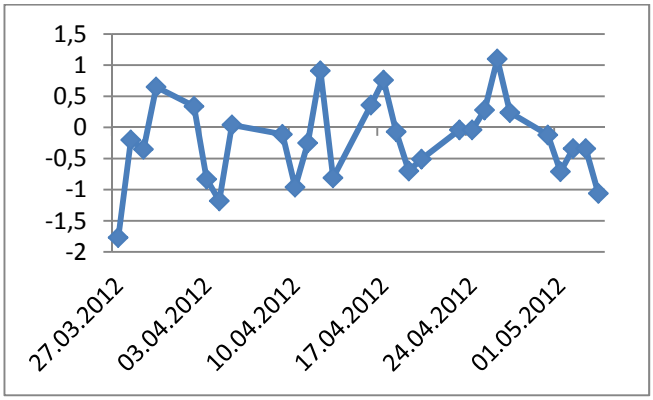


Рис.1(б). Разница цен закрытия

Структура каскадной neo-fuzzy нейронной сети

Исследование проводилось на каскадной neo-fuzzy нейронной сети с настраиваемым количеством слоев. Выбор количества слоев нейросети осуществлялся согласно критерию оптимальной сложности, аналогичного с МГУА. Таблица 1 представлен пример подбора оптимальной сложности для котировок акций ВР

В сети использовались треугольные функции принадлежности, 5 входов для каждого нейрона (согласно количеству дней торговой недели).

Архитектура CNFNN, которая показана на рис. 2, и характеризующее ее отображение имеет следующую форму [1]:

- neo-fuzzy нейрон первого каскада

$$\hat{y}^{[1]} = \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}{}^{[1]}\mu_{ji}(x_i)$$

− neo-fuzzy нейрон m-го каскада

$$\hat{y}^{[m]} = \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}{}^{[m]}\mu_{ji}(x_i) + \sum_{t=n+1}^{n+m-1}\sum_{j=1}^{h} w_{jt}{}^{[m]}t\left(\hat{y}^{[t-n]}\right)$$

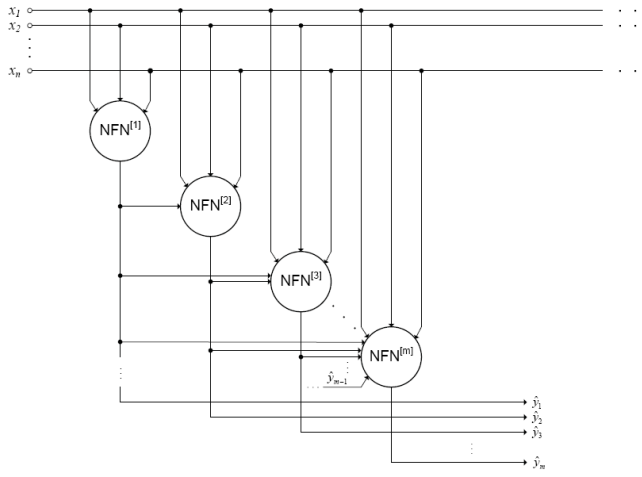


Рис. 2. Архитектура каскадной neo-fuzzy нейронной сети

Обучение нейронной сети

При обучении нейронной сети использовался классический подход к решению данной проблемы, а именно, минимизация оценки (отклонения) при заданной обучающей выборке и методе расчета функции ошибки (критерия обучения) [2]

$$E(k) = \frac{1}{2}\left(y(k) - \hat{y}(k)\right)^2 = \frac{1}{2}\left(e(k)\right)^2 = \\ = \frac{1}{2}(y(k) - \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}\mu_{ji}(x_i(k)))^2$$

Так как рассматриваемый вид нейронных сетей имеет неклассическую структуру, а именно, содержит элементы НМГУА (каскадность), обучение производилось в пакетном режиме с использованием рекуррентного метода наименьших квадратов.

Вектор синаптических весов оценивается по формуле

$$\omega^{[1]}(N) = (\sum_{k=1}^{N}\mu^{[1]}(k)\mu^{[1]T}(k))^{+}\sum_{k=1}^{N}\mu^{[1]}(k)y(k) = \\ = P^{[1]}(N)\sum_{k=1}^{N}\mu^{[1]}(k)y(k))$$

Далее процедура повторяется итеративно для всех слоев с добавлением дополнительного входа для предыдущего слоя на каждом последующем.

Для настройки весовых коэффициентов на последнем слое используются выражения

$$\omega^{[m]}(N) = P^{[m]}(N)\sum_{k=1}^{N}\mu^{[m]}(k)y(k)$$

$$\begin{cases} \omega^{[m]}(k+1) = \omega^{[m]}(k) + \\ + \dfrac{P^{[m]}(k)(y(k+1) - \omega^{[m]T}(k)\mu^{[m]}(k+1))}{1 + \mu^{[m]T}(k+1)P^{[m]}(k)\mu^{[m]}(k+1)}\mu^{[m]}(k+1) \\ P^{[m]}(k+1) = P^{[m]}(k) + \\ + \dfrac{P^{[m]}(k)\mu^{[m]}(k+1)\mu^{[m]T}(k+1)P^{[m]}(k)}{1 + \mu^{[m]T}(k+1)P^{[m]}(k)\mu^{[m]}(k+1)}, P^{[m]}(0) = \beta I \end{cases}$$

Данный алгоритм адаптации приводят к сокращению вычислительной сложности процесса обучения и сокращают время обучения по сравнению с более классическим градиентным методом.

Результаты эксперимента

Для сравнения эффективности прогнозирования были проведены вычисления на данных для различных компаний и индексов [3,4].

Полученные результаты (процент правильных прогнозов) приведены в таблицах 2-5.

Выводы

Проведенные исследования показали, что применение нечетких нейронных сетей позволяет построить более точный прогноз по сравнению со статистическими методами, успешно адаптирован для применения на фондовых рынках и обеспечивает достоверные результаты с погрешностью прогнозирования до 5%.

Показано, что данный метод является универсальным и не зависит от фондового рынка и выбранной ценной бумаги.

В качестве направлений дальнейших исследований планируется доработка структуры сети с использованием в первом слое различных функций принадлежности, увеличение входов сети за счет учета объемов продаж, а также повышение валидности исходных данных на основе автоматизированного анализа новостных блоков.

Табл. 1. Результаты варьирования количества слоев для BP(NYSE)

№ слоя	1	2	3	4	5	6	7	8	9	10
СКО	9,9	9,7	8,6	8,1	7,4	5,4	5,3	4,1	3,9	4,2

Табл.2. Результаты прогноза для 9-слойной каскадной NF сети. Акции BP(NYSE)

Дата	**Цена закр.**	**Изм. цены**	**Прогноз. изменение**	**Откл.**
17.04.2012	43,27	0,76	0,737044471	0,022955529
18.04.2012	43,2	-0,07	-0,076912197	0,006912197
19.04.2012	42,5	-0,7	-0,723448294	0,023448294
20.04.2012	41,99	-0,51	-0,512026521	0,002026521
23.04.2012	41,95	-0,04	-0,045385217	0,005385217
24.04.2012	41,91	-0,04	-0,030691814	-0,009308186
25.04.2012	42,19	0,28	0,295866428	-0,015866428
26.04.2012	43,29	1,1	1,097552911	0,002447089
27.04.2012	43,53	0,24	0,247806517	-0,007806517
30.04.2012	43,41	-0,12	-0,111805251	-0,008194749
01.05.2012	42,7	-0,71	-0,705084907	-0,004915093
02.05.2012	42,36	-0,34	-0,352185712	0,012185712
03.05.2012	42,02	-0,34	-0,343341918	0,003341918
MAPE				5,05228088
СКО				3,9612744

Табл. 3. Результаты прогноза для 12-слойной каскадной NF сети. Акции Oracle (NYSE)

Дата	**Цена закр.**	**Изм. цены**	**Прогноз. изменение**	**Откл.**
17.04.2012	29,28	0,64	0,644837779	-0,004837779
18.04.2012	29,12	-0,16	-0,141646834	-0,018353166
19.04.2012	29,01	-0,11	-0,100081564	-0,009918436
20.04.2012	28,88	-0,13	-0,134767589	0,004767589
23.04.2012	28,48	-0,4	-0,384398393	-0,015601607
24.04.2012	28,69	0,21	0,218971251	-0,008971251
Дата	**Цена закр.**	**Изм. цены**	**Прогноз. изменение**	**Откл.**
25.04.2012	28,87	0,18	0,182256265	-0,002256265
26.04.2012	29,02	0,15	0,15156108	-0,00156108
27.04.2012	29,24	0,22	0,215331992	0,004668008
30.04.2012	29,4	0,16	0,15756039	0,00243961
01.05.2012	29,57	0,17	0,162176113	0,007823887
02.05.2012	29,71	0,14	0,123336467	0,016663533
03.05.2012	29,38	-0,33	-0,31486421	-0,01513579
MAPE				4,513400783
СКО				3,8634555

Табл. 4. Результаты прогноза для 10-слойной каскадной NF сети. Индекс РТС

Дата	Цена закр.	Изм. цены	Прогноз. изменение	Откл.
10.04.0012	1622.07000	-8,71	-8,7626008	0,052601
11.04.0012	1606.91000	-4,5	-4,6160066	0,116007
12.04.0012	1615.69000	7,45	7,5207818	-0,070782
13.04.0012	1618.72000	25,16	24,914922	0,245078
16.04.0012	1612.74000	2,18	2,003641	0,176359
17.04.0012	1587.58000	2,34	2,2713621	0,068638
18.04.0012	1585.40000	-26,5	-26,183145	-0,316855
19.04.0012	1583.06000	-9,97	-9,4627076	-0,507292
20.04.0012	1609.16000	46,65	46,233916	0,416084
23.04.0012	1619.53000	-3,34	-3,2091331	-0,130867
24.04.0012	1572.88000	-7,2	-6,8967989	-0,303201
25.04.0012	1576.22000	8,81	8,9572422	-0,147242
26.04.0012	1583.42000	-19,59	-19,890992	0,300992
MAPE				4,470381366
СКО				1,6754807

Табл. 5. Сравнение результатов прогнозирования

	BP(NYSE), 9 слоев	Oracle (NYSE), 12 слоев	RTS, 10 слоев
MAPE	5,05228088	4,513400783	4,470381366
СКО	3,9612744	3,8634555	1,6754807

Список литературы

1. Bodyanskiy Ye., Kokshenev I., Kolodyazhniy V. An adaptive learning algorithm for a neo-fuzzy neuron // Proc. 3rd Int. Conf. of European Union Soc. for Fuzzy Logic and Technology (EUSFLAT 2003). – Zittau, Germany, 2003. – P. 375–379.
2. Е.В. Бодянский, В.В. Волкова, С.Д. Громов. Каскадная эволюционная нейронная сеть с нео-фаззи-нейронами в качестве узлов [Электронный ресурс]. – Режим доступа: http://www.nbuv.gov.ua/portal/natural/Vejpt/2011_4_3/2011_4_3/55-58.pdf – Последнее обращение 20.05.2011. – Название с экрана
3. Котировки акций по NYSE [Электронный ресурс]. – Режим доступа: http://www.google.com/finance – Последнее обращение 20.05.2011. – Название с экрана
4. Котировки индекса РТС [Электронный ресурс]. – Режим доступа: http://www.rts.ru – Последнее обращение 20.05.2011. – Название с экрана

УДК 683.519

СТЕНИН А.А.,
СТЕНИН С.А.,
ТКАЧ М.М.,
ГУБСКИЙ А.Н.

СИНТЕЗ ИЕРАРХИЧЕСКОЙ СТРУКТУРЫ КРИТЕРИЕВ ОЦЕНКИ ПРИ АНАЛИЗЕ ДЕЯТЕЛЬНОСТИ ОПЕРАТОРОВ СЛОЖНЫХ ТЕХНИЧЕСКИХ СИСТЕМ

В настоящей статье предложен многоуровневый системный подход к формированию иерархической структуры критериев оценки при анализе деятельности операторов сложных систем. Практическая реализация такого метода показана на примере оценки деятельности операторов технологическим процессом подвижных объектов.

The subject of the article is a systematic approach to the multi-level hierarchical structure formation evaluation criteria in the analysis of the activities of the operators of complex systems. The practical implementation of this method is demonstrated by the example of the evaluation of process operators of moving objects.

1. Введение

Результаты многочисленных научно-исследовательских и опытно-конструкторских разработок неоспоримо свидетельствуют, что отличительной особенностью сложных систем, состоящих из многих подсистем, является осуществление взаимодействия иерархическим образом. Как известно, при рассмотрении подобных систем наиболее важную роль играют три основных аспекта[1]:

- принятый уровень абстракции (степень агрегированности модели системы);
- уровень сложности принятия решений;
- уровень приоритета в системе, состоящей из многих подсистем.

В настоящее время уже с полной определенностью можно утверждать, что ни одна из существующих теорий анализа сложных технических систем не может претендовать на то, что единственно она дает адекватное описание процесса ее функционирования. Скорее всего имеется целый спектр теорий, в той или иной мере трактующих проблемы анализа. Это прежде всего обусловлено сложностью и большой степенью неопределенности процесса взаимодействия отдельных подсистем, имеющегося множества аспектов, для исследования которых приходится привлекать знания из различных дисциплин.

Многоуровневый системный подход в теории анализа в первую очередь характеризуется представлением сложной системы в виде взаимосвязанных подсистем, обладающих правом принимать решения и образующих строгую иерархию. Это в значительной мере позволяет решить следующие вопросы:

- установить особенности, характеризующие цели и задачи профессиональной деятельности операторов;
- найти способы отображения функций, выполняемых системой на ее иерархическую структуру;
- определить методы и математические средства, применяемые для оценки деятельности операторов сложных технических систем.

Среди всех тех непосредственных преимуществ, которые представляет применение теории многоуровневых систем к исследованию процесса оценки, можно указать, в первую очередь, следующие:

- создание единой основы для различных подходов посредством введения системы понятий и методов, что позволяет сравнить, противопоставить и взаимно дополнить различные теории;
- строго математическая формулировка как основных понятий, так и получаемых результатов;
- получение отправных точек для исследования различных аспектов и проблем анализа и проектирования эргатических систем с помощью математических методов и моделирования на ЭВМ.

Многоуровневый системный подход, базируясь на математических методах, прежде всего, связан с коммуникацией, управлением, координацией и так далее.

Координирование подразделяется на две части: установление операционных правил, предписывающих законы функционирования для подсистем и практическая реализация этих законов в рамках функционирования всей системы. Первая из них, по сути, соответствует выбору подходящих критериев для оценки эффективности деятельности отдельных подсистем, или в общем смысле, выбору способов координирования при достижении тех или иных формализованных целей.

Следует отметить, что для оценки деятельности оператора в значительной степени характерно качественное определения и слабая формализация таких целей функционирования, как степень эффективности процесса управления . Очевидно, что такое представление цели дает мало конструктивных рекомендаций, как для выработки стратегии ее достижения, так и для интерпретации полученных результатов.

Можно отметить следующие основные способы задания целей.

1. Цель может задаваться посредством перечня внешних требований, которым должен удовлетворять предполагаемый результат.
2. Цель может задаваться путем определения свойств и характеристик, которыми должен обладать предполагаемый результат. Свойства отличаются от требований тем, что относятся непосредственно к своему результату, в то время как требования выступают в качестве внешних условий, которым должны удовлетворять свойства.
3. Цель может быть задана в виде четко сформулированного по содержанию и по форме представления достигаемого результата.

Для многоуровневой системы характерна вполне определенная иерархия целей, формально описываемых соответствующими типами задач: глобальными и решаемыми подсистемами на том или ином уровне иерархии. Очевидно, что для успешной работы всей системы существенно, чтобы цели (задачи) ее подсистем были согласованны с глобальной целью. Совместимость целей формально вытекает из следующих положений:

– только нижестоящие решающие элементы многоуровневой системы являются подсистемами, находящимися в непосредственной связи со всем процессом;
– достижение глобальной цели может быть осуществлено только посредством нижестоящих решающих элементов;
– задачи, решаемые на каждом из таких уровней или расположенные на этом уровне решающие элементы, должны быть координируемы относительно решаемой глобальной задачи;
– вышестоящий решающий элемент, осуществляя координацию, воздействует на нижестоящие элементы, имея в виду свои собственные цели, то есть задачи, решаемые на уровне нижестоящих элементов, должны быть координируемы по отношению к задачам, решаемым вышестоящими элементами;
– глобальная задача, как правило, лежит вне сферы деятельности системы с той или иной иерархией, то есть ни один из решающих элементов внутри иерархии специально не обеспечен полномочиями решать глобальную задачу и тем самым преследовать общую (глобальную) цель, хотя, задача определена в терминах всего процесса.

Таким образом, для совместимости решаемых задач, а тем самым и целей внутри многоуровневой системы, координация задач, решаемых нижестоящими элементами, относительно задачи вышестоящего элемента должна быть соответствующим образом связана с подлежащей решению глобальной задачей.

При разработке системы оценки в качестве глобальной цели следует, очевидно, считать получение некоторого объективного критерия, оценивающего весь процесс деятельности оператора на некотором наборе тестовых режимов работы. Поскольку совокупность тестовых режимов работы, как правило, строится на базе параметрических моделей объекта управления, то формализация процесса оценки действий оператора, по сути дела, сводится к установлению отношений, связывающих параметры динамики объекта управления со значениями обобщенного критерия. При этом обобщенный критерий позволяет дать количественное определение цели оценки, заданной на качественном уровне и указать на эффективные способы их достижения. Если же существует однозначная связь между целью процесса оценки и средствами ее достижения, то критерий оценки может быть задан в виде соответствующего аналитического выражения. В этом случае критерий, оценивающий эффективность деятельности оператора позволяет определить совокупность действий , обеспечивающих достижение заданной цели управления. Эта ситуация может

иметь место исключительно для так называемых "простых" задач управления.

2. Постановка задачи

Естественно, что в сложных системах управления, где глобальная цель носит качественный характер, не представляется возможным определить ее в аналитической форме и единственной альтернативой является формирование многоуровневого критерия. Будучи характерным для системного подхода, критерий такого вида позволяет определить при оценке не только достижение цели управления на заданном наборе тестовых задач, но и контролировать входящие в него как частные режимы, так и отдельные переменные этих режимов.

Такое представление приводит к необходимости рассматривать векторный критерий эффективности обучения:

$$F(\bar{x}) = \bar{f}(\bar{x}) = \{f_1(\bar{x}), f_2(\bar{x}), \ldots, f_n(\bar{x})\}. \qquad (1)$$

В этом случае каждый из частных критериев преследует достижение своей локальной цели. В результате, достижение реальной глобальной цели всегда будет каким-то компромиссом, каким-то сочетанием требуемых качеств, причем заранее неизвестных. В этом и заключается основная проблема многокритериальности (неопределенности целей).

Существует целый ряд способов преодоления неопределенности целей, которая в данном случае заключается в выборе дополнительных гипотез, обеспечивающих минимальное значение критериальных функций $f_1(\bar{x}), f_2(\bar{x}), \ldots, f_n(\bar{x})$ одновременно.

Чаще всего они сводятся к тем или иным способам приведения многокритериальных задач к обычным задачам с одним критерием, поскольку для последних (особенно если целевая функция достаточно гладкая) существуют хорошо разработанные методы решения, либо к исключению из нормального анализа тех вариантов решения, которые заведомо будут плохи.

Второй подход в теории принятия решений называется "принцип Парето" и заключается в том, что в качестве решения следует выбирать только тот вектор $\bar{x}$, который принадлежит множеству Парето. Следует, однако, отметить, что принцип Парето не выделяет единственного решения, он только сужает множество альтернатив. Окончательным выбор все-таки остается за лицом, принимающим решение. Построенное множество Парето лишь облегчает процедуру выбора решения.

В этом случае более приемлемым оказывается первый подход (линейная свертка), когда вместо n частных критериев f_i рассматривается лишь один обобщенный критерий вида:

$$F(\bar{x}) = \sum_{i=1}^{n} \lambda_i f_i(\bar{x}), \qquad (2)$$

где λ_i – некоторые положительные числа, тем или иным способом нормированные (например $\sum_{i=1}^{n} \lambda_i = 1$).

Такой способ свертки вводит, по существу, отношение эквивалентности различных целевых функций $f_i(\bar{x})$, так как величины λ_i показывают, насколько изменяется целевая функция $F(\bar{x})$ при изменении критерия $f_i(\bar{x})$ на единицу:

$$\lambda_i = \frac{\partial F}{\partial f_i}. \qquad (3)$$

Коэффициенты λ_i являются результатом экспертизы и отражают представление оперирующей стороны о содержании компромисса, который она вынуждена принять. Таким образом, содержание компромисса состоит в ранжировании целей, которые вместе с назначением весовых коэффициентов и является той дополнительной гипотезой, позволяющей свести задачу со многими критериями к задаче с единственным критерием.

Очевидно, что количественный результат, полученный согласно указанному выше способу, должен определенным образом отражать качественную сторону процесса оценки, то есть удовлетворять исходной глобальной цели. Это может быть достигнуто, в частности, введением шкалы эффективности, отражающей соответствие шкалы количественного результата принятой шкале балльной оценки действий. Рассмотрим синтез иерархической структуры на примере оценки деятельности операторов технологических процессов и объектов движения.

3. Решение задачи

Нижним *уровнем* иерархии будем считать критерии оценки качества управления объектом по переменным (параметрам) движения; средним *уровнем* – критерий оценки качества выполнения того или иного режима движения на основе ранее сформированных оценок по параметрам движения; верхним *уровнем* иерархии

будет- комплексный критерий, оценивающий действия оператора на заданной совокупности режимов.

Формирование значений частных и обобщенных критериев производится на основе сравнения эталонных (программных) параметров динамики технологических процессов или объектов движения с текущими, полученными в процессе отработки тех или иных тестовых режимов работы.

Пусть рассогласование между текущими и программными значениями по i-той переменной определяется как:

$$\left|\Delta\tilde{x}_i(t)\right| = \left|x_i(t) - x_i^э(t)\right|,\ t = [0, T],\ i = \overline{1, M}, \quad (4)$$

где $x_i^э(t)$ – эталонное изменение i-той переменной;

M – число переменных;

T – интервал управления.

При этом для i-той переменной должен быть задан допустимый диапазон отклонений $\Delta x_{i\,max}$=const, в котором производится оценка качества управления подвижным объектом. Так как рассогласования $\left|\Delta\tilde{x}_i(t)\right|$ i-тых переменных имеют различную физическую природу, а, следовательно, и различный диапазон изменений, необходимо его пронормировать. Если принять:

$$\left|\Delta x_i(t)\right| = \frac{\left|\Delta\tilde{x}_i(t)\right|}{\Delta x_{i\,max}}, \quad (5)$$

то все рассогласования будут лежать в нормированном диапазоне [0, 1]и одинаково влиять на результат последующей суммарной оценки.

Таким образом, критерием контроля и оценки качества управления по i-тому параметру на некотором интервале управления T может служить критерий вида:

$$I_i = \frac{1}{T}\int_0^T \left|\Delta x_i(t)\right| dt. \quad (6)$$

Однако, непосредственное его использование может дать необъективную оценку качества управления, в частности, при наличии кратковременных существенных отклонений i-го параметра от его эталонного значения. Поэтому необходимо модифицировать данный критерий. Пусть считается, что в нормированном диапазоне [0, 1] всегда могут быть условно выделены согласно рис. 2.1 три зоны, соответствующие оценкам “5”, “4”, “3” по принятой пятибалльной системе. В общем случае значения коэффициентов δ_{1i} и δ_{2i} (рис. 1), определяющих эти зоны, определяются либо на основе экспертных оценок, либо в результате анализа тестовых реализаций процесса управления технологическим процессом или подвижным объектом.

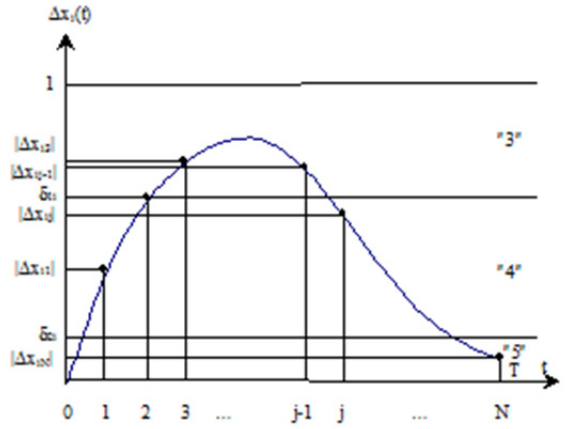


Рис.1. Экспертная градация уровней нормированного диапазона

Такая градация уровней позволяет учитывать при оценке как величину, так и длительность кратковременных отклонений (“выбросов”) переменной относительно ее программных значений на интервале управления T.

При этом формирование значений оценки по параметрам движения производится следующим образом. На интервале управления T через дискретные промежутки времени, длительностью Δt, формируется последовательность отклонений i-той переменной Δx_{ij} $(j = \overline{1, N_i})$, где $N_i = \frac{T}{\Delta t_i}$. Пусть n_{1i} – количество точек, попавших в область “3”, n_{2i} – в область “4” и n_{3i} – в область “5”. При этом должно выполняться условие:

$$n_{1i}+n_{2i}+n_{3i}=N_i. \quad (7)$$

В этом случае суммарная (интегральная) оценка i-той переменной определится как

$$I_i = \frac{1}{n_{1i} + n_{2i} + 1}\left[\sum_{j_1=1}^{n_{1i}} |\Delta x_{ij_1}| + \sum_{j_2=1}^{n_{2i}} |\Delta x_{ij_2}|\right], \quad (8)$$

которая позволяет избежать неопределенности в ситуации, когда $n_{1i}=0$, $n_{2i}=0$, а $n_{3i}=N_i$.

Важным вопросом является выбор шага дискретизации Δt_i, значение которого лежит в пределах $\Delta t_{i\,min}<\Delta t_i<\Delta t_{i\,max}$. В этом случае в качестве верхнего уровня Δt_i может быть выбрана величина:

$$\Delta t_i = \frac{1}{2}\Delta t_{i\,max} = \frac{\delta_{1i}\,|\,\Delta x_{i\,max}\,|}{2\,|\,\Delta \dot{x}_{i\,max}\,|}, \quad (9)$$

что исключает “проскакивание” очередной дискретной точкой минимальной из выбранных областей (рис. 2.1). Требования к нижней границе определяются технологическими характеристиками датчиков и вычислительной аппаратуры, применяемых для формирования и обработки указанных массивов информации.

При управлении большинством существующих технологических процессов и подвижных объектов зачастую возникает необходимость в оценке требуемого сочетания качеств, характеризующих данную цель управления. Являясь, как известно, задачами векторной оптимизации, они решаются, как правило, на основе компромисса, с учетом важности того или иного частного критерия, поэтому используется линейная свертка критериев нижнего уровня по числу переменных, входящих в данный режим.

В результате для данного *l*-го режима ($l=\overline{1,\mathrm{L}}$) критерий эффективности среднего уровня запишется как

$$\theta_l = \sum_{\mathrm{i}=1}^{\mathrm{M}l} \lambda_{l\mathrm{i}} \mathrm{I}_\mathrm{i}, \qquad (10)$$

где M*l* – число переменных в *l*-том режиме;

$\lambda_{l\mathrm{i}}$ – коэффициенты, учитывающие важность i-той переменной в *l*-том режиме.

Для определения значения коэффициентов $\lambda_{l\mathrm{i}}$ для i-тых критериев нижнего уровня предлагается использовать известные методы экспертных оценок [2]

Найденный таким образом критерий эффективности среднего уровня вида (10) оказывается весьма удобным для оценки действий оператора в *l*-м режиме, так как учитывает важность каждой переменной этого режима в их совокупности. Однако, с точки зрения его количественной оценки как в данном *l*-том режиме, так и среди других режимов критерий вида (10) необходимо привести к некоторому единому диапазону его значений. Наиболее удобным оказывается диапазон [0, 1], приведение к которому осуществляется по формуле:

$$\overline{\theta}_l = \frac{\theta_l}{\theta_{l\max}}, \qquad (11)$$

где $\theta_{l\max} = \sum_{\mathrm{i}=1}^{\mathrm{M}l} \lambda_{l\mathrm{i}} \left|\Delta \mathrm{x}_{\mathrm{i}\max}\right|$.

Если ввести нормированное уравнение вида $\sum_{\mathrm{i}=1}^{\mathrm{M}l} \lambda_{l\mathrm{i}} = 1$, то с учетом $|\Delta x_{i\max}|=1$ и $\theta_{l\max}=1$ тогда $\overline{\theta}_l = \theta_l$.

Поскольку существует определенный набор тестовых режимов, возникает необходимость в выборе соответствующего критерия для формирования результирующей оценки по целому ряду режимов. При этом для несмещенности и состоятельности комплексной оценки необходимо выполнение следующих условий.

1. Предельная относительная погрешность суммарной оценки должна лежать в пределах, определяемых значениями максимальной и минимальной погрешностей слагаемых.
2. Результирующая оценка должна определенным образом учитывать важность каждого режима или группы подобных режимов.

Таким условием удовлетворяет критерий вида:

$$\mathrm{J_k} = \frac{\sum_{l=1}^{\mathrm{N_k}} \left(\frac{1}{\theta_l}\right)^{\mathrm{p}_l - 1}}{\sum_{l=1}^{\mathrm{N_k}} \left(\frac{1}{\theta_l}\right)^{\mathrm{p}_l}}, \mathrm{p}_l \geq 1, \qquad (12)$$

значение которого определяется на основе ранее сформированных относительных оценок $\theta_l, (l=\overline{1,\mathrm{N_k}})$ по Nk режимам, а k определяет собой порядковый номер тестового режима.

С помощью выбора соответствующего значения pl можно произвести целенаправленное, с учетом качества выполнения того или иного режима, а также степени его важности, изменение значений критерия в диапазоне от 0 до 1. При этом если все слагаемые имеют одну и ту же (приблизительно одинаковую) предельную относительную погрешность, то и сумма имеет ту же (или приблизительно такую же) предельную относительную погрешность [3]. Иными словами, в этом случае точность суммы (в процентном выражении) не уступает точности слагаемых. Однако при значительном числе слагаемых, сумма, как правило, гораздо точнее слагаемых, так как в этом случае происходит взаимная компенсация погрешностей, поэтому истинная погрешность суммы лишь в исключительных случаях совпадает с предельной погрешностью или близка к ней.

Следовательно, основным фактором, определяющим изменения значений этой суммы в диапазоне 0÷1, является выбор соответствующего для каждого режима pl, что достигается следующим образом. Все Nk режимов данной

k-той учебной задачи объединяются по степени их важности, определенной ранжированием по методу экспертных оценок [4], в ряд групп, соответствующих близким значениям оценок. Например, для k-того тестового режима методом экспертных оценок получено следующее разбиение (см табл. 1).

Таблица 1. Пример разбиение режимов по группам

θ1, θ3, θ6 …	θ2, θ9, θ11 …	… $\theta_{N_k-1}, \theta_{N_k}$
1-я группа	j-я группа	R-я группа

Таким образом, Nk частных критериев $\theta_l, (l=\overline{1,N_k})$ разбиты на R групп по $S_j, (j=\overline{1,R})$ элементов в каждой группе разбиения. Известно [3], что общее число перестановок в этом случае будет равно:

$$P_0 = N_k! = (\sum_{j=1}^{R} S_j)!. \qquad (13)$$

Число перестановок, при которых хотя бы один из Nk частных режимов попадает из своей группы в чужую, что в свою очередь должно штрафоваться, определяется как:

$$P_B = (\sum_{j=1}^{R} S_j)! - \prod_{j=1}^{R} (S_j)!. \qquad (14)$$

Тогда доля этого числа перестановок от общего числа определяется следующим образом:

$$\Delta p = \frac{P_B}{P_0} = \frac{(\sum_{j=1}^{R} S_j)! - \prod_{j=1}^{R} (S_j)!}{(\sum_{j=1}^{R} S_j)!}, \qquad (15)$$

и может служить величиной дискретного шага штрафа pl. При этом, если для некоторого l-го частного режима, оценка по которому должна находиться в j-ой группе $(j=\overline{1,R})$, попадает в i-тую группу $(i=\overline{1,R})$, то соответствующее значение pl определяется согласно выражению:

$$\begin{cases} pl = 1 + (i - j)\Delta p & i > j; \\ pl = 1, & i \le j. \end{cases} \qquad (16)$$

Полученные таким образом количественные значения комплексного критерия (равно как и значения критерия оценки по режимам), должны быть интерпретированы в более приемлемой бальной системе оценок. Бальная система оценок состоит в том, что диапазон изменения какой-либо переменной, в данном случае критериев Jk или θl, разбиваются на ряд интервалов, каждому из которых присваивается оценка (балл). Чаще всего для этих целей используют интервальные шкалы. В этой, как и в других шкалах, соблюдается основное правило шкал, а именно, преобразование числовых значений при сохранении неизменными выполняемых ею функций. Интервальная шкала вводится с точностью до линейного преобразования:

$$\{x_i\} \to \{x_i'\} = ax_i + b \qquad (17)$$

и предполагает трансформации оценок, полученных на одной шкале, в оценки на другой шкале с помощью линейного преобразования.

4. Выводы

Предложенный подход носит универсальный характер и может быть использован для оценки деятельности операторов широкого класса эргодических систем.

Список литературы

1. Вдовин В. М и др Теория систем и системный анализ – М.. Изд Дашков и К. 2010-638с.
2. Бешелев С.Л, Гурвич Ф.Г. Математические методы экспертных оценок – М. Физматгиз 1980-263с
3. Семендяев Н. В. Справочник по высшей математике – М. Физматгиз.1981

УДК 004.724.4(045)

КУЛАКОВ Ю.А.,
КОГАН А.В.,
ПИРОГОВ А.А.

АЛГОРИТМ РАЗДЕЛЕНИЯ И СБОРКИ СЕКРЕТНОГО СООБЩЕНИЯ ДЛЯ МНОГОПУТЕВОЙ МАРШРУТИЗАЦИИ В БЕСПРОВОДНЫХ СЕТЯХ

Предложен алгоритм разделения и сборки секретного сообщения для многопутевой маршрутизации в мобильных сетях. Разбиение сообщения осуществляется на основе использования периодических функций типа y = cos(x) и уравнения «волны». Разработана программа и приведен пример моделирования процесса безопасной многопутевой передачи информации.

The algorithm of division and build a secret message for secure multipath routing. Partition is based on the use of periodic functions of y = cos (x) and the equation of the "wave". The program is an example of modeling and process secure multipath transmission.

1. Введение

В настоящее время актуальным вопросом является безопасность передачи информации в беспроводных сетях. Это связано с тем, что прослушивание передающей беспроводной среды не составляет особого труда. Кроме того, существующие методы повышения безопасности ориентированы в основном на сети фиксированной структуры. В то же время целый ряд беспроводных технологий позволяют обеспечивать высокую мобильность узлов, для которых известные методы защиты информации, используемые в сетях со статической структурой, требуют значительных накладных расходов, а возможно и не применимы вообще [1].

В данной работе предложен способ передачи информации на основе многопутевой маршрутизации с использованием процедуры разделения и сборки секретного сообщения.

2. Обзор существующих решений

До настоящего времени в большинстве случаев решения задачи разделения и сборки секретного сообщения ранее была предложена схема интерполяционных полиномов Лагранжа (схема разделения секрета Шамира). Данная схема позволяет создать (t, n)-пороговое разделение секрета для любых t, n. Основная идея схемы Шамира базируется на том, что полином степени t – 1 может быть однозначно восстановлен по его значениям в t различных точках.

Основной недостаток непосредственного применения схемы – отсутствие способа верификации отдельных ключей, передаваемых участникам, а также возможности проверки достоверности любого их поднабора размера l (для предотвращения подмены отдельных частей секрета).

3. Описание алгоритма

Предлагаемый в настоящей работе алгоритм разделения и сборки секретного сообщения характеризуется тем, что вычисления производятся с определённой точностью, на основе периодических тригонометрических функций. Данные функции имеют постоянную амплитуду, определены и непрерывны на всем промежутке $x \in (-\infty, +\infty)$. Особенность алгоритма заключается в том, что при увеличении точности вычислений период гаммирования может достигать сколь угодно большого значения.

При реализации данного алгоритма использовались периодическая функция y = cos(x) и уравнение «волны» [2]. Особенность данной периодической функции в том, что ее период выражен иррациональным числом, а значению, например, y = 0,5 соответствует бесконечное количество значений x. То есть, при y = 0,5 x принимает значения:

$$\frac{\pi}{3}; \frac{7\pi}{3}; \frac{13\pi}{3} \ \ldots$$

и т. д. до бесконечности, как положительных, так и отрицательных значений.

Применяя уравнение волны y = cos (x + N * Δx), где:

N – целое число, в данном случае определяющее порядковый номер шифруемого символа (0-256);

Δx – приращение функции, задаваемое в секретном ключе (любое число);

x – значение, взятое из открытого текста (0-256).

Отсюда следует, что при y = 0,5 * x может принимать любое значение от $-\infty$ до $+\infty$ (рис.1).

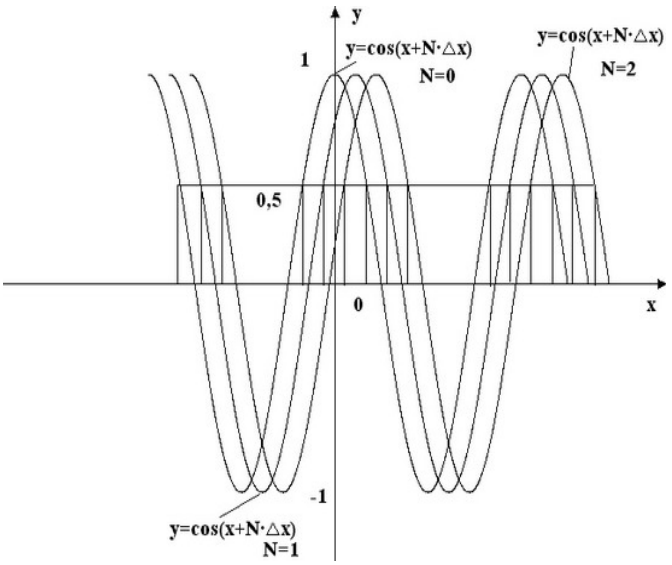


Рис. 1. График функции y = cos (x).

Пример реализации:

По координатной оси x расставляются символы из открытого текста в любом порядке. Каждому символу соответствует свой порядковый номер, например, от 1 до 256. По оси y расставляются символы, используемые в шифротексте в любом таком же или другом порядке. Им так же присваиваются порядковые номера, например от 1 до 256. Три линейные функции описываются как: Y_1, Y_2, Y_3 (рис. 2).

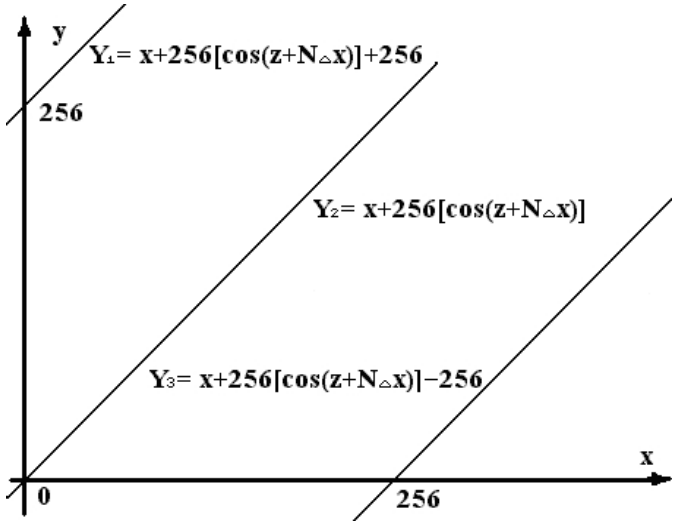


Рис. 3. Линейные функции Y_1, Y_2, Y_3.

$Y_1 = x + 256 * [\cos(z + N * \Delta x)] + 256;$

$Y_2 = x + 256 * [\cos(z + N * \Delta x)];$

$Y_3 = x + 256 * [\cos(z + N * \Delta x)] – 256;$

где:

x – порядковый номер символа из открытого текста;

z – любое число (-∞ до +∞);

N – номер по счету шифруемого символа из открытого текста;

Δx – любое число (-∞ ÷ +∞);

Для примера зашифруем букву А. А – открытый текст. В данном примере секретными являются числа Δx и z. Все остальные параметры не являются секретом. Кроме того, количество секретных параметров можно сделать бесконечное множество.

Знак А занимает промежуток (0÷1) по оси x, знак Б – (1÷2), В – (2÷3) и т.д.. Имея три формулы, подставляем значение 0,5 – середину промежутка (0÷1) – буква А (для буквы Б это значение соответствует 1,5, для В – 2,5 и т.д.).

Пусть, z = 0. Шифруется первый по счету знак из открытого текста, тогда N = 1. Δx = 32.

Подставив цифры, получим формулы следующего вида:

$Y_1 = 0,5 + 256 * [\cos(0 + 1 * 32)] + 256 = 437,6;$

$Y_2 = 0,5 + 256 * [\cos(0 + 1 * 32)] = 217,6;$

$Y_3 = 0,5 + 256 * [\cos(0 + 1 * 32)] - 256 = 38,39;$

Из трех значений Y_1, Y_2, Y_3 выбираем Y_2, так как значение Y_2 попало в промежуток от 0 до 256. И округляем значение Y_2 до большего целого $Y_2 = 218$.

В итоге, открытый текст 0,5 был зашифрован и получил значение 218.

Для расшифровки применяются те же формулы, подставляя уже известное значение 218. Из каждого известного значения необходимо вычесть 0,5 для того, что бы в формулу подставлялось значение из середины отрезка, которому принадлежит данный знак:

$x_1 = 217,5 - 256 * [\cos(0 + 1 * 32)] – 256 = - 255,6;$

$x_2 = 217,5 - 256 * [\cos(0 + 1 * 32)] = 0,4;$

$x_3 = 217,5 - 256 * [\cos(0 + 1 * 32)] + 256 = 265,4.$

Из трех значений Y_1, Y_2, Y_3 выбираем Y_2, так как значение Y_2 попало в промежуток (0÷1).

Преимущества алгоритма:

Данный алгоритм прост в реализации. Особенностью алгоритма является то, что шифротекст представлен в виде иррациональных чисел. В самом процессе шифрования учувствуют промежутки, в которые попадают эти иррациональные числа. Произвести аналитический взлом шифра возможно только точно зная число, а оно иррациональное, то есть бесконечно, и его невозможно вычислить. Соответственно при расшифровании уже участвуют совершенно случайные числа, а не те, которые получили в процессе шифрования.

4. Моделирование процесса безопасной многопутевой маршрутизации

В рамках данной работы была разработана программа моделирования процесса безопасной многопутевой передачи информации.

На первом шаге моделирования (Такт: 0) с помощью модифицированного алгоритма Дейкстры осуществляется поиск множества непересекающихся путей, для каждого из которых вычисляется значение SLi, характеризующее надежность доставки информации по каждому из выбранных путей. Пути и степень их надежности, а также опции вершин отображаются в сервисном окне (рис. 3), при данной топологии сети формируется 3 маршрута с различной надежностью.

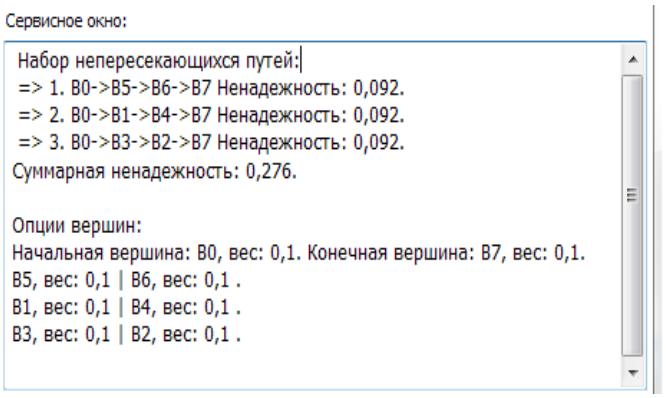


Рис. 3. Сервисное окно программы

На втором шаге осуществляется разбиение исходного сообщения на пары символов (с дополнением символов «0» в начало сообщения в случае несоответствия длины сообщения с требованиями алгоритмов разбиения/сборки сообщения). Далее полученные пары символов преобразуются в соответствии с предложенным алгоритмом (рис 4.). В данном примере исходное сообщение FB7EC5 разбивается на три пакета и шифруется с использованием периодической функции y = cos (x + N * Δx) (рис. 5). Необходимые значения параметров z и Δx задаются в настройках программы, z = 1, Δx = 30. Исходные части сообщения: FB7EC5. В итоге функции шифрования будут иметь следующий вид:

$Y_1 = (251 + 0,5) + 256 * (\cos(1 + 1 * 30))$;
$Y_1 = 485,67$ [+ / - 256] => 229,67;
После округления $Y_1 = 230$;
$Y_2 = (126 + 0,5) + 256 * (\cos(1 + 2 * 30))$;
$Y_2 = 60,43$ [+ / - 256] => 60,43;
После округления Y2 = 61;
$Y_3 = (197 + 0,5) + 256 * (\cos(1 + 3 * 30))$;
$Y_3 = - 57,06$ [+ / - 256] => 198,94;
После округления $Y_3 = 199$.

После шифрования исходного сообщения формируются соответствующие пакеты, которые будут отправлены узлу-адресату по разным маршрутам, которые были найдены ранее:

T=0 Пакет №0 с содержимым:
1|230 сформирован.
T=0 Пакет №1 с содержимым:
2|61 сформирован.
T=0 Пакет №2 с содержимым:
3|199 сформирован.

После того как узлу-получателю пришли все части исходного сообщения, происходит обратная сборка их в исходное сообщение. В итоге функции расшифровании будут иметь следующий вид:

$x_1 = (230 - 0,5) – 256 * (\cos(1 + 1 * 30))$;
$x_1 = - 4,67$ [+ / - 256] => 251,33;
После округления $x_1 = 251$;
$x_2 = (61 - 0,5) – 256 * (\cos(1 + 2 * 30))$;
$x_2 = 126,57$ [+ / - 256] => 126,57;
После округления $x_2 = 126$;
$x_3 = (199 - 0,5) – 256 * (\cos(1 + 3 * 30))$;
$x_3 = 453,06$ [+ / - 256] => 197,06;
После округления $x_3 = 197$;

Исходное сообщение: FB7EC5.

Выводы

Алгоритм разбиения и сборки исходного сообщения на основе тригонометрических функций cos и sin удобен и просто в реализации. Особенность этих функций в том, что они непрерывны и определены на всем промежутке от минус бесконечности до плюс бесконечности на оси x. А на оси y они принимают значения от -1 до 1, причем и на том и на другом промежутке они могут принимать любые значения в этих промежутках. Над этими функциями возможно производить фактически любые математические действия.

Отличительная особенность алгоритма – при увеличении точности вычислений период гаммирования может достигнуть сколь угодно большого значения.

В отличии от предложенной ранее модифицированной схемы Шамира, данный алгоритм позволяет участникам протокола еще до момента восстановления полного сообщения с уверенностью сказать, является ли их часть подлинной.

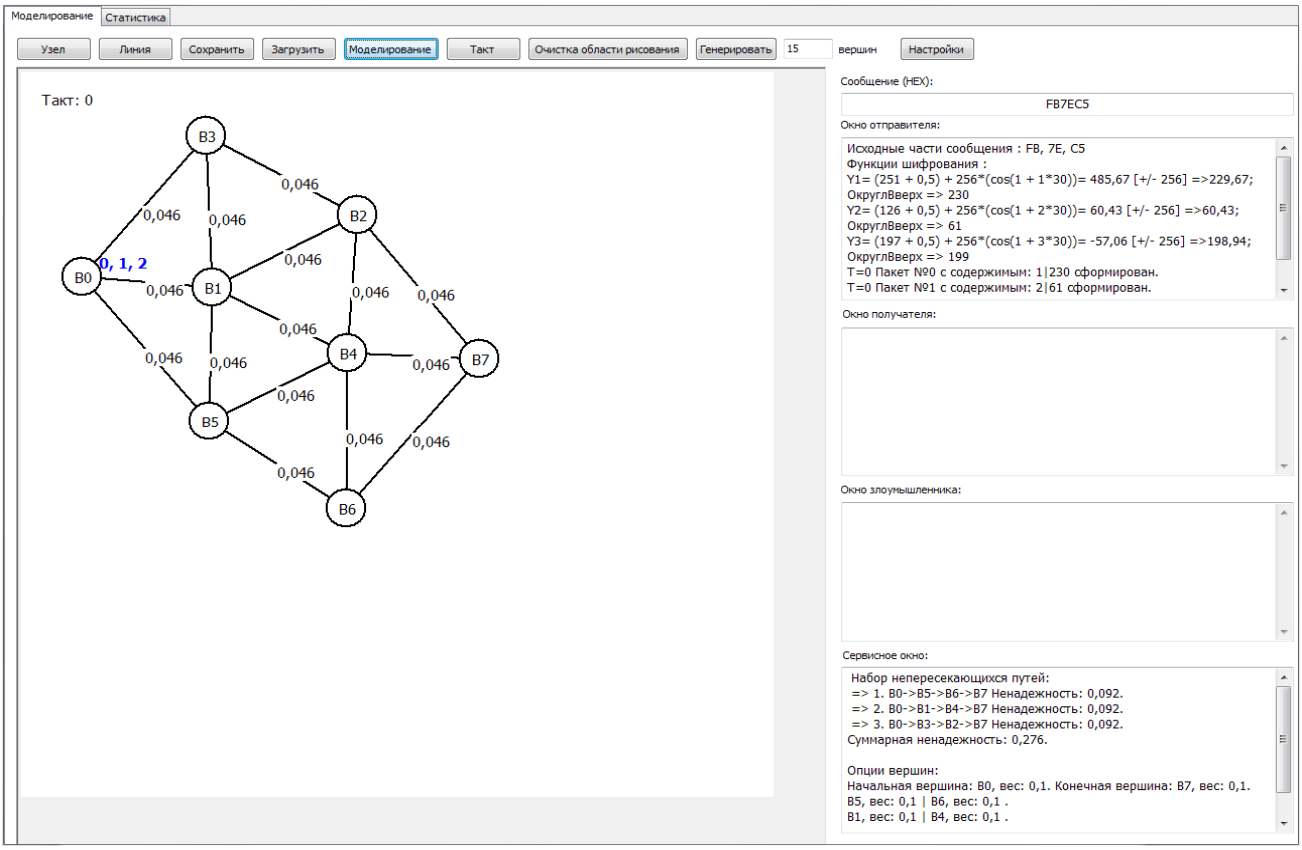


Рис. 4. Поиск набора непересекающихся путей, разбиение исходного сообщения на части и шифрование

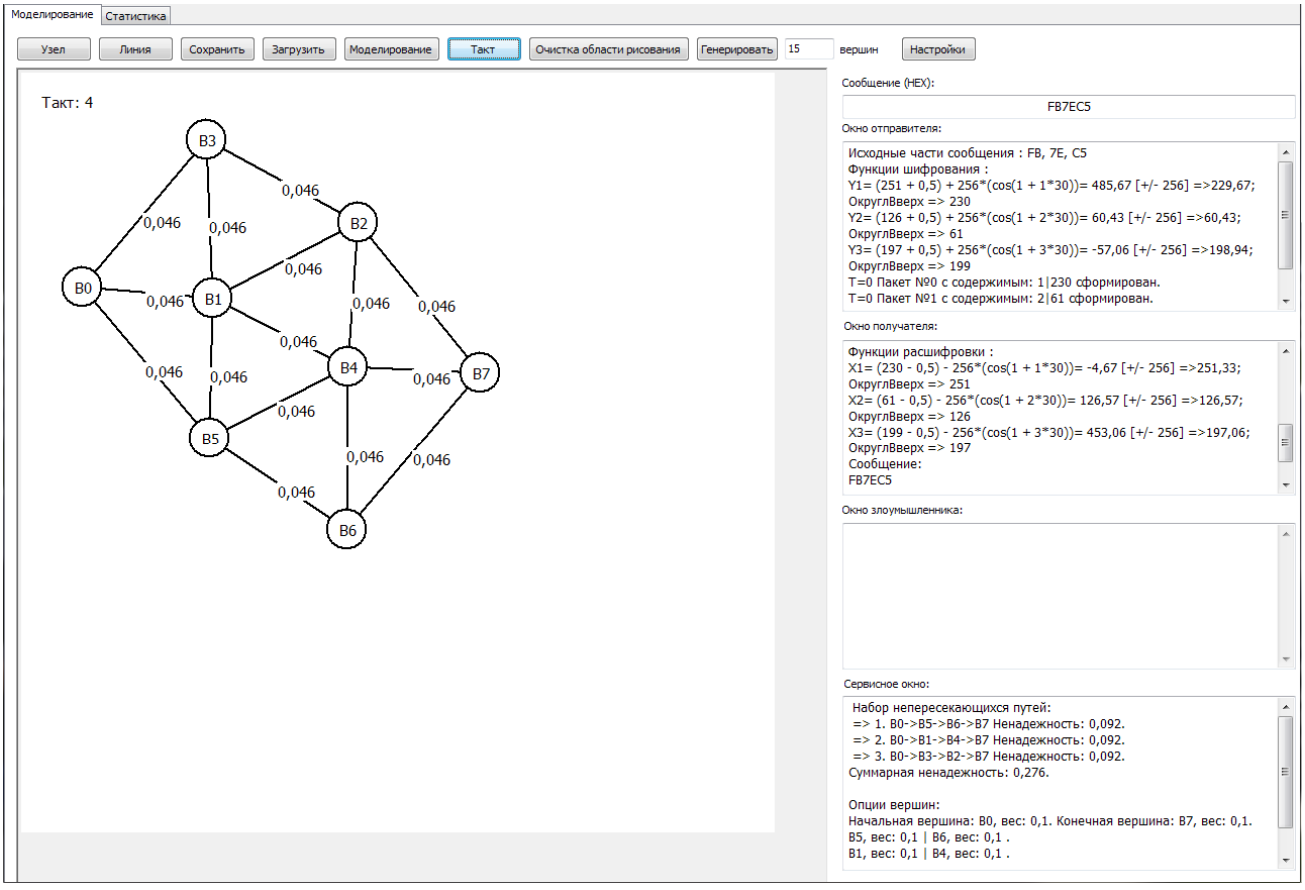


Рис. 5. Сборка частей исходного сообщения и дешифрование

А для того, что бы прочитать сообщение полностью, злоумышленнику необходимо не просто перехватить все части исходного сообщения, но и так же знать значения параметров z и Δx, что осуществить весьма не просто.

Список литературы

1. Кулаков Ю.А. Разработка и моделирование процесса безопасной многопутевой передачи информации в мобильных сетях / Ю.А. Кулаков, А.В. Коган, А.А. Пирогов // Вісник Национального техн. Ун -ту України «КПІ». Інформатика, управління та обчислювальна техніка: зб. наук. праць. – К.: Век+, 2011. – № 54. – С. 145-149.
2. Сизов В. П. «Новый алгоритм шифрования». // Сайт ITSec.Ru – 2007. [Электронный ресурс]. URL: http://www.itsec.ru/articles2/Inf_security/novy-alg-shifrov . – название с экрана.
3. Кулаков Ю.А. Повышение уровня безопасности передачи информации в мобильных сетях / Кулаков Ю.А., Максименко Е.В., Рущак О.А.// Вісник Національного техн. ун -ту України "КПИ": Інформатика, управління та обчислювальна техніка. – К.: ТОВ "ВЕК+", 2007. – Вип. 47. – С.297-304.
4. Кулаков Ю. А. Многопутевая маршрутизация в беспроводных сетях./ Ю. А. Кулаков, А. В. Левчук // Проблеми інформатизації та управління: Зб. наук. пр. – К.: Вид-во Нац. авіац. ун-ту «НАУ-друк», Вып. 4 (26), 2010. – С.142-147.
5. Marti S. Mitigating routing misbehavior in mobile ad hoc networks/ Marti S., Giuli T., Lai K., Baker M. // The 6th annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobi-Com'OO) – 2000 – Boston(MA, USA) – P.255-265.

УДК 519.854.2

ПАВЛОВ А.А.,
СПЕРКАЧ М.О.,
ХАЛУС Е.А.

СУБОПТИМАЛЬНЫЙ ПОЛИНОМИАЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ ОДНОГО КЛАССА МНОГОЭТАПНЫХ СЕТЕВЫХ ЗАДАЧ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ

В статье рассматривается один класс многоэтапной сетевой задачи календарного планирования. Для произвольно заданных конечных директивных сроков (директивных сроков выпуска готовых изделий) необходимо получить допустимое расписание с максимально поздним по времени запуском технологического процесса. При дополнительном ограничении на структуру допустимого расписания излагается точный полиномиальный алгоритм решения сформулированной задачи. Приводится теоретическое и практическое обоснование целесообразности введения дополнительного ограничения на структуру искомого расписания.

In this paper we consider one class of multistage network calendar scheduling problem. For arbitrarily given finite due dates (due dates for finished products' release) it is necessary to obtain a feasible schedule with the latest startup time of the technological process. With the additional constraint on the structure of a feasible schedule the exact polynomial algorithm for solving the formulated problem was stated. The theoretical and practical grounding for expedience of the introduction of additional restrictions on the structure of the desirable schedule was presented.

Введение

В статье рассматривается один класс многоэтапной сетевой задачи календарного планирования. Для произвольно заданных конечных директивных сроков (директивных сроков выпуска готовых изделий) необходимо получить допустимое расписание с максимально поздним по времени запуском технологического процесса. При дополнительном ограничении на структуру допустимого расписания излагается точный полиномиальный алгоритм решения сформулированной задачи. Приводится теоретическое и практическое обоснование целесообразности введения дополнительного ограничения на структуру искомого расписания.

Как показано в [2] первый уровень трехуровневой модели планирования и принятия решений изложенной в [1] (в терминологии [1] – третий уровень модели на котором строится полный план выполнения работ с привязкой к ресурсам – точное планирование) может быть представлен как многоэтапная сетевая задача календарного планирования. В [2] приведены пять элементов из комбинации которых конструируется многоэтапная сетевая модель календарного планирования.

Предметом исследования является следующий класс сетевых моделей:

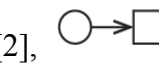
1. Сеть состоит из комбинации элементов (тип 1-4 [2]) и является ориентированным ациклическим графом.

Элементам (тип 2,3 [2]) соответствует в сети оборудование, которое в процессе непрерывной работы в произвольном порядке должны выполнить множество работ без прерывания. В произвольный момент времени может выполняться только одна работа. Элементы отличаются между собой типами связи в сети работ, выполненных оборудованием. Сеть может содержать элемент, являющийся комбинацией элементов (тип 2,3 [2]) (по структуре связей выполненных работ в сети). Элементам (тип 1,4 [2]) соответствует оборудование, выполняющее только одну работу.

2. Сама сеть представляет собой ориентированный граф с двумя типами вершин – *O* – обозначает работу готовую к выполнению, за ней обязательно следует вершина □ – обозначение оборудования на котором выполняется непосредственно предшествующая ему работа.

3. Все ориентированные стрелки направлены слева направо.

4. Всю сеть можно разбить на три части.

4.1 Конечная часть заканчивается элементами вида 1 [2], 

где T_{ik} – время выполнения последней операции (работы) по *i*-ому изделию (*i*-ой серии изделий). В допустимом расписании $T_{ik} \le T_i, i = \overline{1, I}$, где T_i – директивный срок выполнения i-ого изделия (*i*-ой серии изделий).

Примечание. Последней может быть работа, выполненная на элементе (тип 2,3 [2]).

Представляет собой произвольную комбинацию элементов 1-4, которая удовлетворяет следующему условию:

$T_i, i = \overline{1,I}$ однозначно задают для элементов (типа 2, 3[2]) директивные сроки выполнения работ, которые в произвольном порядке могут выполняться на элементах (типа 2,3) [2], у каждого из которых в конечной части сети, нет непосредственно предшествующих элементов (типа 2,3 [2]) (без учета элементов (типа 1,4 [3])).

Примечание. У элемента 1 [2], Т может быть промежуточным сроком выполнения работы (не окончание выполнения изделия (группы изделий))

4.2 Промежуточная часть.

Это произвольная сетевая комбинация элементов (тип 1-4 [2]), удовлетворяющая условию: найденные ниже приведенным алгоритмом наиболее поздние моменты готовности выполнения работ на элементах (типа 2,3 [2]) однозначно задают директивные сроки выполнения работ для непосредственно предшествующих (без учета элементов (типа 1,4) [2]) элементов (типа 2,3 [2]). Непосредственное предшествование означает, что между непосредственно предшествующим элементом (типа 2,3 [2]) и данным элементом (типа 2,3 [2]), существует ориентированный путь содержащий только вершины соответствующие элементам (тип 1,4 [2]), с учетом примечания к п.4.1.

В промежуточной части сети могут находиться элементы (типа 2,3 [2]) непосредственно предшествующие более чем одному элементу (тип 2,3 [2]).

4.3 Начальная часть сети является произвольной комбинацией элементов (тип 1-4 [2] удовлетворяющая следующим условиям:

a) имеются все вершины сети вида $\boxed{1_j}$, $j = \overline{1,p}$, где цифра 1 означает что $\boxed{1_j}$ вершине не предшествует ни одна вершина второго типа (смотри пункт 2).

b) произвольному элементу (тип 2,3 [2]) непосредственно не предшествует элемент (тип 2,3 [2]).

c) рассмотрим в сети полную группу элементов (тип 2,3 [2]). Тогда найденные ниже приведенным алгоритмом наиболее поздние моменты начала выполнения работ элементов этой группы однозначно задают моменты начала выполнения работ всех элементов типа $\boxed{1_j}$, $j = \overline{1,p}$.

Примечание.

1) Сеть точно отражает реальный технологический процесс.

2) Так как выполнение работ может быть разделено во времени, в сети одно и тоже физически существующее оборудование может быть представлено различными вершинами второго типа. Так, например элементы (тип 2,3 [2]) работают непрерывно.

Реально физически существующее оборудование непрерывно выполняет работы для различных групп операций в соответствии с технологическим процессом (время их выполнения на непрерывно работающем оборудовании разнесено во времени), тогда в сети одному и тому же оборудованию соответствует несколько элементов (тип 2,3 [2]).

Приведем на рисунке 1 пример сети, принадлежащей сформулированному классу.

Необходимо решить следующую задачу: найти допустимое расписание выполнения работ ($\forall T_{ik} \le T_i, i = \overline{1,I}$), которому бы соответствовал $\max(\min\limits_{j=\overline{1,p}} T_{1j})$ либо $\max \sum\limits_{j=1}^{p} T_{1j}$, где T_{1j} момент запуска оборудования соответствующего вершине $\boxed{1_j}$. Для примера на рисунке 1 функционалы имеют вид: $\max(\min\limits_{j=\overline{1,18}} T_{1j})$ либо $\max \sum\limits_{j=1}^{18} T_{1j}$.

Для решения поставленной задачи наложим и обоснуем дополнительное ограничение на искомое допустимое расписание. Для этого рассмотрим свойства следующей одноэтапной задачи теории расписаний [3], являющейся моделью элементов (тип 2,3 [2]).

Постановка задачи

Задано множество независимых заданий $J = \{j_1, j_2, ..., j_n\}$, каждое из которых состоит из одной операции. Для каждого задания j известны длительность выполнения p_j и директивный срок выполнения d_j. Прерывания не допускаются. Задания поступают в систему одновременно. Необходимо:

1) найти максимально поздний момент r начала выполнения заданий в допустимом расписании;

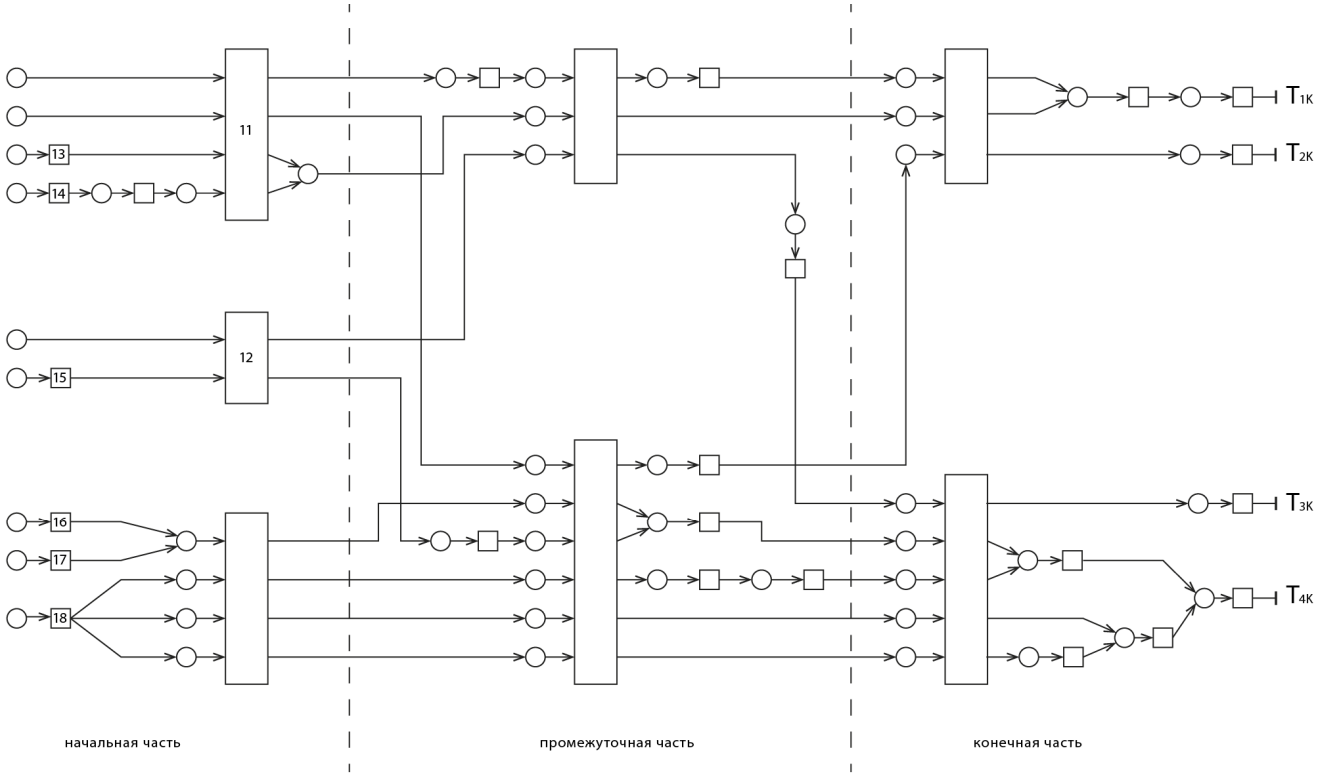


Рис. 1. Пример сети

2) минимизировать суммарное опережение выполнения заданий относительно директивного срока при условии допустимости расписания.

Определение 1. Если для заданного момента выполнения заданий объективно существует расписание, в котором удовлетворены все директивные сроки, то такое расписание называется допустимым.

Теорема 1. [3] Пусть существует произвольная допустимая последовательность выполнения заданий одним прибором $\sigma_1 = \{j_{[1]}, j_{[2]}, ..., j_{[n]}\}$, где $j_{[k]}$ обозначает задание на позиции k в текущей последовательности. Последовательность на этом множестве заданий, упорядоченная по неубыванию значений директивных сроков, также допустима.

Следствие. Если расписание, упорядоченное по неубыванию директивных сроков, является недопустимым, то допустимого расписания не существует.

Алгоритм А [3]

4) Строим, в соответствии с теоремой 1, расписание J = {1, 2, … n}, в котором задания упорядочены по неубыванию директивных сроков.

5) Определяем момент запуска выполнения заданий r, при котором в расписании J хотя бы одно задание j_k будет запаздывающим:

$$\exists k : d_k - r < \sum_{i=1}^{k} p_i .$$

6) Для момента запуска r находим задание j_l с максимальным запаздыванием относительного директивного срока и определеляем новый момент начала выполнения заданий $r_{\max} = r - (C_l - d_l)$, где $C_l - d_l$ – запаздывание по заданию j_l.

Теорема 4.[3] Допустимое расписание, построенное по неубыванию директивных сроков, является оптимальным по критерию минимизации суммарного опережения при выполнении условий:
$d_1 \le d_2 \le \ldots \le d_n, \quad p_1 \ge p_2 \ge \ldots \ge p_n$. Суммарное опережение минимально при моменте запуска $r_{\max}$.

Примечание. Если в допустимом расписании $\sigma = \{1, 2, ..., n\}$ с максимальным моментом начала выполнения заданий для произвольного C_k существуют задания $i \in \overline{1,k-1}$, для которых $C_k \le d_i$, то при упорядочении этих заданий по невозрастанию их длительностей получаем допустимое расписание с максимальным моментом начала выполнения заданий, в котором уменьшается суммарное опережение (в предположении, что в σ существуют как минимум две такие работы различной длительности, не упорядоченные по невозрастанию их значений).

$$\sum_{i=1}^{n-1}\left\lfloor\left(d_i - r_{\max} - \sum_{j=1}^{i} p_j\right)\Big/ \min_{j=i+1,n} p_j\right\rfloor\Delta_i, \quad \Delta_i = \max\left(0, \max_{j=i+1,n} p_j - p_i\right), \qquad (1)$$

Теорема 5. [3] В общем случае верхняя оценка отклонения суммарного опережения допустимого расписания, полученного упорядочением по неубыванию директивных сроков, от оптимального расписания для того же момента запуска по критерию минимизации суммарного опережения, определяется выражением:

где $\lfloor a \rfloor$ обозначает ближайшее снизу целое от a.

С учетом выше приведенных свойств одноэтапной задачи календарного планирования (модели элементов (тип 2,3 [2])) на допустимое расписание многоэтапной сетевой задачи календарного планирования накладывается следующее дополнительное ограничение (Д ограничение):

1. Расписание выполнения работ на элементах (тип 2,3 [2]) всегда является допустимым при известных (произвольно назначенных, найденных алгоритмом) директивных сроках выполнения работ.

2. Моментом запуска непрерывной работы элемента (тип 2,3 [2]) является $r_{\max}$, определяемый Алгоритмом А.

3. Расписание работы элементов (тип 2,3 [2]) определяются расписанием (1,2,…,n) (Теорема 1) с учетом упорядочения работ (смотри примечание к теореме 4, которое реализуется последовательно от меньшего к большему k, для $\forall C_k, k = \overline{2,n}$).

4. Самые поздние сроки моментов готовности работ на элементе (тип 2, 3 [2]), определяемые расписанием их вычисления, однозначно задают директивные сроки выполнения работ на элементах (тип 2, 3 [2]) непосредственно ему предшествующих.

Алгоритм построения субоптимального полиномиального алгоритма решения многоэтапной сетевой задачи календарного планирования

1. По моментам времени $T_i, i = \overline{1,I}$, I – количество изделий (серий изделий) в силу свойств сети однозначно определяются все директивные сроки выполнения работ всех элементов (типа 2,3 [2]) конечной части сети.

2. Расписание работ всех элементов (типа 2,3 [2]) конечной части сети находятся в соответствии с Д ограничением.

3. По построенным расписаниям находятся все самые поздние моменты готовности на выполнения работ непосредственно предшествующие элементам (типа 2,3 [2]) конечной части сети.

4. В силу свойств сети по найденным моментам готовности однозначно определяются все директивные сроки выполнения работ элементов (типа 2,3 [2]) непосредственно предшествующих (связанных хотя бы одним направленным путем не содержащих элементов (типа 2,3 [2])).

5. Расписание элементов (типа 2,3 [2]) у которых определены все директивные сроки определяется Д ограничением.

6. Пункты 4,5 повторяются (не более чем общее количество элементов (типа 2,3 [2]) промежуточной и начальной части сети) до получения всех самых поздних моментов готовности на выполнение работ элементов (типа 2,3 [2]) начальной части сети.

7. В силу свойств сети по этим моментам времени однозначно находятся моменты времени выполнения всех начальных работ. Конец.

Допустимое расписание построено.

Свойство полученного расписания

1. Алгоритм построения допустимого расписания является полиномиальным.

2. В силу логики построения алгоритма с учетом Д ограничения каждый из полученных моментов начала выполнения работ элементами $\boxed{1_j}$, $j = \overline{1,k}$ k – количество начальных вершин сети второго типа, являются максимально поздними , что позволяет определить полученное допустимое расписание как субоптимальное по обоим критериям $\max(\min_{j=1,p} T_{1j})$, $\max\sum_{j=1}^{p} T_{1j}$, (оптимальное с учетом выполнения Д ограничения).

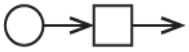
Примечание. Свойство 2 гарантированно имеет место, когда для всех элементов (типа 2,3 [2]) сетевой модели все работы назначенные на выполнение одним элементом (типа 2,3 [2])

имеют различные длительности. Действительно, в этом случае Д ограничение приводит к единственному расписанию для каждого элемента (типа 2,3 [2]).

3. Расписания всех элементов (типа 2,3 [2]) начальной и промежуточной части с учетом назначенных алгоритмов директивных сроков, обладают свойствами вытекающими из теорем (1-5) [3].

4. Свойства полученного субоптимального расписания относительно конечных директивных сроков $T_i, i = \overline{1,I}$ отвечает всем свойствам, вытекающих из теорем (1-5) [3].

Действительно, суммарное опережение построенного расписания является суммой суммарных опережений расписаний всех элементов (типа 2,3[2]) конечной части сети, директивные сроки которых однозначно определяются по $T_i, i = \overline{1,I}$, а сами расписания построены в соответствии с Д ограничением.

Примечание. Эффективность предложенного расписания полностью определяется адекватностью предложенной сетевой модели реальному технологическому процессу и физически существующему оборудованию. Если в сети элементы типа  однозначно задаются технологическим процессом и физически существующим оборудованием, то анализ полученного расписания может показать, что некоторые элементы (тип 2,3[2]) во время непрерывной работы фактически выполняют разнесенные во времени различные группы работ. Тогда необходимо изменить модель сети, введя дополнительные виртуальные элементы (типа 2,3[2]), соответствующие этому оборудованию и заново решить задачу.

Выводы

Сформулирован класс многоэтапных сетевых задач календарного планирования, для которого приведен и обоснован субоптимальный алгоритм, исследованы теоретические свойства полученного решения.

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. Павлов А.А., Мисюра Е.Б., Халус Е.А., Сперкач М.О., Аракелян Г.А. Результирующая формализация первого уровня трехуровневой модели оперативного планирования и принятия решений по критерию минимизации суммарного опережения директивных сроков // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. К.: «ВЕК+», 2012. « No.56.» 2 С. (подано до друку)
3. Павлов А.А., Мисюра Е.Б., Халус Е.А. Полиномиальный алгоритм получения допустимого расписания с максимально поздним моментом начала выполнения одним прибором независимых заданий произвольной длительности с разными директивными сроками, при котором все задания остаются незапаздывающими // Інформаційні технології як інноваційний шлях розвитку України у XXI столітті: Матеріали I Міжнародній науково-практичній конференції молодих науковців 06-08 грудня 2012 р. – Ужгород: Закарпатський державний університет, 2012. – 7 С.

УДК 004.75

БУЛАХ Б.В.,
ЧЕКАЛЮК В.В.,
ЛАДОГУБЕЦ В.В.,
ФИНОГЕНОВ А.Д.

МОДИФИКАЦИЯ СПОСОБА УЧЕТА БАЛАНСА ЗАГРУЗКИ ПРИ ВЫПОЛНЕНИИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ НА ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ

В статье рассматриваются вопросы адаптации параллельных программ, использующих библиотеку PVM, для запуска на кластере НТУУ «КПИ» через планировщик PBS. Разработана модификация способа баланса загрузки, учитывающая архитектуру вычислительных узлов кластера. Даны рекомендации по организации блока анализа конфигурации PVM для параллельных программ.

The article deals with the adapting of parallel PVM programs to run on a cluster of NTUU "KPI" with PBS scheduler. A modification of the load balancing mechanism that takes into account the architecture of computing nodes was developed. Recommendations on the organization of the PVM configuration analysis unit for parallel programs were provided.

1. Введение

Библиотека PVM [1], как средство поддержки параллельных вычислений, позволяет использовать параллельные вычисления на наиболее трудоемких этапах решения задач. Наличие разнообразной техники различной архитектуры и производительности, работающей под управлением различных операционных систем, обусловили популярность использования библиотеки PVM в научной среде при объединении гетерогенных вычислительных средств в единую виртуальную многопроцессорную машину.

В статье [2] рассматривались вопросы разработки параллельных алгоритмов в рамках существующих пакетов проектирования на примере адаптации одного из методов блока параметрической оптимизации – метода случайного поиска с уменьшением интервала поиска (СПУИП). В качестве многопроцессорной системы рассматривались три персональных компьютера, объединенных локальной сетью.

С появлением в НТУУ «КПИ» центра суперкомпьютерных вычислений [3], была проведена модификация пакета NetALLTED [4] и сопутствующей параллельной реализации метода СПУИП [5]. Однако данная реализация рассчитана на запуск в составе предварительно настроенной виртуальной машины. При этом количество slave-программ задавалось статически, что было оправдано в виду однотипности узлов.

В случае запуска параллельных программ с использованием планировщика задач PBS, установленном на кластере НТУУ «КПИ», возникает ряд технических особенностей, которые не позволяют напрямую запускать параллельные программы, использующие библиотеку PVM.

2. Постановка задачи

Запуск программ с использованием планировщика PBS, по сравнению с «ручной настройкой» виртуальной машины, отличается тем, что:
– отсутствует возможность интерактивной работы в консоли PVM;
– заранее неизвестны имена узлов, которые будут предоставлены для решения задачи;
– узлы кластера отличаются по архитектуре и производительности процессоров, а также по количеству доступной оперативной памяти;
– при указании в запросе только количества требуемых ядер отсутствует информация о количестве ядер, выделяемых на узле.

Кроме того, структуры данных конфигурации виртуальной машины хранят только сетевые имена хостов (в качестве хостов рассматриваются машины или узлы, которые имеют IP-адрес), их архитектуру, относительную скорость и идентификатор демона pvmd на данном хосте. Эти данные не позволяют на основании лишь конфигурации виртуальной машины определить количество ядер, выделенных на хосте для решения задачи и запуск соответствующего количества slave-программ.

Необходимость запуска параллельных модулей обусловлена функциональностью междисциплинарного комплекса опти-мального математического моделирования в грид-среде с автоматическим формированием и решением уравнений соответствующих математических моделей (НТР № ДП / 310 – 2011 от 26 июля 2011 г., номер государственной регистрации 0111U005434).

Целью статьи является решение вопросов, связанных с запуском и настройкой PVM с использованием планировщика задач, а также модификация алгоритма баланса загрузки и запуска slave-программ с учетом количества выделенных ядер на узле.

3. Конфигурирование среды выполнения

При описании задачи для планировщика наиболее эффективно указывать лишь количество необходимых ядер, т.к. это увеличивает количество возможных конфигураций и уменьшает время простоя задачи в очереди.

Положим, что для решения задачи необходимо p ядер. Планировщик задач может выделить их как на одном узле, так и на нескольких, в зависимости от текущей загруженности системы. Так для кластера НТУУ «КПИ» при запросе p=10 был получен следующий перечень узлов, хранящийся в файле, указанном в переменной окружения `PBS_NODEFILE`:

```
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n004.kpi
n003.kpi
n003.kpi
```

Таким образом, планировщик выделил 8 ядер на узле `n004.kpi` и 2 на узле `n003.kpi`.

Каждая строка конфигурационного файла PVM выглядит следующим образом:

```
host <id=n> <sp=value>
```

где `id` – идентификатор (уникальный номер) хоста, `sp` – относительная вычислительная мощность хоста по сравнению с другими хостами в PVM, которая может принимать значение от 1 до 10^6. По умолчанию `sp=1000`. Используя дополнительные ключи можно указать логин на удаленном хосте (если он отличается от значения логина на текущем), месторасположение демона PVM и исполняемых файлов (slave), если пути отличаются от принятых по умолчанию и т.д.

Идентификатор `id` используется для формирования уникальных имен временных файлов демона PVM на каждом хосте в случае общей файловой системы. Номера могут задаваться произвольно, но должны быть уникальными в пределах текущей конфигурации.

При формировании файла конфигурации PVM сохранить информацию о количестве ядер, выделенных на узле, можно, используя часть цифр, отводимых для хранения относительной вычислительной мощности хоста. Т.к. на узле в ближайшее время не ожидается больше 99 ядер, а разброс вычислительной мощности между узлами значительно меньше, чем 1000 раз даже с учетом накладных расходов на пересылку данных, то формат можно представить следующим образом:

$$sp_i = sp_{Ri} \cdot 100 + nslaves_i,\ i = 1(1)nhost$$

где sp_i – относительная вычислительная мощность i-го узла; sp_{Ri} – предварительно рассчитанная относительная мощность i-го узла, который может входить в состав PVM из числа узлов кластера; $nslaves_i$ – количество ядер, выделенных на узле для данной задачи; `nhost` – количество выделенных узлов для данной задачи.

Параметры $nslaves_i$ и `nhost` определяются на основании анализа данных переменной окружения `PBS_NODEFILE`.

sp_{Ri} – можно определить с помощью предварительного решения тестовых задач для каждой архитектуры узла:

$$sp_{Ri} = \frac{T_{Rk}}{\max(T_{Rk})} \cdot 1000$$

где T_{Rk} – время решения тестовой задачи на k-й архитектуре узла.

В составе кластера НТУУ «КПИ» присутствуют 2 типа узлов, доступных локальному пользователю:

– узлы (n001-n044) с 2-мя четырехъядерными процессорами Intel Xeon E5440 @ 2.83ГГц и 8 Гб оперативной памяти;

– узлы (n045-n112) с 2-мя двухъядерными процессорами Intel Xeon 5160 @ 3.00ГГц и 4 Гб оперативной памяти.

Грубую оценку можно также проводить, сравнивая тактовую частоту процессоров:

$$sp_{Ri} = \frac{\varphi_{Rk}}{\max(\varphi_{Rk})} \cdot 1000$$

где φ_{Rk} – тактовая частота процессоров узлов. Для кластера НТУУ «КПИ» оценки относительной вычислительной мощности узлов выглядят следующим образом:

$$sp_{R1} = \frac{2.83}{3.00} \cdot 1000 = 943 \qquad (n001 - n044)$$

$$sp_{R2} = \frac{3.00}{3.00} \cdot 1000 = 1000 \qquad (n045 - n112)$$

Для описанной в пункте 3 тестовой задачи файл конфигурации будет выглядеть следующим образом:

```
n004.kpi id=1 sp=94308
n003.kpi id=2 sp=94302
```

Наиболее оптимальным вариантом представляется создание «карты узлов», т.е. конфигурационного файла, содержащего имена узлов и предварительно рассчитанные относительные вычислительные мощности. В этом случае при перестройке конфигурации кластера, его модернизации или расширении не требуется вмешательства в код программы формирования файла конфигурации для PVM.

Формат файла, который содержит вычислительные мощности узлов (hostmap), может иметь вид, аналогичный файлу конфигурации PVM, например:

```
n001.kpi 943
…
n044.kpi 943
n045.kpi 1000
…
n112.kpi 1000
```

4. Учет баланса загрузки

Учет баланса загрузки и распределение slave-программ по выделенным ядрам на основании строк конфигурационного файла необходимо реализовать в главном процессе параллельной программы

Т.к. выделенные ядра могут принадлежать разным по производительности процессорам, то в качестве алгоритма баланса загрузки (АБЗ) можно воспользоваться [6]. Для работы данного алгоритма необходимо определение следующих параметров:

$$p = \sum_{i=1}^{nhost} sp_i \%100$$

$$sp_{Rk} = sp_i / 100, \qquad k = 1(1)p$$

Рассмотрим условие

$$M \geq p$$

где M – количество однотипных групп расчетов, на которые можно разделить исходную задачу. Первым этапом АБЗ является вычеркивание «медленных процессоров», для которых выполняется условие

$$M \cdot sp_{Ri} \leq 1 \qquad (1)$$

Для узлов, которые могут войти в состав PVM при текущей архитектуре кластера НТУУ «КПИ», соотношение между «самым быстрым» процессором и «самым медленным» процессором таково, что данное условие (1) является неактуальным и шаг вычеркивания «медленных процессоров» можно опустить.

В общем случае для узлов, в состав которых входят многоядерные процессоры, необходимо учитывать следующие особенности:

– из перечня процессоров «вычеркивается» не процессор, а ядро, что требует контроля количества «не вычеркнутых» ядер на узле, которое будет соответствовать количеству запускаемых на узле slave-процессов;

– проверка ядер, принадлежащих одному узлу, на условие исключения должна идти последовательно по всем ядрам узла, начиная с узла с наименьшим количеством выделенных ядер. В этом случае при равных относительных вычислительных мощностях, будет минимизироваться количество задействованных узлов, что уменьшает количество необходимых затрат на установление обмена.

5. Адаптированный механизм выполнения параллельных программ

Запуск параллельной программы осуществляется путем постановки в очередь планировщика задач файла сценария (start.sh), который содержит служебную информацию о количестве запрашиваемых ресурсов, вызов программы генерации конфигурационного файла PVM с учетом полученных от планировщика имен узлов, выбор мастер-узла для запуска PVM, и вызов непосредственно требуемой параллельной программы.

В случае если пользователю предоставляется возможность задавать количество требуе-

мых процессоров для параллельного решения, то файл сценария также можно генерировать с подстановкой количества ядер (N) указанных пользователем.

Общая схема запуска представлена на рис.1.

6. Выводы

Разработанная схема выполнения параллельных программ, использующих библиотеку PVM, обеспечивает возможность корректного конфигурирования виртуальной машины при отложенном запуске через планировщик задач PBS.

Использование «карты» вычислительной мощности узлов, а также механизм учета количества предоставленных на узле ядер для решения, позволяет осуществлять баланс загрузки не только в случае использования кластера НТУУ «КПИ», но и при создании параллельной виртуальной машины из разнородных ЭВМ.

Дальнейшее направление работ связано с предоставлением возможности включения параллельной версии пакета ALLTED в маршрут проектирования в рамках проекта создания междисциплинарного комплекса математического моделирования в грид-среде.

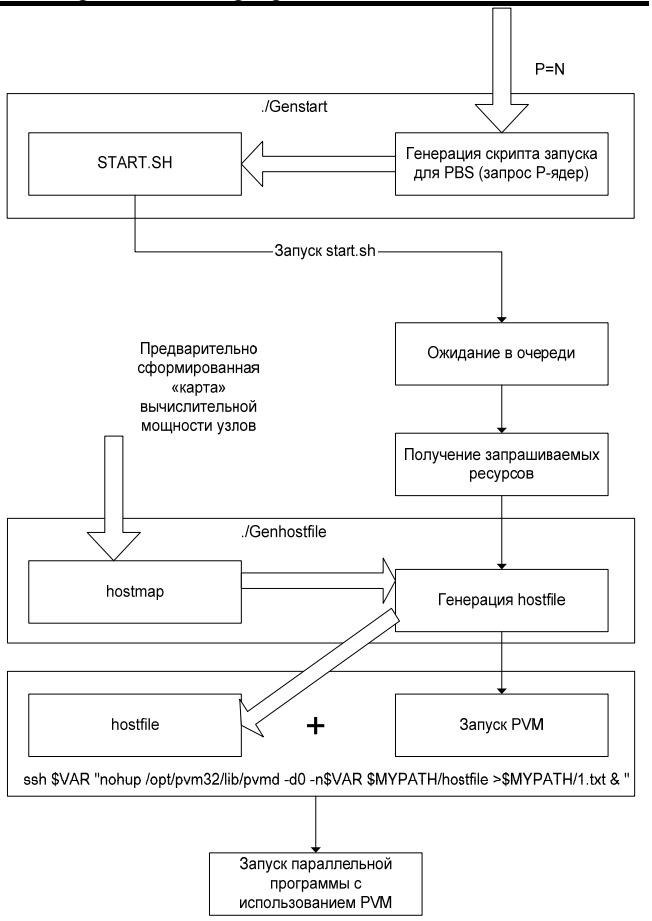


Рис. 1 – Подготовка выполнения параллельной программы

Список литературы

1. Официальный сайт разработчиков библиотеки PVM. – Режим доступа : http://www.csm.ornl.gov/pvm/pvm_home.html. – Дата доступа : 19.02.2012.
2. Ладогубец В.В. Особенности реализации параллельных алгоритмов для однопроцессорных пакетов / Финогенов А.Д., Ладогубец В.В. // Электроника и связь. – 2005. – № 25. – С. 95–98.
3. Официальный сайт центра суперкомпьютерных вычислений НТУУ «КПИ». – Режим доступа : http://hpcc.org.ua. – Дата доступа : 19.02.2012.
4. Сайт разработчиков САПР ALLTED. – Режим доступа : http://allted.kpi.ua. – Дата доступа : 19.02.2012.
5. Ладогубец В.В. Адаптация параллельного алгоритма СПУИП для кластера НТУУ «КПИ» / Ладогубец В.В., Крамар А.В., Финогенов А.Д. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – 2008. – № 48. – С. 99–103.
6. Ладогубець В.В. Алгоритм оптимального балансу загрузки / Ладогубець В.В. // Электроника и связь. – 1999. – Т. 1, № 6. – С. 74–76.

УДК 681.3

СТЕНИН А.А.,
ТИМОШИН Ю.А.,
ШЕМСЕДИНОВ Т.Г.,
КУРБАНОВ В.В.

МОДЕЛИРОВАНИЕ И АНАЛИЗ КОМПЬЮТЕРНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ (КИС) СЕРВИСНО-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЫ (SOA)

В настоящей статье предложен подход к решению задачи моделирования процессов в компьютерных информационных системах с помощью нескольких, дополняющих друг друга моделей работы веб-сервера для анализа в системах с сервисно-ориентированной архитектурой (SOA). Для моделирования применены аппараты марковских цепей, теории массового обслуживания и раскрашенных сетей Петри.

A complex approach to the Computer Information Systems (CIS) modeling problem is presented in this article using multiple complementary models of the web-server for the systems analysis in Service-Oriented Architecture (SOA). An apparatus of Markov Chains, Queuing Theory and colored Petri nets (place/transition nets) are applied for modeling and analysis in mentioned problem.

1. Введение

Эффективность работы современных компаний во многом зависит от организации КИС. Для того, чтобы определить, насколько информационная система соответствует требованиям ведения основных бизнес-процессов компании, необходимо провести анализ КИС.

Для анализа и моделирования компьютерных информационных систем класса «клиент – сервер» наибольшее распространение получили сети Петри и системы массового обслуживания [1].

Известно [2], что сети Петри предназначены для моделирования систем, которые состоят из множества взаимодействующих компонентов. Отсюда сети Петри делают возможным моделирование компьютерных информационных систем математическим представлением на основе теории комплектов, которая представляет собой простое расширение теории множеств. Целью представления компьютерных информационных систем в виде сети Петри для последующего анализа является получение информации о структуре и динамическом поведении компьютерной информационной системы [3].

Использование теории массового обслуживания [1,4] для анализа и моделирования информационных систем с сервисно-ориентированной архитектурой (SOA) [5], позволяет, помимо инструмента имитационного моделирования, использовать для исследования аппарат марковских цепей.

2. Постановка задачи

Известно, что при моделировании и анализе КИС нельзя ограничиться созданием одной модели. Это связано с тем, что сервисно-ориентированные информационные системы имеют сложную конфигурацию взаимодействующих компонентов [6], учет и анализ характеристик которых в рамках одной модели весьма сложен и подчас нецелесообразен [7]. В связи с этим, актуальной является задача построения нескольких дополняющих друг друга моделей работы КИС для ее базовых компонентов, в частности двух моделей работы такого компонента КИС как Web-сервера, построенных на основе сетей Петри и систем массового обслуживания.

При разработке сложных корпоративных КИС для задач, относящимся к большим потокам обрабатываемых данных и высоконагруженным системам, часто используются только лишь статистические данные для расчетов параметров нагрузки и вычислительных ресурсов, таких, как количество потоков обработки запросов, объем оперативной памяти, длина очереди, ожидание и таймауты. Если речь идет о системах, не критичных к задержкам и отказам, то такие подходы приемлемы, но для задач, приближенных к реальному времени, производственных, транспортных и информационно-управляющих систем, связанных с финансовыми, медицинскими и другими критическими по времени и надежности задачами, требуется применение надежных моделей еще до введения систем в реальную

эксплуатацию [3,5]. Такие расчеты желательно проводить еще на этапе проектирования аппаратных и программных средств КИС, чтобы предусмотреть ожидаемые нагрузки, соответствующие расчетным характеристикам и метрикам компонентов ИС на архитектурном уровне. Это позволит применить горизонтальное и вертикальное масштабирование, подготовить соответствующие резервные мощности и перераспределить нагрузку в узких местах, внедрить альтернативные компоненты ИС (например, бездисковые СУБД класса ключ-значение, решить проблему блокирующих операций ввода/вывода, расширить вычислительные ресурсы или применить виртуализацию).

3. Решение задачи

Для разнородных заявок в рамках теории раскрашенных сетей Петри рассмотрим следующий частный случай модели Web-сервера, обслуживающего поток заявок от оконных приложений, АРМов (автоматизированных рабочих мест), веб-интерфейсов пользователей (HTML5, JavaScript) и других прикладных компонентов в корпоративной КИС, построенной в сервисно-ориентированной архитектуре SOA.

Модель содержит три источника заявок, формируемых пользователями клиентских терминалов, сервер приложений и сервер баз данных. Каждый источник генерирует поток заявок одного типа, включая заявки на получение статического содержимого сайта (изображения, статические HTML- страницы, стилевые таблицы); заявки на получение информации из баз данных; заявки сохранения информации в базе данных. Имитационная модель информационной системы в виде раскрашенной стохастической сети Петри приведена на рис 1. Для моделирования обработки сервером заявок разного типа в модель введена раскраска в виде маркеров различной формы <▼ ♦ ■ •>. Типы заявок воответствуют разным типам запрашиваемых URL, указывающих на статические ресурсы (файлы), динамические ресурсы (генерируемые сценариями) и программные обработчики AJAX/JSON (асинхронных вызовов JavaScript, объектной нотации и форматов сериализации объектов XML и JSON).

Маркеры, обозначенные кружком, отражают состояние занятости ресурса сервера приложений – обработки запросов каналами сервера приложений.

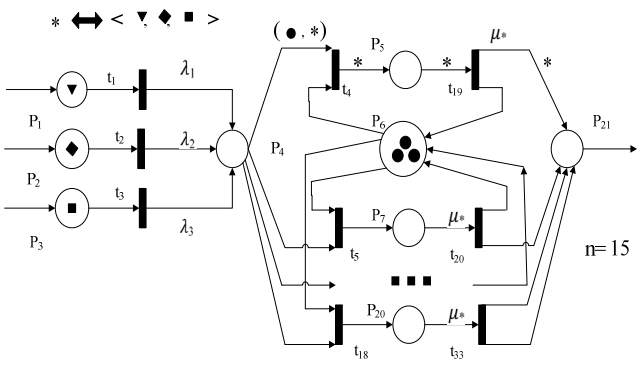


Рис.1. Модель Web- сервера в виде сети Петри

Переходы сети Петри t1,t2,t3, ассоциированы с источником заявок, интенсивность поступления которых на входы сервера приложений λ1 λ2 λ3 соответственно. Множество M каналов сервера разбито на подмножество загруженных каналов M1 и подмножество свободных каналов M2. При этом M1∩M2= ∅ и M=(M1∪M2). Моменты инициирования обработки заявок сервером приложений имитируются переходами t4,t5,…,t18. При поступлении очередной заявки в момент времени τj запускается канал обслуживания Kq∈M2 с наименьшим номером. Время запуска канала Kq определяется временем $\tau_k^q = \tau j$. Окончание обслуживания заявок имитируется переходом t19,…,t20,t33. Время окончания обслуживания с раскраской i каналом Kq задается выражением (1):

$$t_k^q = \tau_k^q + 1/\mu_i; \quad i=1,2,3, \qquad (1)$$

где μ_i – интенсивность обслуживания заявки с раскраской i.

В общем случае моделирование и анализ компьютерных информационных систем на базе сетей Петри очень сложен и трудоемок, потому требует использования специализированных программных продуктов. В частности, для этих целей часто используют специализированный пакет CPN TOOLS [8].

Как уже указывалось ранее, в отличие от сетей Петри, теория массового обслуживания позволяет помимо имитационного моделирования при определенных допущениях относительно входящих потоков запросов и обслуживания получить конечные формулы показателей функционирования компьютерных

информационных систем в аналитическом виде. Так, для простейших потоков запросов и обслуживания анализ компьютерных информационных систем можно выполнить на основе дискретных и непрерывных цепей Маркова [9]. В этом случае работу Web-сервера можно представить следующим графом состояний (рис.2), который соответствует однородности потока запросов и отсутствию раскраски в модели сети Петри, представленной на рис 1. Другими словами в модели циркулирует ординарный поток запросов с экспоненциальным распределением.

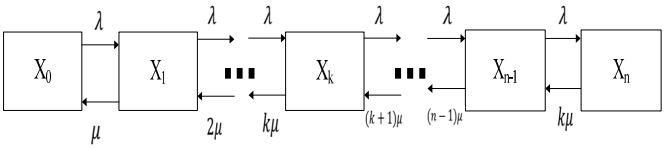


Рис. 2. Граф состояний Web-сервера

Работа такой системы сточки зрения теории массового обслуживания заключается в следующем.

На вход n-канальной системы массового обслуживания подается простейший поток заявок с интенсивностью λ. Интенсивность простейшего потока обслуживания каждого канала μ . Если заявка застала все n каналов занятыми, то она получает отказ (покидает систему не обслуженной). Если заявка застала свободным хотя бы один канал, то она принимается к обслуживанию любым из свободных каналов и обслуживается до конца («терпеливая» заявка).

Согласно рис. 2 в системе возможно следующее множество состояний:

x0 – все каналы свободны, на одна заявка не обслуживается;

x1 – занят ровно один канал (какой именно – не важно), обслуживается одна заявка;

xk – занято ровно k каналов (какой именно – не важно), обслуживается k заявка;

xn – все n каналов заняты, обслуживается n заявок.

В рассмотренном ранее частном случае было принято, что n=15. Это означает, что Web-сервер настроен для работы с 15-тью одновременными соединениями (каналами).

Для графа состояний Web- сервера (рис. 2) в соответствии с мнемоническим правилом составления системы дифференциальных уравнений для вероятностей состояний [4] получим:

чим:

$$\begin{cases} \dot{p}_0(t) = -\lambda p_0(t) + \mu p_1(t); \\ \dots \\ \dot{p}_k(t) = -[\lambda + \kappa\mu]p_k(t) + \lambda p_{k-1}(t) + (k+1)\mu p_{k+1}(t) \\ \dots \\ \dot{p}_n(t) = -n\mu p_n(t) + \lambda pn(t), \end{cases} \quad (2)$$

где: k=1,n-1.

Система (2.2) обычно интегрируется при начальных условиях:

$$\begin{cases} p_0(0) = 1; \\ p_k(0) = 0; p_n(0) = 0 \end{cases} \quad (3)$$

Решение системы (2) при начальных условиях (3) удовлетворяет нормировочному условию:

$$\sum_{k=0}^{n} p_k(t) = 1, \quad (t \geq 0). \quad (4)$$

Уравнения (2) являются уравнениями Эрланга [4].

Заметим, что выражения (2) – (4) справедливы и для случая, когда потоки событий не являются простейшими, а представляют собой нестационарные пуассоновские потоки. В этом случае параметры λ=λ(t) и μ=μ(t) являются некоторыми функциями времени (например, сезонная распродажа).

Для нас представляет интерес стационарный режим работы такой системы, когда λ(t) = const , μ(t) =const при t→ ∞. В реальной жизни установившиеся поток запросов на достаточно длинном интервале времени. В математическом плане такой режим существует, т.к. наша система эргодична, и ему отвечает алгебраическая система уравнений:

$$\begin{cases} O = -\lambda p_0 + \mu p_1; \\ \dots \\ O = -[\lambda + k\mu]p_k + \lambda p_{k-1} + (k+1)\,\mu p_{k+1}; \\ \dots \\ O = -n\mu p_n + \lambda p_{n-1}, \end{cases} \quad (5)$$

где $k = 1, \overline{n-1}$.

Решая систему (5) совместно с (4) получаем формулу для вероятностей состояния рассматриваемой системы массового обслуживания:

$$p_k = \frac{\frac{\left(\frac{\lambda}{\mu}\right)^k}{k!}}{\frac{\sum_{k=0}^{n}\left(\frac{\lambda}{\mu}\right)^k}{k!}}, (k = \overline{0,n}) \quad (6)$$

Введем обозначение: $\bar{\lambda} = \frac{\lambda}{\mu}$,

где $\bar{\lambda}$ – равна среднему числу запросов, поступающих в систему за среднее время обслуживания одного запроса в одном канале. Учитывая выражения для $\bar{\lambda}$ и умножая числитель и знаменатель (6) на $e^{-\bar{\lambda}}$ получим:

$$p_k = \frac{\left(\frac{\lambda^k}{k!}\right)e^{-\lambda}}{\sum_{k=0}^{n}\left(\frac{\lambda^k}{k!}\right)e^{-\lambda}} = \frac{p(k_1\bar{\lambda})}{R(n_1\bar{\lambda})} \qquad (7)$$

где $p(k_1\bar{\lambda})$ и $R(n_1\bar{\lambda})$ табличные функции пуассоновского распределения [4].

На основании формулы (7) можно получить целый ряд показателей работы компьютерной информационной системы: вероятность простоя, вероятность отказа, среднее число занятости каналов и т. д.

В частности для анализа КИС актуальные следующие показатели:

1. Вероятность обслуживания запроса:

$$P_{обсл.} = \frac{R(n-1,\bar{\lambda})}{R(n,\bar{\lambda})}$$

2. Среднее время полной загрузки системы:

$$t_{n.з.} = \frac{1}{n\bar{\lambda}}$$

3. Вероятность занятости хотя бы одного канала:

$$P_{з.к.} = \frac{P(n,\bar{\lambda})}{P(n,\bar{\lambda})}$$

4. Вероятность отказа:

$$P_{отк.} = 1 - P_{обсл.}$$

4. Выводы

Предложенные в работе модели являются достаточно универсальными с точки зрения характеристик потоков заявок и обслуживания в КИС с SOA архитектурой. В некоторых случаях можно получить аналогические выражения для основных показателей работы высоконагруженных КИС. Приведенные подходы можно применять в широком круге задач разработки и развертывания прикладных компьютерных информационных систем корпоративного уровня (внутренних КИС предприятий и организаций), в которых интенсивность потока заявок зависит от заранее известных параметров, таких, как количество пользователей, объем обрабатываемых данных и вычислительная мощность, необходимая для обслуживания элементарных операций в информационных процессах предприятия.

Для систем общего пользования и открытого доступа, таких, как коммуникационные службы и доступ к информационным ресурсам, возможны другие, более простые подходы, позволяющие найти линейные рабочие участки зависимости вычислительных мощностей от потока заявок. Такие задачи характеризуются малым количеством динамических параметров, влияющих на сервер и незначительными отклонениями в использовании ресурсов. Такие задачи не требуют применения архитектурного подхода к разработке ПО для соответствующих КИС, а позволяют решать вопросы устойчивости под нагрузками с применением шаблонов. Однако, сама интенсивность нагрузок в системах с открытым доступом может быть слабо предсказуема. И мы видим проблему в методах оценки и прогнозирования потока заявок в таких системах, как и в системах межкорпоративного взаимодействия.

В любом случае, аналитическое определение параметров ИС в задачах их проектирования, может во многих случаях дополнить или заменить эмпирические и статистические методы, широко использующиеся сейчас в отсутствии достаточно разработанных новых подходов, что показано на примере, рассмотренном в данной статье.

Список литературы

1. Шелухин О.И. Моделирование информационных систем. / Шелухин О.И. М.:Радиотехника.2005.–368с.
2. Питерсон Дж. Теория сетей Петри и моделирование систем / Питерсон Дж. – М.: Мир. 1984
3. Звіт про НДР Тема № 2415-п Розробка принципів та засобів інтеграції розподілених інформаційних систем з динамічною інтерпретацією метамоделей обробки та управління на міжкорпоративному рівні. / О. Стенін Ю. Тимошин, В. Галаган, М. Ткач, В. Ярченко, Т. Шемседінов та інші, 2012. – 309с.
4. Исайченко Д. Измерение процессов управления ИТ / Дмитрий Исайченко – в журн. «Открытые системы» , № 07, 2011, Режим доступа: http://www.osp.ru/os/2011/07/13010493/

5. Шемсединов Т. Слой ИС с динамической интерпретацией метаданных. / Шемсединов Т.Г. Режим доступа: http://blog.meta-systems.com.ua/2011/01/blog-post_28.html
6. Стенін О.А. Розробка фізичних і логічних метрик в задачі багатокритеріальної оптимізації інформаційного навантаження при структуризації корпоративного центру даних. / О.А.Стенін, Ю.А.Тимошин, Т.Г.Шемсединов, С.О.Шуст. Адаптивні системи автоматичного управління – Дніпропетровськ: ДНВП Системні технології, 2009-Вип.12(32).-С.86-91
7. Питер Брукс. Метрики для управления ИТ-услугами / Питер Брукс; Пер. С англ. – М.: Альпина Бизнес Брукс, 2008. – 283с.
8. van der Aalst and. Modeling Business Processes – A Petri Net-Oriented Approach (using CPNTOOLS) / W.M.P. van der Aalst and C. Stahl, The MIT Press, 2011 (ISBN-13: 978-0-262-01538-7). Режим доступа: http://cpntools.org/books/modeling
9. Овчаров Л.А. Прикладные задачи теории массового обслуживания / Овчаров Л.А.– М.: Машиностроение.1969.-324с

УДК 621.384

КУЛАКОВ Ю.О.,
КУЦ В.Ю.

ОЦІНКА СПІВВІДНОШЕННЯ СИГНАЛ/ШУМ В ФУНКЦІОНАЛЬНО-ОРІЄНТОВАНИХ СИСТЕМАХ АНАЛІЗУ ЦИКЛІЧНИХ СИГНАЛІВ

В статті обґрунтовано використання співвідношення сигнал/шум для підсистем вторинної обробки в функціонально-орієнтованих системах аналізу циклічних сигналів.

In the article usage of a signal to noise ratio for subsystems of secondary processing in function – oriented systems of the analysis of cyclical signals is justified

Вступ

Використання функціонально орієнтованих систем дозволяє суттєво збільшити швидкість обробки даних вимірювальних експериментів. Такий ефект досягається завдяки адаптації структур таких систем для вирішення обчислювальних задач певного типу, що мають загальні або подібні алгоритми опрацювання даних. Окремий клас таких систем утворюють функціонально орієнтовані системи аналізу циклічних процесів.

Досвід розробки та застосування таких систем свідчить про зростання вимог щодо підвищення достовірності виявлення та точності вимірювання параметрів та характеристик циклічних сигналів в умовах, коли рівень сигналу близький або навіть менший за рівень завади, а її статистичні характеристики невідомі. Крім того існують задачі, в яких відсутня апріорна інформація про сигнали – відомим є тільки те, що сигнали породжені циклічними процесами.

Для порівняння різних способів та алгоритмів опрацювання сигналів у функціонально-орієнтованих системах аналізу циклічних процесів необхідно обґрунтувати і ввести критерії ефективності їх обробки. Вирішальним фактором, який обмежує застосування тих чи інших відомих способів та алгоритмів обробки в фазовому методі є наявність певного рівня завад у прийнятому сигналі. Тому основою таких критеріїв є співвідношення сигнал/шум (С/Ш).

В статистичній радіотехніці та оптиці при вирішенні задач виділення сигналів на фоні завад та оцінки їх параметрів користуються наступними визначеннями понять «співвідношення сигнал/шум» [1]:

(С/Ш)= (амплітуда сигналу/середнє квадратичне значення шуму),

(С/Ш)= (потужність сигналу/потужність шуму),

(С/Ш)= (пікове значення сигналу/середнє квадратичне значення шуму),

(С/Ш)= (енергія сигналу/енергія шуму).

Ефективність різних способів розв’язання задач виділення сигналів на фоні завад оцінюється покращенням співвідношення (С/Ш) на виході пристроїв, що реалізують дані способи відносно такого відношення на їх вході.

Наведені вище означення співвідношення С/Ш зручні з практичної точки зору, оскільки необхідні для їх визначення величини для широкого кола задач можна отримати шляхом прямих вимірювань такими приладами, як вольтметри амплітудних чи середніх квадратичних значень, ваттметрів і т.п. Використані в цих означеннях величини, по-перше, визначаються через енергетичні параметри сигналу (наприклад, силу електричного струму чи напругу, потужність тощо) і, по-друге, не повною мірою характеризують можливості визначення фазових характеристик сигналів складної форми. Наприклад, перше співвідношення можна застосовувати лише до гармонічних сигналів, воно не враховує тривалості сигналу. Крім того, їх використання для оцінки ефективності тих чи інших способів обробки сигналів в системах аналізу циклічних процесів наштовхується на певні методологічні труднощі: метою обробки сигналів в таких системах є їх фазова характеристика сигналу (або різниця фазових характеристик двох досліджуваних циклічних сигналів) [2], яка не відноситься до числа його енергетичних параметрів та характеристик (виняток становить випадок, коли фазовий зсув сигналів перетворюється в напругу іншими пристроями, наприклад, фазовими детекторами). Тому вказані співвідношення С/Ш можуть бути використані лише для оцінки первинної обробки сигналів, яка може включати підсилення, частотну

фільтрацію, трансформацію спектра частот сигналу тощо.

Мета статті – обґрунтувати відношення сигнал/шум на виході функціонально орієнтовані системи аналізу циклічних процесів.

Постановка задачі. В каналі вторинної обробки сигналів системи визначається різниця фазових характеристик сигналів $u_1(t)$, який уявляє адитивну суміш радіоімпульсного сигналу з частотою заповнення f_c, та гауссівської завади з дисперсією σ^2, та гармонічного сигналу $u_0(t)$ цієї ж частоти f_c. Ці сигнали спостерігаються на часовому інтервалі, що охоплює час їх існування. З метою оцінки ефективності різних алгоритмів обчислення різниці ФХС необхідно обґрунтувати співвідношення С/Ш на вході і виході каналу.

Розв'язок

Для кращого розуміння суті задачі розглянемо узагальнену структурно-логічну схему каналу вторинної обробки функціонально-орієнтованої системи, яка представлена на Рис. 1.

Досліджувані неперервні в часі сигнали $u_1(t)$ та $u_0(t)$ поступають на входи аналого-цифрових перетворювачів АЦП1 та АЦП2. На виході каналів обчислення фазових характеристик сигналів формуються цифрові відкліки ФХС. Суматор Σ визначає їх різницю $\varphi[j]=\Phi_1[j]-\Phi_0[j]$.

Визначення ФХС відбувається на основі дискретного перетворення Гільберта з наступним визначенням ФХС в межах напівінтервалу $[0,2\pi)$, тобто значень $\Phi_1[j](\mathrm{mod}\,2\pi)$ та $\Phi_0[j](\mathrm{mod}2\pi)$. З метою розгортання ФХС без втрати інформації про кількість цілих фазових циклів попередньо виконується цифрова фільтрація послідовностей $\Phi_1[j](\mathrm{mod}\,2\pi)$ та $\Phi_0[j](\mathrm{mod}2\pi)$. В якості такої фільтрації може бути використана обґрунтована в [3] кругова медіанна фільтрація.

Оцінка співвідношення (С/Ш)$_1$ на вході каналу вторинної обробки. Спочатку обґрунтуємо вибір співвідношення С/Ш на вході каналу. Як приклад розглянемо один з характерних видів сигналів фазових ІВС – адитивну суміш $u_i(t)+\xi(t)$ інформативного сигналу у формі радіоімпульса з гауссівською обвідною виду

$$u_i(t)=e^{-at^2}\cos(2\pi f_c t),\ t\in(-\infty,\ \infty), \qquad (1)$$

де α – параметр, який визначає швидкість наростання/спадання радіоімпульсу і має розмірність 1/с, та гауссівської стаціонарної завади $\xi(t)$ з нульовим математичним сподіванням та дисперсією σ^2. Приклади графіків інформативної складової сигналу та його реалізації з гауссівською завадою наведені відповідно на рис. 2,а,б, на якому позначено: 1 – радіосигнал виду (1), 2 – обвідна радіосигналу.

Оскільки амплітуда і миттєва потужність сигналу на інтервалі його існування є величинами змінними, то визначення співвідношення $(С/Ш)_1$ через ці величини не дає однозначного результату. Це саме стосується і визначення співвідношення $(С/Ш)_1$ через пікове значення сигналу: при однаковому піковому значенні сигналу його тривалість, отже і енергія, можуть бути різними.

Рис.1 Узагальнена структурно-логічна схема каналу вторинної обробки системи аналізу циклічних сигналів

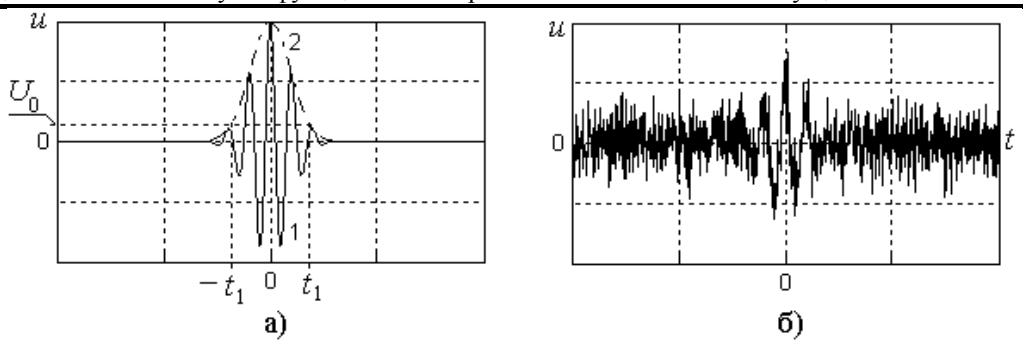


Рис. 2 Графіки інформативного сигналу (а) та його адитивної суміші з гауссівською завадою (б)

Тому найбільш обґрунтованою вбачається оцінка $(С/Ш)_1$ через відношення "енергія сигналу/енергія завади".

Енергія сигналу (1) що виділяється на резисторі R визначається як [4].

$$E_{\text{с}} = \frac{1}{R}\int_{-\infty}^{\infty} u_i^2(t)dt \qquad (2)$$

Оскільки енергія використовується не для оцінки її абсолютного значення, а для порівняння, у виразі (1) вибирають $R=1$ Ом. Визначення енергії сигналу (1) ілюструє рис. 3 (енергії сигналу відповідає заштрихована частина рисунку).

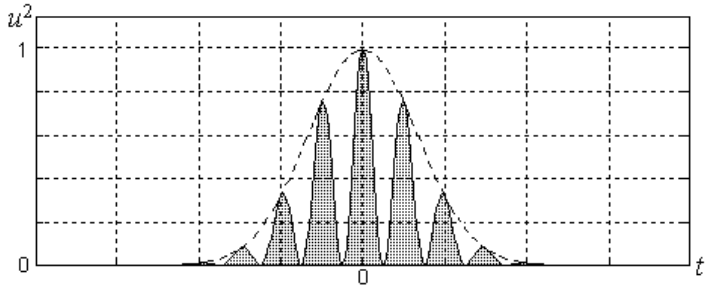


Рис. 3 Визначення енергії інформативного сигналу (1)

В сигналі (1) використана гауссівська обвідна, яка визначена на нескінченному інтервалі часу $(-\infty,\ \infty)$. Енергія шуму на такому інтервалі, навіть при незначній дисперсії, дає нескінченне значення. Такий результат не має практичної цінності і не може бути корисним при визначенні співвідношення $(С/З)_1$. Тому при визначенні енергії сигналу необхідно обмежити його тривалість за певним рівнем. Таким рівнем може бути значення U_0 значно менше за амплітудне значення інформативного сигналу, якому на рис. 3.16,а відповідає інтервал часу $(-t_1,\ t_1)$.

Звичайно, вибір рівня U_0 виглядає дещо штучним. Але з огляду на те, що в задачах порівняння ефективності способів і алгоритмів обробки сигналів важливим є не визначення абсолютного значення співвідношень $(С/Ш)_1$ на вході каналу та $(С/Ш)_2$ на його виході, а встановлення між ним відношення типу "більше/менше" (або встановлення такого відношення для $(С/Ш)_2$ для різних алгоритмів вторинної обробки інформації), такий підхід можна вважати задовільним. Для обраного виду сигналу(1) доцільно задавати інтервал $(-t_1,\ t_1)$ з умови

$$\int_{-t_1}^{t_1}\exp\left(-\alpha t^2\right)dt = 0.95\sqrt{\frac{\pi}{\alpha}}, \qquad (3)$$

що відповідає охопленню 95% площі під кривою обвідної сигналу (1). В цьому випадку інтервал $(-t_1,\ t_1)$ визначається за рівнем $U_0 = \exp(-\alpha t_1^2)$, а значення моменту часу $t_1 = \sqrt{\frac{2}{\alpha}}$.

В часовому інтервалі тривалістю $2t_1$ енергія шуму визначається як $E_{\text{ш}} = \frac{2t_1\sigma^2}{R}$.

Таким чином, на вході вимірювального каналу маємо співвідношення

$$\left(\frac{C}{Ш}\right)_1 = \frac{1}{2t_1\sigma^2}\int_{-t_1}^{t_1} u_i^2(t)dt\,. \qquad (4)$$

У тих випадках, коли точне аналітичне ви-

значення енергії сигналу ускладнено (що має місце, наприклад, для сигналу виду (1)), можна застосувати наближений метод її оцінки, який ґрунтується на заміні обвідної радіосигналу довільної форми обвідною прямокутної форми за умови незмінності площі фігур, утворених обвідними та віссю часу. Для розглянутого вище сигналу (1) гауссівська обвідна в інтервалі $(-t_1,\ t_1)$ може бути замінена прямокутною обвідною зі значенням по осі ординат $A=\dfrac{0.95}{2t_1}\sqrt{\dfrac{\pi}{\alpha}}=\dfrac{0.95\sqrt{\pi}}{2\sqrt{2}}\approx 0.595$, отже енергія інформаційного сигналу оцінюється для сигналу наступного виду

$$u_i(t)=\begin{cases}A\cos(2\pi f_c t),\ t\in[-t_1,\ t_1],\\ 0,\ t\notin[-t_1,\ t_1].\end{cases} \qquad (5)$$

Визначення енергії такого сигналу є тривіальним. Графіки радіосигналу виду (5) та його квадрату наведені на рис. 4 а, б (енергію радіосигналу з прямокутною обвідною визначає заштрихована частина рис. 4,б).

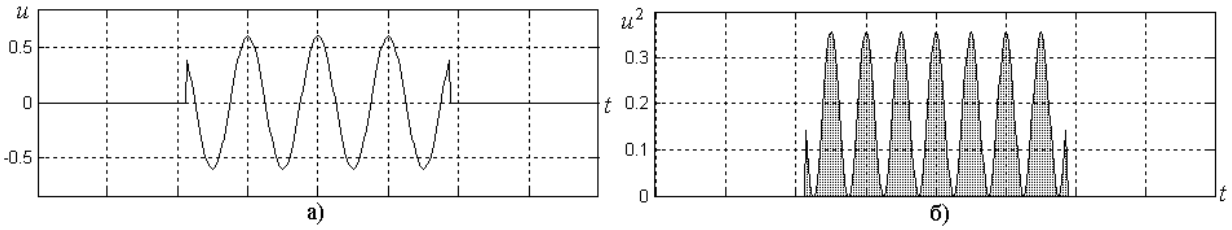


Рис. 4. Наближене визначення енергії радіоімпульсного сигналу з гауссівською обвідною виду (1) через радіосигнал з прямокутною обвідною (а) та його енергію (б)

Оцінка співвідношення $(С/Ш)_2$ на виході каналу вторинної обробки. Розглянемо графік фазової характеристики адитивної суміші сигналу (1) та гауссівської завади, представлений на рис. 5 кривою 1, та представлений кривою 2 графік ФХС гармонічного сигналу з частотою, що дорівнює частоті f_c сигналу заповнення радіоімпульса. Крива 1 отримана за умови фільтрації послідовності $\Phi_1[j]\pmod{2\pi}$ круговим медіанним фільтром (питання вибору апертури кругового медіанного фільтра та особливостей його робот виходить за рамки даної роботи; тут лише зазначимо, що спроба розгорнути ФХС без такої фільтрації вже для співвідношення (С/Ш)1<10 приводить до виникнення хибних стрибкі ФХС величиною 2π).

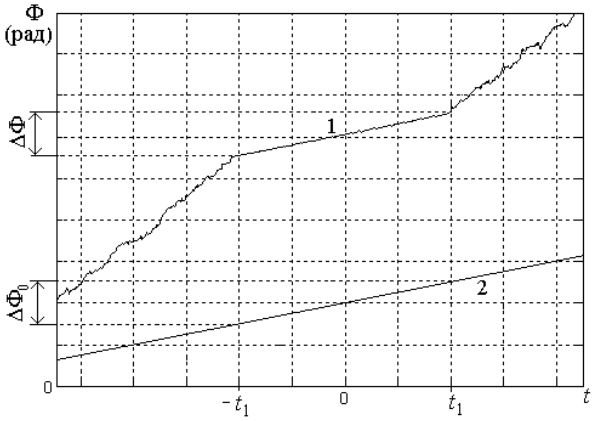


Рис. 5. Графіки фазових характеристик адитивної суміші сигналу (1) та гауссівської завади (крива 1) та гармонічного сигналу частотою f_c (крива 2)

На інтервалі аналізу сигналу $(-t_1,\ t_1)$ фаза досліджуваного сигналу змінюється на величину $\Delta\Phi$, що власне і слід розглядати як корисну компоненту вихідного сигналу при визначенні ФХС. Фаза гармонічного сигналу з частотою несучого коливання за цей же час змінюється на величину $\Delta\Phi_0=4\pi f_c t_1$. За відсутності спотворень ФХС повинна виконуватись рівність $\Delta=\Delta\Phi-\Delta\Phi_0=0$. Відхилення від цього значення повинно трактуватись як зменшення співвідношення (С/Ш)2. Оскільки значення Δ в загальному випадку можуть бути як позитивні,

так і негативні, таке зменшення необхідно врахувати парною функцією, яка обернено залежить від Δ, наприклад, функцією $e^{-\Delta^2}$.

Дія завади на вході ІВС приводить до збільшення розкиду поточних значень різниці ФХС на інтервалі аналізу $(-t_1,\ t_1)$. В роботі [5] обґрунтовано використання для оцінки розкиду кутів вибіркової кругової дисперсії V

$$V = 1 - \sqrt{\left(\frac{1}{J}\sum_{j=1}^{J}\cos\varphi[j]\right)^2 + \left(\frac{1}{J}\sum_{j=1}^{J}\sin\varphi[j]\right)^2}\ , \quad (6)$$

де J – загальна кількість відліків ФХС, яка визначається часовим вікном тривалістю $2t_1$.

З урахуванням цього співвідношення (С/Ш)2 на виході каналу фазової ІВС можна визначити як

$$\left(\frac{\text{С}}{\text{Ш}}\right)_2 = \frac{e^{-\Delta^2}}{V}\ , \quad (7)$$

Таким чином, запропоновані і обґрунтовані формули (4) та (7) дозволяють виконувати оцінку співвідношення сигнал/шум на вході і виході функціонально-орієнтованих систем аналізу фазових характеристик циклічних сигналів, що необхідно для оцінки ефективності різних способів та алгоритмів вторинної обробки інформації в таких системах.

Висновки

Ефективність роботи технічних пристроїв і систем різного призначення визначається збільшенням співвідношення сигнал/шум на виході систем відносно входу. Традиційно це співвідношення визначається як відношення амплітуди(пікового значення, або потужності, або енергії) інформаційного сигналу до середнього квадратичного значення (енергії або потужності) шуму. Таке визначення співвідношення С/Ш має обмеження для використання в системах аналізу циклічних сигналів, на виході яких формуються фазові характеристики, які не належать до енергетичних.

В роботі запропоновано використання співвідношення С/Ш на виході каналів визначення ФХС функціонально-орієнтованих систем аналізу циклічних процесів, яке визначається неенергетичними параметрами сигналу – відхиленням оцінки фазової характеристики сигналів від заданої та вибірковою круговою дисперсією різниці ФХС.

Запропоноване співвідношення може бути використане для оцінки ефективності способів і алгоритмів визначення ФХС циклічних сигналів на фоні адитивної завади.

Література

1. Шестов Н.С. Выделение оптических сигналов на фоне случайных помех / Шестов Н.С. – М.: Советское радио, 1967. – 347 с.
2. Куц Ю.В. Статистична фазометрія [наукова монографія] / Ю.В. Куц, Л.М. Щербак. – Тернопіль: Тернопільський державний технічний університет, 2009р. – 383 с.
3. Куц В.Ю. Аналіз застосування кругової медіанної фільтрації в задачах обробки сигналів. Збірник наукових праць. Інститут проблем моделювання в енергетиці. Випуск 39.ст. 50-56.
4. Сергиенко А.Б. Цифровая обработка сигналов / Сергиенко А.Б. – СПб.: Питер, 2003. – 604 с.
5. Мардиа К. Статистический анализ угловых наблюдений: Пер. с англ. – М.: Главная ред. физ.-мат. лит. изд-ва “Наука”, 1978. – 240 с.

УДК 004.056.5

МАРКОВСЬКИЙ О.П.,
ФЕДОРЕЧКО О.І.,
КОРОТЕНКО А.А.

МЕТОД КОРЕКЦІЇ ОДИНОЧНОЇ ПОМИЛКИ СИНХРОНІЗАЦІЇ В АСИНХРОННИХ ЛІНІЯХ ПЕРЕДАЧІ ЦИФРОВИХ ДАНИХ

В статті запропоновано метод виправлення одиничних помилок, що виникають в асинхронних каналах передачі даних. Основна особливість запропонованого методу полягає в тому, що він дозволяє корегувати як бітові спотворення, так і помилки синхронізації. Метод використовує арифметичні операції. Детально представлено математичну ідею методу та процедури виявлення та корекції помилок обох типів. Використання процедури корекції помилок ілюструється прикладами. Наведено теоретичні та експериментальні оцінки ефективності запропонованого методу.

In article method of single errors correcting in asynchronous digital data transmission channels. The main peculiarity of proposed method consist of that it allows to correct as bit transformation error such synchronization error. Method is based on arithmetic operations. The method mathematical basis and procedure of detecting and correcting both types of errors are presented in detail. The use of the procedure error correction is illustrated via the thorough presentation of an example of erroneous data transmission. The theoretical and experimental estimations of effectiveness of proposed method are given

Вступ

Реаліями сьогоднішнього дня стало розширення використання розподілених обчислень і тенденція до "інтелектуалізації" периферійних пристроїв комп'ютерних систем, що має наслідком різке зростання об'ємів обміну даними. Разом з цим, постійно зростають і вимоги до швидкості передачі даних. Для прикладу, стандарт послідовного інтерфейсу SATA за останні десять років збільшив пропускну здатність у 5 разів, новий стандарт USB 3.0 порівняно з попередньою версією збільшує максимальну швидкість передавання даних відразу в 10 разів, до 5 Гбіт/с [1].

Збільшення швидкості передавання даних неминуче призводить до росту кількості помилок, адже саме передача даних залишається однією з найменш надійних частин комп'ютерних систем. Причиною цьому є низка складних фізичних процесів, що одночасно протікають в середовищі передачі, та призводять до викривлення та пошкодження інформації, що передається. І чим більша швидкість передачі даних застосовується, тим більше фізичних процесів та явищ здатні значно вплинути на процес передачі. Наприклад, при обміні інформацією по оптоволоконним каналам на швидкостях, що перевищують 10 Гбіт/с доводиться зіштовхуватися з новим ефектом, який не відіграє суттєвої ролі на маленьких швидкостях – поляризаційно-модовою дисперсією (PMD), суть якої полягає у тому, що дві поляризовані компоненти світлового імпульсу можуть рухатися із різними швидкостями та спотворювати сигнал. Крім того, при роботі з будь-яким фізичним середовищем передачі доводиться зіштовхуватися із завадами, здатними накладатися на дані, що передаються, та викривляти їх. Ці завади зазвичай є нерегулярними, невпорядкованими та структурно схожими на інформаційні сигнали, що ускладнює їх виявлення [2].

Таким чином, продиктована технічним прогресом необхідність постійного збільшення швидкостей передачі даних має компенсовуватись розробленням нових методів виявлення та корекції помилок, що дозволяють забезпечити ефективність та надійність передачі інформації на великих швидкостях.

Особливо гостро постає проблема забезпечення достовірності інформації, що передається, у обчислювальних системах та мережах, оскільки, на відміну від систем зв'язку або цифрового телебачення, така інформація може бути застосована для керування важливими об'єктами, відповідно до чого зростає ціна як втрати часу при повторному пересиланні даних, так і помилкового визнання пошкодженого пакету даних коректним, що може мати значні негативні наслідки [2].

Таким чином, задача підвищення ефективності виявлення та виправлення помилок синхронізації в асинхронних лініях обміну цифровими даними комп'ютерних систем є актуальною на сучасному етапі розвитку інформаційних технологій.

Аналіз існуючих засобів виявлення та корекції помилок синхронізації

Для будь-якої сучасної обчислювальної системи, передача інформації є одним із базових процесів. Вона відбувається як між окремими вузлами комп'ютера, так і з віддаленими периферійними пристроями, або іншими системами. Для передачі цифрових даних на фізичному рівні розроблено низку методів кодування, таких як NRZ (Non-Return to Zero), AMI (Alternate Mark Inversion), Манчестерське кодування тощо. Недоліком багатьох методів є відсутність властивості самосинхронізації, тобто схильність до утворення помилок синхронізації [1].

Причина утворення помилок синхронізації гарно ілюструється механізмом роботи широковживаного в багатьох послідовних інтерфейсах (таких як USB або FireWire) способу кодування даних – NRZI (Non Return to Zero Invert – метод неповернення до нульового потенціалу). Даний метод використовує лише два види ділянок – з додатним потенціалом та протилежним від'ємним, без нульових областей, при чому передача біта нуля кодується зміною напруги в лінії, а при передачі одиниці значення напруги не змінюється [1].

Спосіб передачі даних окреслює можливі випадки виникнення помилок: якщо передавач надсилає довгу серію одиниць, то потенціал на лінії не змінюється протягом певного часу. Період синхронізації передавача τ_S відрізняється від періоду приймача τ_R на певну випадкову величину μ, так що $\tau_R = \tau_S + \mu$. При передачі серії з n одиниць, часовий інтервал на приймачеві може відрізнятися вже на $n \cdot \mu$, і якщо ця величина стає співрозмірною до τ_S, є певна ймовірність того, що приймач розпізнає серію з n одиниць, як таку, що містить n+1 або n-1.

Важливою особливістю помилок синхронізації є те, що традиційні механізми перевірки коректності даних, такі як використання надлишкових циклічних кодів (Cyclic Redundancy Codes, CRC), виявляються нездатними зафіксувати факт надходження пошкодженого інформаційного пакету [3]. CRC гарантує знаходження "пачки" помилок довжиною не більше степені утворюючого поліному, але оскільки в разі виникнення помилки синхронізації фактично доводиться мати справу зі зсувом надісланого блоку біт та зміною самого розміру отриманого пакета даних, CRC код в багатьох випадках пропускає пошкоджений блок, вважаючи його коректним.

Існує два механізми боротьби з помилками синхронізації: бітовий стаффінг та повторне надсилання пошкодженого помилкою блоку. Другий метод має багато недоліків: по-перше, ніяк не гарантується відсутність помилок у повторному повідомленні, по-друге, декількаразове надсилання одного й того ж самого блоку призводить до затримок у часі, що є суттєвим для систем з роботою у реальному часі [1].

Інший розповсюджений метод – бітовий стаффінг – дозволяє зменшити ймовірність виникнення помилок синхронізації шляхом додавання до кожної послідовності з шести поспіль одиниць нульового біту, тим самим зменшуючи час, протягом якого на лінії не змінюється потенціал. Подібний механізм ускладнює процес передачі та змушує передавати велику кількість додаткових байт (до 17%), при цьому ніяк не гарантуючи відсутність помилок синхронізації, а тільки зменшуючи імовірності їх виникнення [4].

Існуючі корегуючи коди [4] для виправлення помилок синхронізації жорстко прив'язані з помилками, що виникають в USB, тобто зникненню лише одиниць. Разом з тим, на практиці використовується широкий арсенал методів низькочастотного кодування даних в яких помилки носять симетричний характер і кількість бітів, в результаті порушень синхронізації, може як збільшуватися, так і зменшувати.

Таким чином, існує необхідність в розробці спеціальних засобів виявлення та виправлення помилок синхронізації.

Ціллю досліджень є розробка способу ефективної корекції одиночних помилок, що виникають в асинхронних каналах обміну цифровими даними комп'ютерних систем.

Метод корекції помилки синхронізації

Для досягнення поставленої цілі розглядається наступна модель виникнення помилок в асинхронному каналі – при передачі n-бітового блоку B, такого що $B = \{b_1, b_2, \ldots, b_n\}$, втрачається один біт b_l, $l \in \{1, \ldots, n\}$, в наслідок чого

на приймачі отримується інформаційний блок довжиною на один біт менше, а саме $n_R = n - 1$ біт. Можлива також протилежна ситуація: при передачі даних у вихідному блоці на довільній позиції виникає біт (нуль або одиниця), що призводить до отримання на приймачеві інформаційного блоку довжиною $n_R = n + 1$ біт.

В будь-якому випадку, при зникненні або виникненні біту у інформаційному блоці необхідно визначити факт передачі даних з помилкою, визначити точне місцезнаходження помилки у блоці та виконати корекцію отриманих даних.

Для розв'язання поставлених вище задач пропонується використовувати зважений контрольний код $C = \{p, S_1\}$, що складається з двох компонент, які обраховуються за формулами:

$$p = b_1 \oplus b_2 \oplus \ldots \oplus b_n$$
$$S_1 = \sum_{j=1}^{n} b_j \cdot j \qquad (1)$$

Перша компонента – біт парності p, розраховується як сума за модулем два всіх бітів інформаційного блоку.

Значення другої компоненти S_1 контрольної суми розраховується як арифметична сума порядкових номерів усіх позицій, на яких у вихідному блоці знаходяться одиничні біти.

Компоненти контрольного коду, обчислені на передавачеві за формулами (1), позначаються як $C_S = \{p_S, S_{1S}\}$.

В свою чергу, обчислені на приймачеві компоненти контрольного коду позначаються як $C = \{p_R, S_{1R}\}$ та обчислюються за формулами, аналогічними формулам (1).

Безпосереднє виявлення факту передачі інформації з помилкою, визначення її типу, локалізація та корекція відбувається на приймачі шляхом порівняння та аналізу кількості отриманих біт у інформаційному повідомленні з очікуваною кількістю, а також розрядів контрольних кодів: отриманого від передавача, та розрахованого на приймачеві.

Для встановлення наявності помилки синхронізації порівнюється фактично отримана кількість біт у повідомленні n_R з очікуваною кількістю n. Якщо ці кількості збігаються, то констатується відсутність помилок синхронізації при передачі, проте все ще можливими залишаються бітові спотворення у отриманому блоці. Можливі наступні варіанти:

1) біти парності однакові ($p_R = p_S$), другі компоненти контрольного коду збігаються ($S_{1R} = S_{1S}$) – передача інформаційного блоку відбулась без помилок.

2) біти парності однакові ($p_R = p_S$), другі компоненти контрольного коду не збігаються ($S_{1R} \neq S_{1S}$) – передача інформаційного блоку відбулась с помилками парної кратності, виправлення неможливе.

3) біти парності різні ($p_R \neq p_S$) – вважається, що має місце помилка, яка призвела до пошкодження даних. Далі обраховується $j = S_{1R} - S_{1S}$, і в залежності від отриманого значення обирається наступний крок:

а) $j > n$ або $j < -n$ – має місце помилка непарної кратності, більшої за одиницю. Виправлення неможливе, необхідне повторне надсилання даних.

б) $0 < j < n$ – одинична помилка на позиції j, в отриманому приймачем блоці даних одиничний біт на позиції j слід замінити на нульовий

в) $-n < j < 0$ – одинична помилка на позиції j, в отриманому приймачем блоці даних нульовий біт на позиції j слід замінити на одиничний.

Якщо різниця між n та n_R становить більше одиниці, то під час передачі мали місце помилки синхронізації, кратність яких більша або дорівнює двом – такі помилки не можуть бути виправлені запропонованим методом і приймач формує запит на повторну передачу пошкодженого інформаційного блоку. Якщо ж різниця між очікуваною та отриманою кількістю біт дорівнює одиниці, то має місце одинична помилка синхронізації. В такому випадку виокремлюється чотири можливі типи помилок:

1) $n_R < n$ та $p_R \neq p_S$ – біти парності різні та кількість отриманих біт менша за кількість надісланих – має місце зникнення одиничного біту, що знаходиться на позиції m в інформаційному блоці передавача.

2) $n_R < n$ та $p_R = p_S$ – біти парності збігаються, кількість отриманих біт на приймачі менша за кількість надісланих передавачем – під час передачі даних відбулась втрата нульового біту, що знаходиться на позиції m в інформаційному блоці передавача.

3) $n_R > n$ та $p_R \neq p_S$ – біти парності не збігаються, при цьому кількість отриманих біт більша за очікувану кількість – має місце виникнення одиничного біту, що знаходиться на позиції m в інформаційному блоці приймача.

4) $n_R > n$ та $p_R = p_S$ – біти парності однакові, кількість отриманих приймачем біт більша за кількість надісланих передавачем – під час передачі інформаційної послідовності має місце

виникнення нульового біту, що знаходиться на позиції m в інформаційному блоці приймача.

Якщо під час передачі мали місце втрата або виникнення нульового розряду, то процедура локалізації базується на аналізі другої компоненти отриманого та обчисленого контрольного коду, та пропонується у вигляді наступної послідовності дій:

1. Виконується обчислення модулю арифметичної різниці між значеннями другої компоненти контрольного коду передавача та приймача $\Delta_0 = | S_{IR} - S_{IS} |$.

2. Початкове значення порядкового номеру j біту, що потребує корекції визначається як $j = n_R$.

3. Якщо $b_j = 1$,то значення Δ_0 зменшується на одиницю, $\Delta_0 = \Delta_0 - 1$.

4. Якщо $\Delta_0 \neq 0$, то значення індексу зменшується на одиницю $(j = j - 1)$ та відбувається повернення до пункту 3.

5. Якщо кількість отриманих біт менша за очікувану кількість ($n_R < n$), то корекція блоку на приймачеві полягає у послідовному зсуві інформаційних бітів $\{b_{j+1},\ldots,b_{nr}\}$ на одну позицію праворуч та вставці одиничного біту на вільну позицію j, в протилежному ж випадку виконується видалення одиничного біту із позиції j-1, після чого необхідно зсунути на одну позицію ліворуч всі інформаційні розряди $\{b_{j+1},\ldots,b_{nr}\}$

Якщо ж під час передачі мали місце втрата або виникнення одиничного розряду, то алгоритм дій схожий з розглянутим у попередньому випадку:

1. Обчислюється вираз

$$\Delta_0 = \left| \sum_{i=1}^{n} i - S_{1s} - \left(\sum_{i=1}^{n_r} i - S_{1r} \right) \right| = \begin{cases} n+1+S_{1s}-S_{1r}, n_r > n \\ n - S_{1s} + S_{1r}, n_r < n \end{cases} \quad (2)$$

2. Визначається початкове значення індексу j як $j = n_R$.

3. Якщо $b_j = 0$, тоді поточне значення обрахованої в першому пункті Δ_1 зменшується на одиницю, $\Delta_1 = \Delta_1 - 1$.

4. Якщо $\Delta_0 \neq 0$, то значення індексу зменшується на одиницю $(j = j - 1)$ та відбувається перехід до пункту 3.

5. Якщо кількість отриманих біт менша зі очікувану кількість ($n_R < n$), то для виконання корекції блоку на приймачеві необхідно послідовно зсунути інформаційні біти $\{b_{j+1},\ldots,b_{nr}\}$ на одну позицію праворуч та вставити нульовий біт на вільну позицію j, в протилежному випадку видаляється нульовий біт на позиції j-1, після чого виконується зсув на одну позицію ліворуч всіх інформаційних розрядів $\{b_{j+1},\ldots,b_{nr}\}$.

Запропонований метод може бути ілюстрований наступними прикладами:

1) Нехай, з передавача надсилається блок $B_S = \{b_1, b_2, \ldots, b_{16}\} = \{0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1\}$ довжиною 16 бітів (n=16). Контрольний код на передавачі обчислюється у вигляді: $p_S = 0\oplus1\oplus1\oplus0\oplus1\oplus1\oplus1\oplus0\oplus0\oplus1\oplus0\oplus1\oplus0\oplus0\oplus0\oplus1=0$, $S_{1S} = 2 + 3 + 5 + 6 + 7 + 10 + 12=61$.

Таким чином, в доповнення до основного інформаційного блоку передавач надсилає контрольний код $C = \{0, 61\}$.

Під час передачі даних втрачається біт на $m = 7$ позиції, внаслідок чого на приймачі отримується блок довжиною 15 біт: $B_R = \{b_1, b_2, \ldots, b_{15}\} = \{0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1\}$. Компоненти контрольного коду приймача обчислюються наступним $p_R = 1$, $S_{1R} = 2+3+5+6+ 9 + 11+15 = 51$.

Оскільки біти парності приймача та передавача не збігаються ($p_R \neq p_S$), на приймачі фіксується факт отримання інформаційного блоку з помилкою.

Далі, оскільки кількість отриманих бітів менша за очікувану кількість ($n_R < n$), то, згідно запропонованої методики класифікації типів помилок, має місце помилка другого типу, а саме – втрата одиничного біту. Згідно п.1 запропонованої методики для цього випадку спочатку виконується обчислення Δ_0: $\Delta_0 = |n- S_{1s} + S_{1r}| = | 16 – 61 + 51|= = 5$.

Отримане значення вказує на те, що позиція шостого нульового біту із кінця інформаційного блоку відповідає місцезнаходженню позиції m помилки синхронізації. У даному випадку шостому нульовому розряду з кінця відповідає інформаційний біт b_7 і його позиція 7 і є місцем виникнення помилки синхронізації, тобто m=7. Так як кількість надісланих передавачем біт менша за кількість отриманих на приймачеві ($n_R < n$, n_R=15, n=16), то корекція полягає у вставці одиничного біту на позицію m=7, попередньо зсунувши інформаційні біти $\{b_7, b_8, \ldots, b_{15}\}$ на одну позицію праворуч.

Таким чином, скорегований інформаційний блок на приймачі має вигляд $B_R = \{b_1, b_2, \ldots, b_{16}\} = \{0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1\}$ і відповідає інформаційному блоку, що був відправлений передавачем.

2) При надсиланні з передавача блоку біт $B_S = \{b_1, b_2, \ldots, b_8\} = \{0, 1, 1, 0, 1, 1, 1, 0\}$ не виникає помилок синхронізації, проте змінюється значення сьомого біту – з 1 на 0, відповідно до чого на приймачі отримується блок $B_R = \{b_1, b_2, \ldots, b_8\} = \{0, 1, 1, 0, 1, 1, 0, 0\}$. Контрольний код на передавачі обчислюється у вигляді: $p_S = 1$, $S_{1S} = 2+3+5+6+7 = 23$.

Контрольний код на приймачеві обчислюється у вигляді: $p_R = 0$, $S_{1R} = 2+3+5+6 = 16$.

Розмір отриманого приймачем блоку збігається з розміром блоку, надісланого передавачем, проте біти парності та зважені суми різні. Далі обчислюється значення j: $j = S_{1R} - S_{1S} = -7 > -n$, $n = 8$ – одинична помилка, нульовий біт на сьомій позиції необхідно виправити на одиничний.

Скорегований інформаційний блок на приймачі має вигляд $B_R = \{b_1, b_2, \ldots, b_8\} = \{0, 1, 1, 0, 1, 1, 1, 0\}$ і відповідає надісланому передавачем інформаційному блоку.

Таким чином, для досягнення поставленої цілі розроблено метод корекції одиничної помилки синхронізації, оснований на використанні арифметичних операцій.

Окрім помилок синхронізації, запропонований метод дозволяє виправляти звичайні одиничні помилки зміни значення певного біту на протилежне та фіксувати всі аналогічні помилки парної кратності.

Аналіз ефективності

Як зазначалось вище, основними критеріями ефективності засобів корекції помилок є:

- кількість контрольних розрядів;
- клас помилок, що гарантовано виявляються та корегуються;
- обчислювальна складність корекції.

В запропонованому методі контрольний код, що надсилається разом з блоком даних складається з одного біту парності та арифметичної суми порядкових номерів одиниць. Тобто, сумарна кількість контрольних розрядів становить $\lfloor \log_2(1+2+\ldots+n) \rfloor$ біт, де $\lfloor x \rfloor$ – найменше ціле число, яке більше або дорівнює x. Отже, загальна кількість контрольних розрядів становить:

$$e = \left\lfloor \log_2 \sum_{i=1}^{n} i \right\rfloor + 1 = \log_2 \left(\frac{n^2 + n}{2}\right) + 1 \approx 2 \cdot \log_2 n \qquad (3)$$

Характер отриманої залежності вказує на недоцільність використання запропонованого методу у пакетах малої довжини, та зменшення відсотку надлишкової інформації до 10% та менше для пакетів, що мають довжину, більшу за 150 біт.

Основною перевагою запропонованого методу в порівнянні з CRC є гарантування виявлення та виправлення одиничної помилки синхронізації, у той час як CRC не гарантує навіть виявлення помилки. Це може бути проілюстровано наступним прикладом.

Нехай, передається блок, шістнадцятковим відображенням якого є 4003_{16}. Цей блок містить дві серії одиниць: перша містить одну, а друга – дві одиниці. В результаті ділення на передавачі поліному, що відповідає блокові 40030000 на стандартний для CRC [1] утворюючий поліном $x^{16} + x^{15} + x^2 + 1$, формується $7FF8_{16}$. Відповідно, передавач надсилає блок $40037FF8_{16}$. Якщо припустити, що внаслідок помилки синхронізації до другої серії одиниць додається ще одна, то на приймачі отримується блок $80077FF8_{16}$, залишком від ділення якого на утворюючий поліном буде 0, відповідно до чого приймачем отриманий блок визнається коректним.

У разі використання запропонованого методу, до блоку 4003_{16} додається контрольний код $C = \{1, 18\}$, а на приймачеві у разі виникнення аналогічної помилки обраховується код $C = \{0, 22\}$. Обчислення Δ_0 відбувається згідно до п.1 алгоритму виправлення помилок, зв'язаних з виникненням одиничного біту: $\Delta_0 = |n+1+ S_{1s} - S_{1r}| = |16 - 18 + 22| = 9$.

Значення Δ_0 вказує на те, що позиція перед дев'ятим з кінця нульовим розрядом є місцем виникнення помилки синхронізації, відповідно чого з третьої позиції видаляється одиничний біт та відновлюється коректне значення отриманого блоку. Тобто, запропонований метод не тільки виявляє, але й виправляє помилку, яка CRC не виявляється.

Що стосується звичайних помилок, які призводять до зміни значення одного або групи байт на протилежне, то метод практично збігається з відомими методами, в основі яких лежать зважені контрольні суми та гарантує знаходження та виправлення всіх одиничних помилок, оскільки рівняння $S_r + x = S_s$ при невідомому x завжди має одне і тільки одне рішення.

Якщо пошкоджуються два біти, то в такому випадку біт парності залишиться незмінним,

але другі компоненти контрольних кодів будуть різними, отже визначиться помилка, яку не можна виправити.

При виникненні помилок кратності більшої за 2, метод може помилково визнавати отриманий блок коректним, якщо зміна біт не призвела до модифікації біту парності та зваженої суми. Очевидно, що помилка буд-якої непарної кратності призведе до зміни біту парності, отже помилково визнаватись коректними можуть лише блоки, вражені помилкою парної кратності. Відсоток таких помилок для інформаційних блоків різної довжини у разі спотворення чотирьох бітів було аналітично проаналізовано, результати наведено у табл.1.

Табл. 1. Залежність кількості помилок, що не виявляються, від довжини блоку даних

Довжина повідомлення, біт	Відсоток помилок, що не виявляються
128	0,26195%
256	0,13059%
512	0,06519%
1024	0,03257%
2048	0,01628%
4096	0,00819%

Складність корекції помилок з застосуванням запропонованого методу менша в порівнянні з лінійними кодами [4]. Час виправлення помилки не залежить від довжини інформаційного блоку.

Висновки

В результаті проведених досліджень запропоновано метод корекції одиночної помилки синхронізації.

Відмінністю запропонованого методу є те, що в його основі покладено використання зважених контрольних сум, що дозволяє спростити, в порівнянні з лінійними кодами, локалізацію помилки, що важливо для систем обміну даними між компонентами комп'ютерних систем.

Важливою перевагою запропонованого методу в порівнянні з відомими [4] є те, що він забезпечує виправлення помилок, пов'язаних як з втратою нуля, так і з втратою одиниці в блоці даних.

Запропонований метод, крім помилок синхронізації, забезпечує виправлення одиничних помилок в рамках моделі двійкових симетричних каналів.

Запропонований метод орієнтовано для швидкісних каналів обміну даних між компонентами комп'ютерних систем широкого призначення.

Список літератури

1. Агуров П.В. Интерфейс USB. Практика использования и программирования / П.В. Агуров – СПб.:БХВ-Петербург, 2005.- 576 с.
2. Klove T. Error Detecting Codes: General Theory and Their Application in Feedback Communication Systems / T. Klove, V. Korzhik.- Norwell, MA: Kluwer, 1995. – 433 p.
3. Zumbragel J. On the Pseudocodeword Redundacy of Binary Linear Codes / J. Zumbragel, V. Skachek, M.F. Flanagan // IEEE Trans. of Information Theory.-2012.-Vol.58.- № 7.-P.4848-4861.
4. Марковский А.П. Использование взвешенных контрольных сумм для обнаружения ошибок в линиях передачи с асинхронным кодированием данных/ А.П. Марковский , Пуя Солеймани Нежадиан, Мулки Ахмед Яссин Ал Бадайнех. //Современные информационные и электронные технологи: 9-тая междунар. науч.-техн. конф., 19-22 мая. 2008 р.: тезисы докл. – Одесса., 2008.- С.201.

УДК 004.724.4

ГУСЕВ Е.И.

ИССЛЕДОВАНИЕ ОБЛАСТИ ПРИМЕНЕНИЯ НЕБЛОКИРУЮЩЕГО АЛГОРИТМА ФИКСАЦИИ РАСПРЕДЕЛЁННЫХ ТРАНЗАКЦИЙ

Предложена математическая модель, позволяющая сравнить традиционный алгоритм 2х-фазного блокирования с неблокирующим алгоритмом фиксации распределённых транзакций в СУБД. Определение параметров трафика к СУБД, при которых целесообразно использование предлагаемого алгоритма способно существенно повысить эффективность использования распределённых СУБД.

There are proposed a mathematical model that allows to compare the traditional algorithm for 2-phase locking with a non-blocking commit algorithm for distributed transactions in RDBMS. Defining the parameters of the traffic to the database in which it is expedient to use the proposed algorithm can significantly improve the efficiency of distributed DBMS.

1. Введение

Неблокирующий алгоритм фиксации распределённых транзакций [1] представляет собой реализацию принципов ACID [2] для распределённых СУБД [3,4,5]. В исследовании эффективности алгоритма можно выделить основные направления:

- исследование надёжности (устойчивости) алгоритма;
- анализ производительности;
- исследование накладных расходов алгоритма;
- исследование устойчивости системы на коллапс при превышении граничных значений интенсивностей.

Для моделирования работы алгоритма выделим параметры системы, влияющие на её эффективность. Это $t_{лIO}$ – среднее время локального ввода-вывода (с учётом индексаций, кэширования и других техник), t_c – время пересылки сетевого пакета. Также необходимо учитывать граничные показатели работы системы:

$v_{I/O}$– максимальная интенсивность ввода-вывода (шины);

v_{CPU} – максимальная интенсивность ввода-вывода;

v_c – максимальная интенсивность сети.

После превышения какой-то из максимальных интенсивностей система приходит к коллапсу. Стоит правда указать на то, что ν_c может зависеть от интенсивностей $\nu_{I/O}$ и ν_{CPU}. При моделировании стоимостной модели это необходимо учитывать, генерируя пропорционально интенсивности сетевого трафика служебный CPU и I/O трафик.

Можно выделить следующие виды коллапса по:

- блокировкам;
- времени сети;
- вводу-выводу;
- нехватке ресурсов процессора;
- реализации архитектуры СУБД.

Рассмотрев при каких параметрах трафика и применении какого алгоритма возникает какой тип коллапса, можно сравнивать эффективности алгоритмов на одной системе. Например, на заданной системе при входящей интенсивности возникновения конфликта 1000 заявок в секунду применение традиционных алгоритмов приводит к коллапсу по блокировкам, а применение неблокирующего алгоритма фиксации распределённых транзакций не приводит. Увеличение входящей интенсивности возникновения конфликта до 50000 заявок в секунду приведет к коллапсу и в предлагаемом алгоритме. Или перефразировав: граничная пропускная способность системы при традиционном алгоритме: 1000 взаимоблокирующих заявок, а при неблокирующем – 50000.

С другой стороны критерием эффективности может быть среднее время обработки заявки или средняя загрузка системы. Естественно надо понимать, что расчет средних показателей имеет смысл только если коллапс не наступает.

В данной статье мы проведем исследование коллапса блокировок с целью их исключения.

2. Постановка задачи

Предположим, мы имеем систему, на которую поступают заявки с интенсивностью *v*. Система имеет множество общих ресурсов. Обработка заявки занимает время *t*. Если заявка, поступившая в систему, обратившись к общему ресурсу, обнаруживает что общий ресурс занят, она ожидает его освобождения. Если более одной заявки ожидает общего ресурса, то доступ к ресурсу осуществляется в порядке очереди FIFO.

Можно разложить общий поток заявок на *n* непересекающихся потоков, каждый к своему ресурсу и со своей интенсивностью.

Обозначим интенсивность заявок к максимально востребованному ресурсу как $v_к$. Условимся, что вероятность конфликта к любому ресурсу подчиняется закону Пуассона в течение определённого промежутка времени Δt. Тогда $v_к=const$ в течение этого Δt.

Пусть время обработки первой заявки – *t*. Если образовалась очередь из 2 заявок, то время обработки второй заявки это *t* плюс время ожидания в очереди. (обозначим t_x, причем t_x меньше *t*, т.к. вторая заявка поступила после первой) : Если очередь из трех заявок – то аналогично – время обработки 3-й заявки t_x+2t.

Если образовалась очередь из n заявок, то время обработки n –ой заявки равно $t_x+(n-1)t$, время блокирования ресурса – в таком случае – это $t_x+(n-1)t$, при условии что за время $t_x+(n-1)t$ заявок к этому ресурсу не поступит. Если за время $t_x+(n-1)t$ к ресурсу поступит менее чем *n* заявок – то очередь не увеличится, а если более – то увеличится. Если очередь увеличится – то время блокирования ресурса после прохождения времени $t_x+(n-1)t$ также.

Для удобства анализа выделим три состояния системы:

1. Стационарное – $v_к<1/t$ и вероятность превышения $1/t \sim 0$. Очереди не образуется.

2. Переходное – $v_к<1/t$ но вероятность превышения $v_к$ значения $1/t$ ненулевая, то есть отклонение $v_к$ от математического ожидания превышает $1/t$. В системе образуются очереди.

3. Коллапс – $v_к<1/t$. Очередь нарастает и время обработки каждой следующей заявки приходящей в очередь больше чем предыдущей.

Если очередь образовалась и $v_к<1/t$ – то очередь начинает сокращаться, но время обработки заявки в этом случае будет включать ожидание в очереди, то есть в переходном состоянии среднее время отклика увеличивается пропорционально средней длине очереди и зависит от дисперсии.

Таким образом, можно сделать следующие выводы:

1. Уменьшая время обработки заявки, мы при большей интенсивности конфликта останемся в рабочем состоянии.

2. В переходном состоянии возможно существенное (*n* кратное) увеличение времени отклика в случае образования очереди.

3. Пиковое кратковременное увеличение интенсивности способно создать очередь, что скажется на времени отклика последующих заявок в течение еще некоторого времени после завершения пика. Уменьшая время обработки заявки, мы снижаем вероятность увеличения времени отклика при возникновении пика. Т.е. Повышаем устойчивость системы.

4. Расчет $Q_к$ – удельной продолжительности задержек связанных с конфликтом за локальные ресурсы (данные) имеет смысл только в переходном состоянии. В стационарном состоянии она равна 0, а в коллапсе – бесконечность.

3. Расчет времени обработки заявки

Обозначим – все действия системы кроме ожиданий по причине простоя клиента за промежуток времени *dt* величиной *DB*. Также будем полагать, что интенсивность конфликта постоянна в течении этого времени *dt* (стационарный Пуассоновский поток), а значит вероятности событий конфликта *i*-заявок за ресурс ($p_0,p_1,p_2...p_i$) можно будет рассчитывать по закону Пуассона.

Основная формула для расчёта *DB* при асинхронной фиксации распределённых транзакций без блокирования общего ресурса равна:

$$DB = \int_0^t (v(p_0\ t_у + \sum_1^\infty p_{i.н}) + \frac{t_{фк}}{\tau} + \frac{t_{ск}}{\tau} + Q_{к.} + Q_{ф.п.} + Q_{а.})dt,$$

где: τ – время обхода круга;

v – интенсивность заявок;

$t_у$– время обработки в случае отсутствия конфликта;

$t_{iн}$ – время обработки в случае *i* конфликтов за время τ;

$t_{фк}$ – время формирования курьерского пакета;

$t_{ск}$- время сверки пришедшего курьерского пакета с транзакциями, которые прошли с момента предыдущей сверки на локальной ноде;

$Q_к$ – удельная продолжительность задержек связанных с конфликтом за локальные ресурсы (данные);

Q_a – удельные задержки вызванные конфликтом за внутренние ресурсы системы (например ожидание диска, шины ввода-вывода, обновления мета данных системы, конфликтов за доступ к общей оперативной памяти);

$Q_{ф.п}$ – удельная продолжительность задержек фоновых процессов определяется отношением времени работы остальных фоновых процессов, реализующих корректную работу СУБД к прошедшему времени.

Задержки $Q_{ф.п}\,dt$ – происходят в фоновых процессах СУБД и не включаются в суммарное время отклика. Этим показателем нельзя пренебрегать при расчёте коллапса по вводу-выводу или по CPU, однако, при стационарной работе он на время отклика не влияет.

Задержки $Q_a\,dt$ зависят от архитектурных особенностей СУБД. По этим задержкам также могут возникать коллапсы, однако мы не имеем возможности их моделировать, поскольку они очень сильно зависят от инженерных решений разработчика СУБД. Во многих СУБД также есть аппаратные средства для мониторинга и снижения такого рода задержек. Мы будем пренебрегать ими в идеальной модели, а при реальном моделировании рассчитаем погрешность, которую влечет данное допущение. Иными словами, область применения данных алгоритмов будет такой, при которых этих задержек нет или они существенно меньше тех, которые мы учитываем. Таким образом:

$$SE = \sum \int_0^t (v(p_0\ t_у + \sum_1^\infty p_{i.н}) + \frac{t_{фк}}{\tau} + \frac{t_{ск}}{\tau} + Q_{к.})dt.$$

Если система находится в стационарном состоянии – то величиной $Q_к$ можно пренебречь. В этом случае получаем:

$$SE = \int_0^t (v(p_0\ t_у + \sum_1^\infty p_{i.н}) + \frac{t_{фк}}{\tau} + \frac{t_{ск}}{\tau})dt.$$

Главным преимуществом предлагаемого алгоритма является то, что в случае его неэффективности он в неэффективной части легко может быть заменён на традиционные блокирующие алгоритмы [6]. Учитывая это, основной задачей становится определение области применения. То есть разбиение реального трафика на классы, определение классов, для которых применение алгоритма неэффективно и получение для реального трафика результирующего алгоритма работы менеджера транзакций включающего в себя обработку как традиционными методами

Следует заметить что v – это суммарная интенсивность к каждому общему ресурсу. Обозначим интенсивность к каждому общему ресурсу как $\lambda_к$. Тогда:

$$SE = \int_0^t (\sum_{k=1}^\infty (\lambda_k \left(p_0 t_у + \sum_1^\infty p_{i.н} \right) + \frac{t_{фк}}{\tau} + \frac{t_{ск}}{\tau})dt$$

или

$$\frac{d(SE)}{dt} = \left(\sum_{k=1}^\infty \lambda_k \left(p_0 t_у + \sum_1^\infty p_{i.н} \right) + \frac{t_{фк}}{\tau} + \frac{t_{ск}}{\tau} \right),$$

где: $p_0 = e^{-\lambda_k \tau},\ p_i = \frac{(\lambda_k \tau)^i e^{-\lambda_k \tau}}{i!}$.

Можно расписать для каждого общего ресурса:

$$\frac{d(SE_k)}{dt} = \lambda_k (p_0 t_у + \sum_1^\infty p_{i.н} + \frac{t_{фк}}{\tau} + \frac{t_{ск}}{\tau});$$

где: $\frac{d(SE)}{dt} = \sum_{k=1}^\infty \frac{d(SE_k)}{dt}$.

4. Расчёт времени обработки заявки с учётом трафика

Наиболее простой способ классификации – это следовать тем типам команд, которые существуют в языке SQL, а именно: чтение данных (SELECT),чтение предполагающее блокирование (SELECT FOR UPDATE), добавление данных (INSERT),изменение существующих данных (UPDATE), удаление данных (DELETE). Для каждого класса команд распишем (p_0,p_1,p_2…p_i) и времена t_y и $t_{iн}$. Следует указать что расчёт t_y и $t_{iн}$. Для каждого случая следует проводить эмпирическим путём, поскольку разброс

значений, вызванный эффективностью индексации данных и процентом кэширования, может, быть отличаться на порядок и более. Стоит скорее ориентироваться на $t_{лIO}$ предполагая доступ к данным через индекс. Для OLTP систем такое допущение верно. В хранилищах при фул-сканах в грид-системах подход в принципе другой – предполагающий распараллеливание в каждой SQL команде. Имеем:

$$\frac{d(SE_k)}{dt} = \lambda_k(p_0 t_{\text{у}} + \sum_1^\infty p_i t_{i.\text{н}} + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau});$$

где: p_0=1, p_i=0 (i=1,2,3,4…).

Для чтений (sel), поскольку они неблокирующие $t_{усп}=t_{\text{лIO}}$ тогда,

$$\frac{d(SE_{k.sel})}{dt} = \lambda_{k.sel}(t_{лIO} + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau});$$

Для вставок (ins), поскольку они также неблокирующие – аналогично:

$$\frac{d(SE_{k.ins})}{dt} = \lambda_{k.ins}(t_{лIO} + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau});$$

Для операций UPDATE в случае неуспеха необходимо воспроизвести и применить столько изменений, сколько конфликтов произошло. Воспроизведение изменений на момент начала транзакции займёт $t_{лIO}$. Каждое изменение на прошедший конфликт осуществляется в памяти, а значит i-кратное разрешение конфликта менее продолжительно чем $t_{лIO}$. Таким образом можно считать что $t_у=t_{лIO}$, а $t_н=2t_{лIO}$. Также распишем вероятности возникновения 0 конфликтов, 1 конфликта, 2х и т. д.

$$p_0 = e^{-\lambda_{k.upd}\tau},$$

$$p_i = \frac{(\lambda_k\tau)^i e^{-\lambda_k\tau}}{i!} \quad (i = 1,2,3,4,\dots)$$

Имеем:

$$\frac{d(SE_k)}{dt} = \lambda_k(p_0 t_{\text{у}} + \sum_1^\infty p_i t_{i.\text{н}} + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau} =$$

$$\lambda_k(p_0 t_{лIO} + 2t_{лIO}\sum_1^\infty p_i \quad + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau} =$$

$$\lambda_k(p_0 t_{лIO} + 2t_{лIO}\sum_1^\infty p_i \quad + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau} =$$

$$\lambda_k(e^{-\lambda_{k.upd}\tau} t_{лIO} + 2t_{лIO}(1 - e^{-\lambda_{k.upd}\tau}) + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau}) =$$

$$\lambda_k(2t_{лIO} - t_{лIO}e^{-\lambda_{k.upd}\tau} + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau}) \approx$$

$$\lambda_k\left(1{,}5t_{лIO} + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau}\right).$$

Коэффициент 1.5 в данном случае условный, поскольку принимается пятидесятипроцентная вероятность возникновения конфликта. В принципе такое упрощение снимает зависимость от того что поток должен быть пуассоновским (Что существенно расширяет сферу применения алгоритма). Также, если провести более точные измерения этих времён обработки успешной и неуспешной заявки, мы получим то, что экспоненциальная компонента будет складываться с постоянными временами обработки, внося погрешность тем большую, чем больше соотношение интенсивности потока к общему k-тому ресурсу ко времени обхода круга. Аналогично выглядит и ситуация с DELETE и SELECT FOR UPDATE.

Рассчитаем для 2PL(2 Phase Locking Algorithm):

$$\frac{DB}{dt} = 2\nu t_{\text{с}} + Q_{\text{к}} + Q_{\text{ф.п}} + Q_{\text{а}};$$

$$\frac{SE}{dt} = 2\nu t_{\text{с}}.$$

Сравним производные SE для обоих алгоритмов. Поскольку в точке t=0 значения SE в обоих случаях =0 и совпадают, и обе производные всегда больше 0 – то сравнение производных можно применять вместо сравнения SE. Для случаев SELECT и INSERT:

$$\lambda_k\left(t_{лIO} + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau}\right) < 2\nu t_{\text{с}}$$

$$\lambda_k t_{\text{лок}IO}\left(\quad 1 + \frac{t_{\text{фк}}}{\nu\tau t_{лIO}} + \frac{t_{\text{ск}}}{\nu\tau t_{лIO}}\right) < 2\nu t_{\text{с}};$$

$$1 + \frac{t_{\text{фк}}}{\nu\tau t_{лIO}} + \frac{t_{\text{ск}}}{\nu\tau t_{лIO}} \approx 1;$$

$$\lambda_k t_{лIO} < 2\nu t_{\text{с}}.$$

В данном случае мы пренебрегли выражением в скобках, поскольку τ мы можем увеличивать в широких пределах. Распишем для случая UPDATE:

$$\lambda_k(2t_{лIO} - t_{лIO}e^{-\lambda_{k.upd}\tau} + \frac{t_{\text{фк}}}{\nu\tau} + \frac{t_{\text{ск}}}{\nu\tau}) < 2\nu t_{\text{с}};$$

$$2\lambda_k t_{лIO}\left(1 - \frac{e^{-\lambda_{k.upd}\tau}}{2} + \frac{t_{\text{фк}}}{2\nu\tau} + \frac{t_{\text{ск}}}{2\nu\tau}\right) < 2\nu t_{\text{с}};$$

$$\frac{t_{\text{фк}}}{2\nu\tau} + \frac{t_{\text{ск}}}{2\nu\tau} - \frac{e^{-\lambda_{k.upd}\tau}}{2} \approx 1;$$

$$2\lambda_k t_{лIO} < 2\nu t_{\text{с}}.$$

С учетом того, что увеличивая время обхода круга τ отношение $1/\nu\tau$ можно сделать сколь угодно малым, приравниваем выражение в скобках к 1. Поскольку λ_k всегда меньше *v,* неравенство считаем доказанным. Аналогично выглядит и ситуация с DELETE и SELECT FOR UPDATE.

Таким образом, мы показали, что среднее время обработки заявки (а это *SE* разделённое на количество сессий) при использовании неблокирующего алгоритма меньше. Причём учитывая то, что t_c всегда больше $t_{лок.IO}$ на порядок (а то и два) эффективность неблокирующего алгоритма фиксации распределённых транзакций на порядок эффективнее традиционного алгоритма 2PL[6] при сравнении возможности коллапса по блокировкам.

Дальнейшее усовершенствование алгоритма лежит в минимизации задержек на выполнении tфк и tск.

Список литературы

1. Гусев Е.И. Исключение блокирования общего ресурса в распределённых системах / Гусев Е.И.; Кулаков А.Ю. // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. – 2011. Випуск 54. – С.162 – 166.
2. Härder, Theo; Reuter, Andreas (December 1983). "Principles of Transaction-Oriented Database Recovery" (PDF). ACM Computing Surveys 15 (4): 287–317. doi:10.1145/289.291.
3. Open Group Standard DRDA, Version 5, Volume 1: Distributed Relational Database Architecture (DRDA) ISBN: 1-931624-91-7 Document Number: C112
4. Open Group Standard DRDA, Version 5, Volume 2: Formatted Data Object Content Architecture (FD:OCA) ISBN: 1-931624-92-5 Document Number: C113
5. Open Group Standard DRDA, Version 5, Volume 3: Distributed Data Management (DDM) Architecture ISBN: 1-931624-93-3 Document Number: C114
6. Philip A. Concurrency Control and Recovery in Database Systems/ Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman// Addison Wesley Publishing Company, 1987, ISBN 0-201-10715-5

УДК 621.384

КУЦ В.Ю.

СПОСОБИ ОБЧИСЛЕННЯ ФАЗОВИХ ХАРАКТЕРИСТИК ЦИКЛІЧНИХ СИГНАЛІВ

Розглянуто та проаналізовано способи оцінки фазових характеристик циклічних сигналів, та обґрунтовано використання методів статистичної обробки кутових даних для для підвищення точності їх визначення в умовах дії шуму.

The ways of cyclical signals phase characteristics estimation are reviewed and parsed, and usage of methods of the angular data statistical processing for increasing of their definition accuracy in conditions of a noise effect is justified.

Значна частина задач фазових вимірювань пов'язана з аналізом широкого класу циклічних сигналів [1]. Такі сигнали виникають під час аналізу циклічних процесів, які мають місце в багатьох технічних, біологічних та фізичних системах. Ці процеси характеризуються наявністю в них повторювань, а отже і можливістю виділення циклів, тобто сукупності взаємопов'язаних елементів, що утворюють їх кругообіг протягом певного проміжка часу – періоду. Значення періоду в процесі зміни циклів може змінюватись і коливатись в певних межах, розділені періодом значення циклічного процесу можуть не співпадати абсолютно. Незмінним лишається тільки повторюваність в часі чи просторі їх найбільш характерних ознак. Прикладами циклічних процесів можуть бути: в енергетиці – добова та сезонна зміни енергоспоживання, витрати енергоносіїв, в метеорології – зміна напрямку вітру, інтенсивності опадів, в медицині – фізико-біологічні процеси в організмі людини, у телекомунікації – трафік в різні пори року та години доби і т.п.

В роботі [1] для циклічних сигналів обгрунтовано введення понять "фазова характеристика сигналу" та "різниця фазових характеристик сигналів". В технічних системах такі сигнали породжуються циклічними процесами різної фізичної природи (електричними, оптичними, акустичними, і т.і.). Тому аргументами в таких сигналах можуть бути як час, частота, так і координати простору. Фазові характеристики циклічних сигналів функціонально пов'язані зі своїми аргументами (зокрема з часом) залежностями, які зазвичай значно складніші за лінійну, яка має місце для гармонічних сигналів.

У вимірювальній техніці, оптиці, радіолокації, енергетиці, зв'язку тощо існує значна кількість задач, яка може бути розв'язана шляхом аналізу та визначення фазових характеристик циклічних сигналів (далі фазових характеристик сигналів – ФХС). До таких задач належать, наприклад, задачі детектування модульованих сигналів, аналізу інтерферограм, визначення показників якості електроенергії, виділення радіосигналів на фоні завад та оцінки їх фазових характеристик, оцінки стабільності фази високостабільних джерел гармонічних сигналів і т. і.

Метою статті є аналіз способів розгортання ФХС і їх ефективності для дослідження циклічних сигналів в умовах дії шумів і завад.

Розв'язок. Циклічний інформаційний сигнал може бути представлений у загальному виді дійсною функцією часу виду [2]

$$u(t)=U(t)\cos[\Phi(t)],\ t\in(-\infty,\infty),\ \frac{d\Phi(t)}{dt}>0, \quad (1)$$

де $U(t)$ – амплітудна характеристики сигналу (АХС),

$\Phi(t)$ – фазова характеристика сигналу.

Однозначне визначення функцій $U(t)$ та $\Phi(t)$ можливе, наприклад, на основі застосування до функцій виду (1) перетворення Гільберта (ПГ) [1]. Це перетворення визначає спряжений по Гільберту сигнал $\hat{u}(t)$. ПГ є основою математичного апарата, який дозволяє виконувати аналіз характеристик циклічних сигналів виду (1) і визначити їх амплітудну характеристику $U(t)=\sqrt{(u(t))^2+(\hat{u}(t))^2}$ та дробову частину ФХС – функцію $\varphi(t)=\Phi(t)\bmod 2\pi$, $\varphi(t)\in[0,2\pi)$ [1]

$$\varphi(t)=\mathbf{L}[\hat{u}(t),u(t)]=arctg\left(\frac{\hat{u}(t)}{u(t)}\right)+\frac{\pi}{2}\{2-[sign(\hat{u}(t))]\cdot[1+sign(u(t))]\}, \quad (2)$$

де $sign(x)$ – знакова функція,

L – оператор визначення дробової частини ФХС.

Саме повторюваність дробової частини ФХС, тобто її частини в межах напівінтервалу $[0,2\pi)$ визначає циклічність сигналу. Наприклад, для гармонічного сигналу з частотою f_0 функція (2) уявляє періодичну пилкувату функцію з періодом $T_0 = f_0^{-1}$, що має вид показаний на рис. 1.

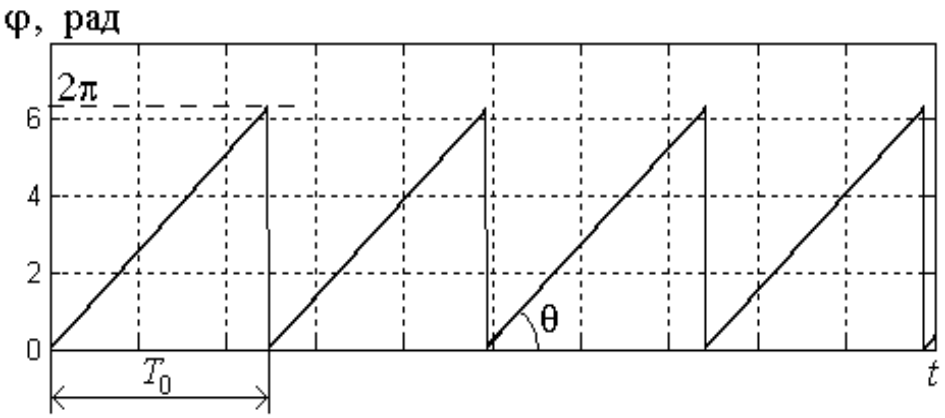


Рис. 1 Приклад графіка функції $\varphi(t)$ ділянки гармонічного сигналу

Функція $\varphi(t)$ періодично змінюється в часі від 0 до 2π з періодом вхідного сигналу T_0. Перехід від максимального до мінімального значень відбувається стрибкоподібно. Кут нахилу θ функції $\varphi(t)$ до осі часу в межах між двома суміжними стрибками однозначно зв'язаний з частотою сигналу: $f_0 = (2\pi)^{-1} tg\theta$, тобто зміні частоти в інтервалі $(0,\ \infty)$ відповідає зміна кута θ в інтервалі $(0,\ 0{,}5\pi)$.

Для реалізації фазового методу вимірювання важливим є не весь вимірювальний сигнал, а його фазова характеристика $\Phi(\nu)$, яка безпосередньо пов'язана з певною фізичною величиною ν (чи вектором). Традиційні алгоритми оцінювання фази сигналу дозволяють визначити лише дробову частину ФХС $\varphi(\nu) = \Phi(\nu) \bmod 2\pi$. В багатьох прикладних вимірювальних задачах необхідним і важливим є отримання всієї ФХС, тобто розгорнутої характеристики. До таких задач належать, наприклад, задачі побудови зображення в магнітному резонансному методі, томографія і спектроскопія, обробка інтерферограм, опрацювання часових рядів тощо. Задача визначення фазових зсувів (характеристик) сигналів за межами півінтервалу $[0,2\pi)$ відома як задача розгортання ФХС або задача усунення багатозначності фазових вимірювань. Аналіз сучасної науково-технічної літератури свідчить, що питання розгортання фази в останнє десятиліття перетворилось на одне з найбільш досліджуваних питань в області цифрової обробки зображень. Цій тематиці присвячено численні наукові праці, в яких розглянуто різні способи та алгоритми розгортання фази для різних практичних застосувань. Проведемо їх аналіз з метою узагальнення способів розгортання фази вимірюваного сигналу.

Способи розгортання фази сигналу систематизовано на рис. 2.

В загальному випадку фазу сигналу можна розглядати як функцію, аргументами якої є час t, частота f та координати тривимірного простору x, y, z, тобто $\Phi(t, f, x, y, z)$. Безпосередньому вимірюванню через вхідний сигнал доступна лише її частина $\varphi(t, f, x, y, z) = \Phi(t, f, x, y, z) \bmod 2\pi$, тобто дробова частина ФХС.

За видом області розгортання способи розгортання фази можна поділити на три групи: способи розгортання фази в частотній області, які дозволяють визначати фазочастотні характеристики електричних кіл та середовищ поширення сигналів; способи розготрання фази в одно-, дво-, та тривимірному просторі, які застосовують для опрацювання, наприклад, інтерферограм; способи розготрання фази сигналів, представлених часовими рядами.

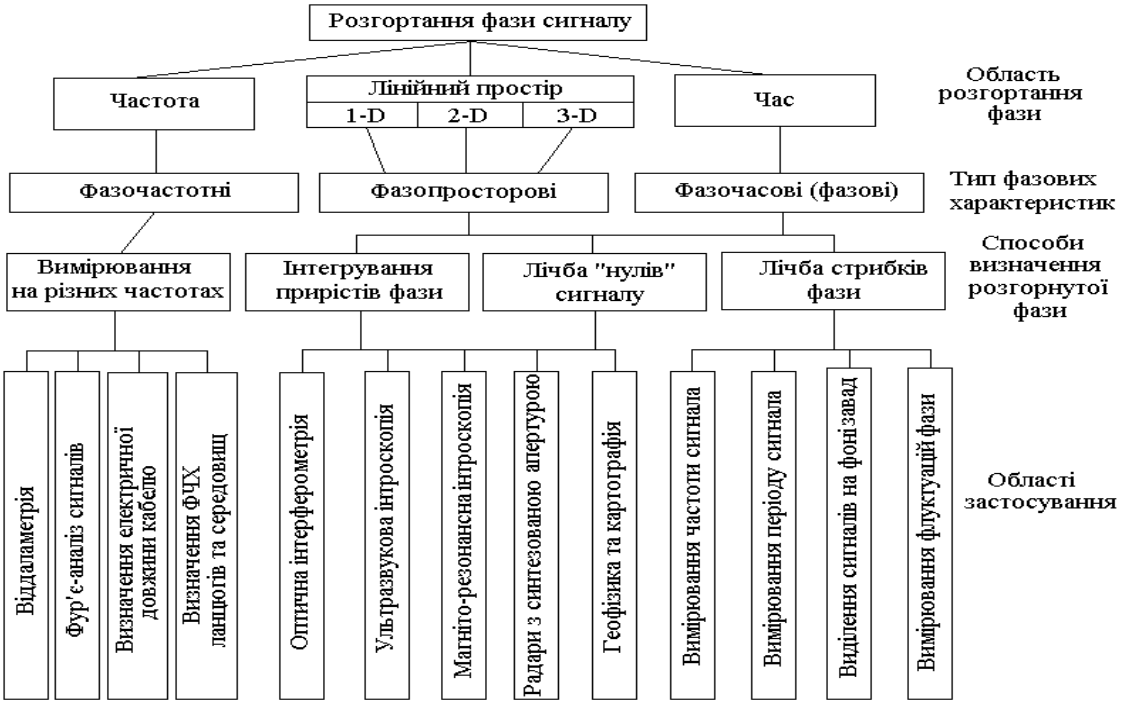


Рис. 2 Систематизація способів розгортання фази сигналу

У першому випадку отримують множину значень фази на певних m частотах, тобто $\{\varphi(f_j),\ j=\overline{1,m}\}$, у другому – значення фази у вузлах сітки, у одновимірному випадку це, наприклад, множина $\{\varphi(j\Delta x)=\varphi[j],\ j=\overline{1,m}\}$, де Δx крок сітки, у третьому – це значення фази у відповідні моменти часу – $\{\varphi(j\Delta T)=\varphi[j],\ j=\overline{1,M}\}$, де ΔT – крок дискретизації сигнала під час отримання часового ряду. Відповідні типи фазових характеристик називатимемо фазочастотними, фазопросторовими та фазочасовими, або просто фазовими.

Фазочастотні характеристики сигналів з областю представлення в інтервалі що перевищує 2π широко застосовуються в задачах віддалеметрії [3], Фурьє-аналізі сигналів [2], визначенні електричої довжини кабелю, в радіолокації [5].

Розподіл фази у просторі застосовують при розшифровці інтерферограм [6], результатів ультразвукової та магніторезонансної інтроскопії в медицині та неруйнівному контролі, при відтворенні рельєфу місцевості та будові форми земної кори в геофізиці та картографії тощо.

Особливості розгортання фази полягають у наступному. Спосіб інтегрування (накопичення) приростів фази ґрунтується на неперервному стеженні за фазою (фазовим зсувом сигналів), яка змінюється в часі чи просторі. За умови, що в початковій точці аналізу ($j=1$) фаза дорівнює значенню $\varphi[1]\in[0,2\pi)$, під час послідовного переходу від першої до j-тої точки простору накопичуються значення

$$\Phi[j]=\varphi[1]+\sum_{k=2}^{j}\Delta\varphi_k=\varphi[1]+\sum_{k=2}^{j}(\varphi[k]-\varphi[k-1])\bmod 2\pi,$$

$$|\Delta\varphi_k|<\frac{\pi}{2}. \qquad (3)$$

Суть способу рахунку числа «нулів» сигналу [7] полягає в тому, що нульові значення знакозмінного сигналу збігаються в часі (чи просторі) з межею фазових циклів, тобто зі значеннями $\Phi(t)\equiv 0\bmod 2\pi$ (іншими словами значення функції $\Phi(t)$ порівнювані з нулем за модулем 2π). Кожний перехід сигналу через нуль відповідає зміні дробової частини ФХС на 2π, що дозволяє побудувати східчасту функцію з величиною сходинок, яка дорівнює 2π. Додавання такої східчастої функції до значення $\varphi(t)=\Phi(t)\bmod 2\pi$ дозволяє виконати розгортання фази. Той факт, що східчаста функція і дробова частина фази визначаються неузгоджено, вимагає додаткових заходів, які спрямовані на уникнення грубих помилок визначення $\Phi(t)$ в околі значень $\Phi(t)\equiv 0\bmod 2\pi$.

За відсутності шумів у досліджуваному сигналі існує можливість узгодженого з $\varphi(t)$ визначення східчастої функції за аналізом стрибків функції $\varphi(t)$. В цьому випадку маємо

$$\Phi(t) = \varphi(t) + \mathbf{K}[\varphi(t)], \qquad (4)$$

де $\mathbf{K}[\cdot]$ – оператор східчастої функції, що визначається за результатами аналізу $\varphi(t)$ і усуває стрибки ФХС в точках $2\pi n$, $n \in N$. В роботі [8] оператор $\mathbf{K}[\cdot]$ визначається як

$$\mathbf{K}[\varphi(t)] = \begin{cases} \mathbf{K}[\varphi(t=0)] = 0, \\ \mathbf{K}[\varphi(t)], \ \lim\limits_{\Delta t \to 0} [\varphi(t-\Delta t) - \varphi(t)] = 0, \\ \mathbf{K}[\varphi(t)] + 2\pi, \ \lim\limits_{\Delta t \to 0} [\varphi(t-\Delta t) - \varphi(t)] = 2\pi. \end{cases}$$

Процес розгортання фази гармонічного сигсигналу згідно (4) наведено на рис. 3.

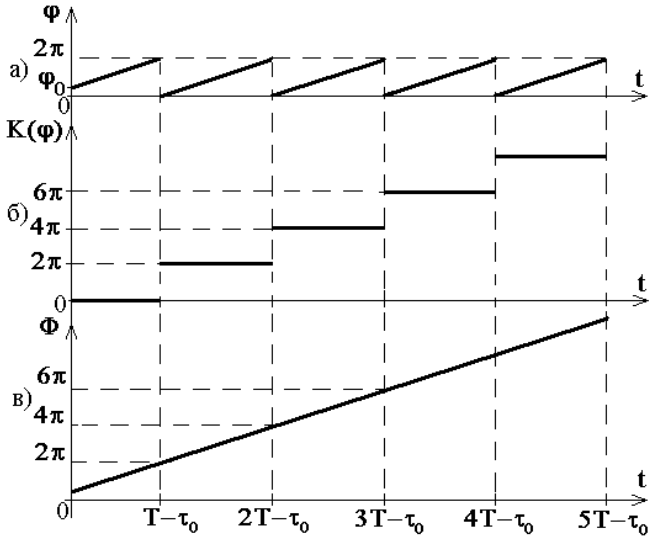


Рис. 3 Графічне представлення процесу розгортання фази гармонічного сигналу за способом лічби стрибків дробової чистини ФХС

Оскільки функція $\mathbf{K}[\varphi(t)]$ отримується з функції $\varphi(t)$, обидві складові в рівнянні (4) є узгодженими, що виключає виникнення грубих помилок визначення розгорнутої ФХС.

Наведений приклад розгортання фази гармонічного сигналу лише ілюструє принцип цієї процедури. Ідея розгортання ФХС є тривіальною і не здається такою продуктивною у випадку, коли йдеться про гармонічні сигнали. Проте для сигналів більш складної структури визначення ФХС надає значно ширші можливості, зокрема можливість статистичного аналізу фазових зсувів сигналів, дослідження динаміки розвитку фази сигналів, аналізу флуктуацій фази сигналів, аналізу фази сигналів у присутності завад тощо.

Слід зазначити, що в практичних дослідженнях зустрічаються і інші, більш складні, випадки відновлення фази, наприклад, коли фаза залежить і від часу, і змінюється у просторі. Цьому випадку відповідають задачі визначення ФХС, в яких гармонічний сигнал,

$$u_1(t,x) = U_1 \cos(2\pi f t - kx - \varphi), x, t \in (-\infty, \infty), \qquad (5)$$

де $k = 2\pi/\lambda$ – хвильове число, що визначає просторовий період коливання;

$\lambda = V/f$ – довжина хвилі,

поширюється у середовищі вздовж координати x з фазовою швидкістю V.

Для сигналу (5) ФХС становить $\Phi_1(t,x) = (2\pi f t - kx - \varphi)$, а її дробова частина – $\varphi_1(t,x) = (2\pi f t - kx - \varphi) \bmod 2\pi$. Функції $\varphi_1(t,x)$ та $\Phi_1(t,x)$ залежать від двох аргументів – часу t і просторової координати x.

Хоча принципи розгортання фази лишаються незмінними незалежно від області її представлення, розгортання фази циклічних сигналів у присутності шумів та завад в часовій області висуває більш жорсткі вимоги щодо щвидкодії і точності способів і алгоритмів його реалізації. В цьому випадку коректне визначення ступінчастої функції $\mathbf{K}[\varphi(t)]$ і відповідно ФХС можливе у випадку, коли $\Phi(t)$ є гладкою функцією, тобто функцією, у якої кожне значення аргументу є глад-

кою точкою [9]. В свою чергу гладка точка функції $f(x)$ визначається як значення аргументу x, для якого виконується умова

$$\lim_{|h|\to 0}\frac{|f(x+h)+f(x-h)-2f(x)|}{|h|}=0. \qquad (6)$$

Умова (6) не виконується у випадку, коли аналізований сигнал спостерігається на фоні адитивного шуму навіть для значних відношень сигнал/шум. В результаті зростає імовірність виникнення грубих помилок визначення ФХС. Проілюструємо цей висновок на прикладі задачі визначення ФХС гармонічного сигналу на фоні адитивного гауссівського шуму з нульовим математичним сподіванням та дисперсією σ^2. На рис. 4,а наведено приклад графіку фрагменту адитивної суміші гармонічного сигналу і гауссівської завади з відношенням сигнал/шум=10 (по потужності), а на рис. 4,б, в – відповідно значення дробової частинии ФХС цієї суміші та її різниці відносно дробової частини гармонічної складової суміші.

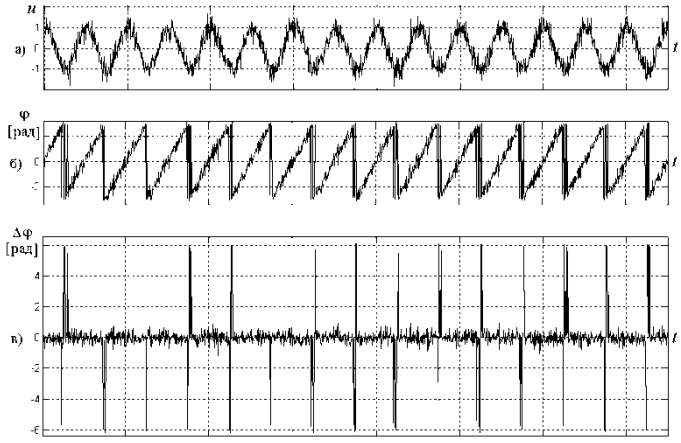


Рис. 4 Графіки вхідного сигналу (а), значень дробової частини ФХС (б) та наявної в ній імпульсної завади $\Delta\varphi$ ***(в)***

З рис. 4 видно, що наявність адитивної завади у сигналі приводить до спотворення дробової частини ФХС гармонічної складової цього сигналу, яке полягає в тому, що: 1) ділянки між стрибками фази спотворені адитивною завадою; 2) з'являється імпульсна завада в околі стрибків фази інформативної частини сигналу. В цьому випадку розгортання ФХС оператором $\mathbf{K}[\cdot]$ супроводжується появою хибних стрибків ФХС величиною близькою до значень $\pm 2\pi$ (тобто суттєвим збільшенням похибки її визначення). Зменшення відношення сигнал/шум веде до збільшення кількості хибних стрибків в околі значень $\Phi(t)\equiv 0 \bmod 2\pi$. Це обумовлює необхідність розробки спеціальних статистичних методів і алгоритмів відновлення ФХС без втрати інформації про її цілу частину. По відношенню до сигналу як "носія" ФХС, таку статистичну обробку можна розглядати як вторинну.

Методи математичної статистики, використовувані для аналізу розподілених на прямій випадкових величин, добре відомі в теорії і практиці вимірювань. Проте застосування таких методів безпосередньо в задачах статистичного опрацювання результатів визначення ФХС має певні обмеження, а в ряді випадків може привести до отримання некоректних результатів. Ці обмеження обумовлені, в першу чергу, замкненим характером простору, в якому розглядається дробова частина ФХС – кола одиничного радіуса, який використовується в статистичному аналізі кутових даних.

Методи статистичного аналізу розподілених на колі випадкових кутів також добре вивчені [10] і суттєво відрізняються від методів статистичного аналізу випадкових величин. Так, наприклад, в статистичному аналізі випадкових кутів як вибіркові характеристики застосовують круговий середній напрямок і кругову дисперсію напрямків, кругову медіану, а як розподіли ймовірності випадкових кутів – гауссівський намотаний, Мізеса та ін.

На сьогодні ще не існує завершеної загальної теорії розгортання фази сигналів і сигнальних полів. Ця проблема знаходиться в стадії активної розробки. В кожному конкретному застосуванні розробляються оригінальні методи розв'язку, які тестуються шляхом комп'ютерного моделювання.

Можна припустити, що попереднє статистичне згладжування фунції $\varphi(t)$ дозволить уникнути або суттєво зменшити кількість хибних стрибків ФХС під час її розгортання. Слід зазначити, що таке згладжування повинно одночасно задовольняти суперечливі вимоги якнайкращого згладжування завади і передавання стрибків дробової частини ФХС з їх мінімальними спотвореннями. Подібна задача розв'язана для розподілених на прямій випадкових величин за допомогою медіанної фільтрації [11]. Можна очікувати, що за аналогією до такого розв'язку ідея застосування кругової медіанної фільтрації на колі [8] для статистичного згладжування дробової частини ФХС також виявиться продуктивною.

Розв'язок поставлених завдань дозволить підвищити точність визначення частотно-часових параметрів циклічних сигналів для функціонально-орієнтованих ІВС їх аналізу за рахунок використання непараметричного статистичного метода опрацювання їх фазових характеристик. Це дозволить розширити функціональні можливості ІВС за рахунок обробки циклічних сигналів з більшою апріорною невизначеністю параметрів сигналу та з меншим відношенням сигнал/шум.

Висновки

Фазова характеристика циклічних сигналів є важливою інформативною характеристикою, яка може бути покладена в основу вирішення широкого кола прикладних задач. Отримання ФХС надає можливості визначення часових і частотних параметрів циклічних сигналів – періоду, частоти, миттєвої частоти, а також можливість статистичного аналізу ФХС, дослідження динаміки її розвитку, аналізу її флуктуацій відносно заданої ФХС.

В реальних умовах ФХС спотворюються шумами і завадами, що діють у пристроях формування і обробки вимірювальних сигналів. Тому визначення ФХС повинно базуватись на статистичних методах обробки, узгоджених з циклічним характером самої ФХС. З значна частина відомих методів і способів визначення ФХС орієнтована на застосування в якості моделі детермінованих сигналів і не повною мірою враховує ймовірнісний характер фазових характеритстик реальних сигналів. Спрощений детермінований підхід для визначення ФХС має суттєві обмеження і приводить до втрат інформації під час її розгортання в умовах низьких відношень сигнал/шум.

Перспективним напрямком розвитку методів розгортання фазових характеристик циклічних сигналів є дослідження та застосування методів статистичного згладжування дробової частини ФХС, зокрема методом кругової медіанної фільтрації, що узгоджений з циклічним характером зміни в часі дробової частини ФХС. Це дозволить розширити функціональні можливості засобів аналізу циклічних сигналів за рахунок можливості обробки і отримання коректних результатів за умови більшої апріорної невизначеності параметрів сигналу та шуму.

Література

1. Куц Ю.В. Статистична фазометрія [наукова монографія] / Ю.В. Куц, Л.М. Щербак. – Тернопіль: Тернопільський державний технічний університет, 2009р. – 383 с.
2. Сергиенко А.Б. Цифровая обработка сигналов / Сергиенко А.Б. – СПб.: Питер, 2003. – 604 с.
3. Михеечев В.В. Геодезические светодальномеры / Михеечев В.В. – М.: Недра, 1979. – 222 с.
4. Кинкулькин И.Е. Фазовый метод определения координат / Кинкулькин И.Е., Рубцов В.Д., Фабрик М.А. – М.: Сов.радио, 1979. – 280 с
5. Денисов В.П. Фазовые радиопеленгаторы / Денисов В.П., Дубынин Д.В. – Томск: Томский госуд. Ун-т систем управления и радиоэлектроники, 2002.–251 с.
6. Гужов В.И. Проблема фазовой неоднозначности и ее решение в лазерной интерферометрии / Гужов В.И., Картавых Е.В.–Автометрия. –2000. –№5. –С.102–107.
7. Жуковский А.П. Теоретические основы радиовысотометрии / Жуковский А.П., Оноприенко Е.И., Чижов В.И. – М.: Сов. радио, 1979.-320 с.
8. Куц В.Ю. Аналіз застосування кругової медіанної фільтрації в задачах обробки сигналів. Збірник наукових праць. Інститут проблем моделювання в енергетиці. Випуск 39.ст. 50-56.
9. Математическая энциклопедия / Гл. ред. И. М. Виноградов. – М.: Советская энциклопедия. Т. 3, 1984. – 1184 с.
10. Мардиа К. Статистический анализ угловых наблюдений: Пер. с англ / Мардиа К. – М.: Главная ред. физ.-мат. лит. изд-ва "Наука", 1978. – 240 с.
11. Быстрые алгоритмы в цифровой обработке изображений / Т.С. Хуанг, Дж.-О. Эклунд, Дж. Нуссбаумер и др.; Под ред. Т.С. Хуанга: Пер. с англ. – М.: Радио и свіязь, 1984. – 224 с.

УДК 004.415.2.043

САЛАПАТОВ В.І.

СТВОРЕННЯ ПРОГРАМ ЗА ДОПОМОГОЮ ІЄРАРХІЧОЇ АВТОМАТНОЇ МОДЕЛІ

У статті пропонується метод новий метод створення програм за допомогою ірархічної моделі кінцевого автомату. При цьому спочатку будується логічна модель програми у вигляді кінцевого автомату, а далі виконується її створення та верифікація. Стани автомату відображають виконання елементарних дій, які можуть бути описані мовою програмування високого рівня.

Ключові слова: верифікація, ієрархічна модель, кінцеві автомати, стек.

In this article new methods of creating programs by means of hierarchical model of the finite automate is proposed. On such model firstly the logical model of program, like for finite automate is created and performed its verification. States of automate reflect elementary actions which may be describes on the high level program language and then be linked into single program.

Вступ

Традиційні технології створення програм передбачають обрання алгоритму вирішення задачі, верифікації, тестування та наступного її впровадження і супроводу. Перші два етапи займають значну частину часу у процесі розробки. Тобто, спочатку програма створюється, а потім доводиться її коректність. Зазначимо, що програми являють собою певні логічні моделі, де дії, які мають виконуватись у конкретних станах моделей програм, являють собою низки елементарних інструкцій у вигляді операторів на одній з мов програмування високого рівня. В подальшому такі окремі інструкції згідно логічної моделі програми компонуються в єдиний закінчений програмний продукт. Така технологія створення програм забезпечує сумісництво етапів створення програм та їх верифікації та можливу корекцію, що значно скорочує та спрощує власно процес розробки.

Сучасні технології створення програм

Традиційні технології програмування надають сучасні мовні засоби побудови програм та їх відлагодження. Наступний етап верифікації передбачає створення моделі вже існуючої програми та її наступну верифікацію. Тобто, такий не зовсім логічний порядок виконання етапів побудови програми додає труднощів для програмістів. Дещо кращою в цьому плані є технологія UML, яка передбачає створення моделі каркасу програми та наступне заповнення усіх елементів каркасу. Але у будь-якому випадку потім виконується верифікація програми. Технологія створення програм MODEL CHTCRING прийнятна для створення розподілених та паралельних програмних комплексів з окремих модулів [2] і не вирішує проблеми створення самих модулів. Тому, більш доцільним є спочатку побудова логічної моделі програми та подальше власно її створення. При цьому стає більш простішим визначення елементарних дій, які необхідно виконати у кожному стані логічної моделі програми. За таким принципом працює комп'ютер, де кожний його стан визначає виконання певної послідовності дій, які він має здійснити. В результаті виконується програма, яка запущена в конкретний момент часу. При створенні програм також доцільно дотримуватись логічної структури алгоритму. Такий принцип забезпечує контроль правильності його виконання. Тобто, якщо коректно побудувати логічну структуру програми з самого початку, то значно спрощується етап верифікації. Логічна структура програми повинна створюватись згідно вимог технічного завдання на розробку. В процесі уточнення технічного завдання програма розбивається на рівні і на кожному рівні визначається логічна структура програми.

Побудова ієрархічної моделі програми у вигляді кінцевого автомату

Великі складні програмні проекти розробляються за принципом згори донизу. Процес розбиття великої програми на окремі частини також передбачає створення логічної структури на кожному рівні. Таким чином, створюється логічний каркас програми або модель. В подальшому на кожному рівні такий процес має по-

вторюватись. Отже модель програми в загальному вигляді можна представити наступним чином M → Mi → Mij ... → Mij...k. Тут M є верхнім рівнем моделі, а Mi, Mij, Mij...k наступні рівні у порядку спадання. На практиці таких рівнів не більше 3-4. На вищих рівнях верхівки автоматної моделі програми визначається логіка програми та часткові дії, тоді як на самому нижньому рівні верхівки автоматної моделі визначають усі необхідні дії. Такі дії на всіх рівнях можуть бути описані на будь-якій мові програмування. При цьому маємо ієрархію рівнів описів верхівок автоматної моделі програми: L → Li → Lij ... → Lij...k, де L – множини описів верхівок на всіх рівнях. Створення програми полягає у зібранні усіх описів верхівок автоматної моделі, починаючи з початкового стану по всіх рівнях і закінчуючи кінцевим станом. Оскільки модель програми створюється у процесі її розробки, то розробка програми та її верифікація здійснюються одночасно. Коректність створеної програми визначається коректністю її автоматної моделі, а також опису усіх її верхівок.

Створення програми обробки виразу

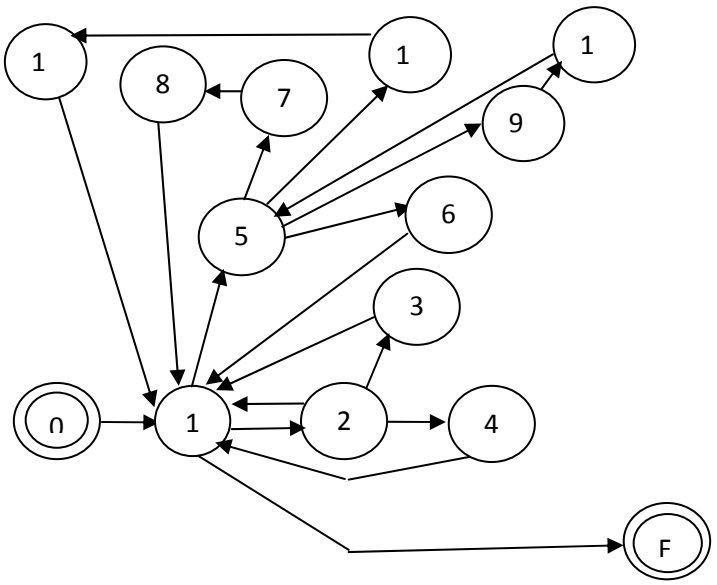
Запропоновану методику створення програм розглянемо на прикладі програми розбору виразу з використанням стекового алгоритму [2], який передбачає аналіз пріоритетності операцій. У початковому стані автоматної моделі мають бути визначені дані для програми у цілому. В даному випадку це буфер, в якому знаходиться текст виразу, буфер результату для проміжного уявлення, а також стек для розміщення в ньому пар лексем: лексем даних та лексеми дії [3]. Лексема дії складається з двох байтів – байту коду та байту приоритету. Отже, початкова верхівка визначає наступні данні.

```
VAR buff:STRING(1024);
buff1: STRING(256);
i, j, k, l : INTEGER;
CDPR  RECORD
RAZD, Pr: BYTE, VAL : INTEGER;
END;
VAR  ZAP1: CDPR;
dstack: ARRAY OF CDPR (32);
j : = 0;
```

Обробка виразу здійснюється за такими правилами.

```
 0 → 1 ; LEXAN
 1 → 2 { TERM = '' }
 2 → 3 {ZAP1.RAZD = '('};'', '('
TO_STACK
 3 → 1
 2 → 1 { ZAP1.RAZD = '+' AND
dstack(i).RAZD = beg}
 2 → 4 { ZAP1.RAZD = '-' AND
dstack(i).RAZD = beg} ;0, '-'
TO_STACK
 4 → 1
 1 → 5 { TERM < > '' }
 5 → 6 { dstack(i).Pr > ZAP1.Pr }
;TO_STACK
 6 → 1
 5 → 7 { dstack(i).Pr ≤ ZAP1.PR } ;
DOING
 7 → 8 ; FROM_STACK
 8 → 1
 5 → 9 ; { ZAP1.RAZD = ')' AND
dstack(i).RAZD # '('}; FROM_STACK
 9 → 10 { dstack(i).RAZD #
'('};DOING, FROM_STACK
 10 → 5
 5 → 11 { dstack(i).RAZD = '('};
FROM_STACK, BREAK
 11 → 12 ; SEARCRAZD
 12 → 1
 1→ F{ dstack(i).RAZD = beg AND
ZAP1.RAZD = fin}
```

Наведені правила описують переходи в автоматній моделі програми з одного стану в інший. У фігурних дужках визначені умови переходів. Якщо умови відсутні, то перехід у верхівку – безумовний. Коментарі вказують на умовні ідентифікатори, які уособлюють низку дій, що повинні виконуватись у зазначених станах. Так *LEXAN* означає дії, які можуть бути описані на нижньому рівні моделювання програми і повязані з пошуком пари лексем – лексеми даних та лексеми дії [2]. Позначки *dstack(i).Pr* та *ZAP.Pr* вказують на пріоритет попередньої та поточної дії, а їх значення beg та fin – початок стеку та кінець виразу. Позначки *dstack(i).RAZD* та *ZAP1.RAZD* вказують на коди попередньої та поточної дії. Ідентифікатори *TO_STACK* та *FROM_STACK* визначають дії у станах, які пов'язані із записом у стек пари лексем – даних та дії, а також читання їх зі стеку відповідно. Позначка *DOING* вказує на верхівку, яка пов'язана з формуванням внутрішнього уявлення частини виразу. Внутрішнє уявлення використовується як при компіляції, так і при інтерпретації. *BREAK* – вказує на верхівку 11, де знищуються ліва та права дужки. Нижче



Мал. Граф моделі програми обробки виразу

наводиться граф автоматної моделі програми обробки виразу згідно описаних правил. Позначка *SEARCRAZD* визначає пошук лексеми дії після правої дужки.

З верхівки 0 виконується перехід у верхівку 1, яка розпочинає цикл з процедури *LEXAN* та аналізу пріоритету і коду лексеми дії. Розкриємо інші введені позначки більш докладно. Верхівка 2 пов'язана з відсутністю лексеми даних та розпізнаванням при цьому припустимих лексем дії. Процедура запису в стек пари лексем даних та дії *TO_STACK*, яка виконується у верхівках 3, 4 та 6 може бути описана наступним чином:

```
dstack(i) : = ZAP1;  i : = i + 1;
```

Протилежна дія у верхівках 8, 9 та 11 *FROM_STACK* описується як:

```
ZAP1 : = dstack(i); i = : i – 1;
```

Ідентифікатор *DOING* визначає обробку формування внутрішнього подання для систем компіляції, а в разі інтерпретації – безпосереднє виконання арифметичних операцій, які обраховують даний вираз згідно коду лексеми дії. Треба зазначити, що в наведеному графі відсутній аналіз помилкових ситуацій, коли не виконується жодна з умов переходу з одного стану в інший. Вважається, що аналіз помилкових ситуацій має відбуватись у тих станах, де присутня умова переходу в інший стан. На графі також не показаний контроль сумісності типів даних. Такі дії повинні виконуватись у процедурах, які об'єднані загальним ідентифікатром *DOING*.

Для аналізу виразу, який записаний в буфері buff, розкриємо виконання дій на мові *PASCAL* в усіх станах автоматної моделі програми аналізу виразу.

Отже, для верхівки 1 маємо цикл аналізу, в якому використовується процедура *LEXAN* для виявлення пари лексем: лексеми даних та лексеми дії.

```
   k : = 0;
  L1:FOR j = k TO 256 DO
  BEGIN
       LEXAN(j);
       k : = j;
IF k = 1 THEN /*наявність лексеми
даних*/
   BEGIN
      IF  ZAP.RAZD = '('  THEN
        /* обробка лівої дужки*/
     ELSE    IF  ZAP.RAZD = '+'
  /*обробка унарного плюса*/
      ELSE   IF  ZAP.RAZD = '–'
  /*обробка унарного мінусу*/
       ELSE  /*обробка помилки*/
  END;
 ELSE
    IF    ZAP1.RAZD    =    ')'    AND
dstack(i).RAZD # '('
       END; /*обробка лексеми дії*/
    ELSE
    IF    ZAP1.RAZD    =    ')'    AND
 dstack(i).RAZD = '('  THEN
 /*BREAK; SEARCRAZD знищення дужок,
 пошук лексеми дії за правою
 дужкою*/
  ELSE    IF dstack(i).PR ≤ ZAP1.PR
THEN
  /*обробка лексеми дії*/
  ELSE    IF dstack(i).PR > ZAP1.PR
THEN
  /*запис у стек пари лексем*/
  IF   dstack(i).RAZD   =   beg   AND
ZAP1.RAZD = fin  THEN
  /*кінець обробки – верхівка F*/
```

Параметр j процедури *LEXAN* визначає позицію у символьному рядку буфера при аналізі виразу. Змінна l вказує на позицію у символьному рядку перед входом у *LEXAN*. Як було зазначено раніше [2] процедура *LEXAN* виконує посимвольний аналіз рядка і виявляє пари лексем: лексеми даних та лексеми дії. В таких парах за певних умов відсутньою може бути тільки лексема дії. На графі моделі ці умови пов'язані з діями у верхівці 2.

На даному прикладі показаний логічний каркас програми обробки виразу. Змістовне наповнення умовно позначено ідентифікаторами, які визначають певні дії. За таким принципом створена програма обробки виразу з додаванням сумісності типів даних, а також обробки вбудованих функцій на мові Pascal і в подальшому була оптимізована на мові Assembler для персональних комп'ютерів.

Висновки

Запропонована логічно зрозуміла технологія створення програмного забезпечення, яка дозволяє перед безпосереднім кодуванням програми створити її модель у вигляді кінцевого автомату. Ця модель уточнюється, корегується і перевіряється її коректність, тобто фактично частково виконується верифікація програми. Остаточна верифікація програми виконується при перевірці змісту дій у кожному стані моделі. В подальшому будується логічний скелет програми, який заповнюється змістовними операторами.

Складні програми можуть розбиватися по рівнях ієрархії, як це показано на прикладі, де на нижньому рівні подана процедура LEXAN. Оскільки процедура LEXAN вже була розглянута раніше, то в цій статті згадується тільки про її призначення. Така технологія поєднує у собі принципи структурного програмування та подання програми у вигляді моделі за допомогою розширеного недетермінованого кінцевого автомату. Така модель зрозуміло уявляє логіку роботи програми, дозволяє легко вносити необхідні корективи в разі потреби.

Список літератури

1. Ахо А., Сети А.Р., Ульман Дж. Компиляторы: принципы, технологии, инструменты. / А. Ахо, А.Р.Сети, Дж.Ульман – М.: Вильямс, 2001. – 768 с.
2. Карпов Ю.Г. MODEL CHTCRING. Верификация параллельных и распределенных программных систем. /Ю.Г. Карпов – СПб.; БХВ-Петербург, 2010.- 560 с.
3. Салапатов В.І. Синтаксичний аналіз із розподілом лексем на групи / В. Салапатов// Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. Київ. – 2008. – № 49. – С. 29-33.

УДК 004.415.2.043

АВРАМЕНКО В.С.

ПАРАМЕТРИЧЕСКИЕ МЕТОДЫ ПОИСКА ИНФОРМАЦИИ

Целью данной работы является создание программных средств для повышения эффективности автоматизированного документального поиска в гипертекстовых ресурсах Интернет. Для этого предлагается метод автоматического индексирования параметрических данных при обработке документов, содержащих числовую информацию с наличием как различных единиц размерностей, так и различных условных обозначений семантически одних и тех же количественных мер.

Ключевые слова: автоматическое индексирование, обработка документов, параметрические данные, числовая информация

An aim hired is creation of programmatic facilities for the increase of efficiency of the automated documentary search in hypertext resources the Internet. For this purpose the method of the automatic code indexing of self-reactance data is offered at treatment of documents, containing numerical information with the presence of both different units of dimensions and different conditional denotations semantically of the same quantitative measures.

Введение

Одним из основных источников информации сегодня являются ресурсы глобальной информационной сети Интернет, и обеспечение доступа к информации обычно связывается с обеспечением доступа именно к ресурсам Интернет. Развитие сети Интернет в наши дни связано в первую очередь с ростом объема информационных ресурсов и повышением качества инфраструктуры физических сетей. В течение следующих нескольких лет основными тенденциями дальнейшего развития будут дальнейший рост объемов ресурсов, накопление информации и увеличение числа пользователей имеющих доступ к глобальной информационной сети.

Развитие исследований в области информационных проблем требует построения таких автоматизированных систем обработки данных, которые извлекают из информационных потоков необходимые сведения. Эти системы, именуемые обычно интеллектуальными роботами, широко применяются в поисковых системах Интернет [4]. Сходная по своей сущности задача возникает при оперативной обработке больших и быстро движущихся потоков текстовой информации, содержащей параметрические данные.

Целью данной работы является повышение эффективности автоматизированного документального поиска в гипертекстовых ресурсах Интернет. Для этого предлагается метод автоматического индексирования параметрических данных при обработке документов, содержащих числовую информацию с наличием как различных единиц размерностей, так и различных условных обозначений семантически одних и тех же количественных мер.

Вспомогательная информация для автоматического индексирования

Метод автоматического индексирования параметрической информации основывается на использовании сведений о внешней (синтаксической) структуре текста и дополнительных сведений, извлекаемых из вспомогательных списков и таблиц.

В процессе исследования текстов и словарей и справочников [3, 5, 6] были получены данные, на основе которых составлены следующие вспомогательные списки и таблицы:

1. Список (словарь) неинформативных терминов (стоп-слов).
2. Таблица слов, входящих в неинформативные выражения.
3. Список окончаний русского языка.
4. Список суффиксов русского языка.
5. Таблица префиксоидов русского языка.
6. Таблица суффиксоидов порядковых и количественных числительных.
7. Таблица порядковых и количественных числительных.
8. Таблица кратных и дольных единиц размерностей физических величин.
9. Таблица единиц размерностей физических величин.

При составлении списков и таблиц были проанализированы словари, образованные из лексики массивов системы АСИОР и МОДИС,

обратный словарь русского языка и другие материалы [1-2].

В словари неинформативных терминов русского и английского языков вошли служебные слова (предлоги, союзы, артикли), сокращения, неинформативные слова и словосочетания. Термины словаря разбиты на группы в зависимости от количества символов, оставшихся после отбрасывания окончаний. Техника отбрасывания окончаний описана ниже (блок лексического анализа). В словарь включена также некоторая информация, необходимая для блока синтаксического анализа.

Списки окончаний и суффиксов разбиты на определенные классы, в состав которых входят положительные и отрицательные элементы (стоп-элементы) с соответствующими морфологическими и семантическими признаками. Во вновь разработанном алгоритме отождествления и отбрасывания окончаний и суффиксов, учитывающем сегментацию суффиксальной последовательности и местоположение отрицательных или положительных элементов этой последовательности, данные списки сведены до минимума. Так, список русских суффиксов содержит только 78 положительных и отрицательных суффиксов длиной от одного до четырех символов, в то время как, например, модифицированный список ИПС АСИОР [1] насчитывал около 400 элементов длиной от 1 до 12 символов.

Списки русских и английских суффиксов разбиты на две группы. К первой группе относятся суффиксы, которые могут быть отброшены, если они занимают только конечную позицию в слове (конечные суффиксы). Во вторую группу отнесены все остальные суффиксы и некоторые конечные суффиксы. Таким образом, в данные списки вошли не сложные суффиксы, а, в основном, неделимые суффиксы (суффиксальные морфы [5]). Это такие суффиксы, при дальнейшем делении которых выделяется хотя бы одна часть, которая не является суффиксом.

При обработке документов, содержащих числовую информацию, возникают затруднения, связанные с наличием как различных единиц размерностей, так и различных условных обозначений семантически одних и тех же количественных мер. Поэтому, для опознавания численной информации, выраженной количественными и порядковыми числительными, а также для перевода единиц измерений в Международную систему единиц СИ (Система Интернациональная) составлены различные таблицы (списки):

– префиксоидов русского языка;

– числительных русского (английского) языка (или их основы);

– единиц измерений всех существующих основных систем (СИ, СГС, МТС, МКГСС) с соответствующими числовыми значениями и коэффициентами пропорциональности.

Список префиксоидов русского языка содержит префиксоиды, являющихся составной частью сложных слов, представляющих числовую информацию.

В таблицу единиц измерений включены также производные единицы размерностей этих систем, имеющие собственные наименования. Производные единицы размерностей, не имеющие собственного наименования и называемые по другим единицам, в данную таблицу не включены, так как такие размерности могут вычисляться алгоритмически на стадии перевода их в систему СИ.

В качестве основных единиц, которые должны входить в любую формулу размерности при ее вычислении, предлагается [3]:

1) семь основных единиц СИ: длина (метр), масса (килограмм), время (секунда), сила электрического тока (ампер), термодинамическая температура (кельвин), сила света (кандела), количество вещества (моль);
2) логарифмическая единица отношения двух величин (десятичный логарифм отношения двух одноименных физических величин, например мощностей, токов, звукового давления), характеризующие интенсивность звука (бел, чаще применяют 0,1 долю бела – децибел);
3) логарифмическая единица, характеризующая единицу количества информации (бит);
4) деньги;
5) относительная величина, представляющая собой безразмерное отношение физической величины к одноименной физической величине (число – штук, сотая доля – процент).

Каждой основной единице присваивается 12-значное 16-ричное число, ненулевой разряд которого в соответствующей позиции закреплен за одной из основных единиц. При составлении формулы размерности каждый разряд (цифра) 16-ричного числа складывается (вычитается) только с соответствующим разрядом по

модулю 5. Отрицательные цифры представляются дополнительным кодом: F (-1), E (-2) и т.д. Цифра соответствующего разряда 16-ричного числа указывает сколько раз входит основная единица в формулу размерности (положительные от 1 до 7 – в числитель, отрицательные от 7 (1001) до 1 (1111) – в знаменатель).

Например, формула размерности для давления, которое в системе СИ измеряется в паскалях и выражается через другие основные единицы (1 Па = 1 Н/м^2) равна

-1 1 -2 0 0 0 0 0 0 0 0 0

(длина-м$^{-1}$, масса-кг, время-сек$^{-2}$), так как Ньютон – производная единица. Исходя из второго закона Ньютона она определяется как сила, изменяющая за 1 с скорость тела массой 1 кг на 1 м/с в направлении действия силы. Таким образом, 1 Н = 1 кг·м/с^2. Поэтому другие единицы размерностей будут иметь следующие коэффициенты пересчета:

1 Паскаль (Pa, Па) = 1 м$^{-1}$·кг·сек$^{-2}$ = 1 Н/м^2 = 10^{-5} Бар (bar, бар) = $10{,}197\cdot10^{-6}$ Техническая атмосфера (at, ат) = $9{,}8692\cdot10^{-6}$ Физическая атмосфера (atm, атм) = $7{,}5006\cdot10^{-3}$ Миллиметр ртутного столба (мм рт. ст., mm Hg, Torr, торр) = $1{,}0197\cdot10^{-4}$ Метр водяного столба (м вод. ст., m H_2O) = $145{,}04\cdot10^{-6}$ Фунт-сила на кв. дюйм (psi)

Лексический анализ

На стадии лексического анализа может использоваться простейший лексический анализатор. Он выполняет вспомогательные функции, и поэтому может быть реализован как самостоятельный блок, или как часть блока морфологического анализа.

Блок лексического анализа принимает исходный текст, который он должен разбить на предложения, или уже готовые предложения. Анализируемое предложение попадает на вход лексического анализатора в виде массива символов, содержащего прописные и строчные буквы русского (английского) алфавита, цифры, знаки пунктуации.

Полученный массив анализатор преобразует в массив лексических единиц. Здесь под этим термином подразумевается слово, число, скобки, знаки препинания. Для каждой лексической единицы формируется отдельная строка, в которую копируются все символы, принадлежащие данной лексической единице, и приписывается тип (класс) лексемы:

– слово естественного языка;

– слово, имеющее в составе русские и английские символы;

– слово, имеющее в составе хотя бы один цифровой символ;

– слово, имеющее в составе различные символы, кроме цифровых символов;

– разделители.

Морфологический и семантический анализ

После лексического анализа термины текста последовательно подвергаются семантическому и морфологическому анализу. На стадии предварительного семантического анализа термины всех типов разделяются на слова естественного языка (русские и английские), числовую информацию и термины, состоящие из смешанных символов.

Числовая информация (числа, числительные) и единицы размерностей приводятся к каноническому виду с учетом соответствующих коэффициентов пропорциональности. Все “слова” естественного языка подвергаются морфологическому анализу, задачей которого является отделение у каждого слова основы и приписывание этой основе морфологической и семантической информации, необходимой для установления текстуальных отношений между терминами на этапе синтаксического анализа. Морфологический анализ выполняется по следующей схеме:

1. Сложные слова разбиваются на самостоятельные единицы: префиксоид(ы) и основу префиксоида. Например, сложное слово “двадцатипятитысячный” разбивается на два префиксоида (“двадцати”, “пяти”) и основу префиксоида (“тысячный”).

2. Различные словоформы одного слова приводятся к одному и тому же парадигматическому коду (основе).

3. Определение порядковых и количественных числительных, неинформативных и полуинформативных слов.

После морфологического анализа входной текст будет представлен последовательностью условных единиц (индексов). Каждая из этих единиц содержит сведения о том, из какой лексической единицы она получена (специальный

знак, неинформативный термин, полуинформативный термин, число, единица размерности, русское слово, английское слово, смешанный термин), а также сведения об этой лексической единицы, извлеченные из вспомогательных таблиц и списков.

Синтаксический анализ

Исходной информацией для этапа синтаксического анализа является цепочка индексов с соответствующей морфологической и семантической информацией. Целью синтаксического анализа является разбиение этой цепочки на определенные конструкции (индексы и сегменты) и установление соответствующих связей между элементами этих конструкций.

Индекс – это набор символов, полученный из значащего термина на стадии лексического анализа. Сегмент – последовательность индексов, ограниченная знаками препинания (запятой, тире, двоеточием, восклицательным знаком, точкой с запятой, точкой, обозначающей конец фразы) или некоторыми служебными словами и другими неинформативными терминами.

Одновременно с сегментацией решаются вопросы лексико-грамматической омонимии, приводится к каноническому виду числовая информация, заданная диапазоном и записанная в сложной (составной) форме.

Лексико-грамматическая омонимия на синтаксическом и морфологическом уровнях распознается с помощью информации, полученной на стадии лексического анализа. Проверке подвергаются только “полуинформативные” термины. Например, при анализе слов “пуст(ой) – пуст(ь)”, “почт(а) – почт(и)”, “вес – вес(ь)” признак “полуинформативности” уничтожится у слов “пустой”, “почта”, “вес”, а признак “неинформативности” будет приписан остальным словам. Для разрешения более сложных омонимов рассматривается левое или правое его окружение, или же левое и правое одновременно

Если при анализе цепочки индексов обнаружатся числовые величины, записанные с указанием интервала (например, “с … по”, “от … до”, “в диапазоне”, “-”), перечислением значений одной и той же величины или в сложной (составной) форме, то они приводятся к каноническому виду с приписыванием признаков нижнего и/или верхнего предела и соответствующей единицы размерности.

На заключительной стадии синтаксического анализа из поискового образа выбрасываются сегменты, которые состоят из неинформативных элементов (индексов). Внутри каждого сегмента устанавливаются связи между его элементами. В частности, для параметрической информации устанавливается атрибутивная связь (число + размерность), дефисная связь (число + число).

Выводы

В отличие от технологий ранних и современных информационно-поисковых систем, базирующихся на технологии текстового поиска и обеспечивающих поиск документов на основе их информационного содержания (набор дескрипторов или значения каких-либо других атрибутов), предлагается дополнить эти технологии автоматическим индексированием параметрических данных при обработке документов, содержащих числовую информацию с наличием как различных единиц размерностей, так и различных условных обозначений семантически одних и тех же количественных мер.

Список литературы

1. Авраменко В.С. Автоматизированная система информационного обеспечения разработок / В.С. Авраменко, В.И. Легоньков В.И., В.Р. Хисамутдинов. – М.: Наука, 1980. – 208 с.
2. Авраменко В.С. Математическое обеспечение диалоговых информационных систем / В.С. Авраменко, В.И. Легоньков В.И., В.Р. Хисамутдинов. – М.: Наука, 1990. – 192 с.
3. Бурдун Г.Д. Справочник по международной системе единиц / Г. Д. Бурдун. – М.: Изд-во стандартов, 1971. – 231 с.
4. Крупник А.Б. Поиск в Интернете: Самоучитель / А.Б. Крупник. – СПб.: Питер, 2006. – 268 с.
5. Зятковская Р.Г. Суффиксальная система современного английского языка / Р.Г. Зятковская. – М.: Высшая школа, 1971. – 186 с.
6. Обратный словарь русского языка. – М.: Советская энциклопедия, 1974. – 944 с.

УДК 004.822:004.75

ПОСПЕШНЫЙ А.С.

OWL 2 EL ОНТОЛОГИЯ ДЛЯ СЕМАНТИЧЕСКОГО ИНФОРМАЦИОННОГО СЕРВИСА ГРИД

Массовому внедрению онтологий и использованию интеллектуальных систем в целом препятствует тот факт, что алгоритмы логического анализа в таких системах имеют чрезвычайно высокую вычислительную сложность. Разрабатывая нашу интеллектуальную систему управления ресурсами Грид, в которой центральную роль играют онтологии, мы столкнулись с недопустимо низкой производительностью логического анализа над базами знаний с большим количеством типизированных утверждений. С целью обеспечить приемлемую скорость классификации и масштабируемость мы преобразовали нашу онтологию Грид-ресурсов к профилю OWL 2 EL и воспользовались модифицированным высокопроизводительным логическим анализатором ELK, что дало многократный прирост производительности логического анализа.

Mass deployment of ontologies and application of intelligent systems in general is hampered by the fact, that reasoning algorithms for such systems have a very high computational complexity. While developing our intellectual Grid resource management system, were ontologies play a fundamental role, we faced an unacceptably low performance of reasoning with knowledge bases that consisted of large amount of datatype expressions. In order to ensure an acceptable speed of classification and scalability, we converted our Grid resource ontology to the OWL 2 EL profile and used a modified high-performance OWL reasoner ELK, which allowed for a multiple performance gains.

Онтология Грид ресурсов

Можно с уверенностью заявить, что Грид-системы стали новым мощным и, в некоторой степени, даже необходимым инструментом для современной науки и инженерии. Однако если посмотреть на структуру любой современной Грид-системы то можно легко заметить её исключительную гетерогенность, так как все структурные составляющие Грид уникальны в отношении их программного и аппаратного обеспечения и к тому же ещё находятся в распоряжении своего непосредственного владельца.

Становится очевидным, что управление и эффективное использование Грид системы невозможно без наличия полной, достоверной и актуальной информации о всех подключенных к ней ресурсах, их характеристиках, состоянии и политике использования. Что не немаловажно, механизм доступа к этой информации обязан быть максимально понятным широкому кругу пользователей и в то же время достаточно гибким и адаптивным для широкого круга задач, так как на практике пользователями Грид могут быть специалисты из совершенно разных областей науки без глубоко понимания принципов и технологий, стоящих в основе Грид.

С целью усовершенствовать интерфейс взаимодействия пользователя с Грид-системой мы решили применить семантические технологии, развивающиеся в рамках концепции Семантической Паутины. С их помощью мы создали “интеллектуальный” информационный сервис Грид – семантический информационный сервис Grid-DL [1].

В основе такого сервиса лежит база знаний о ресурсах Грид, построенная с помощью онтологий. Пользователь может применять высокоуровневое и понятное ему описание решаемой задачи в понятных ему терминах, а необходимые для этого программные и аппаратные ресурсы будут определены и найдены при помощи системы логических выводов. Другим преимуществом такого подхода является надёжный механизм проверки корректности запроса Грид-ресурсов, так как можно проверить запрос на предмет логических несоответствий, что поможет минимизировать ошибочные назначения и простой ресурсов в связи с трудно обнаруживаемыми пользовательскими ошибками.

Упомянутая ранее онтология Грид-ресурсов является краеугольным камнем в нашей концепции семантического информационного сервиса. Будучи основанной на специальной схеме именования ресурсов Грид – GLUE (Grid Laboratory Uniform Environment) [2], эта онтология помогает фиксировать все доступные для существования в Грид компоненты и их характеристики.

Разработанная нами ранее онтология [3] содержит 65 классов, 33 объектных и 106 типизированных свойства и соответствует уровню $\mathcal{SHIF}(\mathcal{D})$ дескриптивной логики, что отвечает OWL-Lite диалекту языка OWL. Однако применив эту онтологию на практике, 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 основных современных логических анализаторов (т.к. Pellet, HermiT, FaCT++), 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 [4] такая задача имеет сложность $\mathcal{NE}xp\mathcal{T}ime$, что в комбинации с очень большим размером онтологии в нашем случае исключает этот подход.

За решением проблемы мы обратились к опыту разработки интеллектуальных системы на основе одних из самых больших на сегодняшний день онтологий – SNOMED CT, GALEN и Gene Ontology.

Один способ обеспечить приемлемую скорость классификации – построить онтологию на основе профиля EL языка OWL 2 [5]. Данный профиль основан на дескриптивной логике $\mathcal{EL}^{++}$ [6], что позволяет достичь необходимых вычислительных характеристик. Так для OWL 2 EL онтологий основные задачи логического анализа могут быть выполнены за полиномиальное время по отношению к размеру обрабатываемой онтологии, сохраняя при этом достаточную экспрессивность языка.

Таким образом, было решено внести необходимые изменения в разработанную ранее онтологию ресурсов Грид таким образом, что бы она соответствовала профилю EL языка OWL 2.

Адаптация онтологии к профилю EL

Согласно спецификации языка OWL 2, Таблица 1 отображает разрешённые в профиле OWL 2 EL выражения.

Таблица 1. Выражения, разрешённые профилем OWL 2 EL

Группа	Тип OWL выражений	Выражение
Определение классов	Квантор существования класса	ObjectSomeValuesFrom
	Квантор существования диапазона данных	DataSomeValuesFrom
	Квантор существования экземпляра	ObjectHasValue
	Квантор существования литерала (типизированных данных)	DataHasValue
	Самоограничение	ObjectHasSelf
	Перечисление единственного экземпляра или литерала	ObjectOneOf DataOneOf
	Пересечение именованных классов и ограничений	ObjectIntersectionOf DataIntersectionOf
Аксиомы	Утверждение подкласса	SubClassOf
	Утверждение эквивалентности классов	EquivalentClasses
	Утверждение непересекаемости классов	DisjointClasses
	Утверждения, формирующие иерархи объектных и типизированных свойств.	SubObjectPropertyOf SubDataPropertyOf
	Эквивалентность свойств	EquivalentObjectProperties EquivalentDataProperties
	Транзитивность объектных свойств	TransitiveObjectProperty
	Рефлексивность объектных свойств	ReflexiveObjectProperty
	Функциональность типизированных свойств	FunctionalDataProperty
	Область определения (домен) свойства	ObjectPropertyDomain DataPropertyDomain
	Диапазон свойства	ObjectPropertyRange DataPropertyRange
	Утверждения индивидов	SameIndividual DifferentIndividuals ClassAssertion ObjectPropertyAssertion DataPropertyAssertion NegativeObjectPropertyAssertion NegativeDataPropertyAssertion
	Ключи	HasKey

Профилем EL предусмотрено использование 19-и типов данных, большинство из которых определённо в рамках спецификации XSD – языка описания структуры XML – документов. Рисунок 1 резюмирует все поддерживаемые в рамках профиля типы данных, устанавливает их наследственность и разрешённые ограничительные аспекты (фасеты).

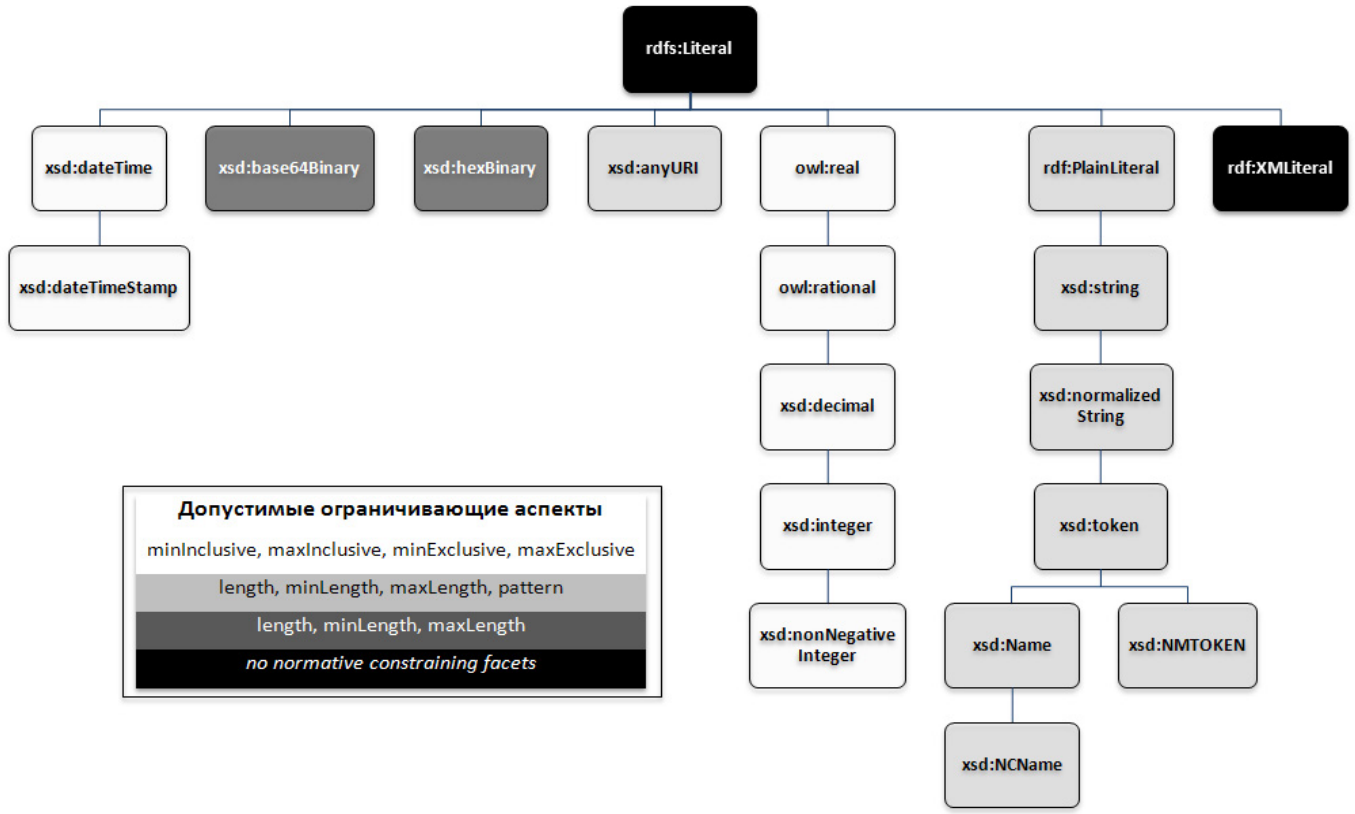


Рис 1: Допустимые типы данных в онтологиях OWL 2 EL профиля

Важно отметить, что профиль OWL 2 EL не предусматривает использование следующих типов данных: *xsd:double, xsd:float, xsd:nonPositiveInteger, xsd:positiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte, xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte, xsd:language и xsd:boolean.* Причина этого заключается в том, что список разрешённых профилем типов был составлен таким образом, что бы пересечение пространств значений каждого из них с любым другим из типов было либо пустым, либо бесконечным. Подобное ограничение помогает обеспечить необходимые вычислительные характеристики логического анализа EL онтологий.

Таким образом, мы смогли сопоставить выражения, используемые в нашей онтологии, с теми, которые разрешены профилем OWL 2 EL и произвели следующие модификации:

1. *Замена запрещённых типов данных.*

В нашей онтологии было использовано запрещённый в профиле EL тип данных xsd:boolean, а именно, в определении класса FileSystem (isReadOnly) и NetworkAdapter (hasInboundIP, hasOutboundIP). Данные выражения были заменены на эквивалентные им, с использование типа xsd:integer, где значение 1 соответствует булевой истине, а 0 – ложному утверждению.

2. *Замена инверсных свойств.*

В OWL онтологиях одно свойство может быть представлено как инверсия другого. Если свойство P_1 представлено как инверсия свойства P_2 и концепт X связан с Y свойством P_2, то отсюда следует, что концепт Y связан с X свойством P_1. Однако профиль OWL 2 EL запрещает использовать подобные выражения.

Данный вид свойств широко используется в нашей онтологии ресурсов Грид, так как помогает использовать более понятные и удобно читаемые выражения. Так, следующие свойства объявлены как инверсные друг другу:

describes	⇆	describedBy
hasServiceData	⇆	isServiceDataFor
hasSoftwareData	⇆	isSoftwareDataFor
hasVOInfo	⇆	isVOInfoFor
hasVOView	⇆	isVOViewFor
managedBy	⇆	manages
partOf	⇆	contains

Для решения этой проблемы мы изъяли из онтологии утверждения инверсности указанных выше свойств и внесли изменения в алгоритм наполнения базы знаний таким образом, что бы при формировании онтологии создавались новые утверждения между экземплярами, связанные упомянутыми объектными свойствами. То есть, если экземпляр *A* и *B* соединены свойством describes, то будет добавлено новое утверждение о том, что *B* describedBy *A*.

3. *Замена функциональных свойств.*

Если свойство объявлено как функциональное, то оно имеет не более одного значения для каждого экземпляра. Эта характеристика известна как наличие уникального свойства.

В нашей онтологии 11 из 32 объектных свойств объявлены как функциональные. К сожалению, профиль OWL 2 EL не предусматривает их использование, поэтому мы вынуждены были извлечь их из онтологии (был удалён признак функциональности, сами объектные свойства остались частью онтологии).

Что бы всё-таки обеспечить проверку уникальности, были внесены изменения в алгоритм наполнения онтологии таким образом, что бы проверка уникальности и консистентности происходила на этом этапе. Таким образом, обеспечивается то же поведение, что и наличие функциональных свойств в онтологии.

4. *Замена обратно-функциональных свойств.*

Если свойство обратно-функционально, то инверсия этого свойства функциональна. То есть, инверсия данного свойства имеет не больше одного значения для каждого индивида. Эта характеристика также известна как недвусмысленное (однозначное) свойство.

Подобные конструкции запрещены к использованию в OWL 2 EL. В нашей онтологии всего одно свойство объявлено как обратно-функциональное – hasLocation. Таким образом, можно было отразить тот факт, что если разные Грид-площадки (Site) расположены в одном и том же месте, то либо это одна и та же площадка, либо это ошибочная запись.

Как и в предыдущем случае, данную проверку, симулирующую поведение обратно-функциональных свойств, было перенесено на этап формирования базы знаний.

5. *Замена симметричных свойств.*

Последнее несоответствие нашей онтологии с профилем EL было наличие симметрических свойств. Если объектное свойство *P* объявлено симметричным, то при объявлении *P(x,y)* автоматически подразумевается *P(y,x)*. В нашей онтологии симметричным являться свойство inRelationshipWith, с помощью которого устанавливаются связи между сервисами Грид (если сервис *A* связан с сервисом *B*, то следовательно сервис *B* тоже связан с *A*).

Аналогично инверсным свойствам, при формировании онтологии создаются новые утверждения между экземплярами, связанные упомянутым симметричным свойством.

Таким образом, после перечисленных выше модификаций, наша онтология ресурсов Грид стала соответствовать профилю OWL 2 EL, не потеряв при этом своих эксплуатационных качеств. Теперь у нас появилась возможность воспользоваться высокопроизводительными логическими анализаторами для EL онтологий в нашем семантическом информационном сервисе Грид, многократно повысив его производительность.

Адаптация онтологии к логическому анализатору ELK

Модифицировав онтологию нашего семантического информационного сервиса Грид таким образом, что бы она соответствовала профилю EL языка OWL 2, мы приступили к выбору наиболее подходящего логического анализатора.

Были рассмотрены такие логические анализаторы как CB, CEL, jCEL, Snoroket и ELK. Последний, благодаря своей продуманной масштабируемой архитектуре стал в основу нашей работы.

ELK является свободно распространяемым высокопроизводительным логическим анализатором с открытым исходным кодом для онтологий OWL 2 EL профиля. Исключительная высокая производительность логического анализа достигается за счёт применения оптимизированного параллельного накопительного алгоритма для рассуждений, работа которого, в общих чертах, заключается в итеративном применении к множеству исходных аксиом трансформирующих правил, порождающих новые аксиомы- следствия.

EKL обладает гибкой модульной архитектурой и может быть использован в разнообразных конфигурациях, в частности возможно использование ELK как утилиты, вызываемой из командной строки операционной системы, как библиотеки, интегрируемой в приложение посредством интерфейса OWL API или как подключаемого расширения к редактору онтологий и баз знаний Protege.

Исчерпывающее описание алгоритмов используемых в ELK, особенности их реализации и доказательства их корректности и полноты можно найти в работах [7-8].

Однако, несмотря на то что ELK позиционирует себя как логический анализатор для OWL 2 EL онтологий, ним поддерживаются далеко не все перечисленные ранее выражения этого профиля.

Самим критическим недостатком ELK было отсутствие поддержки типизированных свойств и выражений с их участием. Так как наша онтология в основном хранит знания о характеристиках Грид-ресурсов именно с помощью типизированных свойств, подобное упущение было недопустимо.

Потому нами была разработана модификация логического анализатора ELK которая добавила полноценную поддержку логического анализа онтологий с типизированными свойствами. Более детально об этих модификациях можно узнать в работе [9].

Другим, менее критическим недостатком было отсутствие поддержки области определения и диапазона свойств (Domain и Range). Природа этого ограничения незначительно влияет на эксплуатационные характеристик нашей системы, однако мы всё же внесли изменения в алгоритм формирования базы знаний таким образом, что бы производить предварительную проверку экземпляров на принадлежность к указанным классам. В скором времени, новая версия ELK будет уже иметь полноценную поддержку упомянутых выражений и данный шаг уже будет не нужен.

Тестирование

Для проверки работоспособности наших модификаций мы воспользовались генератором баз знаний Грид ресурсов и создали несколько онтологий разного размера и сложности.

Наши тестовые базы знаний содержат информацию о различных элементах Грид-системы, которая обслуживает выполнение экспериментов на Большом адронном коллайдере. Онтология представляет собой терминологический блок и утверждения конкретных экземпляров ресурсов Грид и отношений между ними.

В качестве «полезной нагрузки» мы добавили к онтологии несколько определений, который будут выполнять роль пользовательского поискового запроса к базе знаний, а именно, мы будем искать подходящую Грид-площадку расположенную в Соединённом Королевстве, которая на данный момент простаивает и содержит кластер x86_64 архитектуры с узлами, в которых бьём оперативной памяти находится в пределах от 4 до 8 Гб:

```
UK_Site ≡ Site and hasLocation
some (Location and hasName some
  string[pattern ".*, UK"])

Idle_CE ≡ ComputingElement and
  hasState some (CEState and
  hasRunningJobs value 0 and
  hasWaitingJobs value 0 and
hasFreeJobSlots some integer[>0])
 and hasState some (CEState and
  hasStatus value Production)

x64HightMemCluster ≡ SubCluster
    and (describedBy some
    (hasPlatformType value
"x86_64"^^string) and (describedBy
 some (hasRAMSize some integer
    [>= 4096, <= 8192])))

MySite ≡ UK_Site and (contains
some IdleCE) and (contains some
    x64HightMemCluster)
```

По результатам классификации, экземпляры класса MySite будут представлять искомые ресурсы.

В таблице 2 приведены результаты тестирования. Тестовая конфигурация: Intel Core 2 Duo T9300 @ 2.50GHz, 4 Gb RAM, OpenSUSE 12.1 (Linux 3.1.10), Java 1.7.0_05 (-Xmx3200M -Xms3200M). Каждый тест проводился 3 раза. Использовалась обычная модифицированная сборка ELK, без дополнительных оптимизаций (в некоторых случаях возможно достичь двукратного прироста производительности, без потери корректности полученных результатов).

Таблица 2. Результаты тестирования

Конфигурация онтологии	Параметры онтологии	Время классификации, мс
Грид-площадки	13018 аксиом, 2300 экземпляров, 1916 объектных утверждений, 5979 типизированных утверждений.	2 790
Грид-площадки + Грид-сервисы	234729 аксиом, 36367 экземпляров, 68362 объектных утверждений, 93090 типизированных утверждений.	8 062
Грид-площадки + Грид-сервисы + Вычислительные элементы	519653 аксиом, 73900 экземпляров, 110401 объектных утверждений, 260861 типизированных утверждений.	11 776
Грид-площадки + Грид-сервисы + Вычислительные элементы + Кластеры + Подкластеры	822520 аксиом, 80824 экземпляров, 128633 объектных утверждений, 531630 типизированных утверждений.	43 729
Грид-площадки + Грид-сервисы + Вычислительные элементы + Кластеры + Подкластеры + Хранилища данных + Политики безопасности	1117487 аксиом, 133029 экземпляров, 192408 объектных утверждений, 658397 типизированных утверждений.	55 621

Заключение

Полученные результаты позволяют нам с оптимизмом смотреть на перспективы использования семантического информационного сервиса Грид на практике. Модификация базовой онтологий и применение перспективного высокопроизводительного логического анализатора ELK, адаптированного нами для поддержки типизированных выражений, позволило многократно повысить производительность нашей системы и свести время обработки пользовательского запроса к приемлемым величинам.

Описанная онтология доступна для свободной загрузки по адресу http://grid-ontology.googlecode.com/files/GLUE_EL.owl

Программа, производящая формирование базы знаний на основе информационной системы Грид доступна по адресу https://grid-ontology.googlecode.com/svn/trunk/Importer

Список литературы

1. Поспешный А.С., Стиренко С.Г. GRID-DL – семантический информационный сервис ГРИД // Компьютинг, 2011. – Том 10, Выпуск 3. – С. 285 – 294.
2. S.Andreozzi, S. Burke, F. Ehm, L. Field, et al. GLUE Schema Specification v. 2.0. – 2009.
3. Поспешный А.С., Стиренко С.Г. Онтология ресурсов для семантического информационного сервиса Грид, Электронное моделирование. 33 (4) (2011).
4. 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
5. OWL 2 Web Ontology Language Profiles. W3C Recommendation 27 October 2009. http://www.w3.org/TR/owl2-profiles
6. 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.
7. Yevgeny Kazakov, Markus Krötzsch, František Simančík. Concurrent Classification of EL Ontologies**.** Proceedings of the 10th International Semantic Web Conference (ISWC-11). LNCS 7032, Springer, 2011.
8. Yevgeny Kazakov, Markus Krötzsch, František Simančík. The Incredible ELK: From Polynomial Procedures to Efficient Reasoning with EL Ontologies. http://elk.semanticweb.org/papers/elk-journal-2013.pdf
9. Поспешний А.С., Стиренко С.Г. Эффективный алгоритм рассуждений для OWL 2 EL онтологий с типизированными выражениями на базе логического процессора ELK. // Адаптивные системы автоматического управления. – 2012. – № 20 (40). – С. 106-115.

УДК 004.451.24

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

МОДЕЛЬ ОРГАНИЗАЦИИ ВЫЧИСЛЕНИЙ В РАСПРЕДЕЛЁННОЙ СИСТЕМЕ

Рассматриваются существующие модели организации вычислений в параллельных системах, применяемые в различных современных библиотеках и инструментальных средствах. Предложена усовершенствованная модель разбиения на независимые подзадачи, учитывающая размещение данных в локальной памяти узлов вычислительной системы. Приведен математический формализм и семантика операций, программная реализация которых позволит соответствовать математической модели.

Different models used in modern libraries and frameworks to organize computations in parallel systems are studied and reviewed. An improved task-splitting model compatible with work-stealing approach, which allows consideration of data locality in distributed system, is proposed. The mathematical formalism of this model and semantics of operations on the model are described.

Введение

Бурное развитие средств аппаратных средств вычислительной техники в последнее время ориентировано на создание высокопроизводительных систем путём увеличения количества используемых вычислительных элементов. Наибольшее распространение для решения вычислительно-ёмких научных задач получили кластерные системы как подмножество более широкого класса распределённых систем или систем с локальной памятью. Такие системы позволяют достаточно лёгкое горизонтальное масштабирование с целью наращивания вычислительных ресурсов. Однако, программирование таких систем требует значительных усилий по адаптации математически заданных алгоритмов к существующим моделям программирования, что приводит к значительному количеству ошибок при программировании, неэффективному распараллеливанию и, как следствие, неэффективному использованию вычислительных ресурсов [1]. Кроме того, некоторые подходы к представлению задач для систем с локальной памятью предполагают оптимизацию программы для существенно ограниченного объёма аппаратных ресурсов, что может привести к снижению отношения коэффициента ускорения к количеству используемых процессорных ядер при горизонтальном масштабировании [2].

Современные вычислительные средства предлагают несколько различных моделей программирования параллельных систем, в зависимости от доступного аппаратного обеспечения, в частности традиционное параллельное программирование для систем с общей памятью с использованием потоков, программирование распределённых систем через интерфейс передачи сообщений MPI, программирование ускорителей на базе Nvidia CUDA [3] или Intel Many Integrated Core architecture [4], и другие, каждая из которых имеет свои особенности. С другой стороны, предлагается ряд усовершенствованных подходов для описания вычислительных задач с использованием более высокого уровня абстракции, в частности автоматизированное распараллеливание с использованием OpenMP [5,6] или подход деления задач в Intel Threading Building blocks [7]. Однако, эти подходы не учитывают особенности распределённых систем, в особенности необходимость организации пересылок данных и оптимизацию их выполнения.

Таким образом, необходимо предложить модель программного интерфейса, которая бы использовала высокоуровневые абстракции задач и подзадач, тем самым избавляя пользователя от необходимости использовать особенности той или иной вычислительной системы, и учитывала особенности выполнения задач на распределённых системах с возможностью масштабирования. Такая модель предусматривает также возможность адаптивного подхода к изменению количества используемых ресурсов в процессе решения задачи и имеет возможность гибкой адаптации к размерности системы.

Подход к организации вычислений

Текущая ситуация, сложившаяся в сфере высокопроизводительных вычислений, диктует некоторые условия технологической реализации разрабатываемой системы, а именно:

• Доминирующее положение библиотеки MPI, которая обеспечивает коммуникацию между процессами, работающими на различных узлах, широкая поддержка и развитие её реализаций вендорами HPC систем и интерконнектов предопределяют использование MPI в разрабатываемой системе.

• По аналогичным причинам система изначально разрабатывается для Unix систем; тем не менее, размер платформо-зависимого кода небольшой, что в первую очередь обусловлено использованием MPI.

• Система предполагает проектирование и реализацию некоторого количества API, доступных программисту; для этого выбран язык C++ как язык, широко использующийся в HPC и обладающий мощными возможностями по созданию обобщённого кода благодаря шаблонам [8].

• Система представляет собой динамически загружаемую библиотеку, что даёт возможность не прилагать её копию к каждой программе, в то же время не требуя чрезмерных вмешательств в операционную систему как в случае с модулями ядра.

Способ описания заданий, подлежащих параллельному выполнению, сходен с применяемым в библиотеке TBB. Пользователю необходимо:

• выбрать один из готовых классов промежутков, подходящий для задачи, или описать свой (тип Range);

• описать класс, хранящий результат вычислений (тип Value);

• реализовать вычислительный алгоритм (оператор compute);

• реализовать алгоритм разбиения промежутков (оператор split);

• реализовать алгоритм слияния частичных результатов (оператор merge);

• реализовать алгоритм пересылок Range и Value между узлами.

Для запуска задания в параллельном режиме следует передать экземпляры всех этих классов в качестве аргументов функции parallel_reduce().

Не смотря на кажущееся большое количество вспомогательных сущностей и алгоритмов, все их элементы пользователь так или иначе реализовывал бы, выполняя распараллеливание вручную. Особенностью разрабатываемой библиотеки является лишь требование формализации и разделения этих сущностей.

Описанная пользователем задача, механизм её разделения и объединения результатов с помощью механизма шаблонов C++ превращается в код, использующий библиотеки MPI и Pthreads. Такой подход выгоден простотой использования и естественной поддержкой механизмов ООП, что является критичным при больших разработках [9].

Для выполнения пользовательской задачи система использует любой существующий MPI интеркоммуникатор на N процессов, каждый из которых содержит 2 потока: рабочий (worker thread, выполняет вычисления) и вор (thief thread, выполняет пересылки).

Каждый процесс имеет очередь промежутков, готовых для вычисления, а также очередь вычисленных частичных результатов. Когда очередь заданий пуста, рабочий поток пытается объединить какие-либо частичные результаты, иначе он будит поток-вор и блокируется, ожидая момента, когда поток-вор наполнит очередь заданий и разбудит его (подробности алгоритма, по которому наполняется очередь заданий, могут варьироваться в различных алгоритмах, соответствующих данной модели).

Семантика операций

Нижеописанная модель предполагает решение одной пользовательской задачи, описанной с использованием соответствующего программного интерфейса. Этот интерфейс реализует на некотором языке программирование функции, семантически эквивалентные нижеследующим абстрактным математическим операциям.

Под задачей T_0 будем понимать множество последовательностей действий $T_0 = \{t_{0,i}\}$, которые необходимо выполнить над некоторым набором входных данных. Каждая последовательность действий может состоять из произвольного числа действий $t_i = \langle a_0, a_1, a_2, \ldots \rangle$. Действия одной последовательности выполняются последовательно, т. е. так, как они бы выполнялись на однопроцессорной системе. Различные последовательности действий могут выполняться независимо и параллельно.

Исходная задача T_0 может быть разбита на подзадачи $\{T_1^{\langle 0 \rangle}, T_1^{\langle 1 \rangle}, T_1^{\langle 2 \rangle}, \ldots\}$ так, что множества последовательностей действий подзадач не пересекаются, а их объединение даёт изначальное

множество последовательностей действий. Пусть $T_1^{\langle j\rangle} = \{t_{1,i}^{\langle j\rangle}\}$, тогда

$$\bigcap_j T_1^{\langle j\rangle} = T_1^{\langle 0\rangle} \cap T_1^{\langle 1\rangle} \cap T_1^{\langle 2\rangle} \cap \ldots = \varnothing; \quad (1)$$

$$\bigcup_j T_1^{\langle j\rangle} = T_1^{\langle 0\rangle} \cup T_1^{\langle 1\rangle} \cup T_1^{\langle 2\rangle} \cup \ldots = T_0. \quad (2)$$

Более строгой версией (1), которая необходимо соблюдать в процессе разбиения, является

$$\{t_{1,a}^{\langle j\rangle}\} \cap \{t_{1,b}^{\langle j\rangle}\} \quad \forall a,b,j : a \neq b. \quad (3)$$

Подзадачи $T_1^{\langle j\rangle}$ называются *подзадачами первого порядка* или подзадачами исходной задачи. Для общности исходную задачу можем считать *подзадачей нулевого порядка*. Введём дополнительное ограничение на подзадачи: любая подзадача должна содержать хотя бы одну последовательность действий.

$$T_1^{\langle j\rangle} = \{t_{1,i}^{\langle j\rangle}\} \neq \varnothing \quad \forall j. \quad (4)$$

С другой стороны, если подзадача первого порядка содержит более одной последовательности действий, она также может быть разбита на *подзадачи второго порядка* с теми же условиями на множествах последовательностей действий

$$\begin{aligned} &T_1^{\langle j\rangle} = \bigcup_k T_2^{\langle j,k\rangle}, \\ &\bigcap_k T_2^{\langle j,k\rangle} = T_2^{\langle j,0\rangle} \cap T_2^{\langle j,1\rangle} \cap T_2^{\langle j,2\rangle} \cap \ldots = \varnothing. \end{aligned} \quad (5)$$

Вследствие (1), (2), из данных уравнений можем получить обобщённые условия для подзадач первого и второго порядка.

$$\bigcap_{j,k} T_2^{\langle j,k\rangle} = t_2^{\langle 0,0\rangle} \cap t_2^{\langle 0,1\rangle} \cap \ldots \cap t_2^{\langle 1,0\rangle} \cap t_2^{\langle 1,1\rangle} \cap \ldots = \varnothing. \quad (6)$$

Разбиение подзадач может быть продолжено рекурсивно до получения подзадач z-го порядка. Критерием остановки рекурсии является такое разбиение, которое в силу требования (4) может привести только к получению подзадачи $(z+1)$-го порядка из подзадачи z-го порядка, такой что

$$T_{(z+1)}^{\langle j,k,\ldots,l,0\rangle} = T_z^{\langle j,k,\ldots,l\rangle}. \quad (7)$$

Такую подзадачу назовём *неделимой*. Для неделимых задач дальнейшее разбиение не имеет смысла.

Операторы над подзадачами

Введём обобщённый оператор разбиения **gsplit** над подзадачами: данный оператор выполняет разбиение подзадачи z-го порядка на множество подзадач $(z+1)$-го порядка

$$\text{gsplit } T_z^{\langle j,\ldots,k\rangle} \to \{T_{(z+1)}^{\langle j,\ldots,k,i\rangle}\}, \quad i \in \overline{(1, N_{(z+1)}^{\langle j,\ldots,k\rangle})}, \quad (8)$$

где $N_{(z+1)}^{\langle j,\ldots,k\rangle}$ - количество подзадач, на которые была разбита исходная подзадача $T_{(z+1)}^{\langle j,\ldots,k\rangle}$. Оператор может выполнять разбиение любыми доступными способам при соблюдении следующих условий:

$$\begin{cases} T_{(z+1)}^{\langle j,\ldots,k,a\rangle} \cap T_{(z+1)}^{\langle j,\ldots,k,b\rangle} = \varnothing \quad \forall a \neq b; \\ \bigcup_i T_{(z+1)}^{\langle j,\ldots,k,i\rangle} = T_z^{\langle j,\ldots,k\rangle}; \\ N_{(z+1)}^{\langle j,\ldots,k\rangle} > 1; \end{cases} \quad (9)$$

которые являются обобщениями (1), (2) и (4) для произвольной подзадачи.

Концепцию обобщённого оператора разбиения и подзадач можно преобразовать следующим образом для лучшего соответствия практическим применениям. Часто на практике необходимо выполнить одинаковую последовательность действий $\langle a_0, a_1, a_2, \ldots\rangle$ над некоторым множеством данных $\{r_i\}$. Чаще всего набор является упорядоченным и последовательным, т. е. имеет смысл говорить о *промежутке* решения $[r_l, r_h)$. Промежуток эквивалентен конечному счётному множеству данных $\{r_i\}$.

Таким образом, задача может быть представлена в виде пары из последовательности действий и промежутка, над которым необходимо выполнить действие

$$T_0 = \big(\langle t_i\rangle, [r_l, r_h)\big). \quad (10)$$

Такая запись предполагает выполнение одинаковой последовательности действий над всеми элементами промежутка или его частями. Эквивалентность данной записи изначальной концепции задачи может быть достигнута путём использования условных операторов внутри последовательности действий.

В виду достаточно большой сложности последовательности действий и её неизменности целесообразно обобщить её и представить в виде *оператора вычисления* **compute**, выполняющего определённые вычисления над произвольной частью промежутка с целью получения некоторого значения:

$$\text{compute}[r_a, r_b) \to v_{[r_a, r_b)}. \quad (11)$$

Покажем эквивалентность двух форм задания подзадачи. Пусть изначально имелась зада-

ча T_0^a, представленная конечным множеством последовательностей действий $T_0^a = \{t_i\}, i \in \overline{(1,N)}$. Определим промежуток $R = [1, N+1), R \in \mathrm{N}$. И зададим оператор вычисления как

$$\mathrm{compute}\,[a,b) = \bigcup_{i=a}^{b} t_i.$$

Альтернативная форма записи задачи примет вид $T_0^b = (\mathrm{compute}, R)$. Оба вида определения задач после вычисления приводят к одинаковому множеству результатов, таким образом, возможно эквивалентное преобразование между данными двумя формами записи задач.

Определим оператор разбиения **split**, как частный случай обобщённого оператора **gsplit**, для промежутков.

$$\mathrm{gsplit}(\mathrm{compute}, R) \to \{(\mathrm{compute}, R_i)\},$$

где части промежутка $R_i = \mathrm{split}\, R$ определяется при помощи оператора разбиения

$$\mathrm{split}\ R_z^{\langle j,\ldots,k\rangle} \to \{R_{(z+1)}^{\langle j,\ldots,k,i\rangle}\}, \quad i \in \overline{(1, N_z^{\langle j,\ldots,k\rangle})}. \quad (12)$$

Представим каждый промежуток в виде $R_z^{\langle j,\ldots,k\rangle} = \left[r_{z,l}^{\langle j,\ldots,k\rangle}, r_{z,h}^{\langle j,\ldots,k\rangle}\right)$ Тогда условия (9) примут вид

$$\begin{cases} r_{(z+1),l}^{\langle j,\ldots,k,0\rangle} = r_{z,l}^{\langle j,\ldots,k\rangle} \\ r_{(z+1),h}^{\langle j,\ldots,k,N_z^{\langle j,\ldots,k\rangle}\rangle} = r_{z,h}^{\langle j,\ldots,k\rangle} \\ r_{z,l}^{\langle j,\ldots,k,i\rangle} = r_{z,h}^{\langle j,\ldots,k,(i-1)\rangle} \quad \forall i \in \overline{(2, N_z^{\langle j,\ldots,k\rangle})} \\ r_{z,h}^{\langle j,\ldots,k,i\rangle} > r_{z,l}^{\langle j,\ldots,k,i\rangle} \forall i \end{cases} \quad (13)$$

Понятие пути разбиения и слияния

Последовательное применение оператора разбиения к подзадачам, представленным как в виде множеств последовательностей действий, так и в виде промежутков, приводит к формированию у каждой подзадачи некоторой последовательности индексов Γ: $T_z^{\langle \gamma_1, \gamma_2, \gamma_3, \ldots\rangle}$, где длина последовательности определяет порядок подзадачи, а каждый элемент обозначает номер подзадачи данного порядка, из которой путём разбиения была получена данная. *Путь разбиения* – это последовательность индексов подзадач i-го порядка $\Gamma = \langle \gamma_i \rangle$, из которых была получена данная подзадача. Для исходной задачи путь разбиения является пустой последовательностью $\langle\rangle$. Путь разбиения позволяет уникальным образом идентифицировать задачу, определить источник исходных данных, и назначение результирующих данных.

Применение понятия пути разбиения позволяет выявить дополнительные свойства разбиений на основе (9), в частности:

$$T_z^{\langle \gamma_1, \ldots \gamma_n, \gamma_{n+1} \ldots \gamma_N\rangle} \subset T_y^{\langle \gamma_1, \ldots \gamma_n\rangle} \quad \forall n,$$

причём $z = y + N - n$. Аналогично

$$T_z^{\langle \gamma_1, \ldots \gamma_n, \gamma_a, \ldots \gamma_N\rangle} \cap T_y^{\langle \gamma_1, \ldots \gamma_n, \gamma_b \ldots \gamma_N\rangle} = \varnothing$$

$\forall n, a, b : a \neq b.$

Задачу после разбиения можно представить в виде дерева подзадач, где корнем является исходная задача, а узлами и листьями – подзадачи. Тогда для каждого узла можно идентифицировать последовательность подзадач, которым необходим результат вычисления данной подзадачи путём отсечения последних элементов пути разбиения.

Введём оператор слияния **merge**. Данный оператор выполняет объединение результатов вычислений подзадач одного порядка.

$$\mathrm{merge}\ \{V_{(z+1)}^{\langle j,\ldots,k,i\rangle}\} \to V_z^{\langle j,\ldots,k\rangle} \quad i \in \overline{(1, N_z^{\langle j,\ldots,k\rangle})} \quad (14)$$

Объединение возможно только одновременно всех результатов подзадач, порождённых из данной подзадачи. Таким образом, оператор слияния может быть некоммутативен.

Однако, поскольку задача может быть представлена в виде произвольного дерева подзадач и результаты могут быть готовы в произвольные моменты времени, необходимым условием корректной работы оператора слияния является его ассоциативность.

Задание, которое может быть выполнено разрабатываемой системой параллельно, может быть представлено в виде кортежа:

$$(R, \mathrm{compute}, \mathrm{split}, \mathrm{merge}), \quad (15)$$

где операторы вычисления, разбиения и слияния действуют над полями

$$\begin{aligned} \mathrm{split} &= R \to \{R\}; \\ \mathrm{compute} &= R \to V; \\ \mathrm{merge} &= \{V\} \to V; \end{aligned}$$

и R, V – типы промежутка входных данных и результата соответственно.

Способы предварительного разбиения на подзадачи

Рассмотренная модель предполагает использование техники динамической реорганизации графа подзадач: изначально задача или её некоторая часть разбивается на подзадачи, каждая

из которых может быть дополнительно разделена на меньшие подзадачи в процессе выполнения. Широко известно, что граф задачи является её неотъемлемым свойством, а алгоритмы планирования стремятся оптимизировать соответствие этого графа графу вычислительной системы, что приводит к возможности двух базовых решений. Соответственно, граф задачи известен статически. В отличие от такого подхода, данная модель предполагает обобщённый механизм описания возможности разбиения задачи на части, который, тем не менее, не применяется для полной декомпозиции задачи. Модель описывает общий шаблон графа подзадач, а его реальное инстранцирование происходит в процессе выполнения, что позволяет использовать адаптивные подходы. Выполнение одной и той же задачи на одной и той же системе может приводить к различным инстанцированиям графа подзадач, в зависимости от решений, принятых в процессе работы системы.

Для данных целей мы используем две основных реализации оператора разбиения:

• равномерное разбиение – подзадача z-го порядка разбивается на меньшие подзадачи $(z+k)$-го порядка (рис. 1);

• адаптивное разбиение – подзадача z-го порядка разбивается подзадачи $(z+1)$-го порядка, часть их которых разбивается на подзадачи $(z+2)$-го порядка и т. д. до получения неделимых задач (рис. 2).

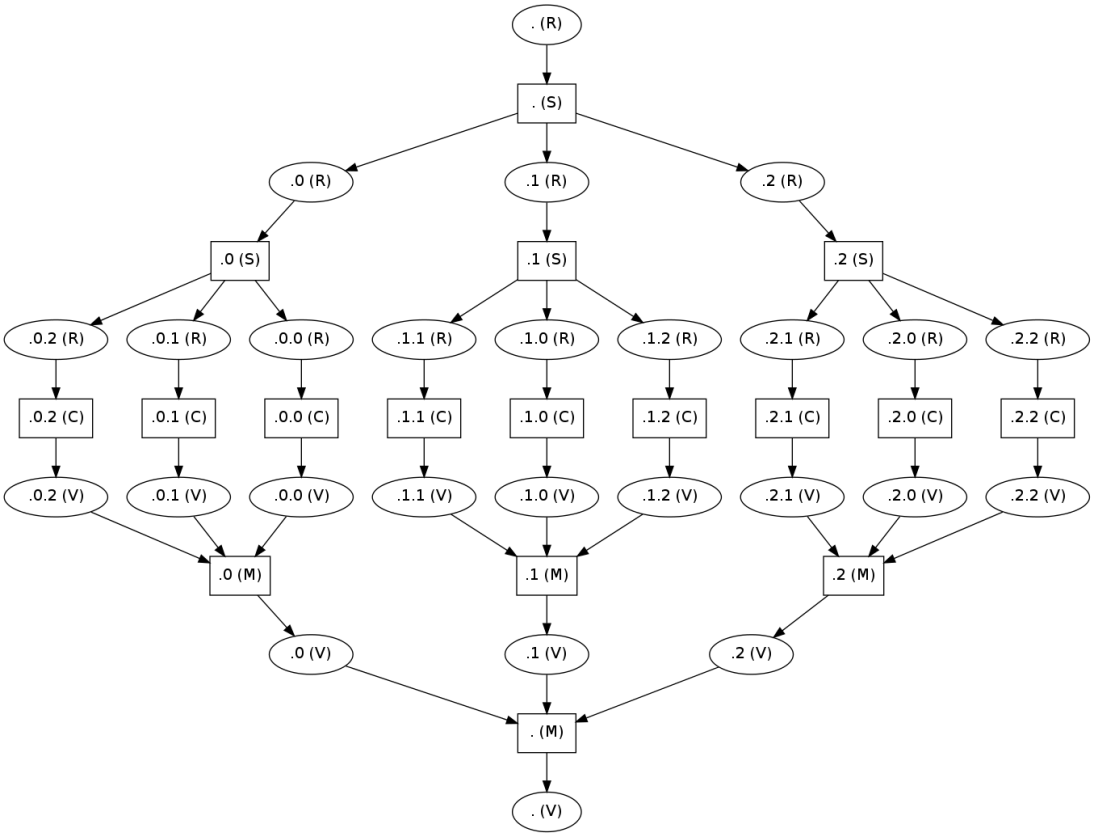


Рисунок 1. Пример разбиения задачи на подзадачи третьего порядка.

Равномерное разбиение

Равномерное разбиение характеризуется числом увеличения порядка подзадач k. Альтернативно его можно выразить через количество требуемых подзадач $k = \log_2 N$.

В данной работе исследуются два возможных варианта количества подзадач:

• разбиение на <<крупные>> подзадачи (PreSplitLargest), т. е. на минимальное количество подзадач N, такое что $N \geq P$, где P – число процессоров. Таким образом каждый процессор в идеальном случае получит ровно одну подзадачу, которая может необходимо выполнить. В таком случае ожидаются высокие коэффициенты ускорения и эффективности за счёт осуществления минимально возможного количества пересылок данных между процессорами. Тем не менее, в случае невозможности разбиения на строго необходимое количество частей ($N = P$) возможна сильная неравномерность нагрузки на процессоры, что может значительно снизить ускорение и эффективность параллельного выполнения.

• разбиение на «средние» подзадачи (PreSplitMid), т. е. на количество подзадач $N \geq P^2$. Этот способ имеет два преимущества: во-первых, часто меньшая по количеству вычислений подзадача требует меньшего количества данных, что позволит начать вычисления как можно раньше; во-вторых, нагрузка на процессоры будет лучше сбалансирована, так как неравномерность нагрузки может составлять не более времени вычисления одной задачи плюс время пересылки исходных данных. Выбор квадратичной зависимости определяется возможностью алгоритмов планирования по запросам забирать задачи из очереди другого процессора. Действительно, если каждый процессор изначально имеет в своей очереди P задач, то одну он может взять на выполнение, а оставшиеся по одной могут забрать остальные процессы остальными процессами.

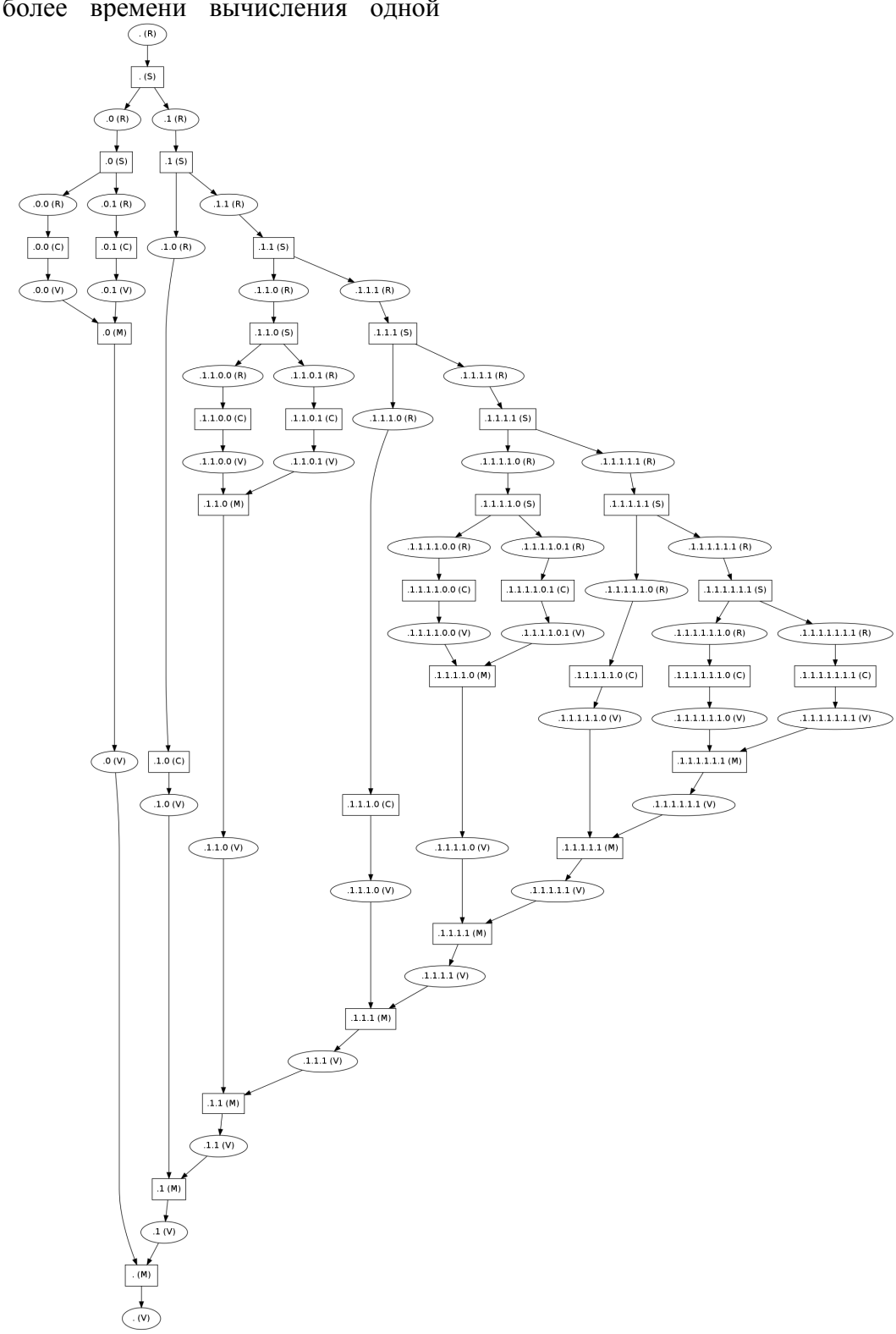


Рисунок 2. Пример динамического адаптивного разбиения задачи.

Адаптивное разбиение

Адаптивный подход к разбиению (PreSplitAdaptive) разработан с целью решения проблемы неравномерности ожидаемой нагрузки на процессоры при предварительном разбиении. Изначально он разбивает исходную задачу на минимально возможное количество подзадач, большее или равное количеству процессоров. После этого на каждый процессор назначается по одной такой подзадаче, а оставшиеся повторно разбиваются по аналогичному принципу. Разбиение заканчивается если не осталось неназначенных подзадач, т. е. каждый процессор получил в сумме одинаковые наборы подзадач, или оставшиеся задачи являются неделимыми. В худшем случае неравномерность нагрузки составит одну неделимую задачу. Данный алгоритм предварительного разбиения может быть достаточно эффективен в случае задач с равномерной вычислительной сложностью на промежутке, однако может привести к посредственным результатам в случае случайного распределения вычислительной сложности по промежутку (впрочем, как и любой другой алгоритм предварительного разбиения).

Варианты операторов разделения и слияния

Оператор разбиения **split** может быть определён для некоторой задачи $T_0 = (\text{compute}, [r_l, r_h))$ на промежутке любым способом, который обеспечивает соблюдение условий (9). Аналогичным образом определяется и оператор слияния **merge**.

В случае представления задачи в виде промежутка исходных данных целесообразно все подзадачи также представлять в виде промежутков для сохранения однородности операторов. Возможно и различное представление подзадач других порядков, однако это существенно усложняет операторы разделения и слияния из-за введения условных ветвлений. Промежуток, как правило, включает все допустимые значения в заданном интервале, что также должно соблюдаться и для промежутков подзадач большего порядка.

Таким образом, целесообразно выполнять разбиение промежутка на ряд последовательных частей. Поскольку алгоритмы строятся в предположении, что вычисления неделимых подзадач приблизительно равны по вычислительной сложности оптимальным представляется разбиение промежутка на части равных размеров.

Наиболее простым подходом является деление промежутка на две части (рис. 3), таким образом всего может существовать до 2^i подзадач i-го порядка (рис. 1). Если представить иерархию подзадач в виде дерева (рис. 2), то видно, что не всегда каждая ветвь данного дерева имеет одинаковую глубину, т. е. не всегда задачи разбиваются до одинакового порядка подзадач. Следовательно, в некоторых случаях возможно получение $N = \sum_i a_i 2^i$ задач, где $a_i \in \mathrm{N}$ – некоторые целочисленные коэффициенты. Можно показать, что это уравнение относительно неизвестных a_i имеет неотрицательное решение для любых положительных N.

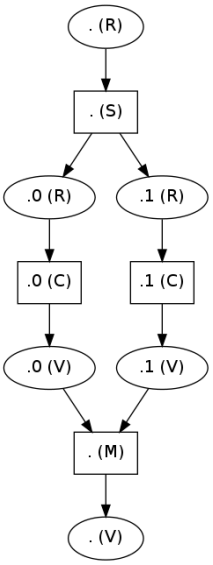


Рисунок 3. Схема разбиения на две подзадачи

Рассмотрим также разделение на три одинаковые части (рис. 4). Аналогично может существовать не более 3^i подзадач i-го порядка. Также возможно образование $N = \sum_i a_i 3^i$ задач. Рассмотрим детально это уравнение. Пусть изначально существует одна подзадача нулевого порядка. Её можно разделить на три части, т. е. заменить одну подзадачу тремя, увеличив таким образом количество подзадач на 2. Аналогичным образом любая подзадача z-го порядка может быть заменена на 3 подзадачи $(z+1)$-го порядка, увеличивая суммарное количество подзадач на 2. Таким образом, N может принимать только нечётные значения, что может сказаться на балансировании нагрузки на процессоры.

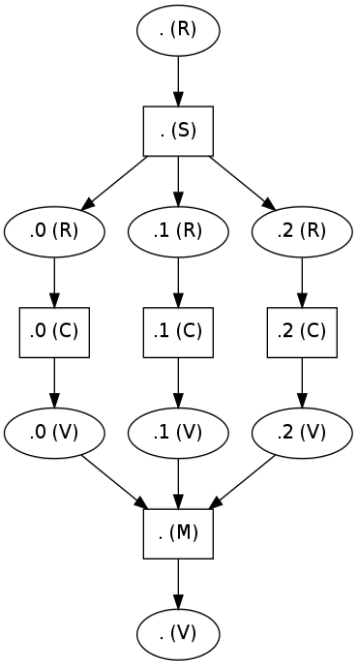


Рисунок 4. Схема разбиения на три подзадачи

Разделение на четыре одинаковые части может быть получено двойным применением разделения на две. Однако, используя рассуждения, аналогичные предложенным для разбиения на три части, можно установить, что суммарное количество подзадач в данном случае может составлять 1,4,7,10,13,….

Обобщая рассмотрим разделение на k одинаковых частей. k конечно, т. к. промежуток конечного размера может быть разделен только на конечное число неделимых частей (в случае существования условия неделимости). Если перед выполнением разделения очередной подзадачи существовало N_a подзадач, то после замены одной из них на k новых суммарное количество подзадач станет равным $N_b = N_a + k - 1$. Таким образом, любое количество разделений данного типа может привести только к $N = N_i + j(k-1)$ подзадачам, где $N_i = 1$ – начальное количество задач, $j \in \mathrm{N}$ – некоторая константа.

Для разделения на k количество подзадач должно удовлетворять условию $N \equiv 1 \bmod (k-1)$. Для $k = 2$ может быть получено любое количество задач, а с ростом k множество возможных значений уменьшается. Следовательно, рассматривать разделение более чем на 3 части нецелесообразно.

Выводы

В последнее время широкое распространение получили модели программирования распределённых систем с использованием высокоуровневых абстракций, например подзадач. Такой подход позволяет с одной стороны скрыть от пользователя сложность внутренней организации аппаратной компоненты современных распределённых вычислительных систем, т. к. пользователь получает возможность описывать задачу в терминах некоторых абстрактных математических операций, а с другой – получить возможность использовать одно и то же описание задачи с использованием некоторого программного интерфейса для работы на вычислительных системах со значительно различающимися архитектурными организациями.

Существующие модели представления задач при помощи подобных абстракций не предполагают использование систем с локальной памятью в виду того, что не учитывают пересылки и локальность данных. Предложенная модель расширяет одну из существующих моделей таким образом, чтобы исправить этот недостаток.

Предложенная модель использует обобщённые математические операции для описания пользовательской задачи и возможности её представления для параллельного вычисления. Данная модель абстрагирует особенности организации конкретной вычислительной системы и позволяет описывать лишь задачу и корректный способ передачи данных для этой задачи. Динамическое конструирование графа подзадач позволит в будущем использовать адаптивные подходы при выборе способа разбиения задачи на подзадачи и использовать эту информацию для улучшения адаптивных алгоритмов динамического планирования.

Предложенная система представляет собой математическую модель, которая может быть использована для анализа и изучения процесса выполнения параллельных программ в распределённых системах кластерного типа. В дальнейшем предполагается расширить данную модель таким образом, чтобы более полно учесть пересылки данных при выполнении задачи, а также учесть вероятностные свойства динамического выполнения подзадач и передачи данных между ними.

Список литературы:

1. *Bastoul Cedric.* Contributions to High-Level Program Optimization // Habilitation (Dr.Sc.) Thesis. Paris-Sud University, France. – 2012.
2. Automatic parallelization and locality optimization of beam forming algorithms [Text] / Albert Hartono, Nicolas Vasilache, Cedric Bastoul *et al.* // High Performance Embedded Computing Workshop (HPEC). – MIT Lincoln Laboratory, Lexington, Massachusetts: 2010.
3. *Kirk D.* Nvidia CUDA software and gpu parallel computing architecture [Text]: Tech. rep.: Nvidia, 2007.
4. 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.
5. Board OpenMP Architecture Review. OpenMP Application Program Interface Version 3.1 [Text]. – 2011.
6. *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.
7. Intel Threading building blocks documentation [Electronic resource]. – Access mode: http://threadingbuildingblocks.org/documentation.php. – Last access: 08.05.2012. – Title from the screen.
8. JTCL.22.32 – ISO/IEC 14882 international standard: Programming language c++ [Electronic resource]. – Access mode: http://www.open-std.org/jtc1/sc22/wg21/. – Last access: 01.03.2013. – Title from the screen.
9. MPI: A message-passing interface standard [Electronic resource]. – Access mode: http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf. – Last access: 08.05.2012. – Title from the screen.

УДК 519.25

НАЗАРЕВИЧ О.Б.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ МОНІТОРИНГУ ГАЗОСПОЖИВАННЯ МІСТА НА ОСНОВІ АДИТИВНОЇ МОДЕЛІ ТА З ВРАХУВАННЯМ МЕТЕОФАКТОРІВ

В даній статті розглянута практична реалізація інформаційної технології моніторингу газоспоживання міста (обласного центра). На базі запропонованої адитивної моделі і з врахування метеофакторів. Обґрунтовані основні вимоги та запропонована автоматизована система моніторингу газоспоживання міста. Описана функціональна схема формування та накопичення бази вимірювань газоспоживання та метеофакторів. Запропонована структура БД для зберігання інформації з кроком накопичення година, доба, тиждень. На конкретному прикладі показано результати роботи даної автоматизованої системи моніторингу газоспоживання міста.

The present article deals with the practical implementation of information technology for monitoring the gas consumption of the city (regional centers) on the basis of proposals additive model and taking meteorological factors. Justified and suggested automating the monitoring system of gas consumption. The functional diagram of the formation of a database of measurements and framework developed of a database to store statistical data measurement of gas consumption and meteorological factors for interval accumulation hour, day and week. On example, demonstrated results applying automate system for monitoring of gas consumption of the city.

1. Вступ

В даній статті розглянута практична реалізація інформаційної технології моніторингу газоспоживання міста (обласного центра). На базі запропонованої адитивної моделі і з врахування метеофакторів. Обґрунтовані основні вимоги та запропонована автоматизована система моніторингу газоспоживання міста. Описана функціональна схема формування та накопичення бази вимірювань газоспоживання та метеофакторів. Запропонована структура БД для зберігання інформації з кроком накопичення година, доба, тиждень. На конкретному прикладі показано результати роботи даної автоматизованої системи моніторингу газоспоживання міста.

2. Постановка задачі

На основі статистичних даних вимірювання газоспоживання та архіву метеоданих міста необхідно запропонувати інформаційну технологію моніторингу газоспоживання міста.

Описати функціонування системи збору інформації з витратомірних комплексів «Флоутек» та архіву метеоданих.

Розробити структуру бази даних для збереження даних вимірювання з кроками накопичення година, доба, тиждень.

Запропонувати автоматизовану систему моніторингу газоспоживання міста та алгоритми статистичної обробки даних на основі запропонованої математичної моделі та з врахування метеофакторів.

3. Задачі моніторингу

Під *моніторингом* процесу газоспоживання міста розуміємо комплекс завдань вимірювання даних газоспоживання, формування БД, визначення статистичних характеристик. Це перш за все, визначення трендових компонент та залишкових компонент часових рядів газоспоживання, завдань поточного та довгострокового моніторингу та прогнозування. Проведення порівняльного аналізу реальних і прогнозованих значень процесу газоспоживання, здебільшого проаналізованих на річному інтервалі з кроками накопичення даних: тиждень, доба, година.

Під задачами моніторингу газоспоживання міста розуміємо таке:

1) поточний моніторинг (оперативний та короткостроковий):

– вимірювання поточних характеристик газоспоживання з інтервалом накопичення: година, доба, тиждень;

– вивід даних вимірювання газоспоживання та метеофакторів у вигляді наочних графіків. Наприклад, графіків газоспоживання, робочого тиску та середньої температури з кроком одна година та доба;

– відображення погодинних та добових графіків у вигляді суми адитивних компонент: тренду, квазігармонійних компонент і стохастичного залишку.

2) довгостроковий моніторинг:

– вимірювання поточних характеристик газоспоживання з інтервалом накопичення: тиждень, місяць, квартал, рік;

– вивід даних вимірювання газоспоживання та метеофакторів у вигляді наочних графіків. Наприклад графіків, добового газоспоживання та середньодобової температури;

– на основі математичної моделі провести аналіз та прогнозування витрат газу врахувавши прогнозовані значення температури повітря на наступну добу.

4. Адитивна модель газоспоживання міста

В результаті апріорного аналізу статистичних даних вимірювання газоспоживання міста (обласного центру) було визначено топологію споживачів газу та основних факторів, що впливають на динаміку газоспоживання на річному інтервалі спостереження [1, 2].

Загальною математичною моделлю вибрано *векторний випадковий процес:*

$$\Theta(\omega,t)=\left(\xi_1(\omega,t),...,\xi_n(\omega,t)\right),\quad t\in T,\quad \omega\notin\Omega$$

де $\xi_n(\omega,t)$ - вектор випадкових процесів пов'язаних з технологічними параметрами газоспоживання: об'єм спожитого газу, робочий тиск в газопроводі, температура повітря в місті, а також зміна топології споживачів.

Тобто стан системи газоспоживання у певний момент часу t характеризується багатьма випадковими величинами. Так як дані вимірювання у нас є дискретними, а зміна топології споживача відбувається на двох основних рівнях (наявність чи відсутність центрального опалення) то така математична модель буде відноситися до класу моделей з дискретними станами системи і дискретним часом.

На етапі попередньої статистичної обробки, запропоновано використання методу «Гусениці-SSA» [3]. Використання такого методу, по суті, є оберненою задачею: на основі аналізу часового ряду газоспоживання необхідно створити математичну модель, оскільки вона апріорно не задається.

В залежності від кроку накопичення статистичних даних газоспоживання буде мінятися і конкретна математична модель. Так, для погодинного часового ряду газоспоживання будемо розглядати адитивну математичну модель:

$$\nu(\omega,t)=A_0(t)+\sum_{j=1}^{k}B_j(t)+\xi(\omega,t)$$

де, $A_0(t)$ – річний тренд; $B_j(t)$ – сума циклічних (квазігармонійних) компонент; $\xi(\omega,t)$ – стохастичний залишок.

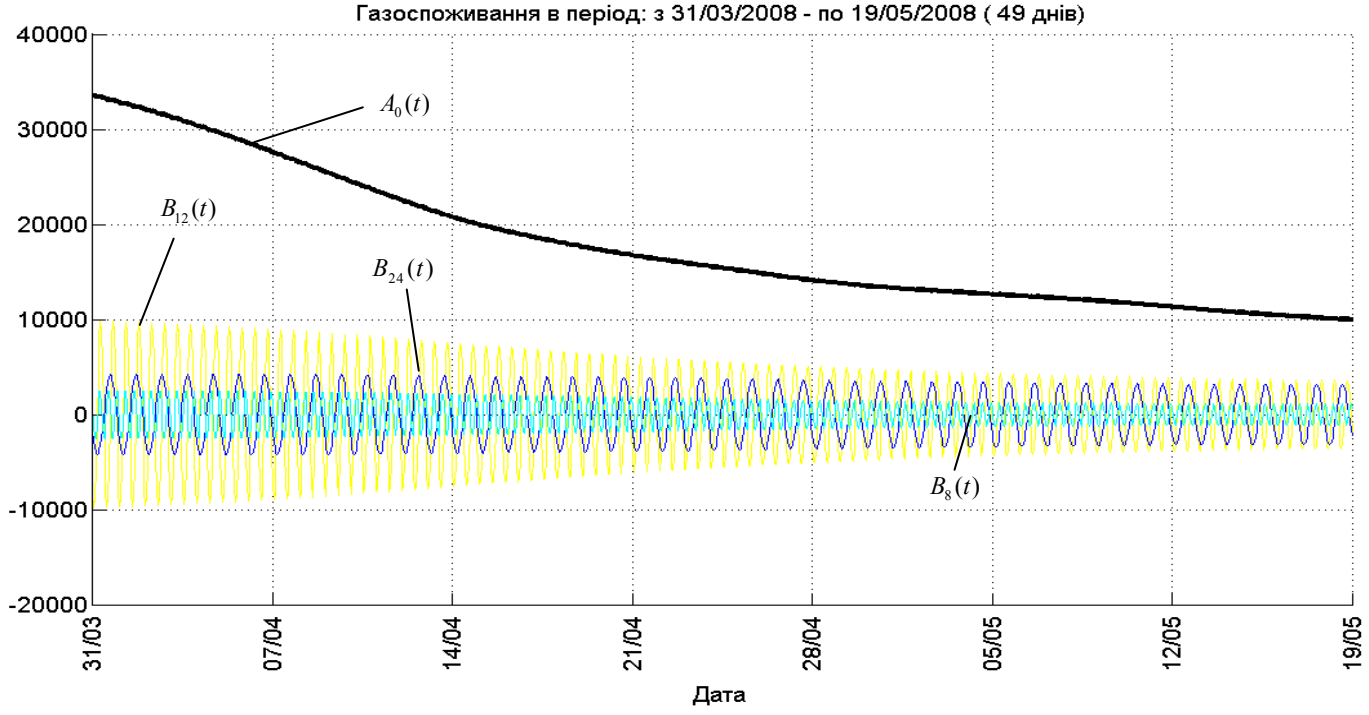


Рис. 1. Графічне зображення адитивних компонент часового ряду газоспоживання: $A_0(t)$ (тренд) та $B_8(t)$, $B_{12}(t)$, $B_{24}(t)$ (квазігармонійні) на інтервалі 5 тижнів у весняний період 2008 року

5. Функціональна схема формування бази вимірювань

Головним робочим параметром технологічного процесу газопостачання на газорозподільній станції є правильне встановлення робочого тиску в системі газопостачання міста. Дана величина напряму залежить від добового споживання газу, яке фактично потрібно передбачити залежно від метеофакторів міста, керуючись прогнозом погоди на наступну добу. Як було зазначено вище, найбільш суттєвим фактором, що впливає на процес газоспоживання є температура повітря, вологість та сила вітру в місті.

В статті запропоновано об'єднати інформацію з геоінформаційних системи (ГІС) обліку газоспоживання та метеоінформацію для міста. Саме таке інформаційне забезпечення є основою розробленої і сформованої БД накопичення результатів вимірювань газоспоживання та метеоданих для міста. Функціональну схему формування БД метеоданих та даних вимірювання газоспоживання міста зображено на рис. 2.

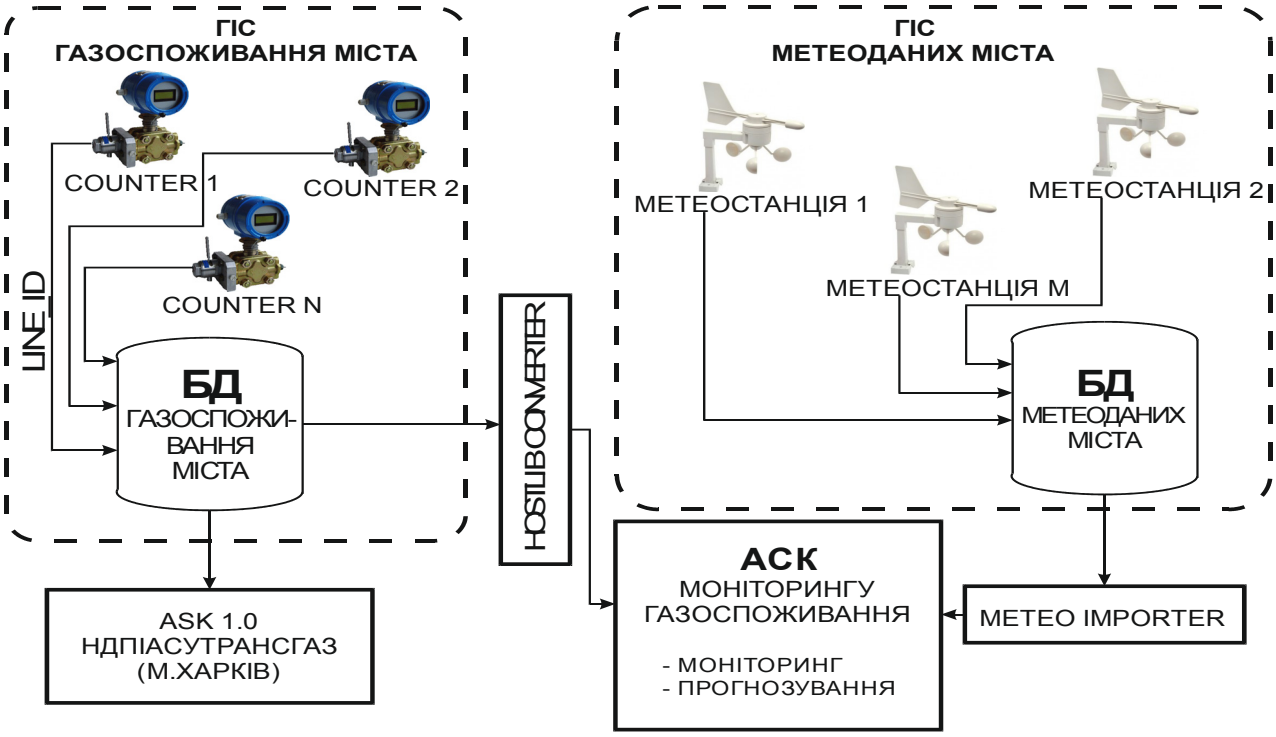


Рис. 2. Функціональна схема формування БД метеоданих та даних вимірювання газоспоживання міста

Таке поєднання інформації з двох ГІС для міста надалі забезпечить комплексний моніторинг та прогнозування газоспоживання щодо автоматизації виробничих потреб диспетчерської служби газопостачання.

6. Структурна схема БД

Вхідними даними для зчитування є двійкові файли з лічильника у промисловому форматі Hostlib для витратомірних комплексів «Флоутек». Для імпорту в БД двійкових даних формату HostLib було розроблено програмне забезпечення «HostLibConverter 1.0» [4]. А для імпорту метеоданих було розроблено програмне забезпечення «MeteoInfoSpaceRu Temperature Importer».

В результаті розробки інформаційної технології запропоновано БД, структура таблиць якої дозволяє зберігати та формувати часові ряди для моніторингу, подальшого статистичного аналізу і прогнозування газоспоживання міста із врахуванням метеофакторів. Перелік основних таблиць та зв'язків БД представлено на рис. 3.

Спроектована БД містить початкову (вхідну) інформацію про газоспоживання та метеофактори для міста, що розміщена в таблицях: T_GAZ_MINUTES, T_GAZ_ HOURS, T_GAZ_DAYS, METEO_HOURS. Таблиці T_GAZ_ SUM_HOURS та T_GAZ_SUM_DAYS є віртуальними таблицями, на основі попередніх, що відображають сумарну інформацію із усіх лічильників (COUNTER) для вказаної інформаційної лінії (LINE_ID). Результат побудови VIEW T_GAZ_SUM_DAYS можна отримати за допомогою наступного SQL-запиту:

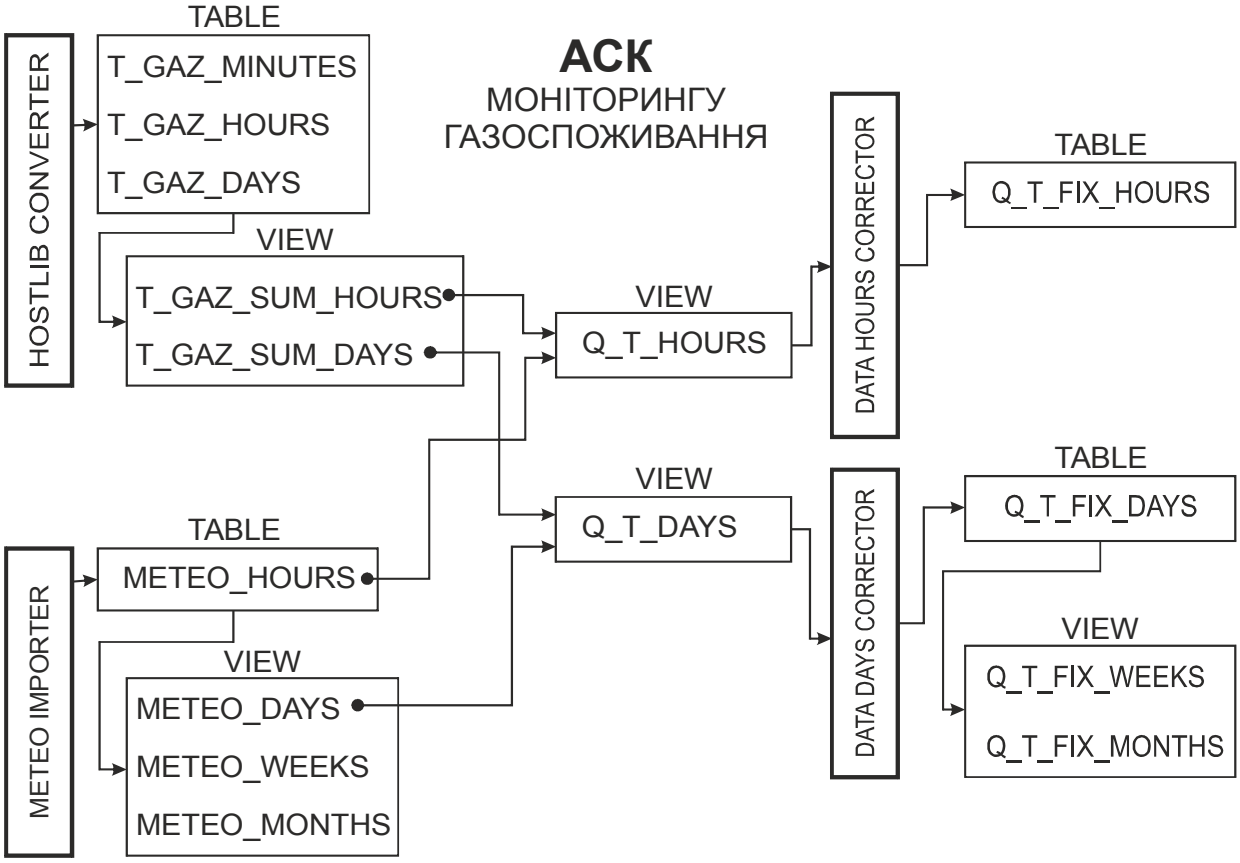


Рис. 3. Структурна схема БД та зв'язків між таблицями

```
CREATE VIEW `t_gaz_sum_days` AS select `t_gaz_days`.`TS`
AS `TS`,sum(`t_gaz_days`.`Q`) AS `Q`,`t_gaz_days`.`LINE_ID` AS `LINE_ID`
from `t_gaz_days` group by `t_gaz_days`.`TS`,`t_gaz_days`.`LINE_ID`
order by `t_gaz_days`.`TS`;
```

Аналогічно створено таблицю для суми погодинних значень газоспоживання міста VIEW T_GAZ_SUM_HOURS.

Наступний етап побудови БД – усереднення температури в місті з погодинних (METEO_HOURS) для середньодобових та середньотижневих даних. Даний алгоритм реалізовано SQL-запитом, що представлений на наступному лістингу.

```
CREATE VIEW `meteo_days` AS select cast(`meteo_hours`.`Timestamp` as date)
AS `Date`,count(`meteo_hours`.`T`) AS `Count`,avg(`meteo_hours`.`T`) AS
`Tcep`
from `meteo_hours` group by cast(`meteo_hours`.`Timestamp` as date)
order by cast(`meteo_hours`.`Timestamp` as date);
```

Таким чином сформовано таблиці усередненої температури за добу, тиждень та місяць: METEO_DAYS, METEO_WEEKS, METEO_MONTHS.

Для проведення регресійного аналізу та короткострокового прогнозування газоспоживання було сформовано об'єднанні віртуальні таблиці Q_T_HOURS та Q_T_DAYS газоспоживання і температури з інтервалами накопичення година та доба.

Дані газоспоживання та середньої температури зберігаються у таблицях t_gaz_hours_te та meteo_hours. Синхронне формування часового ряду реалізовано з допомогою інструкції left outer join. Інтервал спостереження вибірки вказується за допомогою інструкції BETWEEN.

Часові ряди з інтервалом дискретизації тиждень, місяць та рік формуються шляхом сумування первинної погодинної інформації. Наприклад щоб отримати добові дані газоспоживання сумуємо 24 години для кожної доби, а для формування тижневих даних, сумуємо 7 діб для кожного тижня із інтервалу спостереження.

7. Автоматизована система моніторингу газоспоживання

Логічним завершення створення інформаційної технології є реалізація автоматизованої системи моніторингу та прогнозування газоспоживання міста. На шляху до побудови є три етапи (концепція Самарського «модель-алгоритм-програма»):

– побудова математичних моделей та методів;

– створення алгоритмів обробки статистичних даних вимірювань;

– реалізація програмного забезпечення на основі моделей та алгоритмів.

На сьогодні диспетчерською службою міста використовується програмний комплекс «ASK 1.0», розроблений НДПІАСУТРАНСГАЗ (м. Харків), призначений для:

– зчитування даних із автоматичних обчислювачів (АО) витрати газу;

– формування бази даних визначеної структури (так званого Hostlib-формату);

– віддаленого введення в обчислювачі параметрів газу, атмосферного тиску, а також коректування часу;

– перегляду та аналізу отриманої інформації;

– формування, перегляду і друку (чи збереження у файлі) добових і місячних звітів визначеної форми.

Фактично даний програмний комплекс реалізовує геоінформаційну систему (ГІС) опитування автоматичних обчислювачів витрат газу для міста.

Найбільшим недоліком даної системи є відсутність аналізу даних вимірювання в поєднані із метеофакторами. Тому процес аналізу та прийняття управлінського рішення щодо параметрів газопостачання є доволі рутинним та неавтоматизованим процесом. Тому було запропоновано автоматизовану систему моніторингу та прогнозування газоспоживання з врахуванням метеофакторів на основі адитивної математичної моделі.

На Рис. 4 представлено структурну схему автоматизованої системи моніторингу газоспоживання міста, що містить такі основні розділи:

1) моніторинг;

2) попередня обробка та корегування даних вимірювань;

3) статистична обробка даних вимірювань;

4) прогнозування газоспоживання;

5) діагностика аварійних станів.

Моніторинг передбачає побудову графіків робочих параметрів процесу газоспоживання, а саме Q – об'єму спожитого газу ($м^3$), RBS – робочого тиску в системі трубопроводу ($кгс/см^2$), Т – температуру повітря в місті (^oC). Моніторинг будемо поділяти на поточний та довгостроковий, котрі в свою чергу поділяються на оперативний, короткостроковий та тижневий, місячний, річний. Приклад добового моніторингу зображено на рис. 5. Завдання оператора, що контроює робочий тиск газу зберігати заданий диспетчером робочий тиск в газопарової в межах 5% він номінального значення.

Попередня обробка та корегування даних вимірювань передбачає два основних завдання. По-перше, це видалення заздалегідь неправильних значень результатів вимірювання в напівавтоматичному режимі, тобто система виводить «підозрілі» значення газоспоживання, а оператор приймає рішення щодо збереження чи видалення такої інформації.

По-друге, трапляються випадки, коли окремі вимірювачі виходять із ладу й передають неправильні заміри даних газоспоживання чи інших робочих параметрів газопроводу. Враховуючи це, важливо оператором в ручному режимі корегувати дані та вводити правильні показники. Для цього передбачено відповідні таблиці із внесеними коректівами даних Q_T_FIX_HOURS та Q_T_FIX_DAYS для погодинних і добових значень, що відображені на структурній схемі БД (рис. 3).

Статистична обробка даних вимірювань є основним розділом, що містить реалізацію математичних моделей та методів у вигляді конкретних алгоритмів інформаційної технології моніторингу газоспоживання міста.

Цей розділ реалізовує наступні можливості: а) виділення трендів (методами EMD і «Гусениця-SSA»), а також квазігармонійних компонент та стохастичного залишку; оцінка математичного сподівання та дисперсії розсіювання для річних та квазігармонійних компонент; б) поділ часового ряду газоспоживання на сезонні ділянки (аналіз стохастичного залишку за зміною дисперсії [5], та аналіз температури, тобто знаходження дати ввімкнення і вимкнення центрального опалення міста); в) кореляційний аналіз (перевірка кореляційної залежності Q-T на кожній виділеній сезонній ділянці); г) регресійний аналіз (побудова лінії регресії для кожної сезонної ділянки). Зауважимо, що такі розрахунки здійснюються з двома кроками накопичення даних вимірювань: погодинні та добові дані газоспоживання та температури міста (рис. 6).

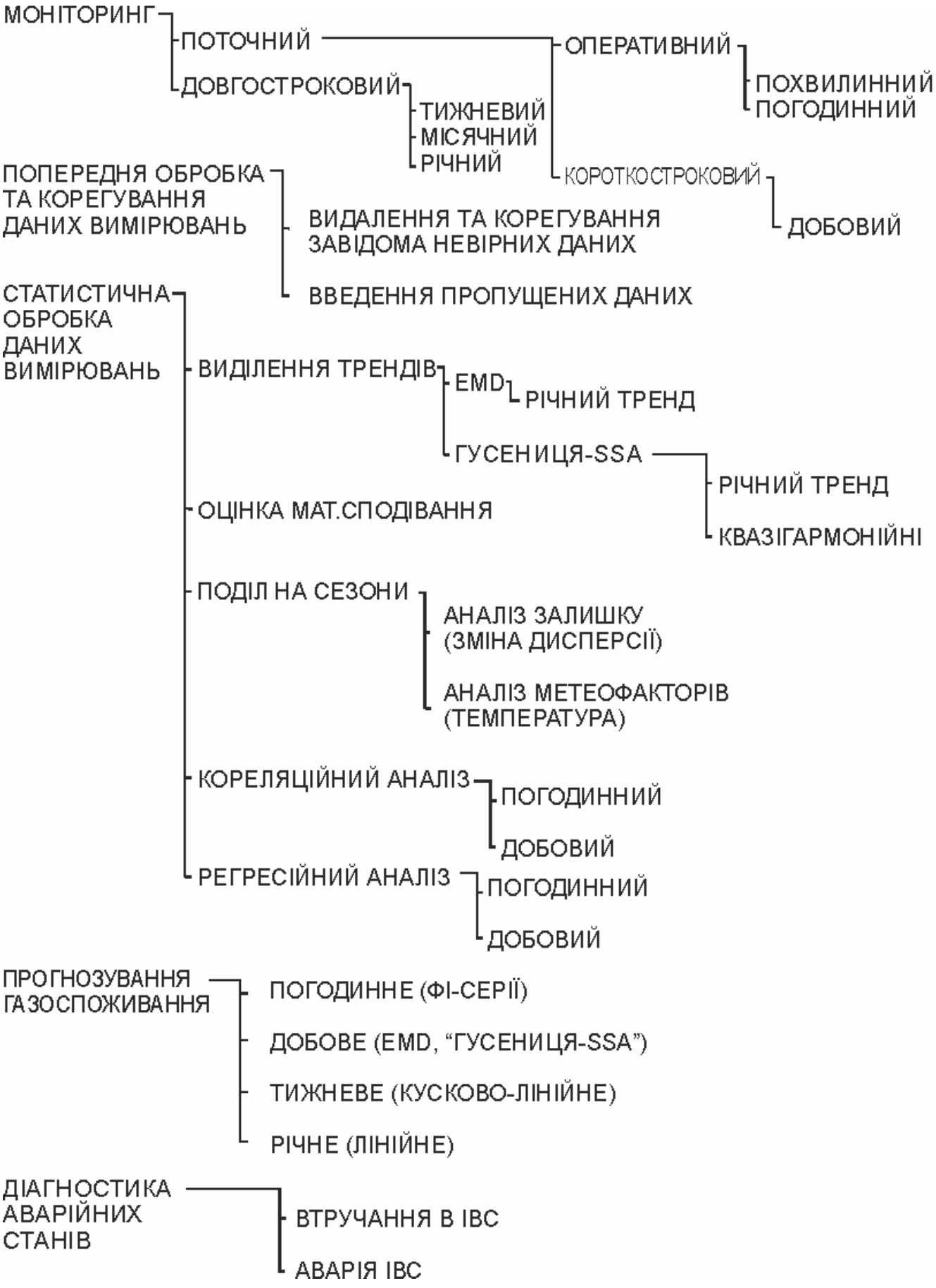


Рис. 4. Структурна схема автоматизованої системи моніторингу газоспоживання міста

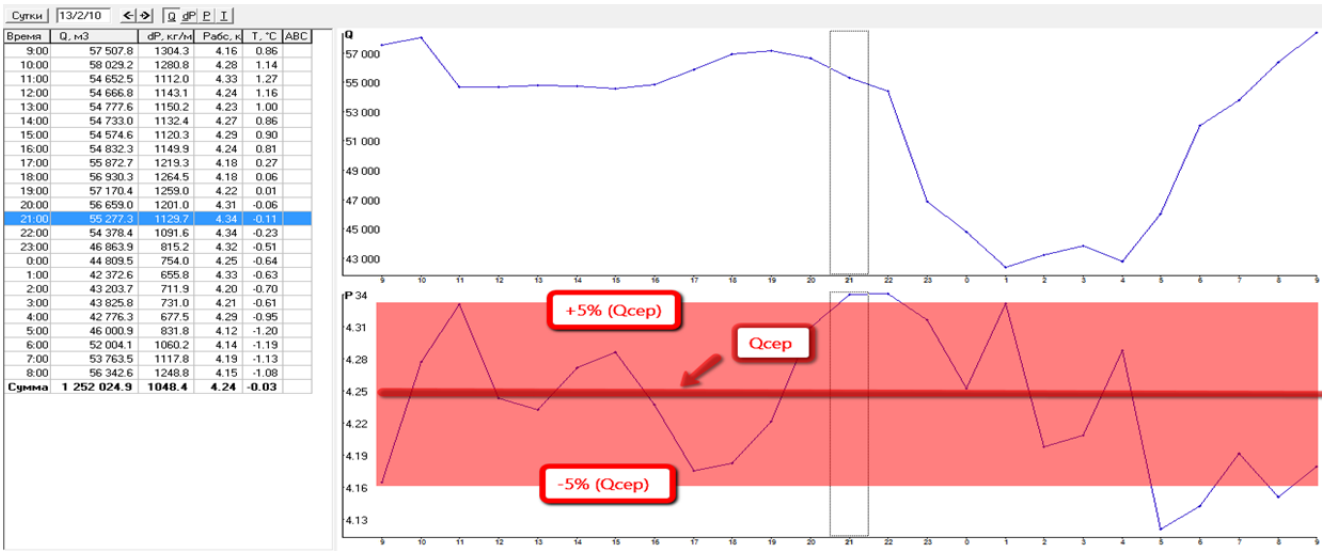


Рис. 5. Графік погодинного моніторингу Q і P упродовж доби

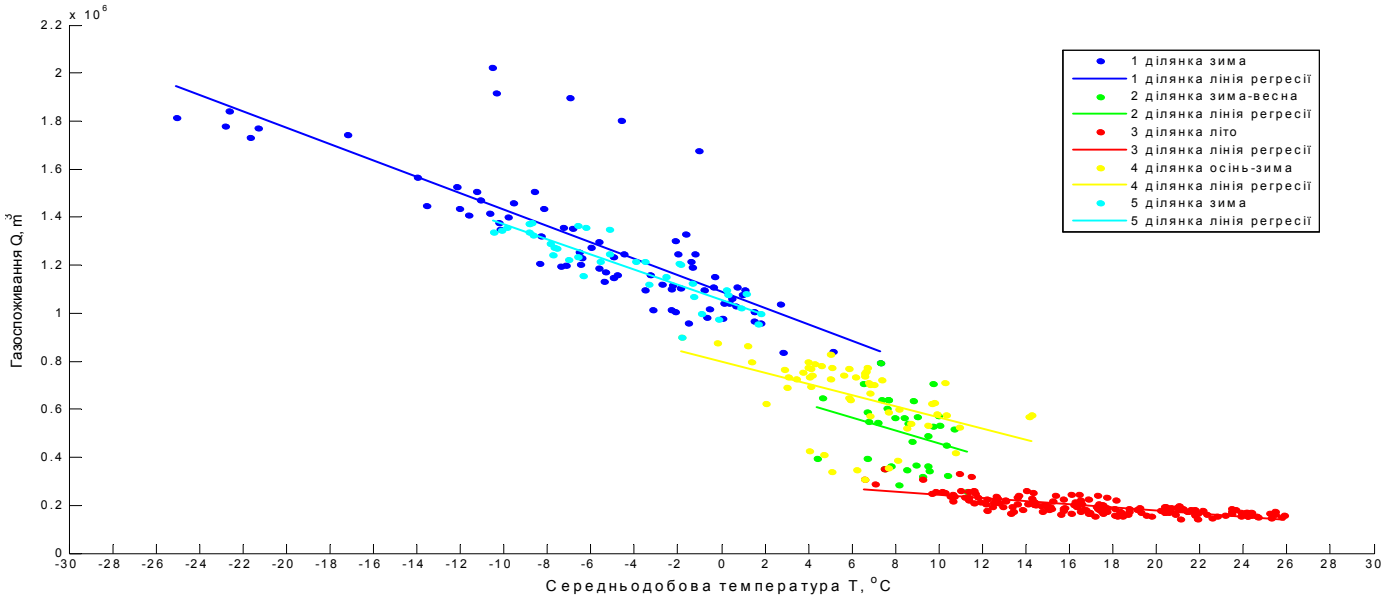


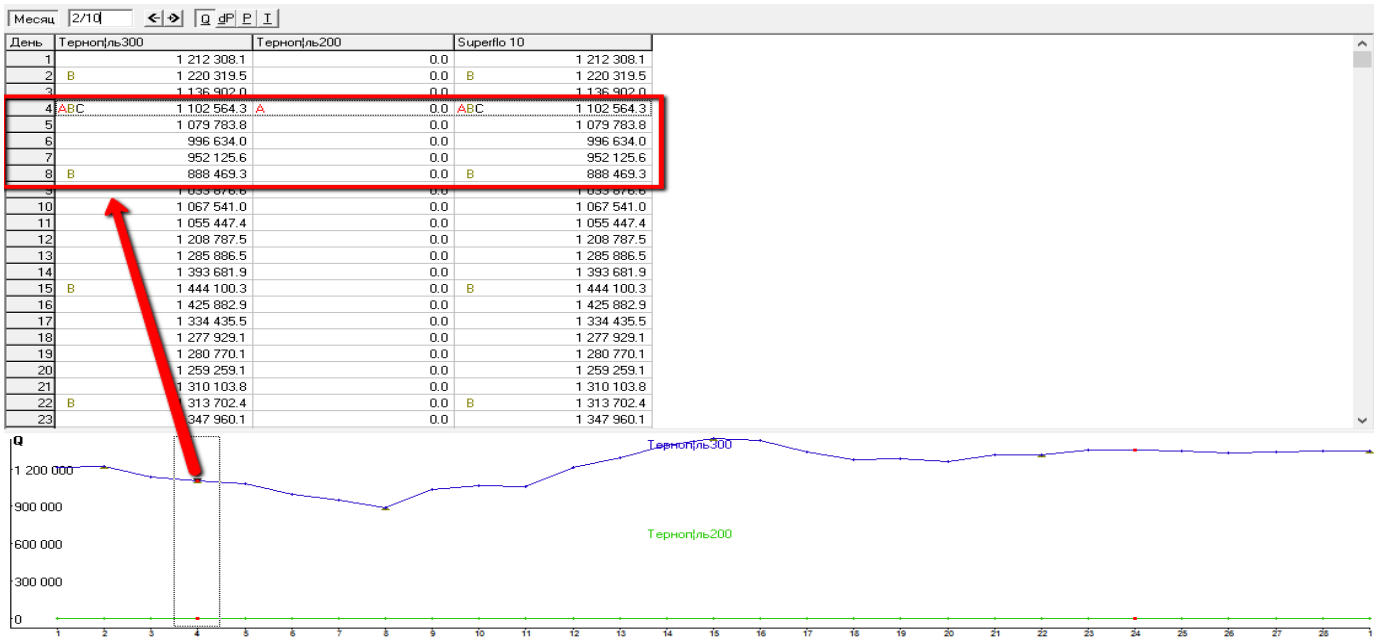
Рис. 6. Графік ліній регресії для п'яти сезонів часового ряду газоспоживання за 2010 рік

Прогнозування газоспоживання є важливим етапом реалізації моніторингу. Газоспоживання міста за попередню добу та прогнозоване значення очікуваної середньодобової температури в місті визначає об'єм газоспоживання на наступну добу. Що задає робочий тиск в газопроводі і безпосередньо впливає на швидкість подачі природного газу в місто. Неправильне встановлення технологічних параметрів подачі газу (а саме, робочого тиску в газопроводі) зумовлює аварійні ситуації, а також незаплановане вимкнення газу для окремих категорій споживачів. Розділ прогнозування газоспоживання містить реалізацію таких алгоритмів: прогнозування погодинного газоспоживання на найближчу добу, добове прогнозування на 3-5 діб, тижневе прогнозування (кусково-лінійне) та річне (лінійне). Такі види прогнозу дають аналітичний матеріал диспетчерській службі та керівництву газопроводу для проведення оперативного, коротко- та довготривалого управління і стратегічного планування продовж наступних років.

Діагностика аварійних станів дає можливість переглянути показники даних за певне число та годину, отримані з деякими порушеннями, а саме: втручанням в автоматичні вимірювачі, несправністю вимірювачів, введенням значень вручну (робота з константами).

У форматі даних, що передаються з автоматичних лічильників передбачено три види аварій: A, B, C. Покази A сигналізують про виник-

нення аварійних ситуаціїй (несправність аналогового датчика, dP – нижче мінімального значення, напруга живлення менше допустимої, збій живлення, несправність в передачі даних). В – несанкціоноване втручання в пам'ять обчислювача. С – робота з константами (ручне введення витрат газу, тиску та інших характеристик, що впливають на покази газоспоживання). Приклад діагностики аварійних станів зображено на рис.7.



День	Термопіль300	Термопіль200	Superflo 10
1	1 212 308.1	0.0	1 212 308.1
2	B 1 220 319.5	0.0	B 1 220 319.5
3	1 136 902.0	0.0	1 136 902.0
4	ABC 1 102 564.3	A 0.0	ABC 1 102 564.3
5	1 079 783.8	0.0	1 079 783.8
6	996 634.0	0.0	996 634.0
7	952 125.6	0.0	952 125.6
8	B 888 469.3	0.0	B 888 469.3
9	1 033 676.6	0.0	1 033 676.6
10	1 067 541.0	0.0	1 067 541.0
11	1 055 447.4	0.0	1 055 447.4
12	1 208 787.5	0.0	1 208 787.5
13	1 285 886.5	0.0	1 285 886.5
14	1 393 681.9	0.0	1 393 681.9
15	B 1 444 100.3	0.0	B 1 444 100.3
16	1 425 882.9	0.0	1 425 882.9
17	1 334 435.5	0.0	1 334 435.5
18	1 277 929.1	0.0	1 277 929.1
19	1 280 770.1	0.0	1 280 770.1
20	1 259 259.1	0.0	1 259 259.1
21	1 310 103.8	0.0	1 310 103.8
22	B 1 313 702.4	0.0	B 1 313 702.4
23	1 347 960.1	0.0	1 347 960.1

Рис. 7. Приклад діагностики аварійної ситуації на газопроводі

8. Висновки

У статті розв'язана актуальна науково-практична задача створення інформаційної технології моніторингу газоспоживання міста на основі адитивної моделі та з врахуванням метеофакторів. Запропоновано структуру БД та сформовано статистичну інформацію результатів вимірювання процесу газоспоживання міста та метеоданих з кроком накопичення даних година, доба, тиждень, місяць та рік.

Розроблена інформаційна технологія моніторингу газоспоживання міста на основі запропонованої математичної моделі. Запропонована система моніторингу автоматизовує функції і завдання виробничого й організаційного управління технологічних процесом постачання природного газу. На реальних статистичних даних вимірювання газоспоживання міста показано результати: моніторингу та статистичної обробки та прогнозування витрат газу на наступну добу, діагностика аварійних ситуацій за допомогою автоматизованої системи моніторингу газоспоживання міста (обласного центру).

Список літератури

Назаревич О.Б. Виділення річного тренду як адитивної складової часового ряду газоспоживання / О.Б. Назаревич // Вісник ТНТУ (Математичне моделювання. Математика. Фізика). – 2011. – Т. 16, № 4. – С. 201-209.

Назаревич О.Б. Статистичний аналіз динаміки газоспоживання міста / О.В. Мацюк, О.Б. Назаревич, Л.М. Щербак // Моделювання та інформаційні технології збірник наукових праць (інститут проблем моделювання в енергетці НАН України ім. Г.Є. Пухова). – Київ, 2011. – № 61. – С. 37-45.

Golyandina N.E. Analysis of Time Series Structure: SSA and Related Techniques / Golyandina N.E., Nekrutkin V.V., and Zhigljavsky A.A. – Boca Raton: Chapman&Hall/CRC, 2000. – 305 p.

Назаревич О. Комп'ютерна програма HostLibConverter 1.0 / О.В. Мацюк, Р.М. Лукавий, О.Б. Назаревич, Н.В. Пйонтко, Г.В. Шимчук // А.с. про реєстрацію авторського права на твір №40120 "Комп'ютерна програма HostLibConverter 1.0" [Текст]. – 2011.

Brodsky B.E. A nonparametric method for the segmentation of the EEG / B.E. Brodsky, B.S. Darkhovsky, A.Ya. Kaplan, S.L. Shishkin // Computer Method and Programs in Biomedicine №60. – 1999. – P.93-106.

УДК 004.932.72'1

ДЬЯКОНОВА С.В.,
ЗАЙЧЕНКО Ю.П.

АНАЛИЗ МЕТОДОВ СЕГМЕНТАЦИИ СПУТНИКОВЫХ ИЗОБРАЖЕНИЙ

В статье проводится сравнительный анализ различных методов сегментации изображений, позволяющих выделять связные области, применительно к задаче обнаружения зданий на спутниковых изображениях. Для оценки качества сегментации используются критерии чрезмерной и недостаточной сегментации. Предложен подход к сегментации спутниковых изображений, основанный на использовании комбинации методов сегментации и нечеткой кластеризации с применением морфологической обработки.

The article presents a comparative analysis of different methods for image segmentation that allows extracting relevant connected regions in the images, as applied to detecting buildings in Satellite imagery. Over and under segmentation criteria used to assess the quality of segmentation. An approach to segmentation of satellite images based on a combination of methods of image segmentation and fuzzy clustering using morphological processing.

Введение

Сегодня городские территории являются наиболее динамично развивающимися областями. Здесь предъявляются самые высокие требования к геодезическим данным в части актуальности и пространственного разрешения. Спутниковые изображения сверхвысокого разрешения обеспечивают средства для поддержания городских геодезических данных в актуальном состоянии и их документирования. Городские администрации, которые занимаются управлением и планированием окружающей среды, все чаще переходят к использованию географических информационных систем (ГИС) для создания планов роста и развития городов, и для изменения зонирования территории, реагируя на демографический рост.

Задача автоматического распознавания зданий на спутниковых изображениях сверхвысокого разрешения является важной частью в решении задачи автоматической интерпретации данных, получаемых из систем дистанционного зондирования Земли (ДЗЗ). Для сбора, хранения, анализа и графической визуализации пространственной информации используется ГИС. Ручная оцифровка и ввод пространственных данных в базу данных ГИС имеет недостатки в виде больших временных и материальных затрат, а также в виде высокого риска допущения ошибок, по причине человеческого фактора. Автоматическая система распознавания зданий на спутниковых изображениях позволит не только сократить временные и материальные затраты на обновление базы данных ГИС, а также повысить точность вводимых данных.

В связи с возрастающими возможностями получения спутниковых данных сверхвысокого разрешения усиливается необходимость в разработке методов их автоматического анализа. Вследствие увеличения разрешения спутниковых данных, все большее развитие получает объектно-ориентированный подход к их обработке, одним из основных этапов которого является сегментация изображения, направленная на выделение объектов.

Подходящий метод сегментации выбирается в зависимости от поставленной задачи. Наиболее часто применяемым при решении задачи обнаружения зданий на спутниковых снимках является метод роста регионов [1] [2] [3]. В работах [4] и [5] для повышения качества сегментации используется метод выделения границ совместно с методом роста регионов.

Цель работы состоит в проведении сравнительного анализа различных методов сегментации изображений, рассмотрении операций морфологической обработки и выборе наиболее подходящего подхода к обработке спутниковых изображений, позволяющего повысить адекватность тематической сегментации.

Методы сегментации спутниковых изображений

Объектно-ориентированный подход к анализу спутниковых изображений высокого пространственного разрешения возник в 90-х го-

дах. В рамках данного подхода происходит анализ и классификация изображений на уровне объектов, т.е. групп пикселей, объединенных на основе определенной совокупности критериев. В качестве признаков для классификации могут использоваться статистические, текстурные и геометрические характеристики объектов. На Рис.1 представлена структура объектно-ориентированного подхода к анализу изображений.

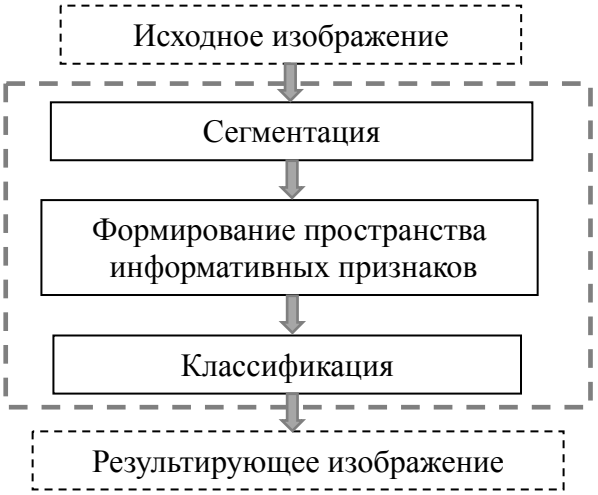


Рис.1 Структура объектно-ориентированного подхода к анализу изображений

Под сегментацией изображения понимается процесс разделения изображения на несколько сегментов. Целью сегментации является выделение сегментов, однородных в определенном заданном смысле. Результатом сегментации является множество сегментов, покрывающих всё изображение, или множество контуров, выделенных на изображении. Объект и фон разделены так, что легко определить число объектов, их местоположение и геометрические характеристики.

Как видно из Рис.1 сегментация является ключевым этапом обработки данных и влияет на эффективность всех дальнейших шагов анализа спутниковых изображений по причине зависимости качества получаемого в результате работы системы распознавания решения от правильно выделенных объектов. Чрезмерная или недостаточная сегментация могут привести к неправильному определению расположения и размеров объектов, что ведет к ошибочным результатам.

К полученным в результате сегментации сегментам предъявляются следующие требования [6]:

1. Области должны быть однородны относительно определенных характеристик;
2. Внутренние части областей не должны содержать большое количество пустот;
3. Смежные области должны существенно отличаться по значению выбранных характеристик;
4. Границы каждого полученного сегмента должны быть четкими.

Проведен сравнительный анализ следующих методов сегментации [7] [8]:

1. метод гистограмм,
2. метод разбиения и слияния регионов,
3. нечеткая кластеризация С-средних,
4. метод роста регионов,
5. метод роста регионов и нечеткая кластеризация С-средних.

В работе использовались спутниковые изображения IKONOS и GeoEye-1. Пространственное разрешение спутниковых данных составляет 0.8 м и 0.5 м соответственно.

Входное изображение представляет собой растровые данные в цветовой модели RGB, преобразуем его в оттенки серого. Каждая точка RGB-изображения воспринимается глазом как более или менее яркая. В образовании этой точки принимают участие все три цветовых канала изображения. Поскольку различные базовые цвета имеют различную воспринимаемую яркость, то будем использовать метод перехода к оттенкам серого, предложенный в [9], который заключается в получении яркости каждой точки по формуле:

$$Y = 0.3*R + 0.59*G + 0.11*B$$

Для проведения сравнительного анализа методов сегментации были использованы критерии, основанные на вычислении меры отличия между результатом сегментации, полученным с помощью алгоритма, и сегментом, построенным экспертом на основе визуального анализа изображения. Оценивается качество сегментации зданий, а не всего спутникового изображения. Необходимо отметить, что присутствие эксперта обуславливает возможность влияния человеческого фактора на результат сравнительного анализа.

Ниже приведены критерии оценки качества сегментации:

1. Показатель чрезмерной сегментации

$$\mathrm{FPR} = \frac{S_1 \cap S_2^c}{S_1 \cup S_2}$$

2. Показатель недостаточной сегментации

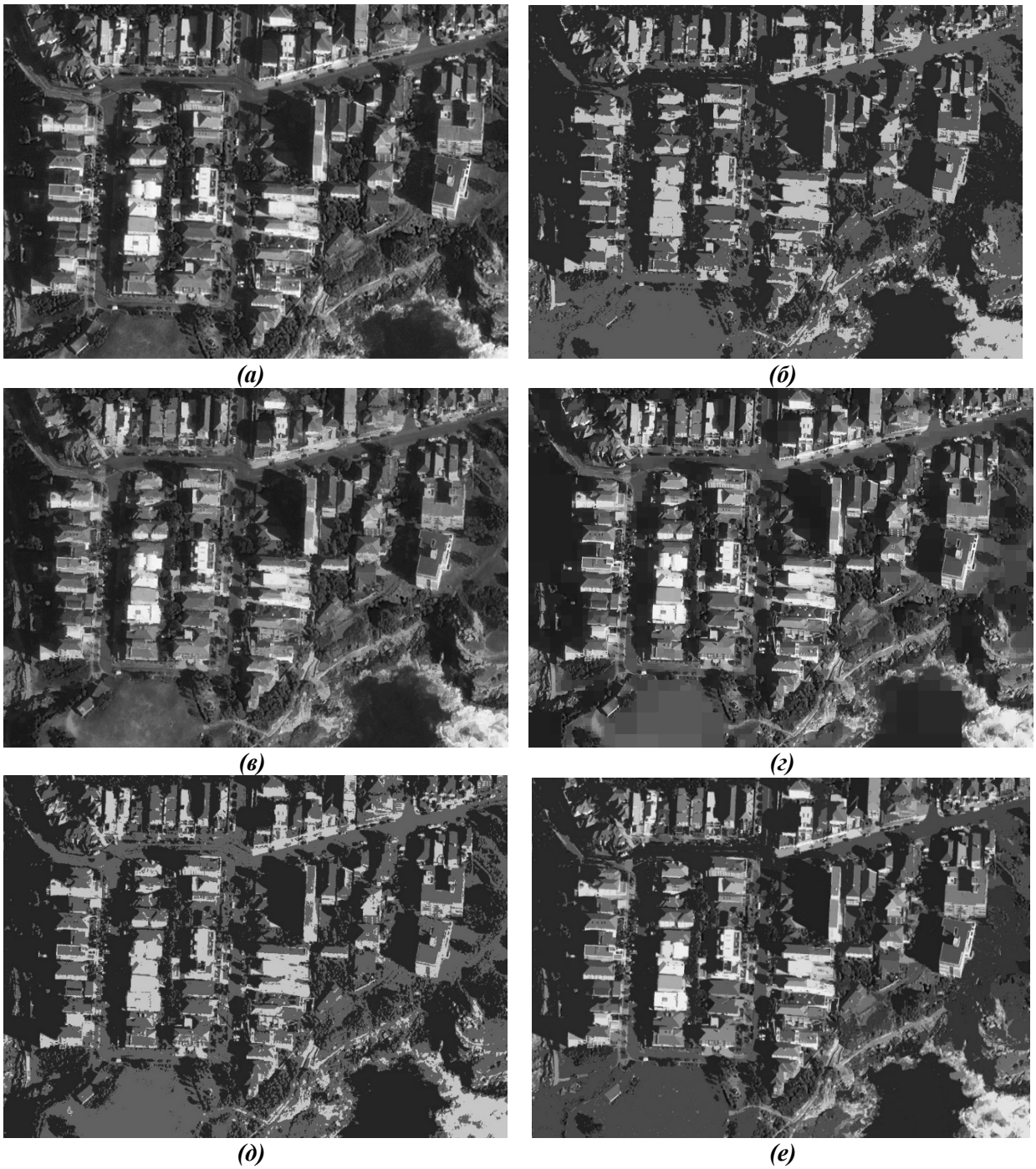
$$\mathrm{FNR} = \frac{S_1^c \cap S_2}{S_1 \cup S_2}$$

3. бщая ошибка сегментации

$$MA = \mathrm{FNR} + \mathrm{FPR}$$

S_1 - результат сегментации, полученный с помощью одного из методов сегментации, S_2 - построенный экспертом сегмент, S_1^c - контур сегмента, полученный с помощью одного из методов сегментации, S_2^c - построенный экспертом контур сегмента.

На Рис. 2 приведены примеры сегментации изображения с помощью каждого из исследуемых методов сегментации. На Рис.3 представлены результаты сравнительного анализа для различных методов сегментации.



(а) ***(б)*** ***(в)*** ***(г)*** ***(д)*** ***(е)***

Рис. 2 Примеры сегментации спутникового изображения IKONOS с пространственным разрешением 0.8 м
(а) – исходное изображение; (б),(в),(г),(д),(е) –сегментированное изображение с помощью различных методов: метод роста регионов + нечеткая кластеризация С-средних, метод гистограмм, метод разбиения и слияния регионов, нечеткая кластеризация С-средних, метод роста регионов

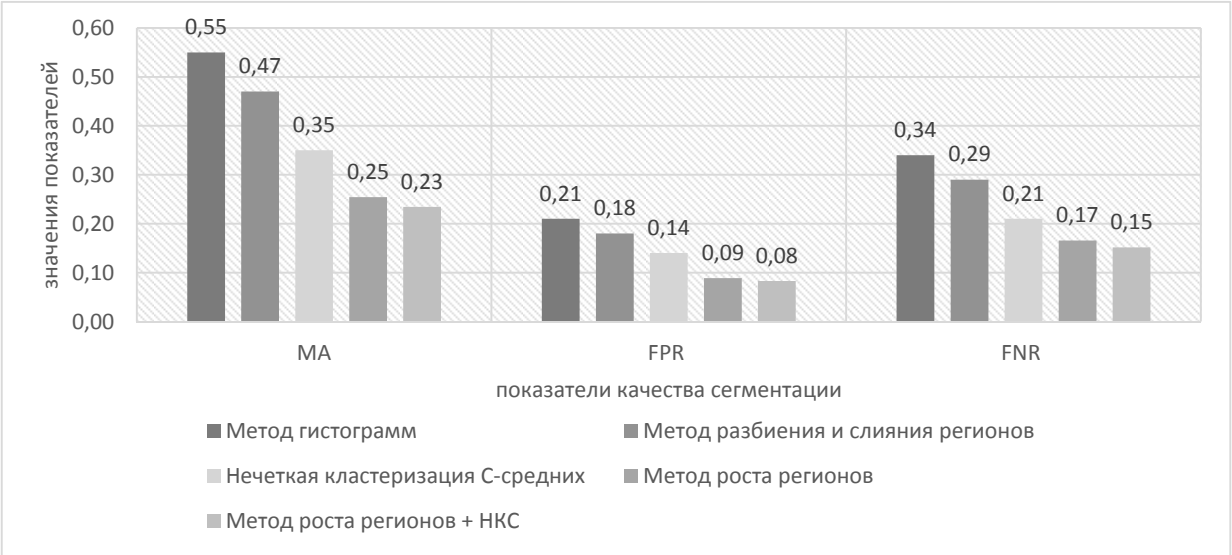


Рис.3 Результаты сравнительного анализа для различных методов сегментации

На основе полученных результатов экспериментальных исследований можно говорить, что наиболее приемлемым при решении поставленной задачи является метод основанный на последовательном применении метода роста регионов и метода нечеткой кластеризации С-средних (общая ошибка сегментации составила 0,23). После него следует метод роста регионов со значением ошибки 0,25. Метод гистограмм, как и метод разбиения и слияния регионов являются не подходящими для сегментации спутниковых изображений с целью обнаружения зданий (общая ошибка сегментации составила 0,55 и 0,47).

В Таблице 1 представлены результаты сравнительного анализа метода роста регионов и предложенного комбинированного метода сегментации (метод роста регионов и нечеткая кластеризация С-средних) относительно территориальных зон различного назначения. В результате экспериментальных исследований получено среднее значение ошибки равное 0,254 при сегментации с помощью метода роста регионов и 0,234 при использовании комбинированного метода сегментации.

Таблица 1. Результаты сравнительного анализа методов сегментации относительно территориальных зон различного назначения.

	MA		FPR		FNR	
Методы сегментации	Роста регионов	Роста регионов + НКС	Роста регионов	Роста регионов + НКС	Роста регионов	Роста регионов + НКС
Урбанизированные территории	0,354	0,307	0,127	0,119	0,227	0,188
Сельская местность	0,237	0,198	0,076	0,064	0,161	0,134
Промышленные зоны	0,221	0,190	0,083	0,075	0,138	0,115
Лесные зоны	0,284	0,242	0,092	0,073	0,192	0,169
Среднее значение ошибки	0,254	0,234	0,089	0,083	0,166	0,152

Морфологическая обработка спутниковых изображений

Для улучшения качества сегментации использовалась морфологическая обработка, состоявшая из последовательного применения двух морфологических операций: размыкание и наращивание [6].

Морфологическая операция размыкание представляет собой последовательное применение операций эрозия и наращивание. В результате сегменты разъединенные узкими

участками пикселей объединяются, контуры объектов становятся более гладкими.

Размыканием бинарного изображения A структурирующим элементом B:

$$A \circ B = \bigcup_{B_x \subseteq A} B_x$$

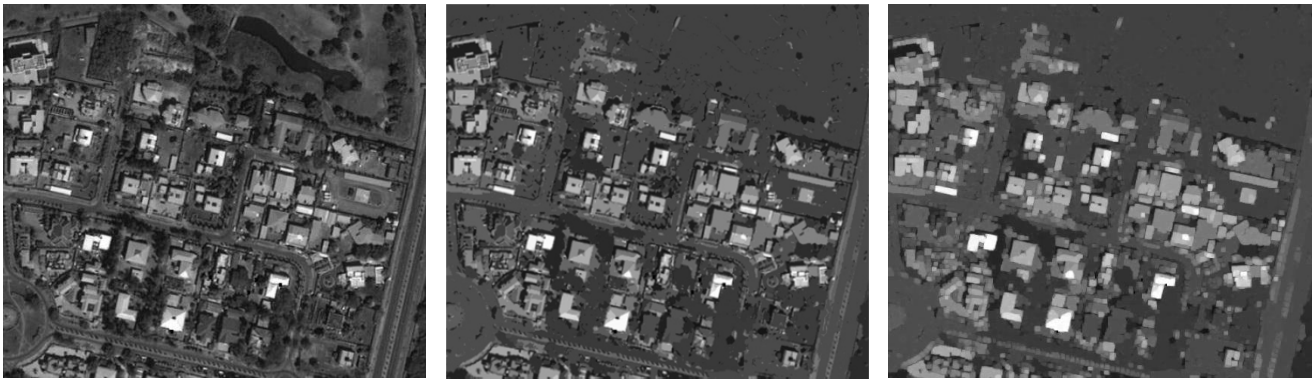
Морфологическая операция наращивание приводит к удалению небольших по площади замкнутых областей фона внутри сегментов.

Наращивание бинарного изображения A структурирующим элементом B:

$$A \oplus B = \bigcup_{b \in B} A_b$$

Результатом применения морфологических операций стало удаление шумов и объединение сегментов, ранее разъединенных узкими участками пикселей.

На Рис. 4 и Рис. 5 показана сегментация спутникового изображения методом роста регионов и методом роста регионов совместно с методом нечеткой кластеризации С-средних с последующей морфологической обработкой.



(а) *(б)* *(в)*

Рис. 4 Сегментация спутникового изображения IKONOS, пространственное разрешение 0.8 м
(а) – исходное изображение; (б) – сегментация методом роста регионов; (в) – морфологическая обработка;

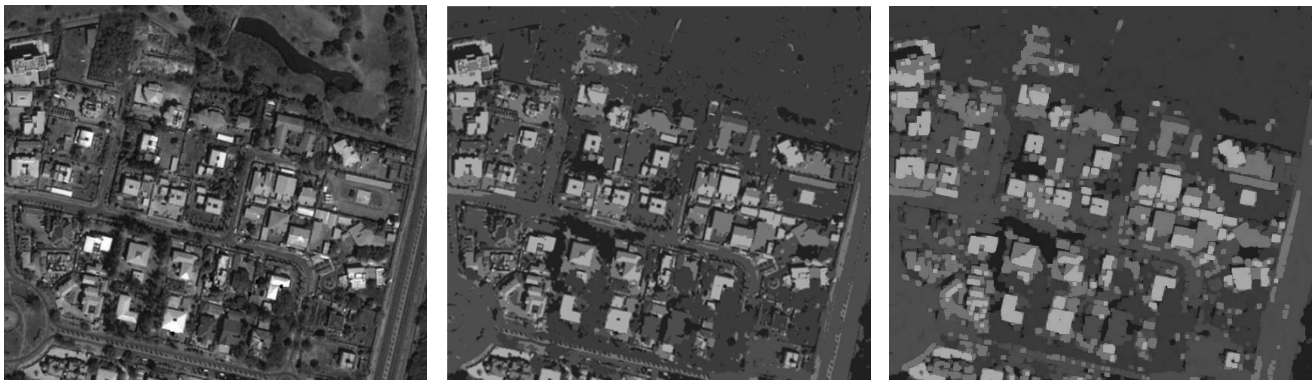


Рис. 5 Сегментация спутникового изображения IKONOS, пространственное разрешение 0.8 м
(а) – исходное изображение; (б) – сегментация методом роста регионов и методом нечеткой кластеризации С-средних; (в) – морфологическая обработка;

Результаты экспериментальных исследований показывают, что подход, основанный на последовательном применении на первом этапе метода роста регионов и метода нечеткой кластеризации С-средних, а на втором этапе морфологических операций является наиболее эффективным при решении задачи обнаружения зданий на спутниковых изображениях.

Выводы

Проведен сравнительный анализ различных методов сегментации изображений применительно к задаче обнаружения зданий на спутниковых изображениях. Для сравнения методов сегментации были использованы критерии, основанные на вычислении оценки близости результатов сегментации, полученных с помо-

щью алгоритма и построенных экспертом на основе визуального анализа.

Путем экспериментальных исследований обнаружено, что метод гистограмм некорректно сегментируют границы объектов и не является приемлемым при решении задачи обнаружения зданий на спутниковых снимках. Общая ошибка сегментации для метода гистограмм составила 0,55. Недостатком метода с использованием гистограмм, является то, что при разделении изображения на однородные области не учитывается пространственное расположение различных областей изображения. Метод роста регионов опирается в первую очередь на пространственные свойства областей, при этом каждый пиксель рассматривается в контексте своих соседей. Проведенный сравнительный анализ показывает, что значения показателей чрезмерной и недостаточной сегментации для метода роста регионов составили 0,09 и 0,17, что значительно меньше, чем значения данных показателей для метода гистограмм 0,21 и 0,34 соответственно. Наилучший результат сегментации был получен с помощью комбинированного метода, заключающегося в последовательном применении метода роста регионов и нечеткой кластеризации С-средних. Применение нечеткой кластеризации позволяет сократить количество регионов и повысить качество сегментации. Общая ошибка сегментации для комбинированного метода составила 0,23.

Рассмотрены операции морфологической обработки. Установлено, что для повышения качества сегментации спутниковых изображений целесообразно проводить морфологическую обработку, которая обеспечивает уменьшение количества анализируемых областей за счет слияния сегментов и удаления несущественных фрагментов с точки зрения рассматриваемой задачи.

Необходимо отметить, что требуются дальнейшие исследования, направленные на усовершенствование методов тематической сегментации, учитывающих пространственные свойства областей и обеспечивающих наилучший компромисс между недостаточной и чрезмерной сегментацией.

Полученные результаты позволяют наметить перспективы использования алгоритмов сегментации при построении автоматических систем обнаружения зданий на спутниковых изображениях.

Список литературы

1. F. Rottensteiner, J. Trinder, S. Clode, K. Kubik. Detecting Buildings and Roof Segments by Combining LIDAR Data and Multispectral Images. - Massey University, New Zealand, 2003. – pp. 60-65.
2. S. M¨uller and D. W. Zaum, Robust building detection from aerial images. – IAPRS, Vol. XXXVI, Part 3/W24, Vienna, Austria, 2005. – pp. 143-148.
3. Adaptive Building Edge Detection by Combining LIDAR Data and Aerial Images. - The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science, vol. XXXVI, Part B1, 2008. – pp. 197-202.
4. S. Tanathong. K. T. Rudahl. S. E. Goldin. Object Oriented Change Detection of Buildings After a Disaster. – ASPRS 2009 Annual Conference Baltimore, 2009. – pp. 1-11.
5. T. Pavlidis and Y. Liow. Integrating Region Growing and Edge Detection. – IEEE, vol.12, №3, 1990. - pp. 208 – 214.
6. Linda G. Shapiro and George C. Stockman. Computer Vision. - New Jersey, Prentice-Hall, 2001. - pp 279-325.
7. Y.Wang. Tutorial: Image Segmentation. - National Taiwan University, Taipei. - pp 1-36.
8. Тараскина А. С. Нечеткая кластеризация по модифицированному методу с-средних. – Новосибирск: Проблемы интеллектуализации и качества систем информатики, 2006. – с.217-228.
9. J.S.Chitode, V.S.Bagad. Communication Systems. – Pune, Technical Publications, 2007. – p.410.

УДК 004.932.75'1

НАДЕРАН Э.,
ЗАЙЧЕНКО Ю.П.

ВЫДЕЛЕНИЕ ИНФОРМАТИВНЫХ ПРИЗНАКОВ РУКОПИСНЫХ МАТЕМАТИЧЕСКИХ СИМВОЛОВ

В статье рассмотрен разработанный подход к выделению информативных признаков рукописных символов. Проведен сравнительный анализ предложенного подхода с подходом на основе доминантных точек. Проведены экспериментальные исследования различных методов обучения нечеткого классификатора NEFCLASS и оценена их эффективность применительно к задаче распознавания рукописных символов.

This article discusses a new approach to discovering informative patterns of handwritten symbols. A comparative analysis of the proposed approach with the approach based on the dominant points. An experimental comparative study of three methods of training NEFCLASS Neuro-Fuzzy Classifier and evaluating their effectiveness applied to the problem of recognition of handwritten symbols.

Введение

Активное развитие планшетных персональных компьютеров, наблюдаемое сегодня, приводит к необходимости ввода данных в ЭВМ без использования клавиатуры. Математические выражения составляют основную часть в большинстве научных и технических дисциплин, однако ввод математических выражений в ПК с помощью традиционных устройств ввода, таких как клавиатура и мышь, является достаточно сложным. Взаимодействие пользователя с ПК с помощью сенсорной функциональности является наиболее естественной, простой и быстрой альтернативой для ввода математических выражений в ЭВМ.

Высокая вариативность в написании одних и тех же символов, и сложность определения структуры математических выражений затрудняют процесс распознавания. Центральной задачей проблемы онлайн распознавания рукописных математических выражений является создание эффективного метода классификации рукописных математических символов, цифр и букв. Основная сложность заключается в существовании большого количества вариаций написания одного и того же символа.

Сегодня наиболее распространенными подходами для классификации рукописных символов являются: алгоритмы сопоставления с образцом, скрытые марковские модели, алгоритмы на основе искусственных нейронных сетей. В работе [1] предложен редактор ввода рукописных математических выражений, позволяющий пользователю вводить цифры и символы в любом порядке. Для распознавания символов, используется алгоритм соответствия образцу, что накладывает определенные ограничения на работу с программой, в частности возникают трудности при работе с различными почерками. В [2] предложен подход, основанный на скрытых Марковских моделях. Предполагается, что пользователь всегда пишет выражение в определенном порядке. Например, при написании дроби, сначала должен быть написан числитель, затем дробная линия и знаменатель. Такое требование может быть легко нарушено в реальном применении из-за высокой изменчивости стиля написания у различных пользователей.

В последнее время начинает активно использоваться аппарат искусственных нейронных сетей с целью решения различных задач классификации. Преимуществом искусственных нейронных сетей является их способность к обобщению полученной информации, таким образом обученная на ограниченном множестве выборочных данных нейронная сеть может вернуть верный результат применительно к данным, которые не участвовали в процессе обучения. Поскольку решение об идентификации рукописных символов принимается системой распознавания в условиях неполной и неточной информации, можно считать подходящим разработку метода распознавания на основе нечеткого нейронного классификатора, сочетающего в себе преимущества нейронных сетей и нечетких систем логического вывода.

Целью данной статьи является рассмотрение разработанного подхода к выделению информативных признаков на основе аппроксимиру-

ющих точек и сравнительный анализ эффективности применения различных подходов к выделению информативных признаков для улучшения качества распознавания рукописных математических символов.

Выделение информативных признаков рукописных символов

В настоящее время одним из наиболее широко применяемых подходов к выделению информативных признаков является подход, основанный на нахождении доминантных точек [3][4][5]. Доминантные точки определяются по следующему правилу: начальная и конечная точки отрезка, точки локальных экстремумов, и точки, лежащие между точками двух типов, описанных выше. Каждый отрезок состоит из точки касания пера, точки отрыва пера и всех точек лежащих между ними. На Рис.1 представлен пример нахождения доминантных точек.

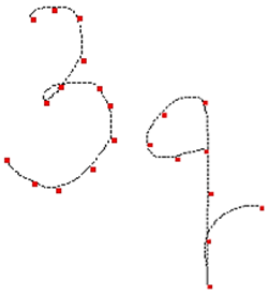


Рис.1 Пример нахождения доминантных точек

В статье предлагается подход к выделению информативных признаков, основанный на использовании точек ломаной аппроксимирующей кривую символа, далее в статье будем называть такие точки «аппроксимирующие точки». Для нахождения аппроксимирующих точек использован алгоритм Рамера - Дугласа – Пекера [6] [7]. Алгоритм применяется для обработки векторной графики и картографической генерализации. Авторами предлагается применение данного алгоритма к аппроксимации кривой с целью выделения информативных признаков. Экспериментальным путем был подобран коэффициент точности сглаживания наиболее оптимальный по соотношению между количеством оставляемых алгоритмом точек и максимальным сохранением очертания символа. На Рис. 2 представлен пример нахождения аппроксимирующих точек.

В качестве классификатора используется нечеткая нейронная сеть NEFCLASS [8].

Рис.2 Пример нахождения аппроксимирующих точек

Общий набор символов содержит строчные буквы латинского алфавита, наиболее часто встречаемые в формулах строчные буквы греческого алфавита, цифры и специальные математические символы (Рис. 3). Отобранные символы позволяют писать тригонометрические и логарифмические функции, интегралы, арифметические корни различной степени и др. Для классификации с помощью нечеткой нейронной сети NEFCLASS рассматриваются символы, написанные без отрыва пера. Такие символы содержат один отрезок. Символы, которые могут быть получены из комбинации символов первой группы и состоят из нескольких отрезков, преобразуются в один символ в процессе структурного анализа на этапе реконструкции символов.

Сравнительный анализ подходов к выделению информативных признаков на основе доминантных и аппроксимирующих точек

Рассмотрим набор информативных признаков [9], значения которых характеризуют рукописные символы математического выражения:

1. Максимальный угол

Для расчета данного признака найти углы β_j, лежащие между сегментами символа, разделенными аппроксимирующими точками, и осью x:

$$\beta_j = \arccos(\vec{x} \cdot \frac{\vec{v_j}}{\| \vec{v_j} \|}) \qquad (1)$$

где $\vec{x} = (1,0)$, а вектор v_j определяется как:

$$\vec{v}_j = ap_i - ap_{i-1},$$

где $2 \le i \le k(ap), 1 \le j \le k(ap) - 1$

$k(ap)$ - общее количество аппроксимирующих точек.

Признак максимальный угол рассчитать по формуле:

Цифры	1	2	3	4	5	6	7	8	9	0	
Латинские буквы	a	b	c	d	e	g	h	k	m	n	
	o	p	q	r	s	u	v	w	y	z	
Скобки	[	]	(	)	{	}					
Греческие буквы	α	β	γ	δ	ε	λ	μ	ξ	ω	ф	∂	Ω
Другие символы	∞	Δ	∇	∑	∏	√	∫	∧	∨	.	,	
	+	-	/	\	|	~	<	>				
Вспомогательные штрихи	ɹ	ɭ	∫								

На этапе распознавания рассматриваются символы, написанные без отрыва пера

±	∀	!	f
i	j	t	k
∅	∉	∈	÷
x	+	=	≠
≈	≥	≤	%
:	θ	π	z
→	7		

На этапе структурного анализа реконструируются символы, состоящие из нескольких отрезков

Рис.3 Рассматриваемый набор символов для написания рукописных математических выражений

2. Среднее значение углов

Используя формулу (1) найти углы β_j. Вычислить разность между расположенными последовательно углами β_j как

$$\varphi_n = \beta_j - \beta_{j-1}, \qquad (2)$$

где $2 \le j \le k(ap)$,

Тогда признак среднее значение углов можно рассчитать по формуле:

$$a_{\text{сред}} = \frac{1}{k(ap)-2} \sum_{n=1}^{k(ap)-2} \varphi_n$$

3. Прямолинейность

Рассчитать значения углов φ_n по формуле (2). Значение признака прямолинейность определить как отношение количества всех $\varphi_n \le |\varepsilon|$ к общему количеству φ_n.

4. Топология

Общее количество раз, где последовательность, состоящая из φ_n изменяет знак с плюса на минус и с минуса на плюс.

5. Гистограмма углов

Разобьем координатную плоскость на восемь секций по 45 градусов каждая. Каждый угол β_j найденный по формуле (1) лежит в одной из восьми полученных секций. Для нахождения гистограммы углов i-той секции разделить количество углов, попавших в эту секцию на общее количество углов:

$$ГУ_i = \frac{кол-во\ углов\ секции\ i}{общее\ кол-во\ углов}, \quad i = 1,\ldots,8$$

6. Площадь сегментов

Для расчета данного признака найти вектора:

$$\vec{u_i} = ap_{i+1} - ap_1,$$
$$\vec{v_i} = ap_{i+2} - ap_1,$$
$$\vec{a_i} = ap_{i+2} - ap_{i+1},$$

где $1 \le i \le k(ap) - 2$

Площадь сегментов вычислить по формуле:

$$S = \sum_{i=1}^{k(ap)-2} \frac{(\vec{u_i} \times \vec{v_i}) \cdot sgn(\vec{u_i} \times \vec{v_i})}{|\vec{a_i}|}$$

Для проведения сравнительного анализа подходов на основе доминантных и аппроксимирующих точек был проведен ряд экспериментов нечеткой нейронной сети NEFCLASS с различными алгоритмами обучения параметров функций принадлежности. С помощью рукописного ввода была создана выборка для обучения, состоящая из 280 рукописных символов. Для каждого символа были получены

значения информативных признаков каждым из исследуемых подходов.

Входам нейронной сети NEFCLASS соответствуют тринадцать информативных признаков, выходам – 70 символов, подлежащих классификации.

Рассмотрим способ формирования базы правил. Каждое правило имеет следующий вид:

Если x_1 является μ_1, x_2 является μ_2, …, x_{13} является μ_{13},

то образец $(x_1,\ldots, x_{13})$ принадлежит классу i;

где $1 \le i \le 70$.

Для обучения параметров функций принадлежности были выбраны три различных алгоритма обучения [10]:

– градиентный алгоритм,
– алгоритм сопряженных градиентов,
– генетический алгоритм.

Для обучения используется гауссова функция принадлежности:

$$\mu(x) = exp\left[-\left(\frac{x-c}{\sigma}\right)^2\right]$$

На Рис.4 показана зависимость погрешности от количества пройденных итераций по каждому методу обучения для нейронной сети NEFCLASS со значениями информативных признаков, полученными с помощью подхода на основе аппроксимирующих точек. По результатам экспериментов можно отметить, что алгоритм сопряженных градиентов сходится значительно быстрее двух других алгоритмов обучения к ошибке, которая ниже установленного порога. Генетический алгоритм сходится намного медленнее и характеризуется этапным улучшением погрешности. Установленный порог был равен 0.07.

Тестирование обученной нейронной сети проведено на проверочной выборке, состоящей из 140 рукописных символов. На Рис.5 приведены результаты классификации, значения информативных признаков были получены с помощью двух исследуемых подходов.

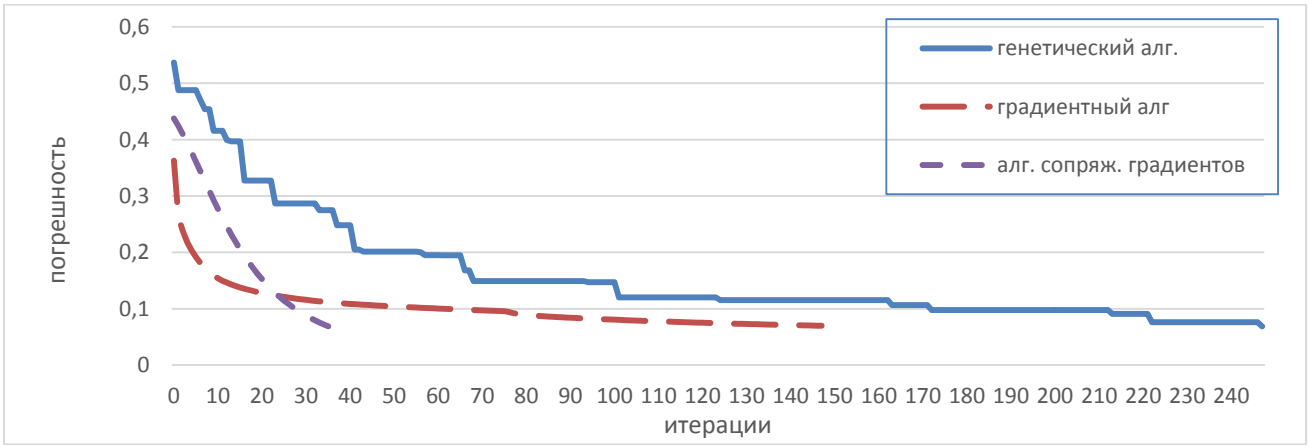


Рис.4 Графическое представление зависимости погрешности от пройденных итераций для различных алгоритмов обучения

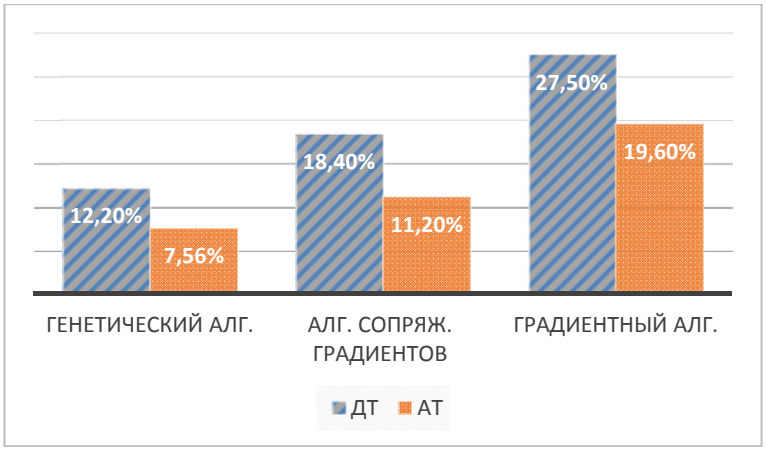


Рис.5 Процент ошибочно классифицированных образцов

Учитывая полученные результаты, можно сказать, что нечеткий классификатор NEFCLASS обученный с помощью генетического алгоритма и со значениями информативных признаков, полученными с помощью аппроксимирующих точек, имеет наименьший показатель по количеству ошибочно классифицированных символов (7,56%). Далее следует алгоритм сопряженных градиентов обучения параметров функций принадлежности нейронной сети (11,20%). Наихудший показатель имеет градиентный алгоритм обучения со значениями информативных признаков, полученными с помощью доминантных точек, 27,50% ошибочно классифицированных символов.

Выводы

В статье был рассмотрен подход к выделению информативных признаков рукописных символов, основанный на использовании точек ломаной аппроксимирующей кривую символа, который позволяет с минимальным количеством точек максимально сохранить очертания символа

Проведен сравнительный анализ подхода на основе аппроксимирующих точек и существующего подхода на основе доминантных точек. Построен нечеткий классификатор NEFCLASS с тремя алгоритмами обучения: генетическим, градиентным и алгоритмом сопряженных градиентов. По результатам экспериментов наилучшее качество распознавания было получено при обучении нейронной сети с помощью генетического алгоритма обучения. Процент ошибочно классифицированных образцов составил 7,56% при использовании подхода на основе аппроксимирующих точек и 12,20% при использовании подхода на основе доминантных точек.

Установлено, что для повышения качества распознавания рукописных символов целесообразно применять генетический алгоритм обучения параметров функций принадлежности на этапе первичного обучения системы и алгоритм сопряженных градиентов на этапе дообучения системы для улучшения временных показателей.

Список литературы

1. Nakayama Y. A prototype pen-input mathematical formula editor. – EDMEDIA, 1993. - pp. 400-407.
2. Kosmala A. Rigoll G. On-line handwritten formula recognition using statistical methods. - International Conference on Pattern Recognition, 1998. - pp. 1306-1308.
3. X. Li , D.Yeung. On-line handwritten alphanumeric character recognition using dominant points in strokes. – Pattern Recognition, vol. 30, no. 1, 1997. - pp. 31-44.
4. J. J. LaViola, R. C. Zeleznik. A Practical Approach for Writer-Dependent Symbol Recognition Using a Writer-Independent Symbol Recognizer. - IEEE Transactions on pattern analysis and machine intelligence, vol. 29, №11, 2007. - pp. 1917-1926.
5. H. Tirandaz, A. Nasrabadi, J. Haddadnia. Curve Matching and Character Recognition by Using B-Spline Curves. - International Journal of Engineering and Technology, Vol.3, №2, 2011. – pp.183-186.
6. Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. - Computer Graphics and Image Processing, vol. 1, 1972. – pp.244- 256.
7. David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. - The Canadian Cartographer, vol. 10, №2, 1973. – pp. 112-122.
8. Detlef Nauck, Rudolf Kruse. NEFCLASS – A neuro-fuzzy approach for the classification of data. – Applied Computing, 1995. – pp. 1-5.
9. J. J. LaViola, R. C. Zeleznik. A Practical Approach for Writer-Dependent Symbol Recognition Using a Writer-Independent Symbol Recognizer. - IEEE Transactions on pattern analysis and machine intelligence, Vol. 29, NO. 11, 2007. - pp. 1917-1926.
10. Зайченко Ю.П. Нечёткие модели и методы в интеллектуальных системах. Учебное пособие для студентов высших учебных заведений. – К.: «Издательский Дом «Слово»», 2008. – с. 344.

УДК 683.519

ДЕМЧИНСКИЙ В.В.,
ДОРОГОЙ Я.Ю.,
ДОРОШЕНКО К.С.

АНАЛИЗ И ПРАКТИКА ВНЕДРЕНИЯ МЕХАНИЗМОВ КАЧЕСТВА ОБСЛУЖИВАНИЯ

В статье рассмотрены вопросы, связанные с анализом и практикой внедрений механизмов качества обслуживания.

The questions related to the analysis and the introduction of quality of service was described in this article.

Введение в проблему

Важность качества предоставляемых услуг для успешного ведения бизнеса очевидна. Рассмотрим одну из сторон этого вопроса в сфере телекоммуникационных услуг - обеспечение требуемых характеристик сервиса для определенных классов трафика. Будем называть трафиками потоки данных по тем или иным признакам выделяемые из совокупного потока данных. Особенность современной сетевой инфраструктуры состоит в совместной передаче трафиков различной природы (видео, голос, данные) по единой сети (конвергенция, TriplePlay). Для каждого типа трафика можно обозначить типичные характеристики и требования, обуславливаемые особенностями приложений, генерирующих данные трафики. Определим понятие качество обслуживания (QoS, Quality-of-Service) как способность сетевых средств обеспечивать требуемые характеристики сервиса для определенных классов трафика. Интересующие характеристики передаваемого трафика - задержка, вариация задержки и потери пакетов.

Основное назначение технологий качества обслуживания состоит в обеспечении приоритетов, необходимой полосы пропускания, контролируемой задержки и вариации задержки, уменьшении процента потерь пакетов при передаче, а также в обеспечении приоритетности обслуживания некоторых потоков с возможностью одновременной передачи других потоков. Зачастую (из-за мультиплексирования трафика или из-за превышения ПС (пропускной способности) входного канала над ПС выходного) сетевые устройства не успевают передавать весь поступаемый трафик, что вызывает накопление пакетов в очередях и сброс пакетов при исчерпании доступной памяти.

Под перегрузкой будем понимать такое состояние сети, при котором основные показатели качества обслуживания заметно ухудшаются. В технологиях QoS применяются 2 стратегии: управление перегрузками и предотвращение перегрузок. Управление перегрузками состоит в управлении очередями, а предотвращение перегрузки осуществляется механизмами контроля за установлением соединений, формирования трафика и управления скоростью передачи.

Цель работы

Целью работы является исследование параметров качества предоставления услуг и их зависимости от применения того или иного типа очередей.

Архитектура QoS

Архитектура QoS [1] включает механизмы классификации, маркировки и формирования (профилирования) трафика, управления очередями и сигнализации QoS (метки DSCP, распространение правил QoS по протоколу BGP, резервирование ресурсов RSVP). В определенном смысле, технология качества обслуживания №1 - пропускная спообность, т.е. увеличение ПС безусловно улучшает характеристики QoS. Однако запросы сетевых приложений эволюционно растут и практически при любой доступной ПС несложно будет ее утилизировать. Т.е. само по себе наращивание ПС (без внедрения других механизмов QoS) будет только временным решением и способно даже усугубить ситуацию. “Возможно купить пропускную способность, но нельзя купить задержку”.

В зависимости от требований приложений и условий соглашения трафик подвергается дифференциации по нескольким уровням приоритета или классам обслуживания. Как правило, механизмы классификации, маркировки и формирования трафика применяются

как можно ближе к границе сети. Классификация трафика на границе сети позволяет освободить ядро (магистраль) сети от этой задачи. Пакеты могут быть классифицированы либо устройством на входе в сеть, либо же породившим трафик приложением. Способы классификации данных можно разделить на явные (метки, заданные в пакетах отправителем) и неявные (выбираемые сетевыми устройствами исходя из заданной политики). Пакеты могут быть классифицированы по любым параметрам заголовков (в общем случае - по спискам контроля доступа) или используя механизм распознавания сетевых приложений.

Как правило, качество обслуживания необходимо обеспечивать как на границе кампусной сети, так и на магистральных соединениях [2]. Все механизмы качества обслуживания должны работать согласованно, поскольку несогласованная работа одного единственного узла сети может нивелировать весь результат. В свете этого полезно исследование поведения трафика при совместной работе маршрутизаторов с поддержкой QoS и без таковой (единственная очередь FIFO).

Заявляемые характеристики обслуживания называют Соглашением об уровне сервиса (Service Level Agreement, SLA) Обычно соглашение об уровне сервиса включает параметры пропускной способности, задержки, потерь пакетов, время реакции/восстановления и условия обслуживания. В целом, требования необходимого уровня QoS определяются решаемыми задачами и используемыми приложениями. При планировании внедрения QoS [3] следует учитывать особенности каждого типа трафика, в первую очередь, трафика реального времени. Интерактивный трафик наиболее чувствителен к параметрам задержки и вариации задержки. Обычно для комфортной работы интерактивных приложений передачи голоса или видео полная задержка не должна превышать 150 мс, а вариация задержки - до 10..30 мс [4]. Для потоковой передачи в реальном времени требования менее жесткие, соответственно 1 с и 100 мс. Суммарная (коммулятивная) задержка передачи состоит из задержек сериализации, прохождения сигнала по среде передачи и задержки ожидания в очередях. Вариация задержки определяется размерами пакетов других приложений. Рассмотрим характеристики основных механизмов управления очередью [1,2]. Аппаратно практически все механизмы управления очередями работают между буфером интерфейса и очередью на отправку (передающим кольцом интерфейса).

Общая очередь FIFO. Пакеты всех потоков рассматриваются как равноправные и обслуживаются по мере их поступления. По сути, механизм FIFO не предполагает никакого активного управления трафиком. Быстрота обработки пакетов в очередях зависит только от очередности их поступления и скорости обработки.

Взвешенная справедливая очередь (Weighted Fair Queue, WFQ). Весь трафик одного уровня приоритета попадает в классовую очередь, в которой все потоки обрабатываются равномерно и получают приблизительно равную задержку. Предполагается, что потоки с меньшим объемом данных будут иметь приоритет над потоками с большим объемом данных. При увеличении количества потоков доля полосы каждого потока будет уменьшаться. Считается, что планировщик WFQ улучшает работу протокола TCP (медленый старт и контроль за перегрузкой). Это приводит к более равномерной загрузке каналов и стабильному времени ответа TCP приложений. Существуют механизмы взвешенной справедливой очереди, основанные на потоках или на классах. В первом случае сетевые устройства автоматически распределяют трафик на потоки и обеспечивают равномерное распределение ПС между ними. Во втором случае классификация трафика является настраиваемой. При большом количестве обрабатываемых потоков механизм WFQ увеличивает нагрузку на процессор сетевого устройства, поэтому его применение на высокоскоростных интерфейсах ограничено. Взвешенная справедливая очередь позволяет динамически разделить полосу пропускания между потоками согласно весу каждого потока, определяющего порядок обработки пакетов в очереди, и дает возможность снижать задержки потоков трафика.

Взвешенная справедливая очередь на базе классов (Class-Based Weighted Fair Queue, CBWFQ) позволяет обеспечивать минимальную гарантированную полосу пропускания и предотвращает захват полосы пропускания низкоприоритетными потоками при передаче высокоприоритетных данных. Если один из классов не полностью использует заданную долю ПС, то она равномерно распределяется между другими потоками. Аппаратно взвешенная справедливая очередь реализуется механизмом очередности с весами и с учетом

длины пакетов. По сути, для всех технологий QoS должен выполняться закон сохранения задежки.

Приоритетная очередь (Priority Queue, PQ). Механизм PQ обеспечивает безусловный приоритет трафика. Реализуется 4 очереди со строгим приоритетом. Приоритетные очереди целесообразно применять лишь в тех случаях, когда трафик имеет иерархическую структуру и трафики с более высоким приоритетом не должны задерживаться нижними трафиками. Недостаток приоритетной очереди – пагубное воздействие на неприоритетные трафики и непредсказуемость их времени ожидания. Потоки данных, требующие большой ПС, не должны попадать в высокоприоритетную очередь, а непродуманная настройка приоритетов может привести к блокированию низкоприоритетных трафиков.

Настраиваемая очередь (Custom Queue, CQ). Планировщик CQ делит ПС между потоками пропорционально их весу. Очереди обслуживаются циклически, из каждой очереди берется число байт пропорционально весу очереди. ПС канала в результате динамически распределяется между очередями. Выделяемая очереди полоса пропускания задается неявно посредством счетчика байт и длины очереди. Настраиваемые очереди используются в тех случаях, когда необходимо гарантировать определенные минимальные характеристики для каждого класса трафика (неявно резервировать полосу пропускания) и не дают возникнуть ситуации, когда пакет будет находиться в очереди в течение неопределенного времени. Настраиваемые очереди позволяют более равномерно обслуживать различные потоки и, по сравнению с приоритетными очередями, не блокируют низкоприоритетный трафик. Аппаратно настраиваемые очереди реализуются с помощью механизма весов с заимствованием. Обычно параметры настраиваемых очередей задаются статически и не могут автоматически адаптироваться к изменению ситуации в сети.

Очередь с малой задержкой (Low Latency Queue, LLQ) - комбинация приоритетной очереди и классовой взвешенной справедливой очереди. В очереди с малой задержкой для определенного класса резервируется минимальная полоса пропускания, а в случае доступности дополнительных ресурсов класс способен их утилизировать. Если класс не использует гарантируемую полосу - ее могут использовать другие потоки.

Формирование трафика - процедура манипулирования данными для повышения качества передачи. Средства выполнения правил (policing) сбрасывают пакеты, выходящие за пределы оговоренной скорости, а средства профилирования пытаются буферизировать и сгладить выбросы трафика, и таким образом создать равномерный поток. Механизм профилирования трафиков требует больше ресурсов и применяется, как правило, на пограничных клиентских устройствах, в то время как механизм выполнения правил – между сетями.

Эксперименты и результаты

Рассмотрим примеры поведения трафика при использовании различных механизмов QoS. Смоделируем следующую топологию: каналы Ethernet с ПС 10 Mbit/s, последовательные каналы - 800 Kbit/s. Для наглядности сравнения относительных характеристик трафика возьмем 4 потока с равной интенсивностью. Средняя скорость каждого потока 44 пакета/секунду, длина пакетов равномерно распределена от 200 до 1000 байт. Интенсивность трафика подобрана таким образом, чтобы наиболее узкий канал в топологии (800 Kbit/s) постоянно находился в состоянии насыщения (минимально возможные потери). При моделировании использовался генератор трафика TGN [5]. Заданный поток трафика передается от R4 через R1, R2, R3, R5 и поступает на другой порт R4, где и происходит сбор его статистик (рис. 1).

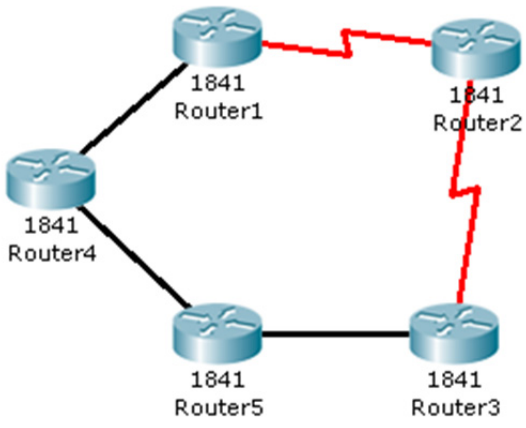


Рис.1. Топология сети

Будем изменять механизм управления очередью на выходном интерфейсе маршрутизатора R1. На остальных маршрутизаторах - очереди FIFO. Эксперименты показали, что повторное применение механизмов QoS к сформированному трафику с заданными характеристиками не улучшает его характеристики. Понаблюдаем за метриками трафика в

зависимости от применяемых на маршрутизаторе R1 механизмах QoS. Интерес представляет относительное изменение метрик при разных сценариях моделирования. Базовый сценарий (сценарий 1) - отсутствие управления трафиком на R1 (общая очередь FIFO). Сценарий 2 - использование WFQ очереди. Сценарий 3 – использование CBWFQ очереди, веса всех 4-х классов равны. Сценарии 4 и 5 – CBWFQ очередь, в которой веса 1-го класса увеличены, соответственно, до 0.30 и 0.35 (за счет уменьшения веса 4-го класса трафика). Сценарий 6 - очередь LLQ, веса всех классов равны, 1-й трафик приоритетный.

Результаты моделирования (табл. 1) свидетельствуют, что за счет организации очередей в планировщике WFQ (сценарий 2) немного увеличилась задержка, но уменьшился разброс (вариация) задержки - трафик стал более предсказуемым.

Табл. 1. Результаты моделирования

Сценарий	1		2		3		4		5		6	
Трафик	avg-delay	stdev-delay	avg-delay	stdev-delay	avg-delay	stdev-delay	avg-delay	stdev-delay	avg-delay	stdev-delay	avg-delay	stdev-delay
1	44,58	8,70	39,64	19,67	54,53	11,54	40,49	10,05	14,28	6,93	2,13	0,86
2	47,01	8,81	40,03	19,55	54,15	10,73	54,12	11,09	53,75	10,84	63,71	14,35
3	45,74	8,86	39,29	19,69	55,01	11,46	54,17	10,11	53,40	11,00	64,97	15,93
4	46,31	7,38	39,68	19,69	54,64	10,26	69,48	12,94	90,37	17,84	63,45	15,43

Сценарии 3, 4 и 5 показывают - увеличение доли потока можно рассматривать как увеличение относительного приоритета, что влечет уменьшение задержки данного потока за счет увеличения задержки "класса по-умолчанию". Сценарий 6 - использование приоритетной очереди для 1-го класса трафика значительно уменьшает его задержку (практически до минимально возможного уровня - постоянной составляющей задержки) и перераспределяет задержки между остальными классами трафика ("закон сохранения задержки").

Выводы

Таким образом, существующие механизмы управления очередями способны обеспечивать равномерное обслуживание различных потоков трафика и приоритет их обслуживания. Относительный приоритет трафика может быть задан как вес потока или посредством реализации отдельной очереди потока.

При абсолютной приоритетности трафика обязательно следует учитывать его максимально возможную интенсивность т.к. трафик с абсолютным приоритетом способен полностью подавить остальные трафики.

Предпочтительнее использование LLQ очереди, где для приоритетного трафика ограничена максимальная доля используемой ширины канала. В этом случае также следует оценивать максимально возможную интенсивность приоритетного трафика, чтобы обеспечить ему достаточную полосу пропускания. Следовательно, для максимального использования технологий QoS целесообразно формирование характеристик трафика на входе в сеть и применение скоростных механизмов управления очередями на магистральных соединениях.

Список литературы

1. Кучерявый Е.А. Управление трафиком и качество обслуживания в сети Интернет. - СПб.: Наука и Техника, 2004. - 336 с.: ил.
2. Вегешна Шринивас. Качество обслуживания в сетях IP. :Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 368 с.
3. Степанов С.Н. Основы телетрафика мультисервисных сетей. - М.: Эко-Трендз, 2010. - 392 с.: ил.
4. Девидсон Джонатан и др. Основы передачи голосовых данных по сетям IP. 2-е издание. :Пер. с англ. – М.: Издательский дом «Вильямс», 2007. – 397 с.
5. TGN Traffic Generation Using Cisco Pagent IOS. – [Електронний ресурс]. – URL: http://www.ciscoconsole.com/nms/traffic-generation-using-cisco-pagent-ios.html/

УДК 519.237.8:681.518.2

РОЛИК А.И.,
ЛАНГЕ Т.И,
ПОКОТИЛО А.А.,
МАРТ Б.А.,
ЯСОЧКА М.В.

МЕТОД ПОТЕНЦИАЛЬНЫХ ФУНКЦИЙ В ЗАДАЧАХ ОЦЕНКИ УРОВНЯ ТЕЛЕКОММУНИКАЦИОННЫХ СЕРВИСОВ

Предложен метод оценки уровня телекоммуникационных сервисов по результатам анализа значений совокупности метрик, измеренных в узлах телекоммуникационной сети. Оценивание уровня сервиса производится с применением методов теории распознавания образов, в частности, метода потенциальных функций. Проанализированы особенности применения метода потенциальных функций при решении задач оценки уровня сервиса IPTV.

The method of estimating the level of the telecommunications services using a set of metrics, measured at the nodes of the telecommunication network is proposed. The service level estimation is performed using the pattern recognition methods and particularly the potential functions method. The features of the potential functions method application to the IPTV service level estimation problems are analyzed.

Введение

В настоящее время на рынке телекоммуникационных услуг происходит острая конкурентная борьба между операторами телекоммуникационных сервисов (ОТС), предлагающих клиентам сервисы высокоскоростного доступа в интернет, VoIP, IPTV и пр.

Ухудшение качества сервисов, длительное время устранения неисправностей вызывают недовольство абонентов, негативно сказываются на имидже ОТС и приводят к сокращению доходов операторов из-за оттока абонентов. Поэтому для сохранения клиентов и поддержания конкурентоспособности ОТС должны стабильно предоставлять сервисы на высоком уровне с минимальными затратами ресурсов. Для этого необходимо непрерывно получать сведения о текущем уровне предоставляемых сервисов, предпринимать управляющие воздействия по поддержанию согласованного уровня сервиса, выявлять тенденции ухудшения качества и своевременно осуществлять восстановительные мероприятия до появления жалоб пользователей. Поэтому данная работа, в которой предлагается применять распознавание образов на основе метода потенциальных функций для оценки уровня сервисов ОТС по обобщенным показателям функционирования телекоммуникационной сети (ТКС), является актуальной.

Постановка задачи исследования

Масштабы современных ТКС вынуждают ОТС использовать современные системы управления (СУ) [1, 2], фокус управления которых смещается от управления ТКС к управлению сервисами и пользователями. СУ призваны автоматизировать процессы мониторинга, анализа состояния, управления распределением нагрузки, поиска и устранения неисправностей, а также решения множества других задач, связанных с предоставлением высококачественных сервисов.

Для управления качеством сервисов необходимо оперативно получать информацию о значениях параметров функционирования ТКС, осуществлять в СУ сведение и анализ ключевых метрик уровня сети, сервисов и пользователей.

Наиболее чувствительными к девиациями значений параметров функционирования ТКС являются сервисы, связанные с передачей аудио и видео данных. Поэтому в данной работе основное внимание уделено вопросам оценки уровня таких сервисов как VoIP или IPTV. Причем оценка уровня сервисов производится по анализу значений ключевых метрик уровня сети.

В зависимости от того, пользуются ли эталонами методы, применяемые для оценивания качества передачи аудио и видео потоков, их можно отнести к одному из двух типов [3, 4]. Методы первого типа основаны на использовании эталонов, т. е. образов передаваемого контента. Оценка качества производится путем сравнения передаваемого и принимаемого потоков данных, что делает невозможным ис-

пользование этих методов для непрерывного мониторинга качества сервиса на стороне абонента. Методы второго типа не требуют знания эталона, поэтому представляют наибольший интерес для данных исследований.

Для оценки качества сервисов, связанных с передачей аудио и видео данных, применяются субъективные и объективные методы. Первые основаны на использовании MOS-подобных оценок [5–7] качества голоса или изображения, когда абонент после прослушивания аудио или просмотра видео контента выставляет субъективную оценку. Необходимость присутствия экспертов в том месте, где производится оценка качества сервисов, делает невозможным применение подобных методов при оперативной оценке качества сервиса, предоставляемого большому количеству пользователей, а также для автоматизации процессов непрерывного мониторинга и управления уровнем таких сервисов. Поэтому ОТС широко используют методы объективного оценивания качества сервисов VoIP или IPTV, такие как NIQA [4], LCQA [8] и др.

Для объективного оценивания качества IPTV без использования эталона применяются алгоритмы EPSNR, TVQM, MPQM, основанные на оценке четкости контуров изображения, сравнении текущего и предыдущего кадров изображения, анализе значений джиттера, величины потери пакетов и длительности задержки и других метрик [9].

Оценивание качества сервисов объективными методами требует установки специального оборудования на стороне абонента, что при большом количестве абонентов осуществить практически невозможно. Ситуацию может спасти установка оборудования только у отдельных абонентов с последующей интерполяцией вычисленного уровня сервиса на остальных абонентов. Однако такой метод приводит к увеличению сложности и стоимости СУ ввиду большого количества подсетей, на которые распадается ТКС большого ОТС.

Особый интерес представляет исследование возможности оценки ожидаемого качества сервиса в отдельно взятой подсети для всех абонентов на основании анализа значений показателей суммарного трафика, генерируемого абонентами этой подсети и проходящего через сетевой узел, соединяющий данную подсеть с другими подсетями. Преимущество такого подхода очевидно – трафик анализируется не на стороне большого количества пользователей, а в одном месте, каковым является сетевой узел, через который проходит весь трафик. Однако погрешность оценки уровня сервисов, которую дает такой подход, может быть недопустимо большой.

В то же время значения ключевых метрик и параметров функционирования ТКС, таких как вероятность потери пакетов, задержка, загруженность каналов связи, скорость передачи данных, джиттер и т. д., которые оказывают непосредственное влияние на качество телекоммуникационных сервисов, могут предоставить современные узлы ТКС, например, маршрутизаторы Cisco. Узлы получают такую информацию в результате самостоятельно проводимого анализа каналов, соединяющих узлы, и выполнения необходимых расчетов для определения значений метрик. Эта информация может быть удаленно считана СУ и использована для оценки ожидаемого уровня сервиса у абонентов, трафик которых проходит через эти узлы. В этом случае отпадает необходимость применения дополнительного специального оборудования. Недостатком подхода может быть существенная погрешность оценки качества сервисов VoIP и IPTV. Однако, учитывая тот факт, что для ОТС гораздо важнее поддерживать высокий уровень качества сервисов для большинства пользователей, получая оценку качества с минимальной стоимостью, чем точно определять величину этого уровня для каждого абонента, то такой подход вполне оправдан, поэтому взят за основу в данной работе.

Предельные значения некоторых ключевых метрик уровня сети, при которых качество аудио или видео потоков еще соответствует допустимому уровню качества, регламентируется рядом стандартов, например [10–15]. При этом оговариваются только пороговые значения для отдельных метрик без учета негативного влияния значений других метрик. В то же время трафик, проходящий через узлы ТКС, характеризуется значениями большого количества параметров, поэтому представляет интерес установить зависимость качества сервисов, связанных с передачей аудио и видео данных, от совокупности параметров трафика и провести исследование зависимости уровня сервиса от совокупности значений метрик уровня сети.

Таким образом, необходимо разработать метод оценки уровня сервисов, связанных с передачей мультимедийной информации, на основании анализа значений совокупности метрик, измеренных в узлах ТКС. Для реализации метода необходимо определить, как сопоставить значения ключевых метрик трафика, проходящего через узел, с ожидаемым уровнем услуг, предоставляемых абонентам подсети. Решение

данной задачи может быть осуществлено с использованием методов теории распознавания образов.

Целью данной работы является разработка метода оценки уровня сервисов, связанных с передачей аудио и видео информации, на основании анализа значений совокупности метрик, измеренных в узлах телекоммуникационной сети. Метод основан на теории распознавания образов и использует потенциальные функции.

Применение методов распознавания образов в задачах оценки уровня сервисов

Задачу оценки уровня сервисов, предоставляемых ОТС, на основе анализа значений ключевых метрик, измеренных в узлах ТКС, можно рассматривать как задачу распознавания образов. В этом случае классами будут выступать оценки уровня сервиса: «неудовлетворительный», …, «отличный», признаками – значения метрик уровня сети, а в качестве объектов – зафиксированные ситуации, характеризующиеся значениями метрик и относящиеся к одному из классов с определенным уровнем сервиса. В геометрической интерпретации каждому объекту соответствует точка с координатами, отвечающими численным значениям признаков, расположенная в подпространстве, отвечающему классу, к которому относится объект. Соответствие описанных понятий отображено в таблице 1.

Таблица 1. Соответствие понятий

Задача оценки уровня сервиса	Распознавание образов	Геометрическая интерпретация
Уровень сервиса	Класс, образ	Подпространство, область
Метрики, параметры трафика	Признаки	Координатные оси
Ситуация, замер: в некоторые моменты времени определяются значение метрик и фиксируется уровень сервиса	Объект: характеризируется признаками, относится к определенному классу	Точка: имеет координаты, относится к подпространству

Обучив классифицирующие модели по значениям ключевых метрик узла ТКС, можно в процессе автоматического мониторинга определять ожидаемый уровень предоставляемого сервиса в подсети без использования аппаратуры, устанавливаемой на клиентской стороне, как этого требуют методы объективной оценки [3, 4, 8, 9].

В используемом методе в процессе мониторинга ТКС накапливается статистика значений ключевых метрик уровня сети и параметров функционирования ТКС. Путем периодических опросов, сбора жалоб пользователей, проведения тестов с участием экспертного комитета составляются записи о периодах, когда был известен уровень сервиса. Таким способом формируются статистические выборки для подсетей, на которых обучаются математические модели, сопоставляющие ожидаемый уровень сервиса со значениями параметров ТКС. В процессе последующего мониторинга ТКС текущие значения параметров и ключевых метрик уровня сети используются для оценки ожидаемого уровня сервиса посредством полученных и обученных математических моделей распознавания ситуации.

Метод потенциальных функций

Для решения поставленной задачи в работе использован метод потенциальных функций, разработанный в 60-х годах прошлого века [16]. Метод применим к задачам распознавания образов с учителем и без учителя, а также для восстановления функций. Оба аспекта важны при оценке уровня сервисов и рассматриваются в данной работе.

Свое название метод получил из-за сходства вида этих функций с картиной распределения потенциалов электростатического поля, создаваемого частицами с разноименными зарядами. В последнем случае потенциал поля в определенной точке соответствует сумме потенциалов, создаваемых каждой из частиц. Причем потенциал поля максимальный вблизи частицы и уменьшается по мере отдаления от нее.

Потенциальная функция $K(x, y)$ – функция двух переменных, где x и y – точки пространства X [16]. Функция определена на всем пространстве X и положительна $(K(x, y) > 0, \forall x, y \in X)$. Закрепив источник потенциала в точке $y = x'$, получим функцию $K(x, x')$ одной переменной x. Значение функции $K(x, x')$ зависит от расположения x относительно источ-

ника потенциала x', принимает максимальное значение, когда точка x совпадает x', $\max K(x, x') = K(x', x')$, и убывает при удалении точки x от x', причем $\rho(x_1, x') > \rho(x_2, x') \rightarrow K(x_1, x') < K(x_2, x')$, где ρ – расстояние между точками.

Таким образом, основными свойствами потенциальной функции являются:

1. $K(x, y) > 0$, $\forall x, y \in X$;
2. $\max K(x, x') = K(x', x')$;
3. $\rho(x_1, x') > \rho(x_2, x') \rightarrow K(x_1, x') < K(x_2, x')$.

Поверхность потенциальной функции имеет колоколообразный вид. На рис. 1 функция изображена для двухмерного пространства. Закрепленная точка x' является источником потенциала, а значение потенциала в точке x убывает по мере удаления от x'.

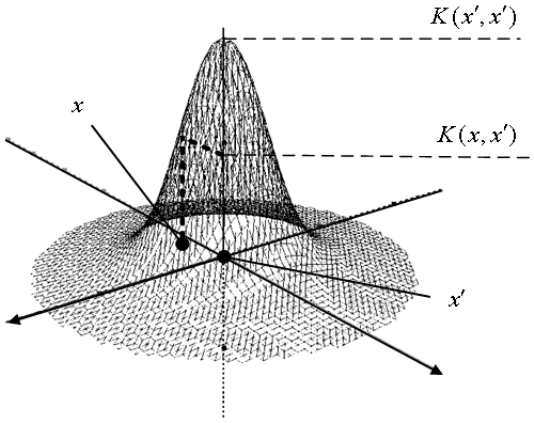


Рис. 1. Вид потенциальной функции в двухмерном пространстве

При распознавании образов A_j, $j = 1, 2, \ldots$, когда $j \geq 2$, целесообразно суммировать потенциальные функции $K(x, x_i)$ в точках $x_i \in A_j$, определяемых учителем как относящиеся к *j*-му образу. В этом случае потенциал $\Phi_{A_j}(x)$ каждого образа A_j будет определяться так:

$$\Phi_{A_j}(x) = \sum_{x_i \in A_j} K(x, x_i).$$

При распознавании точка x^* относится к тому образу A_j, $j = 1, 2, \ldots$, потенциал которого $\Phi_{A_j}(x^*)$ в заданной точке максимальный – $\max_j \Phi_{A_j}(x^*)$. На рис. 2 показан пример определения потенциалов образов A_1 и A_2.

При распознавании только двух образов, например, *A* и *B*, целесообразно ввести разделяющую функцию f, которая принимает положительные значения для точек образа *A* и отрицательна для точек образа *B*. Тогда потенциалы точек, относящиеся к образу *A*, следует прибавлять к разделяющей функции, а относящиеся к *B* – отнимать.

Для разделяющей функции *n*-е приближение f^n определяется следующим образом:

$$f^n = \sum_{x_i \in A} K(x, x_i) - \sum_{x_j \in B} K(x, x_j).$$

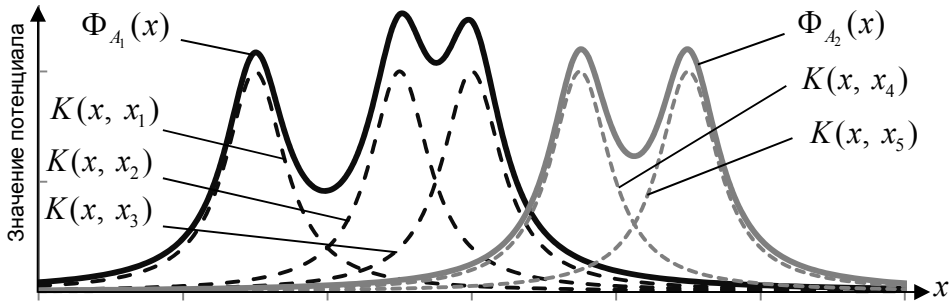


Рис. 2. Пример определения потенциалов двух образов, причем $x_1, x_2, x_3 \in A_1$, $x_4, x_5 \in A_2$

Причем восстанавливаемая функция f^*, соответствующая высказыванию учителя относительно определения класса точки, принимает значения $f^* \geq 0$ при $x \in A$ и $f^* < 0$ при $x \in B$.

Метод является итерационным. Точки предъявляются по одной в случайном порядке и после каждого показа происходит коррекция восстанавливаемой или разделяющей функции. При этом находится следующее приближение f^{n+1}. Особенностью геометрической интерпретации метода при распознавании двух образов является то, что исходное пространство признаков *X* дополняется осью, по которой откладывается значение потенциала. Работа происходит в этом расширенном пространстве, а разделяющая гиперповерхность проходит через точку пересечения решающей функцией оси нулевого потенциала.

Рекуррентная процедура нахождения очередного приближения разделяющей функции задается следующим образом [16]:

$$f^{n+1} = q^n f^n + r^n K\left(x, x^{n+1}\right),$$

при этом параметр q^n чаще всего принимается равным единице, а параметр r зависит от значений f^n и f^* в данной точке. Причем

$$r^n = \gamma_n\left(r\left(f^n, f^*\right) + \varepsilon\right),$$

где ε – помеха; коэффициент γ_n является либо константой, либо стягивающим множителем; $r\left(f^n, f^*\right)$ – функция, указывающая направление корректировки, определяемая следующим образом:

$$r\left(f^n, f^*\right)\begin{cases} < 0, & f^* < f^n; \\ > 0, & f^* > f^n; \\ = 0, & f^* = f^n. \end{cases}$$

При распознавании образов будем считать разделяющей функцией такую функцию, которая принимает положительные значения для объектов одного образа и отрицательные для другого:

$$f^*(x_i)\begin{cases} \geq 0, & x_i \in A; \\ < 0, & x_i \in B. \end{cases}$$

Тогда рекуррентная формула определения очередного приближения разделяющей функции принимает следующий вид:

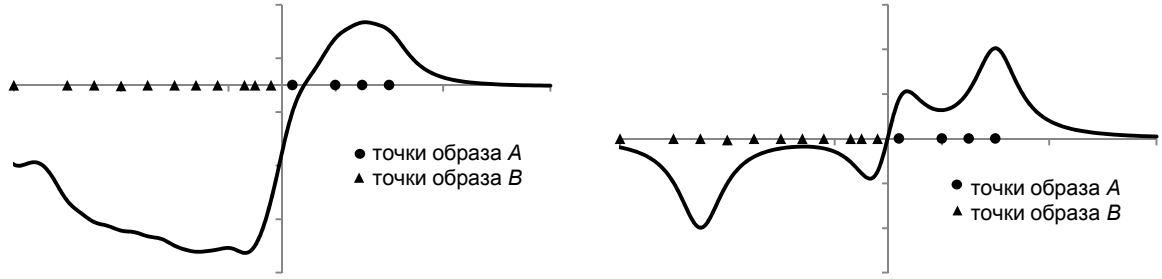
$$f^{n+1} = f^n + r^n K(x, x^{n+1}),$$

где

$$r^n = \begin{cases} 0, \text{при отсутствии ошибки}; \\ +1, f^n(x^{n+1}) < 0, \quad x \in A; \\ -1, f^n(x^{n+1}) \geq 0, \quad x \in B. \end{cases}$$

Корректировка функции производится только в тех точках, в которых происходят ошибки распознавания (см. рис. 3,*а*). Использование только этих точек для обучения позволяет производить эффективное обучение на несбалансированных выборках, когда количество объектов одного образа значительно превышает количество объектов другого образа (см. рис. 3,*б*). В противном случае, при преобладании объектов одного образа в обучающей последовательности граница разделения будет смещаться в сторону области второго образа, вследствие чего часть объектов второго образа будет ошибочно распознаваться как объекты первого образа (см. рис. 3,*а*). Кроме того, игнорирование точек, в которых функция правильно осуществляет классификацию, существенно снижает время обучения. В итоге после *n*-го приближения разделяющая функция представляет собой сумму потенциальных функций в данных точках:

$$f^n = \sum_{x_i \in A} K(x, x_i) - \sum_{x_i \in B} K(x, x_i).$$



a) *b*)

Рис. 3. Вид разделяющей функции: а) с учетом всех объектов обучающей выборки; б) с учетом только тех объектов, для которых происходят ошибки распознавания

Стоит заметить, что время классификации моделями, обученными методом потенциальных функций, линейно зависит от количества точек, использованных для обучения, то есть тех точек, для которых решающая функция модели допускала ошибки при обучении. Во время обучения модели на каждом шаге происходит классификация с учетом точек, использованных для обучения на предыдущих итерациях, таким образом, время обучения не превышает $n(n-1)/2$. Большинство ошибок распознавания происходит вблизи границы разделения, поэтому количество точек, использованных для обучения, примерно соответствует количеству точек, расположенных в непосредственной близости к границе разделения.

При решении задачи восстановления функций может быть реализовано две процедуры восстановления. В первом случае, когда не известно лучшее разложение функции f^*, применяется процедура, которая может как восстанавливать, так и приближать функцию:

$$f^{n+1} = f^n + \gamma_n sign(f^*(x^{n+1}) - f^n(x^{n+1}))K(x, x^{n+1}),$$

$$\sum_{i=1}^{\infty}\gamma_i = \infty; \sum_{i=1}^{\infty}\gamma_i^2 < \infty.$$

При известном наилучшем разложении восстанавливаемой функции может быть использована процедура, восстанавливающая функцию. Реализация такой процедуры для метода потенциальных функций не содержит стягивающего множителя γ:

$$f^{n+1} = f^n + \frac{1}{\Lambda}\left(f^*\left(x^{n+1}\right) - f^n\left(x^{n+1}\right)\right)K\left(x, x^{n+1}\right),$$

где $\Lambda > \frac{1}{2}\sup K\left(x, x\right)$, $\gamma = 1/\Lambda = \text{const}$;

$r\left(f^n, f^*\right) = (f^*(x^{n+1}) - f^n(x^{n+1}))$.

Благодаря тому, что метод является итерационным, проявляется его преимущество – простота переобучения моделей при появлении новых обучающих данных. В этом случае сохраняются все старые данные и только исправляются ошибки, возникающие при классификации новых данных. Поскольку модель заново не обучается, то переобучение происходит быстро.

Обобщенный алгоритм решения задач методом потенциальных функций. Предлагается следующий обобщенный алгоритм решения задач классификации и восстановления функций методом потенциальных функций. Алгоритм состоит из следующих этапов:

1. Если вид функции f^* известен, выбирается $K(x,y)$, раскладывающаяся по той же системе функций, что и f^*. Если вид функции f^* неизвестен, то $K(x,y)$ подбирается интуитивно или экспериментально.

2. Задается первое приближение функции: $f^0 \equiv 0$.

3. Вычисляется следующее приближение:

а) Вычисляется значение функции в новой точке $f^n\left(x^{n+1}\right)$.

б) Определяется r в зависимости от поставленной задачи – восстановление функции или распознавание.

4. В случае, если коэффициент r^n не равен нулю, сохраняются значения x^{n+1} и r^n.

5. Производится показ очередной точки x^{n+1} и осуществляется вычисление следующего приближения.

6. Остановка может производиться через L показов после последнего исправления (количество L задается), либо после показа всей обучающей выборки.

7. Качество полученной модели проверяется путем показа проверочной выборки.

Пример применения метода потенциальных функций

Работу метода потенциальных функций продемонстрируем на разделении двух образов в одномерном пространстве. Пусть образу A соответствуют все точки, значение координаты которых $x \geq 0$, образу B – точки, значение координаты которых $x < 0$. В качестве потенциальной функции выберем

$$K\left(x, x^*\right) = \frac{1}{1+10(x-x^*)^2}.$$

Изначально $f^0 \equiv 0$ на всем диапазоне, поэтому все объекты определяются, как относящиеся к образу A, и такая функция дает ошибки распознавания во всех точках образа B. После первого предъявления объекта из образа B, например, в координате $x = -5$, разделяющая функция становится отрицательной $f^1 < 0$ на всем диапазоне x, и все объекты относятся к образу B (рис. 4,*а*). Далее предъявляется объект образа A в координате $x = 3$ (рис. 4,*б*). Сплошной линией на графике показана разделяющая функция f, равная сумме потенциалов, создаваемых предъявленными точками. Функция разделяет два образа в точке $x = -1$ $(f = 0)$. На рис. 4,*в* – рис. 4,*е* показаны следующие итерации, причем пунктирными линиями показаны потенциалы, создаваемые каждой из точек обучающей выборки.

После 13 исправлений разделение происходит в точке $x = -0,04$ (рис. 4,*ж*), а при использовании такой разделяющей функции f происходят ошибки классификации только в полуоткрытом интервале $-0,04 \leq x < 0$.

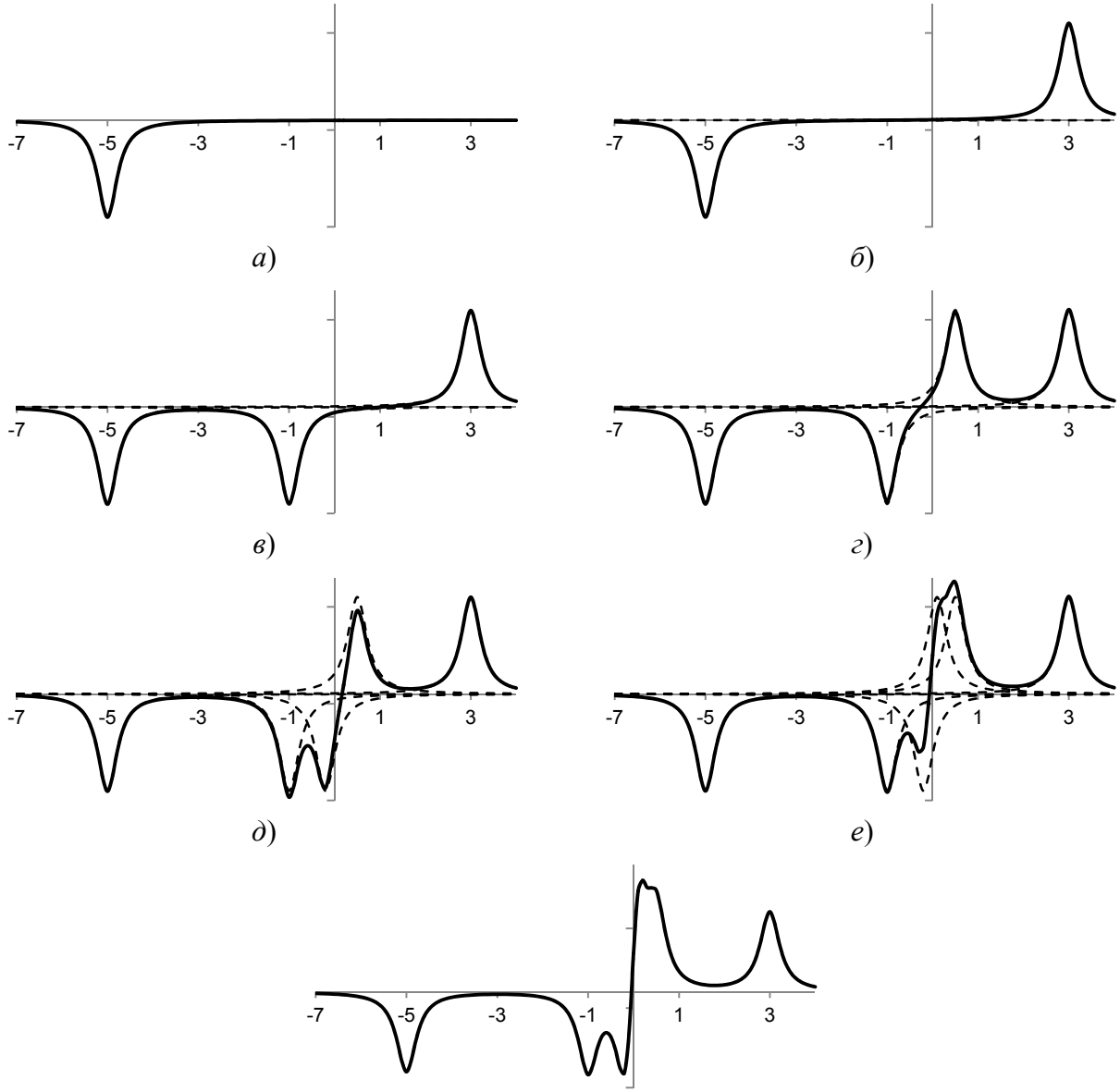


Рис. 4. Вид разделяющей функции после: а) одного исправления; б) двух; в) трех; г) четырех; д) пяти: е) шести; ж) тринадцати исправлений. Сплошной линией обозначена разделяющая функция, а пунктирной – потенциалы, создаваемые точками

Применение метода потенциальных функций для оценки уровня сервисов

В данной работе с использованием метода потенциальных функций решалась задача оценки уровня сервиса IPTV, предоставляемого абонентам ОТС. Оценка уровня сервиса производилась в зависимости от значений таких ключевых метрик как джиттер, потеря пакетов и задержка. Согласно [10–15] эти показатели функционирования ТКС имеют наибольшее влияние на качество сервиса IPTV.

Влияние этих параметров на качество работы сервиса IPTV заключается в следующем:

Задержка. Благодаря использованию буферизации качество видео в IPTV не имеет строгих ограничений на максимальную величину задержки, однако увеличение задержки приводит к увеличению времени переключения каналов, поскольку требуется больше времени для буферизации.

Потери пакетов. Потеря пакетов проявляется для пользователя в виде шумов в видеоизображении и звуке. При анализе этой метрики необходимо учитывать различное влияние на качество изображения типа потерянного кадра (*I*, *P* или *B*) и устойчивость кодека к потере медиа-данных. Поскольку при формировании передающим оборудованием дейтаграммы UDP в нее, как правило, инкапсулируется до семи 188-байтовых пакетов, то потеря хотя бы одной дейтаграммы UDP приведет к появлению видимых дефектов изображения. Стандартом

ETSI TS 102 034 допускается интенсивность потери дейтаграмм не более одной в час.

Джиттер. Увеличение значения джиттера может привести к переполнению или высвобождению буфера на приемной стороне. Следствием большой величины джиттера может быть ситуация, когда пакеты будут приходить вне очереди, что может приводить к тому, что они будут отброшены. Это вызовет эффект, схожий с дефектами изображения от потери пакетов.

При проведении экспериментальных исследований уровень сервиса оценивался оценкой MOS [5–7]. При разных значениях метрик в ходе оценки формировалась таблица данных, в которой каждому набору значений метрик сопоставлялась оценка уровня сервиса. Для обучения моделей и проверки их характеристик полученные данные делились на две выборки – обучающую и проверочную. На обучающей выборке производилось обучение моделей, а проверочная использовалась для определения адекватности, устойчивости и стабильности полученных моделей, а также времени классификации. Для сравнения были построены модели с помощью следующих методов: логистической линейной регрессии, нелинейной регрессии, наивного байесовского классификатора, метода потенциальных функций, деревьев решений с решающим правилом, построенным на основе логистической линейной регрессии, и метода потенциальных функций. В таблице 2 приведены средние показатели времени обучения, требуемой оперативной памяти, адекватности и стабильности моделей, обученных указанными методами.

Таблица 2. Средние экспериментальные показатели для используемых методов

Метод распознавания	Время обучения, мс	Требуемая память, кб	Адекватность	Стабильность
Наивный байесовский классификатор	60	569	0,78	0,55
Логистическая линейная регрессия	55	3539	0,79	0,58
Нелинейная регрессия	236	6497	0,65	0,30
Метод потенциальных функций	3	104	0,93	0,80
Деревья решений (решающее правило – метод потенциальных функций)	7	247	0,87	0,78

При проведении экспериментов по оценке качества сервиса IPTV лучшие результаты показали метод потенциальных функций, дерево решений на основе метода потенциальных функций и наивный байесовский классификатор.

Вид разделяющих поверхностей, построенных методом потенциальных функций для разных обучающих выборок, приведен на рис. 5. По осям отложены нормированные значения метрик джиттера, задержки и потери пакетов, по значениям которых производилась оценка уровня сервиса для двух градаций: «удовлетворительно» и «не удовлетворительно». Значения метрик нормировались относительно значения 60 мс для джиттера, 2300 мс для задержки и 0,06 для потерянных пакетов. Поверхность соответствует точкам пространства с нулевым потенциалом и отделяет область с хорошим качеством сервиса от области с неудовлетворительным качеством, причем область, расположенная под поверхностью, соответствует оценке «удовлетворительно», а над поверхностью – «неудовлетворительно».

На рис. 5 изображены поверхности для двух градаций уровня качества. Продолжая разбивать область с удовлетворительным уровнем сервиса на подобласти, можно легко получить большее количество градаций уровня сервиса. Полученная в результате такого процесса многослойная фигура будет отображать значения метрик в шкалу оценки уровня сервиса от «неудовлетворительно» до «отлично».

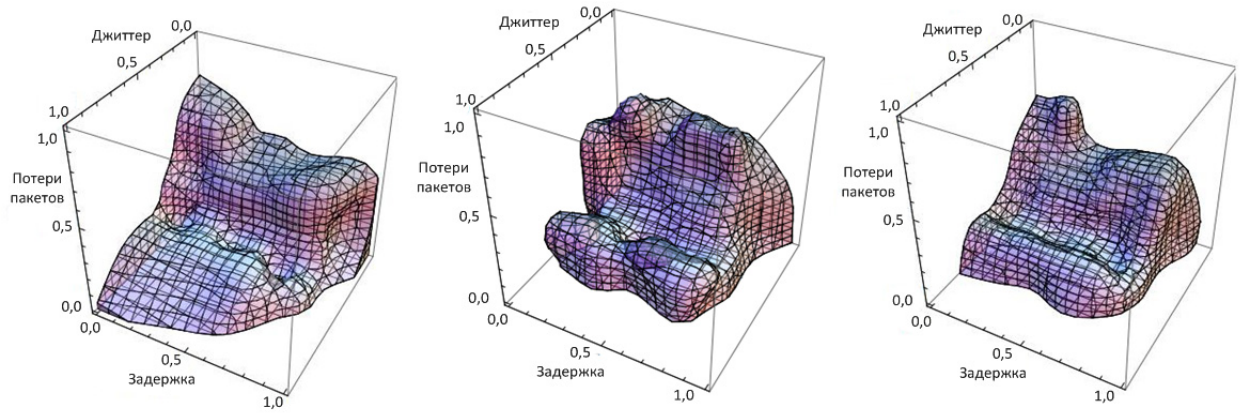


Рис. 5. Примеры разделяющих поверхностей, полученных методом потенциальных функций для различных обучающих выборок

Рекомендации использования метода потенциальных функций для оценки уровня телекоммуникационных сервисов

При проведении исследований, посвященных применению метода потенциальных функций для решения задач оценки уровня телекоммуникационных сервисов, были обнаружены преимущества и вскрылись недостатки метода.

Метод потенциальных функций при обучении и переобучении исправляет разделяющую функцию только в тех точках, в которых происходят ошибки. Данная особенность является одновременно достоинством и недостатком метода. С одной стороны, метод является менее чувствительным к несбалансированным выборкам, в которых количество точек одного класса значительно превышает количество точек другого класса. Имея статистику жалоб пользователей и данные мониторинга ТКС, легче выявить моменты времени, когда сервис предоставлялся с неудовлетворительным качеством, чем периоды, когда качество сервиса было на согласованном уровне. В связи с эти возникает дисбаланс в сторону точек с неудовлетворительным качеством.

С другой стороны, результаты работы метода потенциальных функций зависит от порядка предъявления точек. Модель корректируется при обучении только тогда, когда возникают ошибки распознавания. Поэтому необходимо, чтобы точки, принадлежащие к разным классам, из обучающей выборки поступали бессистемно. В противном случае многие точки, являющиеся полезными для обучения модели, окажутся проигнорированными. Это требование соответствует указаниям, изложенным в [17], где говорится о том, что точки в обучающей последовательности должны быть беспорядочно распределены по образам и подаваться в случайном порядке.

Целесообразно построить несколько моделей, перемешивая выборку перед построением каждой модели. После чего следует сравнить модели и отобрать лучшие по имеющимся критериям. Проведенные исследования показали, что путем двух- и более кратного повторного показа проигнорированных при обучении точек можно улучшить адекватность полученных моделей. На рис. 6 и рис. 7 показаны результаты переобучения модели, обученной с точкой-выбросом.

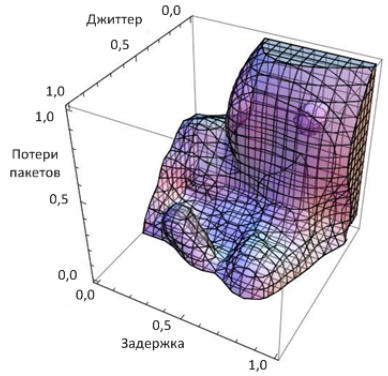


Рис. 6. Модель, обученная с выбросом

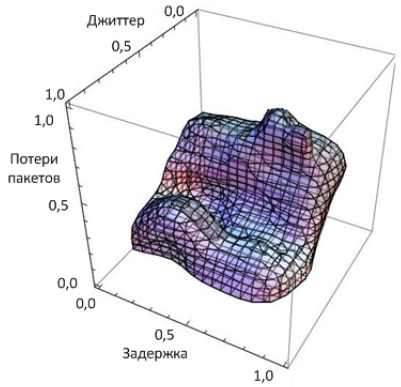


Рис. 7. Исправление ошибок выброса путем переобучения

Появление точки-выброса посредине процесса обучения и его неокончательное последующее исправление привело к тому, что адекватность модели составила всего 0,75 (рис. 6). «Раковина» на рисунке соответствует искажению разделяющей функции, возникшему вследствие появления точки-выброса, «раздутость» вокруг соответствует появлению исправляющих точек в окрестностях точки-выброса, однако такое исправление еще сильнее исказило разделяющую поверхность. После исправления ошибки, созданной выбросом, путем переобучения с показом ранее игнорированных точек адекватность модели возросла до 0,9 (рис. 7).

Параметры ТКС, оказывающие влияние на качество сервисов могут изменяться в различных диапазонах. В результате чего метод может быть нечувствительным или наоборот, слишком чувствительным к девиациям значений отдельных параметров. В случае, если ядро потенциальных функций является функцией от расстояния между двумя точками либо имеет иную, симметричную относительно различных параметров форму, для достоверного распознавания методом потенциальных функций необходимо, чтобы диапазоны изменения параметров примерно совпадали, а точки были распределены со схожей частотой относительно каждого из параметров. С этой целью производится масштабирование или нормирование значений параметров. Кроме того, можно подбирать форму ядра, соответствующую масштабу изменений значений параметров.

Если неизвестен вид разделяющей функции, то отсутствует возможность определения точного вида и параметров ядра потенциальных функций. На практике не обязательно определять наиболее близкое разложение разделяющей функции, можно выбрать в качестве потенциальной функции любую гладкую функцию от двух точек, которая соответствует указанным выше требованиям [17]. Удобно в качестве потенциальной функции взять колоколообразную функцию от расстояния между двумя точками, принимающую максимальное значение в центре. От выбора формы и параметров потенциальной функции зависит степень остроты колокола, и, соответственно, степень влияния отдельного исправления на всю разделяющую функцию. При выборе потенциальной функции, имеющей вид острого колокола, при каждом исправлении корректируется только малый локальный участок, поэтому размер обучающей выборки должен быть большим. Также существует риск получить участки, на которых разделяющая функция принимает вид «гребенки», часто меняя решение об отнесении объектов к тому или иному образу на небольшом промежутке. При выборе потенциальной функции, имеющей вид пологого колокола, исправляется широкий участок результирующей функции, а граница разделения передвигается скачкообразно на существенное расстояние. При этом в итоге можно получить неадекватную функцию, если в конце обучения предъявить алгоритму неудачную точку.

Выбор формы потенциальной функции также зависит от частоты расположения точек обучающей выборки. Чем выше плотность точек, тем более узким может быть колокол потенциальной функции. Если же выбрать потенциальную функцию, имеющую вид острого колокола, при выборке с разреженными точками каждое исправление будет влиять лишь на небольшой локальный участок разделяющей функции. Поэтому целесообразно обучать сразу несколько моделей, выбирая разные потенциальные функции, после чего сравнивать их адекватность и отдать предпочтение модели, которая дает лучшие результаты.

Модели на основе потенциальных функций можно упростить, убрав из разделяющей функции отдельные точки, которые мало влияют на вид функции или являются выбросами. Одним из способов упрощения моделей является графическое построение функции для определения точек выброса, либо точек, не влияющих на результирующую функцию, и их последующего удаления из точек модели. Представляет интерес разработка методов автоматического выявления точек выброса и прореживания не влияющих на результирующую функцию точек для моделей метода потенциальных функций.

Выводы

Предложен метод оценки качества сервисов, связанных с передачей аудио и видео информации, на основании анализа значений совокупности ключевых метрик, измеренных в узлах телекоммуникационной сети. Оценивание уровня сервиса производится с применением методов теории распознавания образов, в частности, метода потенциальных функций. Проанализированы особенности применения метода при решении задач оценки уровня сервиса

IPTV. Проведены исследования для сравнения эффективности метода потенциальных функций с другими методами распознавания образов. Показано, что предложенный метод оценки уровня сервиса обладает большой точностью и быстродействием, а также не требователен к вычислительным ресурсам. Метод быстро обучается и хорошо классифицирует предъявляемые объекты. При получении новых данных не требует полного повторного обучения, ограничиваясь только исправлением ошибок классификации на новых данных. Отмечены недостатки метода и предложены варианты их устранения при оценке уровня телекоммуникационных сервисов.

Модуль оценки уровня сервиса с использованием методов распознавания образов, в том числе метода потенциальных функций, реализован в составе системы управления SmartBase ITS Control, разработанной в НТУУ «КПИ».

Список литературы

1. 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.
2. Ролик А.И. Система управления корпоративной информационно-телекоммуникационной инфраструктурой на основе агентского похода / А.И. Ролик, А.В. Волошин, Д.О. Галушко, П.Ф. Можаровський, О.О. Покотило // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: «Век+», – 2010. – № 52. – С. 39–52
3. Harte L. Service Quality Monitoring, Analyzing, and Diagnostics for IP Television Systems and Services / L. Harte. – Althos Publishing, 2008. – 116 p.
4. NIQA – Non-Instrisuve voice Quality Analyzer. Режим доступа: http://www.sevana.fi/non-intrusive-voice-quality-testing-software.php
5. Rec. ITU-R BS.1387-1. Method for objective measurements of perceived audio quality. – Recommendation ITU-R. – 2001. – 100 p.
6. Rec. ITU-T J144. Measurement of the quality of service. Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference. – ITU-T Recommendation. – 2001. – 70 p.
7. Rec. ITU-R BT.1683. Objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a full reference. – Recommendation ITU-R (Question ITU-R 44/6). – 2004. – 107 p.
8. Grancharov V. Non-intrusive speech quality assessment with low computational complexity / Volodya Grancharov, David Y. Zhao, Jonas Lindblom, W. Bastiaan Kleijn // In INTERSPEECH 2006 – ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17–21, 2006. ISCA. – 2006.
9. Kratochvíl T., Slanina M. Digital Video Image Quality / Tomáš Kratochvíl, Martin Slanina // Digital Video, Floriano De Rango (Ed.), ISBN: 978-953-7619-70-1, InTech, (2010). Режим доступа: http://cdn.intechopen.com/pdfs/8532/InTech-Digital_video_image_quality.pdf
10. Rec. ITU-T G.107. The E-model: a computational model for use in transmission planning. – ITU-T Recommendation. – 2011. – 18 p.
11. Rec. ITU-T G.114. One-way transmission time. – ITU-T Recommendation. – 2003. – 12 p.
12. Technical Specification ETSI TS 102 034. Digital Video Broadcasting (DVB); Transport of MPEG¬2 TS Based DVB Services over IP Based Networks. V1.4.1. – ETSI. – 2009. – 229 p.
13. TR148. Technical Report: Managing the Quality of Customer Experience, Release 1.0. – TM Forum. – Nov., 2009. – 121 p.
14. Rec. ITU-T J.244. Measurement of the quality of service. Full reference and reduced reference calibration methods for video transmission systems with constant misalignment of spatial and temporal domains with constant gain and offset. ITU-T Recommendation. – 2008. – 53 p.
15. Rec. ITU-R BT.1359. Relative timing of sound and vision for broadcasting. – Recommendation ITU-R (Question ITU-R 35-4/11). – 1998. – 3 p.
16. Айзерман М. А. Метод потенциальных функций в теории обучения машин / М.А. Айзерман, Э. М. Браверман, Л. И. Розоноэр. – М.: «Наука», 1970. – 384 с.
17. Васильев В.И. Распознающие системы: [справочник] / В.И. Васильев. – К.: Наукова думка, 1983. – 422 с.

УДК 681.5.13

КРАВЕЦЬ П.І.,
ЛУКІНА Т.Й.,
ШИМКОВИЧ В.М.,
ТКАЧ І.І.

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ТЕХНОЛОГІЇ ОЦІНЮВАННЯ ПОКАЗНИКІВ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ MIMO-ОБ'ЄКТІВ УПРАВЛІННЯ

У даній роботі розглядається комплексний формалізований підхід до реалізації багатоетапної процедури ідентифікації складних динамічних об'єктів з використанням нейромережевих модельних структур, в роботі розроблено програмний пакет «MIMO-Plant» в середовищі MatLab, для оцінювання показників нейромережевих моделей MIMO-об'єктів управління, представлено його детальний опис та приклади дослідження.

In this paper describes an integrated formal approach to implement multistage procedure for identification of complex dynamic objects using neural network model structures in the developed software package «MIMO-Plant» in the environment MatLab, to assess the performance of neural network models MIMO-object of control, presented details and examples of research.

Реальні промислові об'єкти управління зазвичай є багатовимірними, тобто мають кілька входів і кілька виходів MIMO (multi input, multi output) [1, 2]. У ряді випадків такі об'єкти можна промоделювати, нехтуючи другорядними впливами, а також другорядними керованими величинами. В результаті такого обґрунтованого спрощення можна отримати просту модель з двома (керуючим і збурюючим), а в деяких випадках і з одним впливом, і з однією керованою величиною. Це істотно спрощує як аналіз об'єкта, так і розробку системи автоматичного управління ним. Однак у багатьох випадках необхідно забезпечити управління декількома керованими величинами об'єкта, його вихідними величинами, шляхом впливу на декілька його входів, тобто зміни керуючих величин, плюс до того слід врахувати ще й кілька збурень. Особливість управління та моделювання такого, досить складного в описі об'єкта, полягає в тому, що може статися, що вплив по одному входу призводить до зміни не однієї, а відразу декількох керованих величин. Наприклад, вертоліт, якщо пілот вертольота бажає прискоритися, то він, природно збільшує подачу газу. Однак тільки це не призводить до збільшення швидкості: несучий гвинт починає обертатися швидше, і вертоліт прагне піднятися. Більш того, прискорення обертання несучого гвинта змушує вертоліт ще й повертатися навколо вертикальної осі, змінюючи напрямок руху. Отже, щоб прискорити політ вертольота на тій же висоті і по тому ж напрямку, пілот повинен збільшити подачу палива і одночасно збільшити тягу заднього гвинта, змінюючи кут атаки його лопатей, а також нахилити вертоліт вперед (точніше, перекосити площина обертання лопатей головного гвинта), щоб додана збільшенням газу потужність витрачалася на політ вперед.

Нейромережеві системи управління являють собою новий високотехнологічний напрямок в теорії управління та відносяться до класу нелінійних динамічних систем [3]. Однією з головних рис нейронних мереж є універсальні апроксимінаційні властивості [4,5]. Нейронні мережі дозволяють з будь-якою точністю обраховувати довільну неперервну функцію багатьох змінних $f(s_1,...,s_n)$. Отже, з їх допомогою можливо як завгодно точно апроксимувати функцію, породжену будь-якою неперервною системою в тому числі і MIMO-об'єктами.

Метою даної роботи є розробка технології дослідження та оцінювання нейромережевих моделей MIMO-об'єктів управління для вибору оптимальної структури нейромережі.

Для вирішення завдань ідентифікації і управління найбільш адекватними є багатошарові нейронні мережі прямого поширення (БНМПР) [6-9]. При використанні БНМПР для реалізації процедури ідентифікації динамічних систем необхідно визначити «зовнішню» і «внутрішню» структури нейронної мережі. «Зовнішня» структура нейромережевої моделі

повністю визначається вектором входу і вектором виходу. При виборі «внутрішньої» структури нейромережевої моделі необхідно визначити:

- кількість прихованих шарів;
- кількість нейронів у кожному схованому шарі;
- вид активаційної функції для кожного шару.

Збільшення числа нейронів у схованому шарі і збільшення числа прихованих шарів підвищують репрезентативні можливості нейронної мережі, тобто дають можливість моделювати більш складні взаємозв'язки, але призводять до значних труднощів при практичній реалізації [10], збільшення часових витрат як на навчання, так і на роботу в режимі прогнозування. Це і пояснює факт прагнення використовувати мінімальну реалізацію БНМПР в більшості технічних додатків.

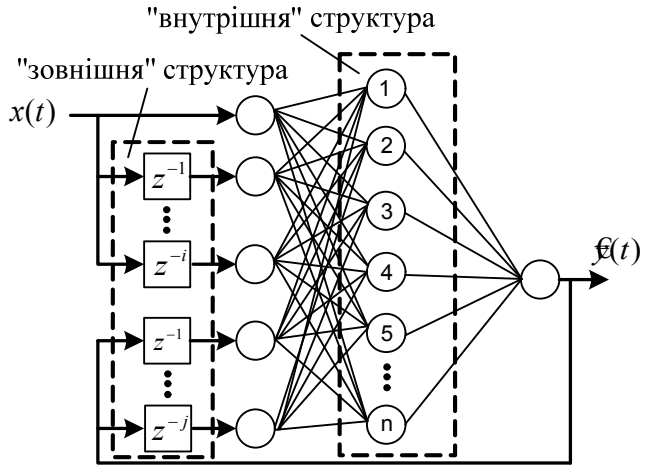


Рис. 1. Структурна схема багатошарової нейронної мережі прямого розповсюдження

Існує ряд теорем [4,6,11,12], які математично обґрунтовують апроксимуючу властивість нейронних мереж, з яких випливає що, багатошарові нейронні мережі з одним прихованим шаром, сигмоїдальними (сигмоїдальна, гіперболічний тангенс) функціями активації та лінійною функцією активації вихідного шару, може апроксимувати функції з заданою точністю в разі відсутності обмеження на число нейронів у схованому шарі. На рисунку 1 представлена структура БНМПР. Виходячи з вищевикладеного, завдання побудови нейромережевої моделі зводитися до визначення «зовнішньої» і «внутрішньої» структури, а саме визначення кількості затриманих входів i, затриманих сигналів виходу мережі j, і кількості нейронів прихованого шару n.

Основна ідея роботи полягає в побудові оптимальної, в силу деякого критерію, моделі за результатами спостережень над вхідними та вихідними змінними системи. На практиці реалізація процедури ідентифікації (рисунок 2) вимагає вирішення цілого ряду допоміжних завдань, основними з яких є:

- планування/проведення експерименту і попередня обробка експериментальних даних;
- вибір модельної структури;
- оцінка (оптимізація параметрів) моделі;
- прийняття рішення про адекватність моделі.

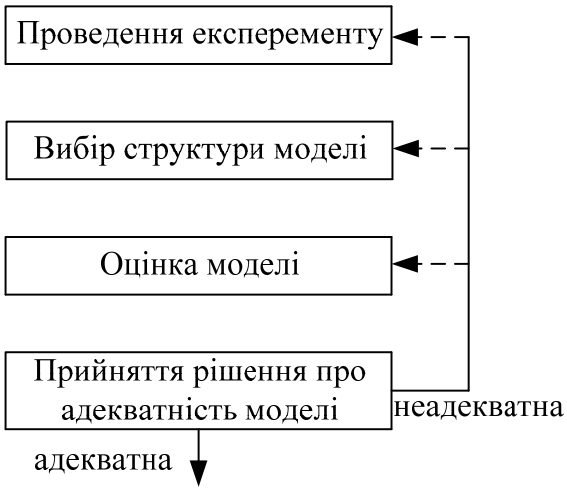


Рис. 2. Загальна схема реалізації процедури ідентифікації

Для вирішення даної задачі було розроблено програмний пакет «MIMO-Plant» в середовищі MatLab.

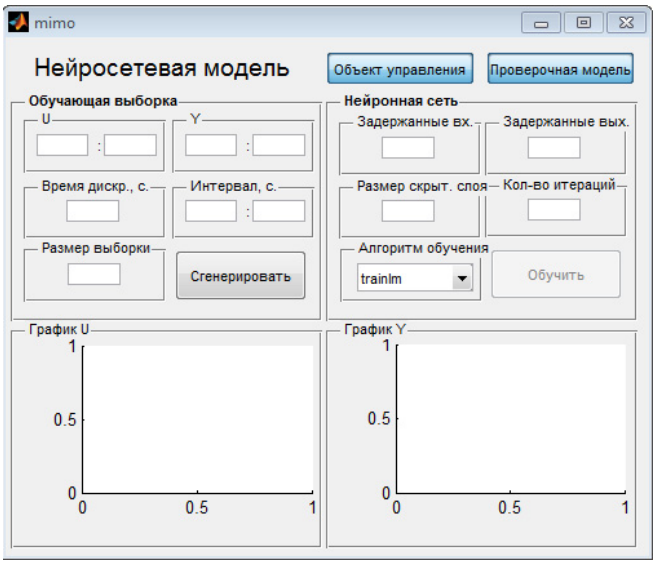


Рис. 3. Головне вікно програмного пакету «MIMO-Plant»

При натисканні на кнопку «Объект управления» відкриється модель об'єкта управління.

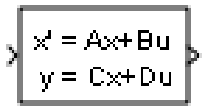


Рис. 4. Математична модель об'єкта управління

Завдання об'єкта відбувається за допомогою блоку State-Space. Цей блок реалізує систему, поведінку якої можна визначити як:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$

$$y = \mathbf{C}x + \mathbf{D}u,$$

де x – вектор станів, u – вхідний вектор, а y є вихідним вектором.

Матриця коефіцієнтів повинна мати такі характеристики:

- матриця **A** повинна бути n на n матриця, де n-число станів;
- матриця **B** повинна бути n на m матриця, де m-число входів;
- матриця **C** повинна бути r на n матриця, де r-число виходів;
- матриця **D** повинна бути r на m матрицю.

Ширина вхідного вектора залежить від кількості стовпців в матрицях **B** і **D**. Ширина вихідного вектора залежить від кількості рядків у матрицях **C** і **D**.

Підготовчим етапом створення нейромережі є генерація навчальної вибірки, яка відбувається при натисканні кнопки «Сгенерировать».

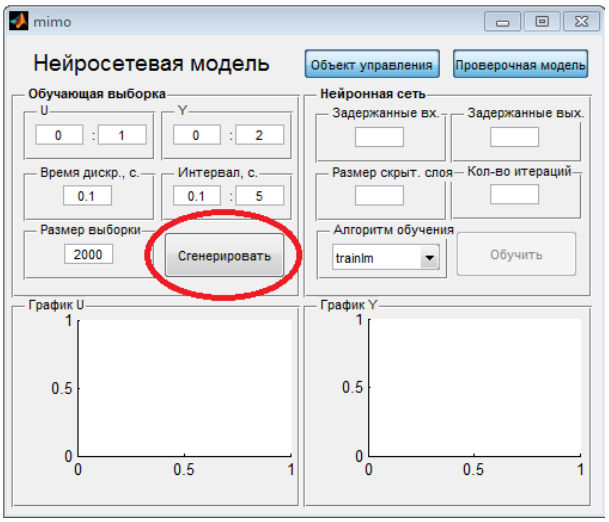


Рис. 5. Генерація навчальної вибірки

У полі навчальної вибірки необхідно задати параметри необхідні для генерації навчальної вибірки:

1. Задати діапазон зміни входу ($U_{\max}$, $U_{\min}$) і виходу ($Y_{\max}$, $Y_{\min}$) ОУ. При виборі $Y_{\max}$, не потрібно обмежувати вихід об'єкта управління, тобто $Y_{\max}$ вибирати великим усталеного значення при $U_{\max}$.

2. Розмір навчальної вибірки слід вибирати таким, щоб вона містила всі можливі комбінації амплітуд і частот з робочого діапазону системи (при часу дискретизації 0.1 рекомендуємо брати розмір навчальної вибірки порядку 2000-4000).

3. Інтервал визначає мінімальну і максимальну тривалість тестового сигналу. Причому максимальне значення повинно бути не менше часу встановлення перехідного процесу ОУ при стрибкоподібному впливі. Якщо всі поля групи «Навчальна вибірка» заповнені, то в області *PlantInput* і *PlantOutput* повинні відображатися вид вхідного впливу на ОУ та реакція ОУ на вхідні дії.

Після того як дані готові, переходимо до завдання структури МНСПР і її навчанню. Для цього необхідно заповнити поле «Нейронна мережа».

1. В полі «Розмір прихованого шару» вказуємо кількість нейронів n.

2. В полі «Кількість затриманих входів» вводимо значення i.

3. В полі «Кількість затриманих виходів» вводимо значення j.

4. В групі параметрів «Нейронна мережа», вибираємо алгоритм навчання (trainlm) і кількість навчальних ітерацій (200).

5. Натискаємо кнопку «Обучить».

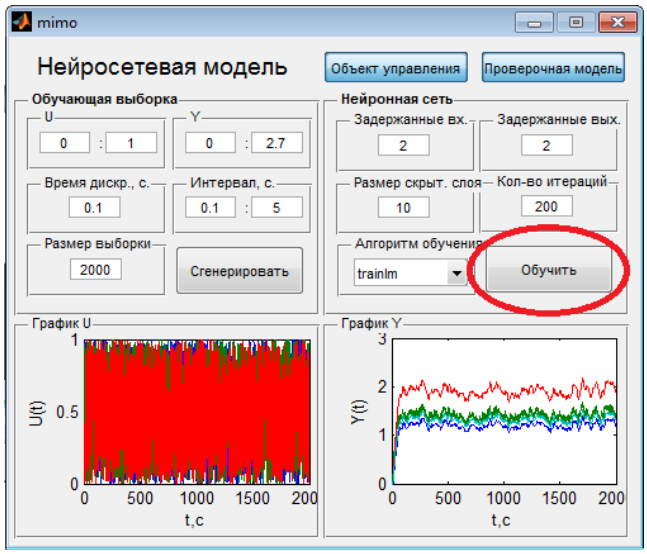


Рис. 6. Навчання нейромережевих моделей МІМО-об'єктів управління

Результатом виконання процедури навчання буде згенерована нейронна мережа, яка являє собою набір нейромережевих моделей ОУ (рисунок 7).

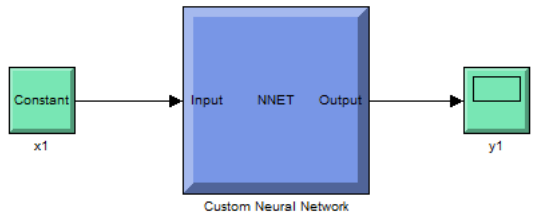


Рис. 7. Згенерована нейронна мережа

Для того, щоб подивитися перехідні процеси нейронної мережі та багатовимірного об'єкту управління, необхідно скористатися перевірочною моделлю, яка визивається при натисканні кнопки «Перевірочна модель». В результаті цих дій ми відкриємо наступну модель:

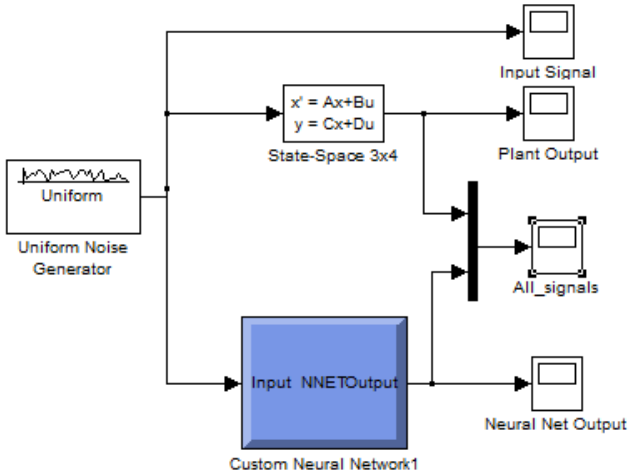


Рис. 8. Перевірочна модель

Нейронна мережа «Custom Neural Network1», яка зображена на рисунку 8, була отримана в результаті копіювання з вікна що з'являється після навчання нейронної мережі та підставлена у перевірочну модель.

При натисканні на блок «All_signals» перевірочної моделі ми можемо побачити графічне зображення виходу нейронної мережі та стандартного об'єкту управління, а також візуально порівняти отримані результати (рисунок 8).

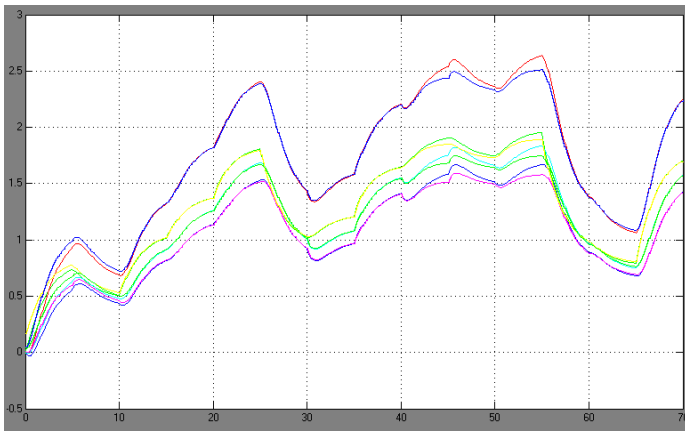


Рис. 9. Графічне зображення виходу нейронної мережі та стандартного об'єкту управління

Для обчислення сумарної середньоквадратичної помилки потрібно скористатися розробленою моделлю, де:

1) На вхід подається вхідний сигнал генератором сигналу.

2) Об'єкт управління задається блоком State-Space.

3) Під об'єктом управління знаходяться 16 варіацій структури нейромережі.

4) У блоці «Display» відображається сумарна середньоквадратична помилка, яка математично описується наступною формулою:

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^{z}(\hat{y}-y)^2}{z}},$$

де z – кількість значень похибки (вибирати рівне кількості елементів вектора *tout* з workspace Matlab); y – вихід моделі об'єкта або системи; $\hat{y}$ – вихід нейромережевої моделі об'єкта або системи.

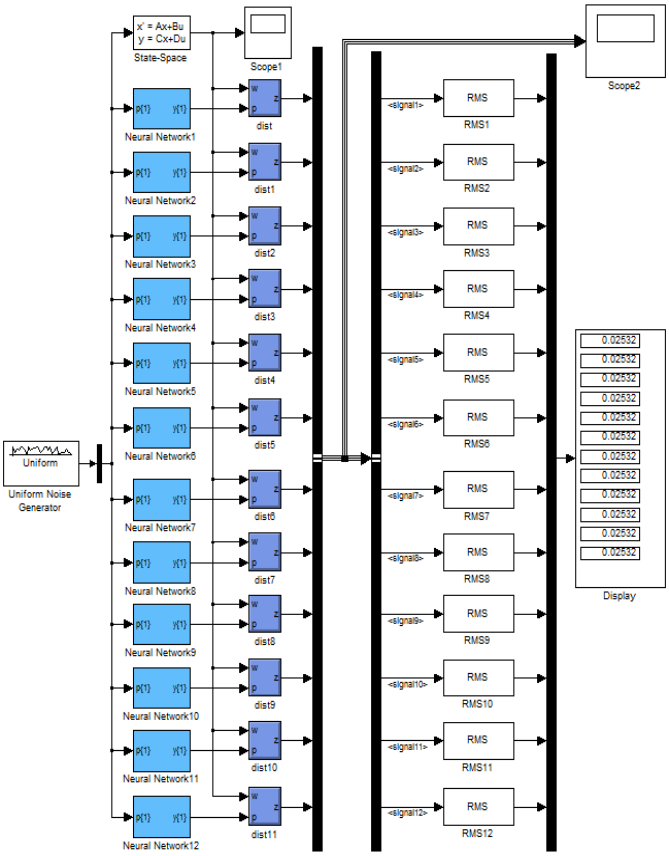


Рис. 10. – Модель обчислення сумарної середньоквадратичної помилки

Далі розглянемо приклад дослідження нейромережевих моделей MIMO-об'єкту управління, який має 3 вхідних та 4 вихідних параметри, в програмному пакеті «MIMO-Plant».

Крок 1. Задаємо MIMO-об'єкт:

$$\mathbf{A}=\begin{pmatrix}-1.23 & 0 & 0\\ 2.5 & -0.56 & 0\\ 0 & 3 & -2.3\end{pmatrix},\quad \mathbf{B}=\begin{pmatrix}0.2 & 1 & 0.3\\ 0 & 0.4 & 0.1\\ 0 & 0.2 & -1\end{pmatrix},$$

$$\mathbf{C}=\begin{pmatrix}0 & 0 & 0.3\\ 1 & 0 & 0.2\\ 0.1 & 0.2 & 0.3\\ 0.2 & 0 & 0.3\end{pmatrix},\quad \mathbf{D}=\begin{pmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}.$$

Для того, щоб задати об'єкт управління, вводимо матриці A,B,C,D як параметри у вікно блоку «State-space», який визивається при натисканні кнопки «Объект управления» головного меню програми.

Крок 2. Вводимо необхідні параметри для генерації навчальної вибірки, а саме:

1) Задаємо діапазон зміни входу від 0 до 1 та діапазон зміни виходу від 0 до 2.7.

2) Обираємо час дискретизації 0.1 с.

3) Обираємо інтервал від 0.1с до 5с.

4) Обираємо розмір навчальної вибірки 2000.

5) Натискаємо кнопку «Згенерувати».

Крок 3. Створюємо та навчаємо нейронну мережу. Для цього заповнюємо поля блоку «Нейронная сеть» головного меню програми, а саме:

1. У полі «Розмер скритого слоя» вказуємо кількість нейронів n=5.

2. У полі «Количество задержаних входов» вводимо значення i=0.

3. У полі «Количество задержаних выходов» вводимо значення j=1.

4. У полі «Алгоритм навчання» обираємо алгоритм навчання (trainlm) і кількість навчальних ітерацій (200). Нажимаємо кнопку «Обучить».

Крок 4. Після того, як нейронна мережа створена та навчена, підставляємо її у модель обчислення сумарної середньоквадратичної помилки. У блок «State-space» цієї моделі задаємо MIMO-об'єкт. В блок «Uniform Noice Generator» задаємо матрицю нижнього значення вхідного сигналу([0 0 0]) та матрицю верхнього значення вхідного сигналу ([1 1 1]). Кількість параметрів генератора залежить від обраного об'єкту управління. В даному випадку, оскільки маємо об'єкт управління з 3 вхідними параметрами, записуємо відповідно наведені вище параметри блоку «Uniform Noice Generator».

Крок 5. Повторюємо крок 2-3 і створюємо 108 нейронних мереж з різними параметрами затриманих входів та виходів, а також різною кількістю нейронів у прихованому шарі. Для цього ми варіюємо вводимі патаметри поля «Розмер скритого слоя» від 5 до 16 нейронів, поля «Количество задержаних входов» від 0 до 2 входів, а також поля «Количество задержаних выходов» від 1 до 3 виходів.

Крок 6. Після того, як всі нейронні мережі створені, навчені та підставлені у модель обчислення сумарної середньоквадратичної помилки, запускаємо модель та отримуємо значення середньоквадратичної помилку у блоці «Display»

Табл. 1. Результати дослідження нейромережевих моделей

i_j / n	0_1	0_2	0_3	1_1	1_2	1_3	2_1	2_2	2_3
5	0.593	0.6415	0.663	0.02072	0.03368	0.01907	0.007993	0.03166	0.934
6	0.6336	0.565	0.6564	0.012	0.01389	0.05195	0.01004	0.878	0.04302
7	0.6814	0.912	0.6008	0.007073	0.02914	0.0373	0.01277	0.789	0.00856
8	0.7195	0.4416	0.8858	0.02093	0.02749	0.0159	0.01474	0.3705	0.06729
9	0.7148	0.5647	0.7339	0.01271	0.01793	0.01482	0.01554	0.02248	0.927
10	0.6226	0.899	0.3688	0.009068	0.01324	0.04971	0.01829	0.7073	0.956
11	0.675	0.5014	0.841	0.03134	0.01825	0.03142	0.07851	0.913	0.00784
12	0.5785	0.50435	0.7661	0.007977	0.003114	0.02115	0.004742	0.03676	0.03685
13	0.6581	0.936	0.856	0.003391	0.01355	0.01059	0.0186	0.3052	0.03426
14	0.783	0.3992	0.887	0.009263	0.009352	0.01895	0.02771	0.002146	0.878
15	0.981	0.731	0.734	0.01303	0.01212	0.05857	0.02063	0.02022	0.02057
16	0.837	0.6442	0.558	0.001715	0.001401	0.003874	0.005804	0.1045	0.1761

З таблиці 1 видно, що оптимальною структурою для даної моделі з 3 вхідними та 4 вихідними параметрами є структура з 16 нейронами у прихованому шарі, 1 затриманим входом та 2 затриманими виходами. При цьому отримано значення середньоквадратичної помилки 0.001401. Також у роботі було досліджено різні багатовимірні об'єкти 2x2, 4x2, 4x3, 3x3. Для системи 2x2 оптимальною структурою виявилась структура з 5 нейронами у прихованому шарі,1затриманим входом та 3 затриманими виходами (5-1-3). Середньоквадратична помилка при цьому становила 0.004196. Для моделі 4x2 – структура (5-2-3) з помилкою 0.001372, моделі 4x3 – структура (16-1-2) з помилкою 0.001452, моделі 3x3 – структура (16-1-3) з помилкою 0.005575.

Висновок

У даній роботі розглядається комплексний формалізований підхід до реалізації багатоетапної процедури ідентифікації складних динамічних об'єктів з використанням нейромережевих модельних структур. Розроблено програмний пакет «MIMO-Plant» в середовищі MatLab, для оцінювання показників нейромережевих моделей MIMO-об'єктів управління, представлено його детальний опис та приклади дослідження.

Перелік посилань

1. Е.М. Воронов. Оптимизация многообъектных многокритериальных систем. − М.: Изд-во МГТУ, 2001. − 576 с.
2. Егупов Н.Д. Методы робастного, нейро-нечеткого и адаптивного управления. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 744 с.
3. Терехов В.А. Нейросетевые системы управления: Учеб.пособие для вузов. – М: Высшая школа. 2002. - 183с.
4. Cybenco G. Approximation by superposition’s of a sigmoidal function // Math. Of Control, Signal and Systems. 1989. № 2. P. 303 – 314.
5. Funahashi K. On the approximate realization of continuous mapping by neural networks. Neural Networks. 1989. № 2. P. 182 – 192.
6. Barron 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.
10. Кравець П.І., Шимкович В.М., Зубенко Г.А. Технологія апаратно-програмної реалізації штучного нейрона та штучних нейронних мереж засобами FPGA / Вісник НТУУ «КПИ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2012. – №55. 174-180с.
11. Hornik K., Stinchcombe M., White H. Multilayer feedforward networks are universal approximators // Neural Networks. 1989. № 2. P. 359 – 366.
12. Гаврилов А.И. Перспективы применения нейросетевых технологий в системах автоматического управления // Вестник МГТУ им. Н.Э. Баумана. Приборостроение. - 1998. -№1.-С. 119-126.

УДК 681.327.12.001.362

ДОРОГИЙ Я.Ю.

УСКОРЕННЫЙ АЛГОРИТМ ОБУЧЕНИЯ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

В статье рассмотрен вопрос построения нового алгоритма обучения для сверточных нейронных сетей. В ходе исследования предложено новое правило для вычисления ошибки нейрона.

In the article the question of designing of new training algorithm for the convolution neural networks. The study found the new rule for calculation of neuron's error.

Введение в проблему

Обучение нейронных сетей является одной из важных проблем при использовании их при решении различных проблем. Наличие качественного алгоритма обучения позволяет улучшить получаемые результаты, снизить ошибку построенной сети.

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 [1-3].

Данный тип сетей хорошо зарекомендовал себя для решения проблемы распознавания человека по фотопортрету, но задача разработки качественного алгоритма обучения для такого вида сетей является актуальной проблемой.

Цель работы

Целью данной работы является разработка ускоренного алгоритма обучения для сверточных нейронных сетей на базе классического алгоритма обратного распространения ошибки.

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.

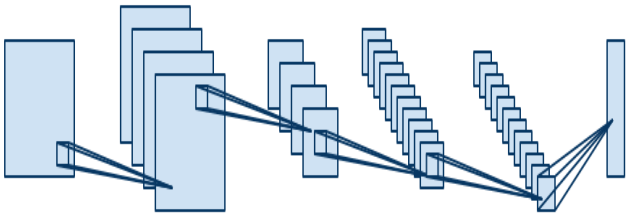


Рис. 1. Структура СНС

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.

Адаптация алгоритма обратного распространения ошибки для сверточных нейронных сетей

Рассмотрим алгоритм обратного распространения ошибки. В алгоритме можно выделить несколько основных шагов процесса обучения. Обычно именно эти шаги выделяют при практической реализации НС и данного алгоритма обучения.

Рассматриваемая НС имеет сигмоидальную активационную функцию (1).

$$f(u) = \frac{1}{1+e^{-u}}, \quad (1)$$

где

$$u = w_0 + w_1 x_1 + ... + w_n x_n \quad (2)$$

Введем следующие обозначения: множество учебных данных $X_s = \{x_i \mid i = 1,..., N\}$ и соответствующие им ожидаемые значения $Y_s = \{y_i \mid i = 1,..., M\}$, где N – число входов НС, M – число выходов НС, $s = 1,..,S$, а S – размерность обучающей выборки. Обозначим множество значений вывода нейронов НС через $F_i = \{f_j^i\}$, где i – номер слоя, а j – номер нейрона в слое. Для входного слоя $j = 1,2,...,N$, для первого скрытого слоя – $j = 1,2,...,C_1$, второго – $j = 1,2,...,C_2$ и т.д., для последнего скрытого слоя – $j = 1,2,...,M$. Общее число слоев НС равно К и число нейронов во всех скрытых слоях будет равно $C = \{C_0, C_1, ...C_K\}$, где $C_0 = N, C_K = M$.

Алгоритм обратного распространения ошибки можно представить следующим образом:

1) Для входного слоя установим значение каждого элемента, равное соответствующему элементу входного вектора. Значение вывода каждого элемента установим равным вводу.

$$F_0 = X_1 \quad (3)$$

2) Для элементов первого слоя вычисляем совокупный ввод и вывод:

$$u_j^1 = w_{j,0}^1 + \sum_{i=1}^{N} f_i^0 w_{j,i}^1, \quad (4)$$

где

$$f_j^1 = \frac{1}{1+e^{-u_j^1}} \quad (5)$$

где $j = 1,2,..., N_1$ – номер нейрона первого скрытого слоя, $w_{j,i}^l$ – i-й весовой коэффициент j-го нейрона l–слоя НС, f_i^0 – значение i-го элемента входного слоя.

3) Повторим первый шаг для всех скрытых слоев НС, включая выходной слой нейронов, причем формулы (4) и (5) немного изменятся, и будут иметь вид:

$$u_j^k = w_{j,0}^k + \sum_{i=1}^{C_k} f_i^{c-1} w_{j,i}^c, \quad (6)$$

где

$$f_j^c = \frac{1}{1+e^{-u_j^c}}, \quad (7)$$

где $c = 2,...,K$.

4) Сравниваем значения векторов Y_0 и F_K, если разница между вектором - образцом Y_0 и реальным выводом НС F_K находится в допустимом пределе, то переходим к шагу 5, если нет, то переходим к шагу 6.

5) Для входного вектора устанавливаем значение каждого элемента, равное соответствующему элементу $X_2 (F_0 = X_2)$ и переходим вновь к шагу 2. Если же на вход НС подан последний вектор X_S, то в зависимости от того, требовалось ли на этой эпохе обучение, либо вновь подаются на вход все обучающие векторы, начиная с первого, либо считается, что НС обучена и обучение заканчивается.

6) Для каждого нейрона выходного слоя вычисляется его ошибка. Поскольку рассматривается НС с сигмоидальной активационной функцией, то значение ошибки равно

$$\delta_j^K = f_j^K (1 - f_j^K)(y_j^0 - f_j^K), \quad (8)$$

где $j = 1,2,...,M$.

7) Для предпоследнего слоя вычисляется ошибка каждого нейрона

$$\delta_i^{K-1} = f_i^{K-1}(1 - f_i^{K-1}) \sum_{j=1}^{C_K} \delta_j^K w_{i,j}^{K-1}, \quad (9)$$

где δ_j^K – значение ошибки j-го элемента *K-го* слоя, $w_{i,j}^{K-1}$ – вес связи

между i-м элементом *K-1* слоя и j-м элементом *K*-го слоя, f_i^{K-1} – значение i-го элемента *K-1*-го слоя.

8) Для всех остальных скрытых слоев ошибка вычисляется по формуле (9).

9) Далее для всех слоев обновляются значения весовых коэффициентов каждого нейрона

$$\Delta w_{i,j}^c (t+1) = \eta(\delta_i^c f_j^{c-1}) + \alpha \Delta w_{i,j}^c (t), \quad (10)$$

где η – скорость обучения, α – инерция, или влияние значения предыдущего изменения, обычно берется значение $\alpha < 1$, t – номер итерации. И тогда устанавливаются новые значения весовых коэффициентов, равные

$$w_{i,j}^c = w_{i,j}^c + \Delta w_{i,j}^c (t). \quad (11)$$

10) Для входного вектора устанавливается значение каждого нейрона, равное соответствующему элементу $X_2 (F_0 = X_2)$ и выполняется переход вновь на шаг 2. Если уже подан последний вектор X_S на вход НС, то вновь подается первый обучающий вектор.

Замечание: Если НС является сверточной (СНС), то необходимо учесть факт совместного использования весов множеством межнейронных связей. Такие связи используются в СНС между парами слоев свертки-субдискретизации, а также внутри этих пар. Для учета этих особенностей необходимо привести формулу расчета весов (10) к виду (12):

$$\Delta w_{i,j}^c (t+1) = \frac{\eta(\delta_i^c f_j^{c-1})}{N_w} + \alpha \Delta w_{i,j}^c (t), \quad (12)$$

где N_w- количество связей, что совместно используют весовой коэффициент w.

Модификация алгоритма обучения

Теперь рассмотрим значения ошибки для нейронов последнего слоя, вычисляемые на шаге 6 алгоритма обратного распространения ошибки в соответствии с формулой (8). Зависимость значения ошибки нейрона δ_j^{K-1} последнего слоя от выходного значения этого нейрона f_j^{K-1} для различных ожидаемых значений, представлена на рис. 1-3. Необходимо отметить, что форма графика будет меняться при различных ожидаемых значениях y_j^0.

В соответствии с формулой (8), значение y_j^0 может находиться в диапазоне от 0 до 1, так как сигмоидальная функция определена именно на этом диапазоне. Рассмотрим три графика для значений $y_j^0 = 1$ (рис.1), $y_j^0 = 0$ (рис.2) и $y_j^0 = 0.5$ (рис.3).

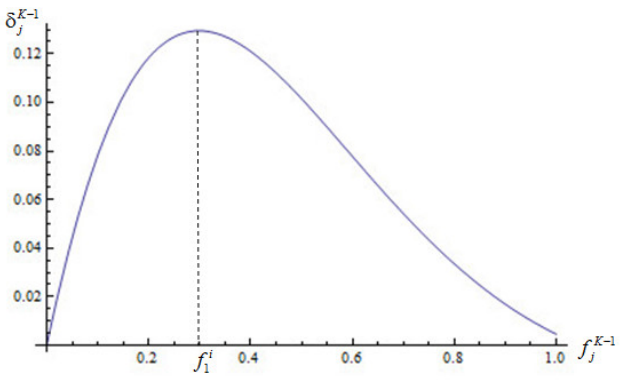


Рис.1. График ошибки при $y_j^0 = 1$

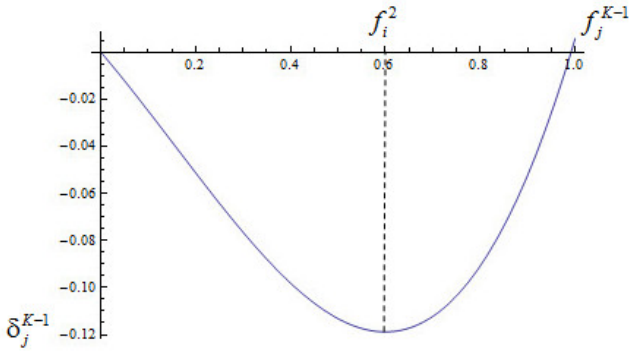


Рис.2. График ошибки при $y_j^0 = 0$

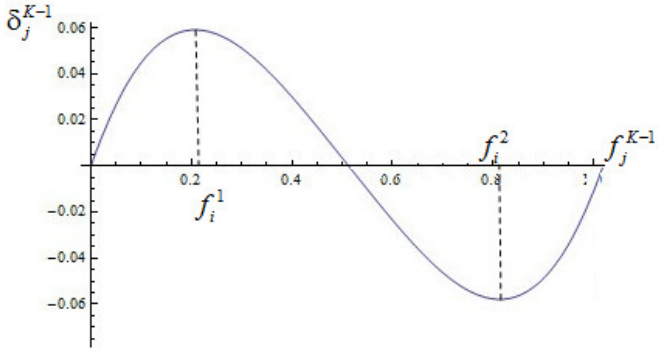


Рис.3. График ошибки при $y_j^0 = 0.5$

По графикам видно, что значение ошибки стремится к 0 в трех случаях, когда:

- значение f_j^0, полученное от НС, близко к ожидаемому y_j^0;
- значение f_j^0, полученное от НС, близко к 0;
- значение f_j^0, полученное от НС, близко к 1.

Это значит, что в случаях, когда значение f_j^0 приближается к 0 или 1, а не к ожидаемому y_j^0, процесс обучения останавливается и изменений весовых коэффициентов НС уже не происходит или происходят, но весьма незначительные. Таким образом, получается, что одна итерация обучения оказывается «потерянной» и никак не влияет на общий ход обучения.

Если вернуться к логике функционирования НС, то этот эффект в изменении значения ошибки можно объяснить. Значение ошибки, стремящееся к 0 в граничных точках графика, объясняет ее возможность обучаться даже на зашумленных данных, когда по ошибке некоторые учебные образы могли быть отнесены к неверному классу. Но необходимо помнить, что для сигмоидальной функции выполняется условие:

$$\frac{1}{1+e^{-u}} \to 0 \text{, при } u \to -\infty \qquad (13)$$

и в то же время

$$\frac{1}{1+e^{-u}} \to 0 \text{, при } u \to +\infty \qquad (14)$$

Из этого следует, что значение ошибки может быть сколь угодно малым, но не станет равным 0. Следовательно, остается возможность «доучить» НС и на этот учебный образ. Однако чрезвычайно малые значения, которые может принимать ошибка в этих точках, означают, что на обучение может потребоваться значительное время, а в зависимости от размеров НС это могут быть часы или даже дни.

Для исключения подобной ситуации предлагается произвести модернизацию алгоритма, ускорив процесс обучения. Для этого изменим функцию (8) определения ошибки нейронов последнего слоя. Теперь значение ошибки δ_i будет стремиться к 0 только в том случае, когда полученное на выходе нейрона значение будет близко к ожидаемому.

Точки локальных экстремумов на диапазоне [0,1] можно найти как значения первой производной по функции (8) на этом диапазоне. Это будут точки:

$$f_i^{'1} = \frac{1}{3} + \frac{1}{3}y_i - \frac{1}{3}\sqrt{1 - y_i + y_i^2}, \qquad (15)$$

$$f_i^{'2} = \frac{1}{3} + \frac{1}{3}y_i + \frac{1}{3}\sqrt{1 - y_i + y_i^2}, \qquad (16)$$

где y_i – ожидаемое значение от i-го нейрона последнего слоя.

Теперь формула (8), применяемая на шестом шаге обучения НС, для вычисления ошибки нейрона изменит свой вид и примет следующую форму:

$$\delta_i = \begin{cases} f_i^{'1}(1 - f_i^{'1})(y_i - f_i^{'1}) \, при \, f_i < f_i^{'1} \\ f_i(1 - f_i)(y_i - f_i) \, при \, f_i^{'2} < f_i < f_i^{'1} \\ f_i^{'2}(1 - f_i^{'2})(y_i - f_i^{'2}) \, при \, f_i > f_i^{'2} \end{cases}, \qquad (17)$$

где f_i – полученное значение j-го элемента выходного слоя; $f_i^{'1}, f_i^{'2}$ – точки локальных экстремумов для ожидаемого значения y_i на диапазоне [0,1].

Эксперименты и результаты

Проведенные эксперименты показали, что при использовании данного модифицированного алгоритма время, необходимое для обучения НС, уменьшается на 24%. Это связано с тем, что количество выполняемых элементарных операций данного алгоритма увеличивается на 17 для каждого исходящего нейрона, а число эпох обучения, необходимых для тренировки сети, сокращается на 32%.

Выводы

В ходе исследования был разработан новый алгоритм обучения для сверточных нейронных сетей. Также, проведен ряд экспериментов для определения работоспособности предложенного алгоритма и получено значительное повышение скорости обучения нейронных сетей.

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. Дорогий Я.Ю. Модифицированный алгоритм обучения сверточных нейронных сетей. //Сборник материалов. VII Международная научно-практическая конференция “перспективы развития информационных технологий”. 18 апреля 2012, г. Новосибирск: Издательство НГТУ. – с 34-38.

.

УДК 004.91

ПОЛТОРАК В.П.,
ГОЛКОВ В.Б.

ПІДВИЩЕННЯ ШВИДКОДІЇ ПРОЦЕДУРИ МНОЖЕННЯ ТОЧКИ НА ЧИСЛО У АЛГОРИТМАХ ЕЛЕКТРОННОГО ЦИФРОВОГО ПІДПИСУ НА ЕЛІПТИЧНИХ КРИВИХ

Робота присвячена підвищенню швидкості постановки та верифікації електронного цифрового підпису на еліптичних кривих за рахунок використання запропонованого удосконаленого алгоритму виконання операцій над точками еліптичних кривих. Проведено огляд та аналіз відомих алгоритмів електронного цифрового підпису, розглядаються питання реалізації алгоритму електронного цифрового підпису на еліптичних кривих.

The paper is devoted to the increasing the speed of creation and verification of elliptic curve digital signature by using the proposed improved algorithm of operations on elliptic curve points. Known algorithms of digital signature are reviewed and analyzed, the questions of implementation of elliptic curve digital signature algorithm are examined.

1. Вступ

Розвиток глобальних комунікацій в діловому і повсякденному житті привів до появи нової області взаємовідносин, предметом яких є електронний обмін даними.

Проблема збереження електронних документів від копіювання, модифікації і підробки вимагає для свого вирішення специфічних засобів і методів захисту. Одним з поширених в світі засобів такого захисту є Електронний цифровий підпис (ЕЦП), який забезпечує автентичність повідомлення та неспростовність застосування особистого ключа (автентифікація власника цифрового підпису)[1]. За допомогою спеціального програмного забезпечення ЕЦП підтверджує достовірність інформації документу, його реквізитів і факту підписання конкретною особою.

У даній роботі розглядаються методи створення ЕЦП та питання їх реалізації. ЕЦП являє собою реквізит електронного документу, що дозволяє встановити відсутність спотворення інформації в електронному документі з моменту формування ЕЦП і перевірити належність підпису власнику сертифіката ключа ЕЦП.

Цифровий підпис провадить наступні функції [2]:

- автентифікація: для отримувача повідомлення повинна бути можливість переконатися в його достовірності;
- неможливість відмови від відправлення: відправник не повинен мати можливості заперечити відправлення та підпис повідомлення протягом деякого часу після відправлення;
- цілісність: отримувач повідомлення повинен мати можливість перевірити той факт, що повідомлення не було змінене при передачі.

Цифровий підпис повинен гарантувати наступні ознаки:

- немає можливості підробки підпису;
- підпис може бути перевіреним, засвідченим;
- після того, як повідомлення підписане, відправник не зможе відмовитися від факту його передачі.

2. Актуальність проблеми

На сьогоднішній день ЕЦП широко використовується юридичними та фізичними особами, активно впроваджується в державних установах і органах державної влади. У системах документообігу використовуються різні алгоритми ЕЦП, і незважаючи на те, що загалом всі алгоритми виконують свої функції, між ними існує значна різниця, у кожного алгоритму є свої переваги та недоліки, які потребують дослідження.

До основних характеристик систем електронного цифрового підпису можна віднести довжину ключа, необхідну для забезпечення

заданої криптостійкості, та швидкість створення та верифікації ЕЦП. Віддається перевага алгоритмам, які забезпечують задану криптостійкість використовуючи ключі меншої довжини, так як це зменшує кількість даних, які потребують обробки, зберігання та передачі по каналам зв'язку. З поширенням технологій ЕЦП на мобільні пристрої, смарт-карти та інші пристрої з жорсткими обмеженнями обчислювальної спроможності, об'єму пам'яті та використовуваного трафіку, дані характеристики стають вирішальними при виборі алгоритму ЕЦП.

Всі алгоритми ЕЦП виконують операції над великими числами від 2^{160} до 2^{4096} і більше, в залежності від математичного апарату, який використовується в кожному конкретному алгоритмі. У системах ЕЦП дані обчислення займають майже весь процесорний час, отже витрати на апаратні складові таких систем практично повністю залежать від швидкодії обраного алгоритму.

У даній роботі проаналізовані відомі алгоритми ЕЦП, обрано алгоритм, що забезпечує достатню криптостійкість при найменшій довжині ключа у порівнянні з іншими алгоритмами, так як менша довжина ключа потребує виконання операцій над числами меншого розміру, що підвищує швидкодію. Також запропоновано удосконалений алгоритм виконання математичних операцій над точками еліптичних кривих, який значно зменшує кількість необхідних обчислень, а також його модифікований варіант з паралельними обчисленнями, здатний виконувати операції швидше при використанні в багатоядерних та багатопроцесорних обчислювальних системах.

3. Огляд існуючих рішень

Ця робота зосереджена на дослідженні асиметричних схем цифрового підпису з доповненням (appendix) [2]. «Доповнення» означає, що використовується криптографічна хеш-функція для створення дайджесту повідомлення, і перетворення підпису повідомлення відбувається над дайджестом повідомлення, а не над власне повідомленням.

Сучасні схеми цифрового підпису можуть бути класифіковані таким чином відповідно до складної математичної проблеми, що лежить в їх основі, та забезпечує їх безпеку [3]:

1) Схеми факторизації цілих чисел (ФЦ), безпека яких залежить від складності розв'язку проблеми розкладу на множники (факторизації) цілих чисел. Прикладами цих схем є схеми підпису RSA та Рабіна.

2) Схеми дискретного логарифму (ДЛ), безпека яких залежить від складності розв'язку проблеми знаходження (звичайного) дискретного логарифму у скінченному полі. Прикладами є схеми підпису Ель-Гамаля, Шнорра, DSA.

3) Схеми на еліптичних кривих (ЕК), безпека яких залежить від складності розв'язку проблеми знаходження дискретного логарифму на ЕК. Прикладами є схеми підпису ECKDSA, ECSS, ECDSA.

Схема цифрового підпису RSA. Для формування електронного підпису відправник виконує над контрольною сумою документу h ті ж самі дії, що і при шифруванні, але використовує не відкритий ключ одержувача, а свій власний закритий ключ, тобто $sign_i = (h_i^d \bmod n)$. Відкритий і закритий ключі просто міняються місцями. На приймальній стороні одержувач зводить підпис у степінь відкритого ключа e відправника і отримує $(sign_i^e \bmod n) = (h_i^{de} \bmod n) = h_i$ (згідно з тими ж формулами, що і в асиметричному шифруванні RSA).

Якщо після зведення в степінь значення співпадає з обчисленою незалежно на приймальній стороні контрольною сумою документу, то перевірка вважається виконаною, а документ - справжнім. Ніхто, крім відправника, не знаючи d, не може обчислити такий підпис $sign_i$, щоб зведення її до рівня відкритого ключа e дало необхідну контрольну суму - це те ж саме важковирішуване завдання, що і в асиметричному шифруванні RSA. Отже, забезпечити документ таким підписом $sign_i$ міг тільки справжній власник закритого ключа. Схема ЕЦП наведена на рис. 1.

Схема цифрового підпису DSA. Згідно з FIPS Pub 186–2 (Federal Information Processing Standards (FIPS), 2000), DSA використовує наступні параметри:

1) p – простий модуль, де $2^{L-1} < p < 2^L$ для L=1024;

2) q – простий дільник p - 1, де $2^{159} < q < 2^{160}$;

3) $g = h^{(p-1)/q} \bmod p$, де h будь-яке ціле число в межах $1 < h < p-1$ таке що $h^{(p-1)/q} \bmod p > 1$ (g має порядок q mod p);

4) x = a випадково або псевдо випадково згенероване ціле число в межах $0 < x < q$;

5) $y = g^x \bmod p$;

6) k = а випадково або псевдо випадково згенероване ціле число в межах $0 < k < q$;

Цілі числа *p, q,* та *g* можуть бути відкриті та загальновідомі групі користувачів. Закритий та відкритий ключі користувача - це *x* і *y,* відповідно. Параметри *x* та *k* використовуються лише для генерації підпису, та повинні триматися в таємниці. Параметр *k* повинен змінюватися для кожного підпису.

Підписом для повідомлення *m* буде пара чисел *r* та *s* обчислених згідно з рівняннями: $r=(g^k \bmod p) \bmod q$, та $s=(k^{-1}(\mathrm{SHA}{-}1(m)+x\,r)) \bmod q$.

Вище згадане k^{-1} є мультиплікативною інверсією *k,* mod *q,* тобто, $(k^{-1} * k) \bmod q = 1$ та $0 < k^{-1} < q$. Значенням SHA−1(m) є 160-бітовий рядок, що є виходом алгоритму Secure Hash Algorithm, визначеного у стандарті FIPS 180-1, та повинен бути перетворений у ціле число. Підпис разом з повідомленням передається отримувачу.

До процедури верифікації підпису необхідно отримати доступ до *p, q* та *g*. Як тільки відправник підписав повідомлення, він відправляє повідомлення, *m,* разом з цифровим підписом *r* та *s*.

Нехай *m', r',* та *s'* – це отримані версії *m, r,* та *s,* відповідно, та нехай *y* буде відкритим ключем відправника. Щоб перевірити підпис відправника, отримувач спершу перевіряє чи лежать в заданих межах $0 < r' < q$ та $0 < s' < q$. Якщо ця умова порушена, то підпис відкидається. Якщо ці дві умови виконуються, тоді перевірник обчислює:

$$w = (s')^{-1} \bmod q$$

$$u_1 = ((SHA-1(m')w) \bmod q$$

$$u_2 = ((r')w) \bmod q$$

$$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$$

Якщо $v=r'$, тоді підпис є вірним та перевірник може бути певним, що це повідомлення дійсно було відправлене особою, що має закритий ключ, *x,* що відповідає відкритому ключу *y*.

Якщо *v* не дорівнює *r',* тоді повідомлення могло бути змінене, або невірно підписане Алісою, або повідомлення могло бути підписане зловмисником, та, таким чином, повідомлення повинне бути визнане недійсним [2].

Схема цифрового підпису ECDSA. ECDSA є одним з варіантів схеми підпису Ель-Гамаля. ECDSA з'явився як модифікація алгоритму DSA, з метою використання еліптичних кривих в якості основи обчислень. Згодом алгоритм знайшов визнання і на цей час використовується у декількох стандартах [3].

Розглянемо базову схему алгоритму:

1) Загальновідомі параметри: в алгоритмі можливе використання скінченних полів двох видів: поля характеристики два або великого простого поля. Надалі розглянемо в якості базового велике просте поле. Відкриті параметри складаються з поля порядку *p*, еліптичної кривої визначеної двома елементами поля *a* та *b*, простого числа *n* більшого ніж 2^{160} , та елемента *G* порядку *n*, що належить еліптичній

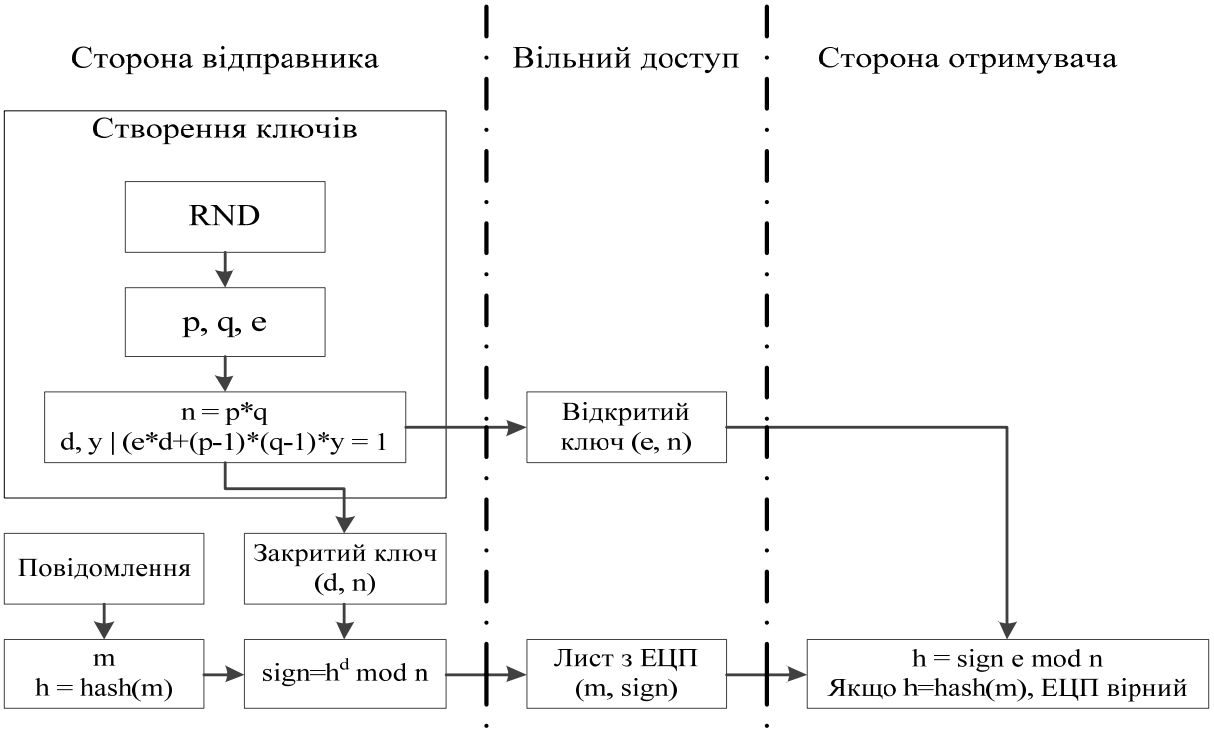


Рис. 1. Схема ЕЦП RSA

кривій. Рівняння еліптичної кривої має вигляд $y^2 = x^3 + ax + b$. Відкриті параметри є об'єктом для багатьох критеріїв безпеки.

2) Підготовка: обрати випадкове ціле число d в межах [1, n - 1], обчислити $Q = dG$. На виході отримаємо $(K_p, K_s) = (Q, d)$.

3) Генерація підпису:обрати випадкове ціле число k в межах [1, n - 1] та обчислити:

$$(x_1, y_1) = kG;$$

$$r = \bar{x}_1 \bmod n;$$

$$s = \frac{H(M) + ar}{k} \bmod n.$$

Тут $\bar{x}_1$ є певним чином перетворений в ціле число елемент скінченного поля x_1 . Якщо $r = 0$ або $s = 0$, знову переходимо на крок вибору k. Виходом процедури генерації підпису буде пара $\sigma = (r, s)$

4) Верифікація підпису: перевірити, що $Q \neq O$, $Q \in C$, та $nQ = O$. Перевірити, що r та s лежать в межах [1, n - 1] і $r = x_1 \bmod n$ для $(x_1, y_1) = u_1 G + u_2 Q$, $u_1 = \frac{H(M)}{s} \bmod n$, та $u_2 = \frac{r}{s} \bmod n$.

Складність алгоритму ECDSA полягає в необхідності роботи з об'єктами різних типів: точки еліптичної кривої, елементи поля, цілі числа, та рядки бітів. В стандартах, що використовують даний алгоритм, наведений докладний опис про способи представлення цих об'єктів та способи маніпуляції ними.

Самою складною операцією даного алгоритму є операція множення точки на скаляр, практично весь час обчислень витрачається саме на неї, тому у даній роботі зроблено акцент саме на підвищення швидкості даної операції. Розроблено алгоритм множення точки еліптичної кривої на скаляр зі значно зменшеною кількістю необхідних математичних операцій, та з можливістю виконання обчислень паралельно. Запропонований алгоритм описано нижче.

4. Порівняльний аналіз криптостійкості алгоритмів підпису

З трьох проблем, що використовуються в асиметричних схемах підпису, проблема факторизації цілих чисел та дискретного логарифму можуть бути вирішені алгоритмами, які виконуються за субекспоненційний час. Це означає, що проблема визнана досить важкою, але не настільки важкою як ті проблеми, що вирішуються алгоритмами лише за повністю експоненційний час. З іншого боку, найкращий алгоритм загального спрямування для проблеми дискретного логарифму в групі точок еліптичної кривої (ПДЛЕК) виконується за повністю експоненційний час. Це означає, що ПДЛЕК наразі визнана більш складною ніж проблема факторизації цілих чисел або проблема дискретного логарифму (ПДЛ).

Маючи на увазі цю властивість криптосистем на еліптичних кривих, висока безпека при відносно невеликому розмірі ключа, дослідимо властивості систем на еліптичних кривих порівняно з іншими криптосистемами ЕЦП.

Проблема факторизації цілих чисел (ПФЧ) полягає в наступному: маючи складене число n, що є добутком двох великих простих цілих чисел p та q, знайти p та q. В той час як знаходження двох великих простих чисел є досить простим завданням, проблема розкладання добутку двох таких чисел на множники визнана обчислювально складною, якщо прості числа обрані належним чином. Схема ЕЦП RSA базується на цій проблемі.

Існує два основних типи алгоритмів факторизації [4]: загального спрямування та спеціалізованого спрямування. Алгоритми факторизації спеціалізованого спрямування намагаються використати спеціальні ознаки числа n, яке розкладається. На відміну від них, час виконання алгоритмів факторизації загального спрямування залежить лише від розміру числа n.

Одним з найбільш потужних алгоритмів факторизації спеціалізованого спрямування є метод факторизації з використанням еліптичних кривих (ECM), що був винайдений в 1985 році Хендріком Ленстрою. Час виконання цього алгоритму залежить від розміру простих множників числа n, тому в першу чергу алгоритм намагається знайти малі множники. Найбільшим простим множником знайденим на цей час за допомогою ECM є множник з 54 цифр (180 біт) числа з 127 цифр (422 біта) [4].

До розробки криптосистеми RSA, кращим алгоритмом факторизації загального спрямування був алгоритм неперервних дробів, який міг розкладати на множники числа до 40 десяткових цифр (133 біта). Цей алгоритм заснований на ідеї використання базового множника простого числа та генерації відповідної множини лінійних рівнянь, вирішення яких

обов'язково приведе до розкладання на множники. Ця ідея лежить в основі і кращих алгоритмів, що використовуються на даний момент: квадратичне решето (QS) та загальне решето числового поля (NFS). Обидва алгоритми легко можуть виконувати факторизацію паралельно у розподілених мережах робочих станцій.

Квадратичне решето було використане, щоб розкласти на множники число з 129 десяткових цифр (429 біт), що було числом-викликом RSA поставленим в 1977 році.

Загальне решето числового поля, розроблене в 1989 році, найкраще працює на числах спеціальної форми (512+1). Згодом воно було розширене до алгоритму факторизації загального спрямування. Недавні експерименти довели, що NFS є дійсно потужним алгоритмом для розкладання на множники цілих чисел, що мають щонайменше 130 цифр (432 біта) [4].

Проблема дискретного логарифму в простому полі. Якщо p є простим числом, тоді Z_p визначає множину чисел {0, 1, 2, …, p-1}, де додавання та множення виконуються по модулю p.

Проблема дискретного логарифмування в скінченному полі (ПДЛ) полягає в наступному: маючи просте число p, генератор α Z_p, та ненульовий елемент $\beta \in Z_p$, знайти унікальне ціле число $l, 0 \le l \le p-2$, таке що $\beta \equiv \alpha^l \pmod p$. Ціле число l називається дискретним логарифмом β за базою α [4].

Базуючись на складності цієї проблеми, Діффі та Хеллман запропонували відому схему обміну ключами в 1976 році. З того часу багато інших криптографічних протоколів, чия стійкість залежить від ПДЛ, були запропоновані, включаючи DSA, схему Ель-Гамаля та схему Шнорра. Через інтерес до цих застосувань ПДЛ інтенсивно вивчалась математиками останні 20 років.

Як і з проблемою факторизації цілих чисел, існують два типи алгоритмів для вирішення проблеми дискретного логарифму. Алгоритми факторизації спеціалізованого спрямування намагаються використати спеціальні ознаки числа p, яке розкладається. На відміну від них, час виконання алгоритмів факторизації загального спрямування залежить лише від розміру числа p.

Найшвидші відомі алгоритми загального спрямування для вирішення ПДЛ засновані на методі, що називається індексне або диференційне обчислення. В цьому методі, створюється база даних з малих простих чисел та відповідних їм логарифмів, згідно з якою можуть бути отримані логарифми випадкових елементів поля. Це нагадує методи з базовим множником для проблеми факторизації. Тому якщо відбудеться прогрес в алгоритмі вирішення однієї з проблем, скоро після цього схожий покращений алгоритм буде знайдено і для іншої проблеми.

Найкращим відомим алгоритмом для вирішення ПДЛ на даний момент є загальне решето числового поля. Він має такий ж асимптотичний час виконання, що й відповідний алгоритм для факторизації цілих чисел. З цього можна зробити висновок, що проблема знаходження логарифмів у випадку простого числа p розміром k біт має приблизно таку ж складність, що й розкладання складеного числа n розміром k біт. В 1998 році було оголошений розв'язок Виклику Діффі-Хеллмана, який включав вирішення ПДЛ по модулю простого числа з 129 цифр. Використане просте число мало спеціальну форму, тому був використаний алгоритм загального решета числового поля, який був дуже ефективним для цього виклику. Ці результати є доказами довгострокової безпеки, в криптосистемах, що використовують ПДЛ, слід використовувати модуль p розміром 1024 біта або більше [4].

Проблема дискретного логарифму групі точок еліптичної кривої, визначеної над простим полем. Якщо p є простим ступенем, то F_p визначає скінченне поле, що містить p елементів. В застосуваннях p зазвичай є ступенем 2 (2^m) або непарним простим числом (p).

Проблема дискретного логарифму в групі точок еліптичної кривої (ПДЛЕК) полягає в наступному: маючи еліптичну криву E визначену над полем F_p, точку $G \in E(F_p)$ порядку n, і точку $Q \in E(F_p)$, необхідно визначити ціле число $l, 0 \le l \le n-1$, таке що $Q = lG$, за умови, що таке ціле число існує.

Базуючись на складності цієї проблеми, було запропоновано використання групи точок еліптичної кривої визначеної над скінченним полем для реалізації різних криптосистем.

Один з таких протоколів є аналог DSA на еліптичних кривих, алгоритм ECDSA. Таким чином, ПДЛЕК може розглядатися як схожа на проблему ПДЛ, але в іншому алгебричному середовищі.

З 1985 року ПДЛЕК привернула значну увагу провідних математиків з усього світу. Алгоритм Полліга та Хеллмана [3,5] скоротив визначення l до визначення кожного з множників n по модулю l. Тобто, щоб досягнути максимального рівня безпеки, необхідно щоб n було простим числом. Кращим алгоритмом загального спрямування для вирішення ПДЛЕК на даний момент є метод Полларда, який з прискоренням запропонованим Галлантом, Ламбертом та Венстоном, має близько $\sqrt{\pi n}/2$ кроків, де під кроком маємо на увазі операцію додавання в групі точок еліптичної кривої.

Найважливішим є те, що для ПДЛЕК не відомо жодного алгоритму типу індексного обчислення на відміну від ПДЛ. Тому вирішення ПДЛЕК вважається складнішим ніж проблеми факторизації цілих чисел або ПДЛ, оскільки невідомо алгоритму загального спрямування з субекспоненційним часом виконання.

В 1991 році Менезес, Окамото та Венстон (MOV) [4] показали, як ПДЛЕК може бути скорочено до ПДЛ в розширенні полів F_p, де можна використати методи індексного обчислення. Тим не менш, цей алгоритм скорочення MOV ефективний тільки для дуже особливого класу кривих, відомих як суперсингулярні криві. Крім того, існує простий тест, за допомогою якого можна перевірити, що окрема еліптична крива не є вразливою до цієї атаки. Суперсингулярні криві є спеціально забороненими в усіх стандартах з використанням систем на ЕК, таких як IEEE P1363, ANSI X9.62, та ANSI X9.63.

Іншим так званим слабким класом еліптичних кривих є так звані аномальні криві – це криві E визначені на полем F_p, котре має кількість точок, що дорівнює точно p. Атака на ці криві була винайдена незалежно Семаєвим, Смартом та Сато і Аракі, та узагальнена Рюком. Як і у випадку з суперсингулярними кривими, існує простий тест, за допомогою якого можна перевірити, що окрема еліптична крива не є вразливою до цієї атаки.

Дослідження ПДЛЕК та пов'язаних з нею проблем та пошук нових рішень продовжується і на цей час. Необхідно відзначити, що в вище описаних програмних та апаратних атаках, обчислення єдиного дискретного логарифму в групі точок еліптичної кривої має ефект розкриття одного закритого ключа користувача. Ті самі зусилля необхідні для визначення другого закритого ключа користувача.

На графіку на рис. 2 порівнюється час, потрібний для злому ECDSA з часом необхідним для злому RSA або DSA для різних розмірів ключа з використанням найкращих відомих алгоритмів [5]. Значення обчислені в MIPS роках. MIPS рік представляє час обчислення, що дорівнює 1 року на машині спроможній виконувати 1 мільйон інструкцій в секунду. Для порівняльного тесту, загальноприйнятим є те, що 10^{12} MIPS років представляє значну безпеку на даний момент, оскільки таке значення потребує, щоб більшість обчислювальної потужності на планеті працювала значний проміжок часу. Щоб досягти такого рівня безпеки RSA та DSA необхідно використовувати ключ розміром 1024 біт, в той час як для алгоритму ECDSA достатньо ключа розміром 160 біт.

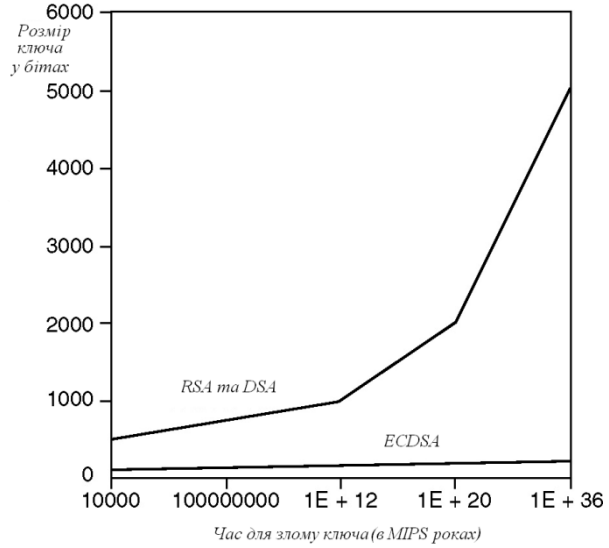


Рис. 2. Порівняння рівнів безпеки криптосистем ЕЦП

Зауважимо, що різниця між системами зростає по мірі збільшення розміру ключа. Наприклад, помітимо, як значення обсягу обчислень зростає для ECDSA з ключем розміром 300 біт порівняно з RSA та DSA з ключами розміром 2048 біт.

Порівняння трьох складних математичних проблем, на яких базуються відомі асиметри-

чні криптосистеми підпису виявило той факт, що жодна з них не є доказово надійною. Роки інтенсивних досліджень призвели до загальноприйнятого переконання, що ПДЛЕК значно складніша ніж проблема факторизації цілих чисел та ПДЛ. Загальним висновком є те, що ПДЛЕК потребує повністю експоненційного часу для вирішення.

Розглянувши основні схеми постановки електронного цифрового підпису та аналізу їх переваг та недоліків, отримані результати можна звести до наступної табл. 1 для значення 10^{11} MIPS років необхідних для злому закритого ключа.

Табл. 1. Порівняння сучасних алгоритмів цифрового підпису

Ознака порівняння	**Алгоритм ЕЦП**		
	RSA	DSA	ECDSA
Проблема що лежить в основі алгоритму	ПФЧ	ПДЛ	ПДЛЕК
Розмір системних параметрів в бітах	1024	2208	481
Розмір отриманого підпису в бітах	1024	320	320
Розмір відкритого ключа в бітах	1088	1024	161
Розмір закритого ключа в бітах	2048	160	160

5. Підвищення швидкодії процедури множення точки на число у алгоритмах ЕЦП на еліптичних кривих

Запропонований алгоритм підвищення швидкодії процедури множення точки на число, яка лежить в основі алгоритму ECDSA, був досліджений на спеціально розробленому макеті системи ЕЦП, що реалізує алгоритм ECDSA, так як даний алгоритм має ряд переваг, описаних вище, у порівнянні з іншими алгоритмами ЕЦП.

До розроблюваної системи ставились наступні вимоги:

- Можливість встановити порядок p еліптичної кривої у діапазоні $[3; 2^{521}]$, що надає гнучкі засоби для вивчення та дослідження алгоритму;
- Можливість використання так званих аномальних та суперсингулярних кривих для дослідження їх уразливості;
- Побудова графіку еліптичної кривої невеликого порядку, наочне виконання операцій над точками побудованої еліптичної кривої;
- Поетапне відображення процедур постановки та верифікації ЕЦП;
- Можливість підстановки зміненого документу та його підпису на етапі верифікації підпису у цілях дослідження.

На рис. 3 представлена блок-схема реалізованого алгоритму постановки ЕЦП на основі алгоритму ECDSA.

У блоках 3 і 6 блок-схеми, представленої на рис. 3, виконується операція множення точки еліптичної кривої на число. Дана операція потребує значної кількості обчислень, так як в математичному апараті еліптичних кривих над скінченим полем не існує операції множення точки на число, більше 2, тому необхідне багаторазове виконання операції додавання точок. При такому підході у разі виконання операції множення точки P на число k, необхідна одна операція знаходження точки $2P$ та k - 2 операцій додавання, які потребують великої кількості часу при числа k порядку 2^{160} і більше.

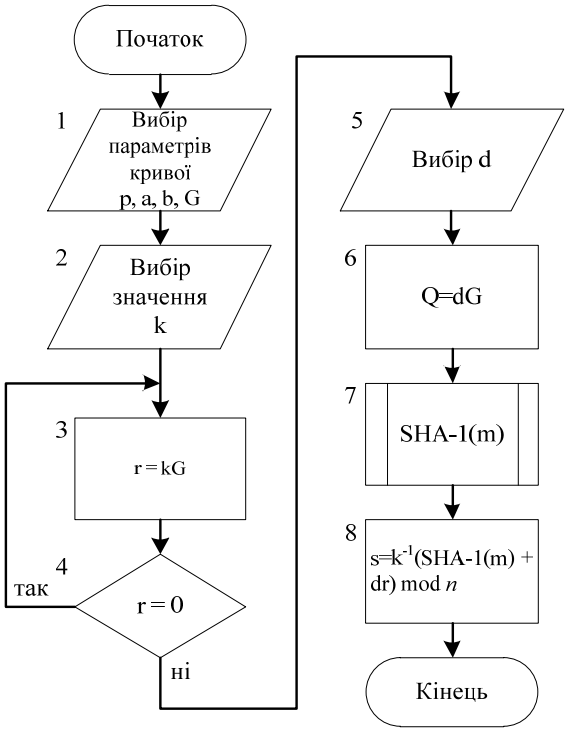


Рис. 3. Блок-схема реалізованого алгоритму постановки ЕЦП

На заміну даного алгоритму множення точки на число було використано наступний алгоритм. Для виконання операції множення точки P на число k послідовно знаходяться точки $2P$, $2(2P) = 4P$, $2(4P) = 8P$, …, 2^iP,…,

$2(2^{n-1}P) = 2^nP$, де n – кількість розрядів числа k у двійковому представленні. Точка kP знаходиться як сума тих точок 2^iP, для яких i-й розряд числа k у двійковому представленні рівний 1. Нехай двійкове представлення числа k містить r одиниць, тоді для виконання операції множення точки на число необхідно r - 1 операцій додавання та n - 1 операція множення точки на 2. У найгіршому для швидкодії випадку, коли r = n, необхідні n - 1 операція додавання та n - 1 операція множення точки на 2. Так як число n не перевищує $\log_2 k + 1$, досягається висока швидкість виконання операції множення точки на число. Блок-схема алгоритму множення точки на число представлена на рис. 4.

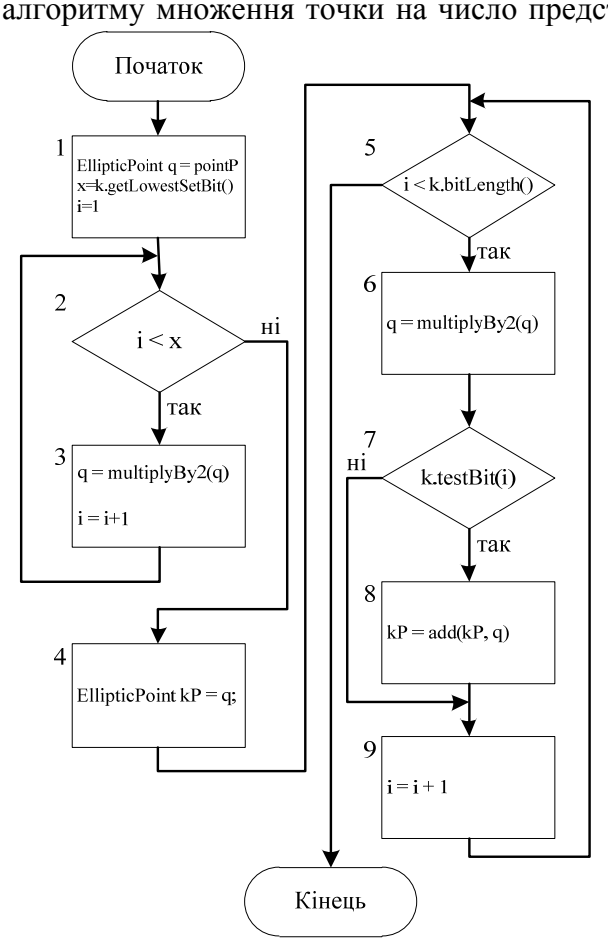


Рис. 4. Блок-схема алгоритму множення точки еліптичної кривої на число

Алгоритм, блок-схема якого представлена на рис. 4, послідовно виконує дві основні задачі на кожній ітерації: знаходження 2^iP шляхом виконання операції множення результату попередньої ітерації на 2, та додавання точки 2^iP до загального результату, якщо вона до нього входить. Таким чином, можна виділити дві окремі підзадачі, що дає можливість створити модифікований алгоритм множення точки на скаляр з паралельними обчисленнями. Було створено алгоритм, у якому наявні два потоки обчислень, що виконуються паралельно. Перший потік приймає на вхід задану точку еліптичної кривої P, та послідовно виконує операцію множення точки на 2. Всі обчислені 2^iP, для яких i-й розряд числа k у двійковому представленні рівний 1, додаються у чергу точок, які необхідно додати для отримання результату. Другий потік отримує виконує додавання всіх точок, які поступають у чергу від першого потоку, та видаляє з черги вже додані точки, та додає до результату точку P, якщо молодший розряд числа k рівний 1. Так як операції множення точки еліптичної кривої на 2 та додавання двох точок еліптичної кривої виконуються паралельно, і їх виконання займає приблизно однакову кількість часу, досягається зменшення часу виконання операції множення точки на скаляр майже вдвічі. Блок-схеми частин розробленого паралельного алгоритму представлені на рис. 5.

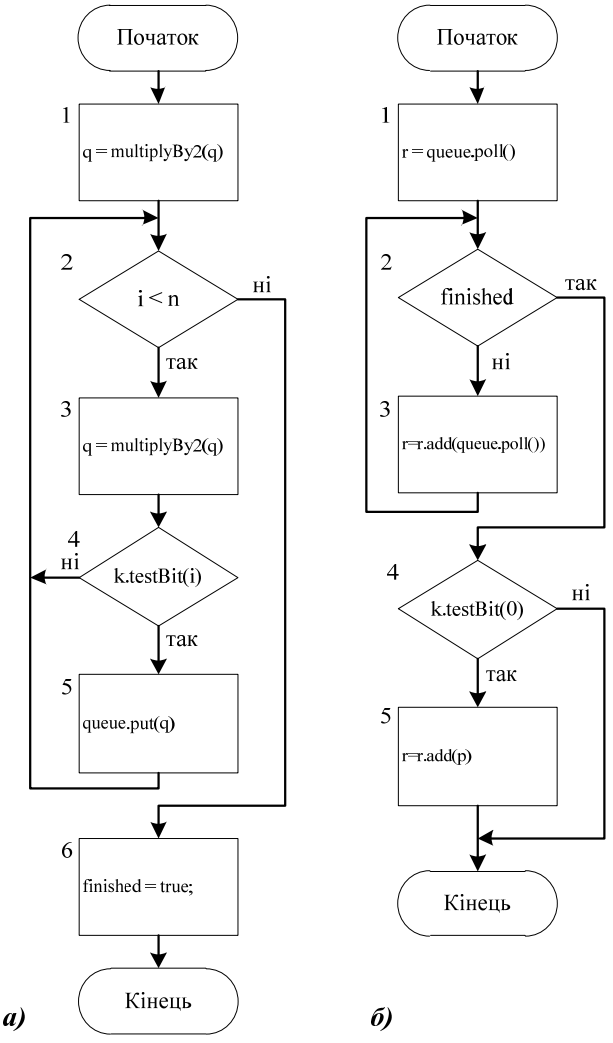


Рис. 5. Блок-схеми паралельного алгоритму множення точки еліптичної кривої на число: а) знаходження та зберігання точок 2^iP; б) обчислення суми точок 2^iP

6. Висновок

У даній роботі були досліджені сучасні схеми постановки електронного цифрового підпису та проведений порівняльний аналіз найбільш поширених схем.

В результаті дослідження схем постановки електронного цифрового підпису була отримана таблиця, у якій вказано розміри параметрів, що використовуються в обчисленнях, зберігаються на носіях а також передаються по каналам зв'язку. Згідно з отриманими результатами, алгоритм ECDSA при менших розмірах параметрів, що використовуються в обчисленнях та передаються по каналу зв'язку, здатний забезпечити більшу криптостійкість порівняно з іншими алгоритмами ЕЦП. Більша криптостійкість забезпечується завдяки використанню проблеми знаходження дискретного логарифму в групі точок ЕК (яка аналогічна проблемі пошуку закритого ключа), для якої досі не знайдено алгоритму з реальним часом пошуку результату.

На основі алгоритму ECDSA реалізовано систему створення та верифікації ЕЦП на базі еліптичних кривих, що, на відміну від існуючих реалізацій, дозволяє варіювати параметри еліптичної кривої в широких межах, та надає засоби їх дослідження, серед яких побудова графіку точок еліптичної кривої з заданими параметрами, відображення результатів операцій над точками графіку, поетапне виконання постановки та верифікації ЕЦП, можливість використання суперсингулярних та аномальних еліптичних кривих для досліджень їх недоліків.

Важливим результатом дослідження є розроблений алгоритм множення точки еліптичної кривої на скаляр, який призводить до значного підвищення швидкодії системи. В роботі описана модифікація даного алгоритму, у якій виділено дві окремі підзадачі, призначена для додаткового підвищення швидкодії алгоритму на багатопроцесорних системах на пристроях з багатоядерними процесорами, які стають розповсюдженими навіть у мобільних пристроях. У реалізації системи ЕЦП на еліптичних кривих використовується запропонований алгоритм множення точки еліптичної кривої на скаляр, дійсне підвищення швидкодії перевірено експериментально. Використання запропонованого алгоритму призводить до зменшення витрат часу на обчислення на порядки, в залежності від розміру системних параметрів, а використання паралельних обчислень у модифікованому алгоритмі призводить до додаткового зменшення часу обчислень до 50%, в залежності від кількості розрядів двійкового представлення числа, на яке множиться точка еліптичної кривої, що містять одиницю.

Порівняльний аналіз існуючих схем ЕЦП показав, що саме реалізований алгоритм підпису ECDSA є найбільш перспективним, так як довжина його ключа, з підвищенням вимог до криптостійкості, не так стрімко зростає, як у випадку інших існуючих алгоритмів, таких як RSA та DSA.

Список літератури

1. ДСТУ4145-2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка, Київ: Держстандарт України, 2003.
2. Serge Vaudenay. A Classical Introduction to Cryptography: Applications for Communications Security, Springer, 2006.
3. D.Johnson and A.Menezes. The Elliptic Curve Digital Signature Algorithm (ECDSA), Univ. of Waterloo, 1999.
4. D.Johnson and A.Menezes. The Elliptic Curve Cryptosystem. Remarks on the security of the elliptic curve cryptosystem, Univ. of Waterloo, 2000.
5. Harold F. Tipton and Micki Krause. Information Security Management Handbook, Sixth Edition, 2006.

УДК 004.3

ЖАБИНА В.В.,
КЕДА А.Ю.

ИССЛЕДОВАНИЕ МОДЕЛИ ПОТОКОВОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ С ПАРАЛЛЕЛЬНЫМ ФОРМИРОВАНИЕМ КОМАНД

Предложена имитационная модель потоковой вычислительной системы с формированием параллельных потоков команд. Исследуется зависимость времени реализации мелкозернистых алгоритмов в зависимости от количества вычислительных модулей, сред формирования команд, а также соотношения числа длинных и коротких операций. Показана возможность автоматического выявления скрытого параллелизма задач.

Simulator of the data-flow system with parallel instruction flows generation is proposed. Dependence of fine grained algorithm realization time on number of processors and instruction generation units, as well as on short and long instructions proportion is investigated. Possibility to discover hidden parallelism is shown.

Введение

При решении задач управления и моделирования в реальном времени возникает необходимость реализации вычислительных алгоритмов с мелкозернистой структурой. В качестве примера таких алгоритмов можно указать алгоритмы интерполяции функций, расчета траектории объектов в многомерном пространстве и т. д. Ускорения вычислений в этом случае можно добиться распараллеливанием алгоритмов на уровне операций.

Большинство из современных технологий параллельного программирования относятся к средствам статического распараллеливания процессов [1]. Задачи распараллеливания в этом случае решаются на этапе разработки программ. При статическом анализе алгоритмов не всегда удается выявить параллельные ветви, то есть скрытый параллелизм, что объясняется недостатком информации о динамике процессов в системе.

Перспективным подходом, позволяющим устранить ряд недостатков статического планирования, является разработка средства динамического распараллеливания вычислений. В этом случае назначение заданий на вычислительные узлы осуществляется системой в процессе решения задач. Такой подход дает возможность достичь большей степени параллелизма, так как позволяет выявить параллельные ветви, которые возникают непосредственно в процессе вычислений.

Поскольку динамическое распределение заданий осуществляется средствами самой системы, то важной задачей является уменьшение непроизводительных расходов времени на этот процесс.

Одним из подходов для динамического распределения заданий между вычислительными узлами является использование модели вычислений, управляемых потоком данных (потоковой модели). Распределение операций между вычислительными узлами в этом случае может быть реализовано автоматически на аппаратном или микропрограммном уровне, что сокращает затраты времени.

Потоковая модель вычислений

В системах, управляемых потоком данных (СУПД), команды выполняются не в заданной программой последовательности, а при наличии готовых данных (готовности всех операндов), то есть определяющим в данном случае является не порядок выполнения команд, а доступность данных для команды. Подготовка вычислений осуществляется на основе графа, каждой вершине которого соответствует операция (функция), а каждой дуге – данные.

Операция для каждой вершины графа описывается информационным словом, которое называется актором (actor). Акторы связаны между собой только по данным, которые соответствуют дугам графа.

Из соответствующих элементов акторов и данных в среде формирования команд (СФК) составляется команда, которая выполняется в свободном вычислительном модуле (ВМ) или поступает в очередь.

Для реализации данной модели вычислений могут быть использованы ПЛИС, которые

содержат вычислительные ядра, модули памяти и средства коммутации. Использование такой элементной базы и технологии SoC (System on chip) дают возможность эффективной реализации параллельных вычислений на уровне операций. При большой частоте тактирования современных ПЛИС основную задержку вычислений вносит общая СФК, построенная на основе памяти.

Известны различные способы формирования команд [2-8], эффективность которых определяется достигаемой интенсивностью потока готовых команд. Введение в систему нескольких СФК позволяет формировать параллельные потоки команд. Однако простое дублирование СФК приводит к недостатками статических методов, то есть к ручному распараллеливанию процессов: программист должен предварительно разрезать граф задачи на подграфы, сформировать идентификаторы для акторов и данных, которые приписывают их к определенным СФК.

В работе [9] показан способ автоматической идентификации объектов для систем с несколькими СФК. На основе графа задачи формируется матрица, с помощью которой определяются ветви команд, зависимых по данным (команды из разных ветвей могут выполняться одновременно).

Формат акторов имеет вид

$$A_i =< M_i, I_i, F_i, K_i, L_i, T_i >, \qquad (1)$$

где M_i – идентификатор СФК (номер потока команд); I_i – уникальное имя данного актора; F_i – функция преобразования данных (код операции); K_i – количество акторов, для которых i-й актор подготавливает операнд; $L_i =<I_j,M_j>$ – список имен акторов I_j с идентификаторами M_j, для которых i-й актор подготавливает операнд; T_i – тип данных. Данные определяются кортежем

$$D_i =< M_i, Q_i, N_i, L_i, T_i >, \qquad (2)$$

где Q_i – значение операнда; N_i – количество акторов, для которых передается данный операнд.

Из объектов (1) и (2) в разных СФК формируются параллельные потоки команд. Каждая команда составляется из объектов, имеющих одинаковые идентификаторы СФК.

Имитационная модель системы

В соответствии с указанным методом построена имитационная модель, позволяющая исследовать зависимость времени вычислений от параметров системы и характеристики алгоритмов.

Модель является комплексом программных средств, позволяющих подготавливать данные для задачи, моделировать ход вычислений и формировать данные для его анализа. Взаимодействие программных модулей системы представлено на рис. 1.

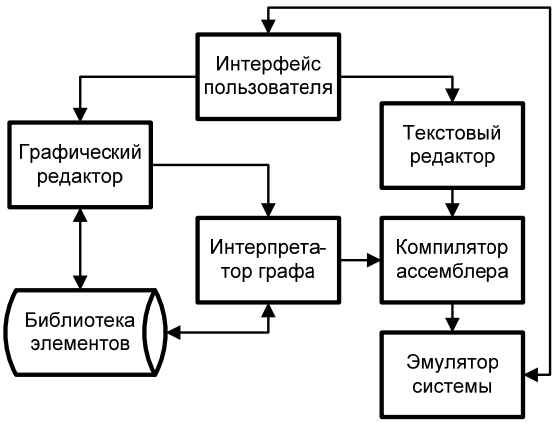


Рис. 1. Схема взаимодействия программных модулей системы

Интерфейс пользователя позволяет выбрать средство ввода алгоритма - графический или текстовый редактор.

Модуль графического редактора предоставляет возможность ввода и редактирования графа задачи, в котором вершинам соответствуют акторы, а дугам графа – данные. Модуль графического редактора передает структуру данных в модуль интерпретатора графа для дальнейшей обработки. В библиотеке элементов хранятся объекты, соответствующие определенному набору функций (преобразования данных и управления).

Модуль текстового редактора позволяет вводить и редактировать программу на специально разработанном ассемблере. Программа хранится в памяти в виде набора строк. Ссылка на программу передается компилятору ассемблера.

Модуль интерпретатора графа выполняет конвертацию графического представления, полученного графическим редактором, в код ассемблера. Для каждой вершины графа интерпретатор генерирует команду ассемблера. Данный модуль, как и текстовый редактор, передает компилятору ассемблера ссылку на программу.

Лексический анализатор представляет собой первую фазу компилятора. Его основная

функция состоит в чтении новых символов и выдаче последовательности токенов, используемых синтаксическим анализатором. Лексический анализатор построен как детерминированный конечный автомат.

Синтаксический анализатор получает строку токенов от лексического анализатора и проверяет, может ли эта строка порождаться грамматикой исходного языка. Он также сообщает обо всех выявленных ошибках. В случае удачного разбора входного потока терминалов синтаксический анализатор передает в модуль эмулятора список акторов и данных.

Эмулятор позволяет отображать состояние компонентов системы на всех стадиях обработки данных, включая процессы ввода данных, этапы формирование команд в СФК, распределение команд между ВМ и устройствами вывода данных. Фиксируется продолжительность этапов обработки данных.

Исследование модели системы

Для сравнительной оценки производительности реальных систем используются тестовые задачи с разной частотой команд, имеющих различную продолжительность выполнения, что учитывает область применения систем. При сравнении учитывается реальное время решения задачи.

На стадии проектирования системы можно сравнивать с помощью моделей по среднему времени выполнения команд $T_{\text{К}} = \sum_i p_i t_i$, где t_i – время выполнения команды i-го типа, а p_i – частота команды в программах. Частоты команд называют также смесями Гибсона. Для оценки продолжительности вичислений используются условные единицы (например, такты).

Для сравнения систем широкого применения на этапе проектирования используют упрощенную формулу $T_{\text{К}} = 0{,}7 p_{\text{К}} t_{\text{К}} + 0{,}3 p_{\text{Д}} t_{\text{Д}}$, где $p_{\text{К}}, t_{\text{К}}$ соответствуют короткой, а $p_{\text{Д}}, t_{\text{Д}}$ – длинной операции [10]. Для проблемно-ориентированных и специализированных систем могут использовать другие частоты команд.

В процессе моделирорвания определялась продолжительность вычислений в зависимости от следующих параметров:

– количество ВМ;

– количество СФК;

– продолжительность выполнения команд.

В качестве длинных рассматривалась команда умножения, а в качестве коротких – сложения.

В качестве тестовой использовалась задача, граф которой представляет бинарное дерево с высотой $h = 6$. Такой граф имеет последовательно-параллельную структуру, что соответствует большинству алгоритмов реализации численных методов при решении задач реального времени. Кроме того, при задании графа легко обеспечить требуемое соотношение длинных и коротких операций.

Исходя из соотношения длительностей выполнения длинных и коротких операций с использованием методов ускорения второго порядка для длинных операций (матричная схема умножения) [11], длительность выполнения короткой операции принималась равной длительности формирования команды в СФК, а длинной – в два раза больше. Такт моделирования принимался равным десяти условным единицам времени.

Для большого класса практических задач считаются весьма устойчивыми частоты коротких и длинных операций соответственно 0,7 и 0,3. При таких значениях частот на рис. 2 сведены результаты моделирования, при котором изменялись такие параметры системы, как количество ВМ (от 1 до 16) и количество СФК (1, 2, 4).

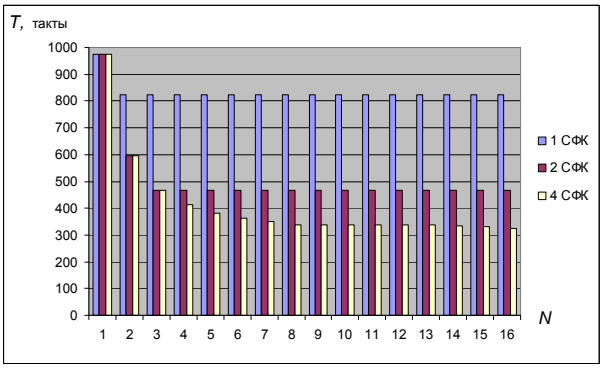


Рис. 2. Изменение времени T выполнения алгоритма при количественном соотношении коротких и длинных операций 7:3 (N – число ВМ)

В специализированных системах, связанных с решением задач цифровой обработки сигналов, аэродинамики, гидроакустики, сейсморазведки, метеорологии, управления быстрыми динамическими объектами, моделирования сплошной среды и ряда других задач, ча-

стота появления длинных операций значительно выше, чем в системах общего применения. При решении вышеуказанных задач усредненное соотношение коротких и длинных операций может изменяться в сторону увеличения частоты длинных операций.

На рис. 3 сведены результаты моделирования бинарного дерева с одинаковыми частотами для длинных и коротких операций при таких же параметрах системы, как в предыдущем случае.

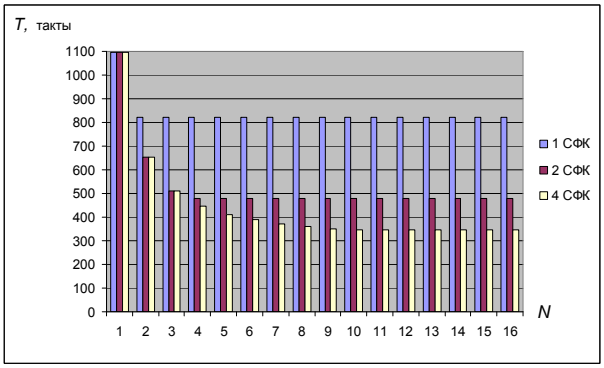


Рис. 3. Изменение времени T выполнения алгоритма при одинаковой частоте коротких и длинных операций (N – число ВМ)

Из приведенных на рисунках диаграмм можно сделать вывод, что при увеличении количества СФК время выполнения алгоритмов может быть уменьшено до определенного предела. Значение этого предела зависит, в первую очередь, от соотношения количества ВМ и СФК в системе, а также от степени параллелизма задачи. Для рассмотренных двух вариантов моделирования видно, что добавление даже одной СФК может улучить временные характеристики системы. Естественно, чем «шире» граф задачи и больше СФК, тем больший выигрыш можно получить в скорости обработки данных.

Для оценки эффективности использования нескольких СФК было рассмотрено изменение коэффициента относительно ускорения вычислений, определяемого по формуле

$$K_{\text{у}} = \frac{T_{1'} - T_2}{T_1} 100\%,$$

где в рассматриваемом случае T_1 – время выполнения алгоритма с одной СФК, а T_2 - время выполнения алгоритма с несколькими СФК.

Из приведенного на рис. 4 графика видно, что использование двух СФК позволяет ускорить вычисления более чем на 40 %, а при 4-х – почти на 60 %.

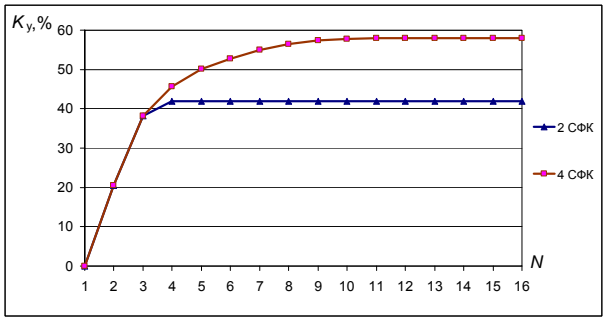


Рис. 4. Зависимость коэффициента ускорения вычислений от числа СФК и количества N ВМ

Скрытый параллелизм задач

На весьма простом примере можно показать, как в системе реализуется скрытый параллелизм задачи, который весьма трудно определить при статическом анализе алгоритма.

Из ярусно-параллельной формы алгоритма (рис. 5) видно, что на каждом ярусе находится не более двух вершин.

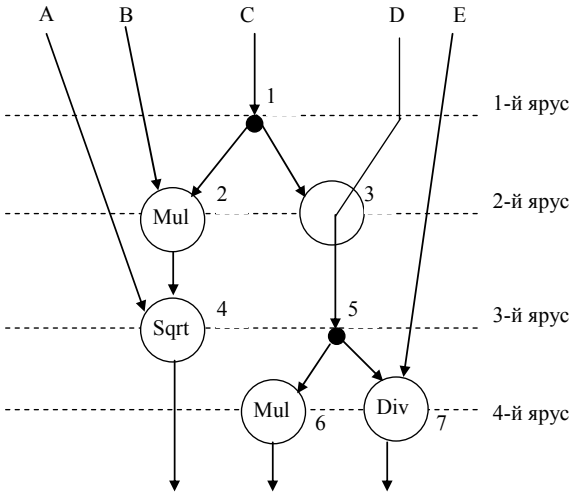


Рис. 5. Ярусно-параллельная форма алгоритма

В результате моделирования получена временная диаграмма (рис. 6), из которой следует, что при наличии в системе трех ВМ одновременно выполняются 3 операции (4, 6, 7). Это связано с разным временем выполнения операций. В частности, операция 5 выполняется быстрее операции 4.

На таком простом алгоритме, зная все времена выполнения операций и затраты времени на пересылки, несложно выделить параллель-

ные операции. Но реальные алгоритмы могут быть намного сложнее, что делает задачу распараллеливания процессов в статике очень трудной. В таких случаях применение СУПД является весьма эффективным.

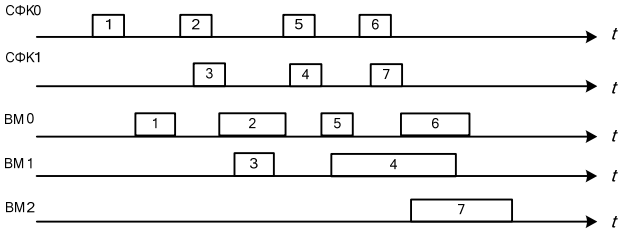


Рис. 6. Временная диаграмма выполнения алгоритма

Оценивалась возможность автоматического обнаружения скрытого параллелизма при решении систем линейных алгебраических уравнений (СЛАУ) методом Гаусса.

Метод Гаусса состоит в последовательном исключении неизвестных до тех пор, пока не останется одно уравнение с одним неизвестным (прямой ход метода). При этом матрица СЛАУ приводится к треугольному виду, где ниже главной диагонали располагаются только нули.

Обратный ход начинается с решения последнего уравнения и заканчивается определением первого неизвестного.

Оценим возможность распараллеливания вычислений при прямом и обратном ходе метода Гаусса.

Имеем $Ax = b$, где $A = [a_{ij}]$ – матрица размерности $n \times n$, $\det A > 0$, $b = (b_1, b_2, ..., b_n)$.

В предположении, что $a_{11} \neq 0$, первое уравнение системы

$$\sum_{j=1}^{n} a_{ij}^0 x_j = b_i^0,\ i = 1,...,n$$

делим на коэффициент a_{11}, в результате получаем уравнение

$$x_1 + \sum_{j=2}^{n} a_{1j}^1 x_j = b_1^1 .$$

Затем из первого уравнения вычитается каждое из остальных уравнений, деленное на соответствующий коэффициент a_{i1}. В результате эти уравнения преобразуются к виду

$$\sum_{j=2}^{n} a_{ij}^1 x_j = b_i^1,\ i = 2,...,n.$$

Первое неизвестное оказалось исключенным из всех уравнений, кроме первого. Далее предполагаем, что $a_{22}^1 \neq 0$, делим второе уравнение на a_{22}^1 и исключаем неизвестное x_2 из всех уравнений, начиная со второго, и т.д. В результате последовательного исключения неизвестных система уравнений преобразуется в систему уравнений с треугольной матрицей

$$x_i + \sum_{j=i+1}^{n} a_{1j}^i x_j = b_i^i,\ i = 1,...,n$$

.

Из n-го уравнения определяем x_n, из $(n-1)$-го – x_{n-1} и т.д. Последним определяется x_1. Совокупность таких действий называется обратным ходом метода Гаусса.

Реализация прямого хода требует $N \approx 2n^3/3$ арифметических операций. При этом параллельно могут выполняться n^2 операций. При обратном ходе требуется $N \approx n^2$ арифметических операций, причем, параллельно можно выполнять $n-1$ операцию.

При статическом распараллеливании для системы с тремя неизвестными максимальное число ветвей графа равно девяти. Моделирование проводилось с использованием 16 ВМ. Соотношение длительности коротких и длинных операций при моделировании принята такой, как в предыдущих примерах.

Как видно из результатов моделирования (табл. 1), увеличение числа СФК позволило более эффективно загрузить ВМ. При двух СФК одновременно выполняют операции 11 ВМ, а при четырех – 12 ВМ. Скрытый параллелизм связан с разбросом времени выполнения операций на параллельных ветвях программы.

Табл. 1. Результаты моделирования

Количество СФК	**Количество параллельно работающих ВМ**	**Число тактов**
1	8	509
2	11	335
4	12	255

Выводы

Метод автоматического формирования параллельных потоков команд в СУПД дает потенциальную возможность ускорить решение задач, когда интенсивность обработки данных

в ВМ превышает интенсивность формирования команд одной СФК. Акторы и данные автоматически снабжаются идентификаторами СФК на этапе компиляции программы. Нет необходимости выполнять распараллеливание в ручном режиме.

Динамическое распределение операций позволяет выявить непосредственно в процессе моделирования скрытый параллелизм задачи, связанный с различными длительностями обработки данных в различных ветвях алгоритмов, что весьма затруднительно при статическом анализе алгоритма.

Моделирование с использованием симулятора позволяет для рассматриваемых задач определить параметры системы, которые обеспечивают эффективную реализацию целевой функции. Может быть определено время решения задачи при разном количестве ВМ и СФК, что дает возможность в каждом конкретном случае выбрать необходимую конфигурацию системы.

Список литературы

1. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.
2. *Dennis J. B., Missunas D. P*. A preliminary architecture for basic data flow processor// Proc. 2nd Annual Symp. Comput. Stockholm, May 1975. N. Y. IEEE. – 1975. – P. 126 – 132.
3. *Silva J.G.D., Wood J.V.* Design of processing subsystems for Manchester data flow computer // IEEE Proc. N.Y. – 1981. – Vol. 128, N 5. – P. 218 – 224.
4. *Watson R., Guard J.* A practical data flow computer // Computer. – 1982. – Vol. 15, N 2.– P. 51 – 57.
5. *Hogenauer E.B., Newbold R.F. Inn Y.T.* DDSP – a data flow computer for signal processing/ Proc. Int. Conf. Parall. Process. Ohio, August 1982. N.Y. // IEEE. – 1982. – P. 126 – 133.
6. *Johnson D.* Data flow machines threaten the program counter// Electronic Design. – 1980. – N 22. – P. 255-258.
7. Функционально ориентированные процессоры */ Водяхо А.Н., Смолов В.Б., Плюснин В.У., Пузанков Д.В. /* Под ред. *В.Б.Смолова.* – Л.: Машиностроение. Ленингр. отд-ние, 1988. – 224 *с.*
8. *Жабина В.В.* Повышение эффективности параллельной обработки данных на уровне операций в потоковых системах // Вісник Національного технічного університету України “Київський політехнічний інститут”, Інформатика, управління та обчислювальна техніка,: Зб. наук. пр. – К.: „ВЕК+”. – 2008. – №49. – С. 112-116.
9. *Жабина В.В.* Параллельное формирование команд в потоковых системах // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка: Збірка наукових праць. – К.: Век+. – 2009. – № 50. – С. 113-117.
10. *Лощилов И.Н.* Перспективы роста производительности ЭВМ // Зарубежная радиоэлектроника. – 1976. – №5. – с. 5-25.
11. *Карцев М.А.* Брик В.А. Вычислительные системы и синхронная арифметика. – М.: Радио и связь, 1981. – 360 с.

Відомості про авторів

Авраменко В.С. НТУУ «КПІ»
Булах Б.В. асистент кафедри СП ІПСА НТУУ "КПІ"
Грибенко Д.В. студент кафедри ОТ НТУУ «КПІ»
Гемба Н.В. магістр кафедри ММСА «ІПСА» НТУУ «КПІ»
Голков В.Б. студент НТУУ «КПІ»
Горобец А.Н. НТУУ «КПІ»
Губский А.Н. аспірант кафедри ТК НТУУ «КПІ»
Гусев Е.И. асистент кафедри ОТ НТУУ «КПІ»
Демчинский В.В. НТУУ «КПІ»
Дорогой Я.Ю. асистент кафедри АУТС НТУУ «КПІ»
Дорошенко К.С. асистент кафедри АУТС НТУУ «КПІ»
Дьяконова С.В. аспірант НТУУ «КПІ»
Жабина В.В. старший преподаватель кафедры ПОКС НТУУ «КПИ»
Зайченко Ю.П. д.т.н., профессор кафедры ММСА УНК «ИПСА» НТУУ «КПИ»
Зиненко А.И. студент кафедри ОТ НТУУ «КПІ»
Кеда А.Ю. студент кафедры ПОКС НТУУ «КПИ»
Коган А.В. асистент кафедри ОТ НТУУ «КПІ»
Кононенко Ю.А. НТУУ «КПІ»
Коротенко А.А. НТУУ «КПІ»
Кравець П.І. к.т.н., доцент кафедри АУТС НТУУ «КПІ»
Кулаков Ю.А. д.т.н., професор кафедри ОТ НТУУ «КПІ»
Курбанов В.В. аспірант кафедри ТК НТУУ «КПІ»
Куц В.Ю. НТУУ «КПІ»
Ладогубец В.В. к.т.н., доцент кафедри САПР НТУУ «КПІ»
Ланге Т.И. кафедра ВТ НТУУ «КПІ»
Лукіна Т.Й. кафедра АУТС НТУУ «КПІ»
Марковський О.П. к.т.н., доцент кафедри ОТ НТУУ «КПІ»
Март Б.А. НТУУ «КПІ»
Мисюра Е.Б. к.т.н, с.н.с. кафедри АСОІУ НТУУ «КПІ»
Надеран Э. аспірант НТУУ «КПІ»
Назаревич О.Б. асистент кафедри комп'ютерних наук, Тернопільський національний технічний університет
Павлов А.А. д.т.н., профессор кафедры АСОИУ НТУУ "КПИ", декан ФИВТ НТУУ «КПІ»
Павлов А.В. м.н.с., Международный научно-учебный центр информационных технологий и систем НАНУ
Пирогов А.А. студент кафедры ВТ НТУУ «КПИ»
Покотило А.А. кафедра АУТС НТУУ «КПІ»
Полторак В.П. к.т.н., доцент кафедри АУТС НТУУ "КПІ",
Поспешный А.С. аспірант кафедри ОТ НТУУ «КПІ»
Ролик А.И. к.т.н., доцент кафедри АУТС НТУУ «КПІ»
Салапатов В.І. к.т.н., доцент кафедры кибернетики, ЧНУ
Свірін П.В. асистент кафедри СП ІПСА НТУУ «КПІ»
Симоненко В.П. д.т.н., професор кафедри ОТ НТУУ «КПІ»
Сперкач М.О. асистент кафедри АСОІУ НТУУ «КПІ»
Стенин А.А. д.т.н., професор кафедри ТК НТУУ «КПІ»
Стенин С.А. Начальник сектора Центрального таможенного управления лабораторных исследований и экспертной работы
Стиренко С.Г. к.т.н., доцент кафедри ОТ НТУУ «КПІ»
Тимошин Ю.А. к.т.н., с.н.с., доцент кафедри ТК НТУУ «КПІ», зав. відділом НИИ "Системных технологий"
Ткач І.І. студент кафедри АУТС НТУУ «КПІ»
Ткач М.М. к.т.н., и.о. заведующего кафедрой ТК НТУУ «КПИ»
Федоречко О.І. студент кафедри ОТ НТУУ «КПІ»
Финогенов А.Д. к.т.н., кафедри СП ІПСА НТУУ «КПІ»
Халус Е.А. асистент кафедри АСОІУ НТУУ «КПІ»
Чекалюк В.В. асистент кафедри СП ІПСА НТУУ «КПІ»
Шемсединов Т.Г. научный сотрудник НИИ "Системных технологий" НТУУ «КПИ»
Щербина О.В. НТУУ «КПІ»
Шимкович В.М. аспирант кафедри АУТС НТУУ «КПІ»
Ясочка М.В. кафедра АУТС НТУУ «КПІ»

SUMMARY