З М І С Т

Згуровский М.З., Павлов А.А., Мисюра Е.Б., Мельников О.В Методы оперативного планирования и принятия решений в сложных организационно-технических системах 3

Згуровский М.З., Павлов А.А., Мисюра Е.Б., Мельников О.В Методология построения эффективного решения многоэтапных задач календарного планирования на основе принципа иерархии и комплекса взаимосвязанных моделей и методов 8

Безносик О.Ю., Кот Д.М Зменшення неоднозначності алгоритму скорочення розмірності RLC-схем 19

Ладогубец В.В., Крамар А.В., Финогенов А.Д Расчёт времени выполнения параллельного метода численного интегрирования на основе разностей высших порядков 23

Петренко А.И., Малявко А.С Программа моделирования процесса экспонирования резиста в электронно-лучевой литографии 30

Жданова О.Г., Падалко О.Б Про проблему генерації планів тестування для автоматизації процесу верифікації програмного продукту 34

Корочкин А.В., Березовский А.Н Модели вычислений для масштабируемых кластерных систем 42

Зайченко Ю.П., Лабжинский В.А Сравнительный анализ двух методов прогнозирования загрязнения воздушного бассейна 49

Молчановський О.И., Степанов И.О., Ковтун Д.И Порівняння алгоритму жадібного рядкового замощення і методу відбитків при виявленні плагіату у вихідних кодах програм 56

Дишлевий О. П Застосування предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення 64

Сергієнко А.М., Лепеха В.Л Деякі особливості проектування мікроконтролерів для СНК 70

Сергиенко А.М Применение рациональных дробей в специализированных вычислителях 74

Лепеха В.Л., Лесик Т.М., Франко Р.А Реализация декодера Рида-Соломона в ПЛИС 78

Давиденко І.М., Кравченко С.О., Бабій К.Ю Децентралізована модель керування GRID-системою 82

Блінова Т.О Ділення в кодах золотої пропорції в порозрядному режимі, починаючи зі старших розрядів 86

Зюзя А.А Способ противодействия реконструкции ключей блоковых алгоритмов защиты информации анализом динамики потребляемой мощности 89

Кулаков А.Ю., Клименко И.А Способ формирования структуры виртуальной GRID системы 97

Ковальчук О.В Методи та засоби передбачення перевантаження комп'ютерних мереж 101

Турченко Ю.А Использование арифметических взвешенных контрольных сумм для коррекции ошибок в каналах со спектральной

модуляцией 106
Жабина В.В Параллельное формирование команд в потоковых системах 113
Марковский А.П., Виноградов Ю.Н., Поважная Н.С. К проблеме оценки качества двоичных псевдослучайных последовательностей для систем защиты информации 118
Болдак А.А., Невдащенко М.В., Демьяненко М.В Система факторов для компаративного анализа программных проектов 125
Дорогой Я.Ю., Яшин В.Е., Яцук С.В Многопоточная эмуляция искусственных нейронных сетей произвольной топологии 130
Kussul O., Novikov O Utility-based reputation model for VO in GRIDs 137
Мухин В.Е., Волокита А.Н Анализ событий информационной безопасности для проведения корректирующих действий по управлению безопасностью 146
Скакун С.В Структурно-функциональный анализ компонентов систем Sensor Web 152
Кондрашова Н.В Метод Грама-Шмидта-Гаусса для решения систем линейных уравнений в МГУА 158
Теленик С.Ф., Ролик А.И., Букасов М.М., Ясочка М.В Определение распространения влияния неисправностей в сети доступа на качество предоставляемых сервисов 164
Харук Р.В Обработка данных из журнала аудита в системах мониторинга событий безопасности 174
Зайцев В.Г., Лан Чуньлинь Проектирование и внедрение системы автоматической классификации документов для конечного множества языков 179

УДК 519.816:854.2

ЗГУРОВСКИЙ М.З.,
ПАВЛОВ А.А.,
МИСЮРА Е.Б.,
МЕЛЬНИКОВ О.В.

МЕТОДЫ ОПЕРАТИВНОГО ПЛАНИРОВАНИЯ И ПРИНЯТИЯ РЕШЕНИЙ В СЛОЖНЫХ ОРГАНИЗАЦИОННО-ТЕХНОЛОГИЧЕСКИХ СИСТЕМАХ

Рассматривается алгоритмизация процесса принятия решений по выбору наилучшей альтернативы в блоке принятия решений, включенном в иерархическую модель планирования и управления сложными системами с сетевым представлением технологических процессов и ограниченными ресурсами.

The algorithm of decision making on selection the best alternative in the decision unit is considered. This unit is included to the hierarchical model of planning and control of complex systems with network representation of technological processes and limited resources.

Иерархическая модель планирования и управления сложными системами с сетевым представлением технологических процессов и ограниченными ресурсами

В [1, 2] представлена иерархическая модель планирования и управления сложными организационно-технологическими системами, состоящая из трех уровней (рис. 1): прогнозного, согласованного и точного планирования. В основу представленной модели планирования положен созданный на основе новой конструктивной теории математический аппарат решения труднорешаемых задач календарного планирования за критериями максимизации прибыли предприятий [1, 3].

В [4] указано, что возможность трехуровневой моделью планирования получать близкие к оптимальным расписания работ по различным критериям нельзя считать завершающим этапом планирования. Возникает задача выбора лицом, принимающим решение, наилучшей альтернативы в соответствии со слабо формализованной глобальной целью. Этот выбор осуществляется в блоке принятия решений, включенном в иерархическую модель (рис. 1).

В соответствии с трехуровневой моделью, построение распределения работ по ресурсам осуществляется в три этапа. Первый состоит в построении агрегированной модели. Если какие-либо работы выполняются в одном мультиресурсе в рамках одного захода в мультиресурс по одному заданию, то они агрегируются в одну агрегированную работу. Продолжительность выполнения агрегированной работы определяется ее критическим путем. Для каждого комплекса работ определяется критический путь выполнения агрегированных работ. На основе агрегированной информации строится граф на критических путях заданий. Вершины полученного графа – это агрегированные работы, дуги отображают связи между мультиресурсами, регламентирующие технологию выполнения заданий.

Некоторые работы, принадлежащие разным заданиям, требуют выполнения в специализированных уникальных мультиресурсах, желательно в рамках одного захода в мультиресурс. В этом случае при выполнении некоторых условий, описанных ниже, формируется объединенная агрегированная работа, что на графе связности отображено общими вершинами. Для определения приоритетов заданий при построении согласованного плана выполнения заданий соответственно критериям оптимальности, важным является решение на первом уровне задачи «Минимизация суммарного взвешенного момента окончания выполнения заданий» (МВМ) для случая, когда весовые коэффициенты всех вершин графа связности, кроме конечных, равны нулю (см. [1]). В результате решения этой задачи формируется приоритетно-упорядоченная последовательность, определяющая очередность запуска агрегированных работ на выполнение.

Второй этап заключается в построении согласованного плана выполнения заданий с учетом указанных выше критериев оптимальности. Полученные на первом уровне приоритеты агрегированных работ служат дополнительной информацией, позволяющей

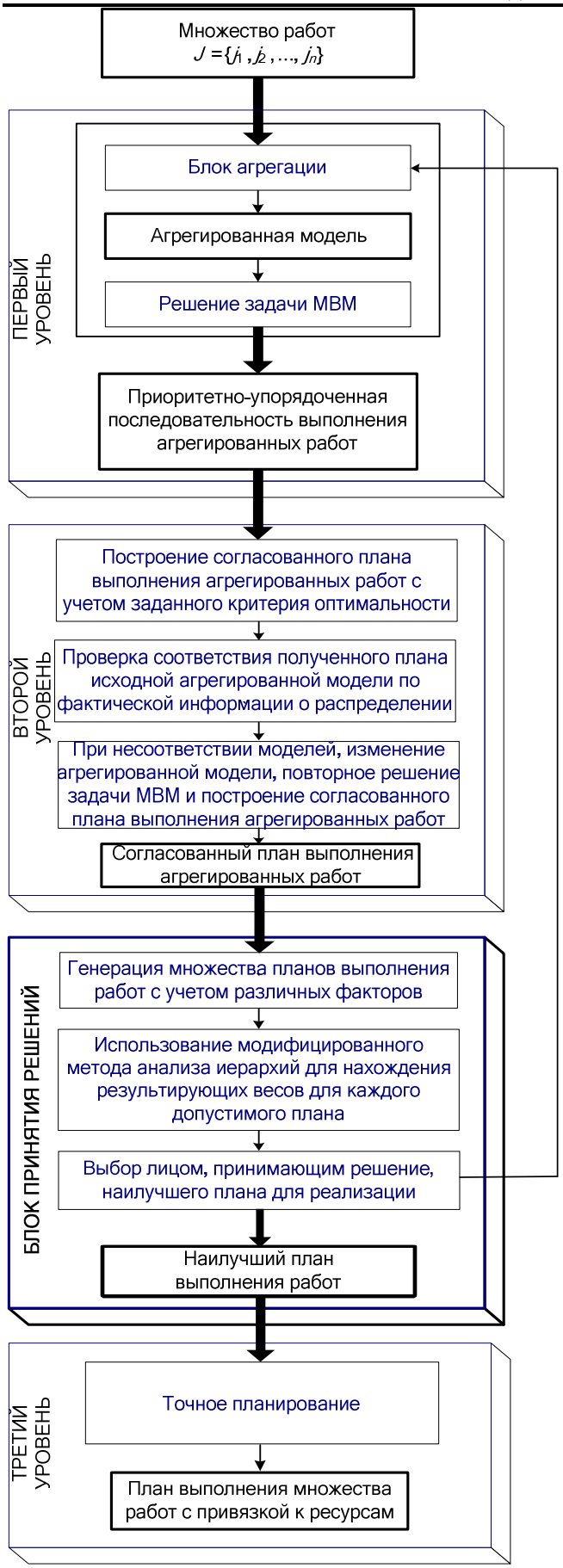


Рис. 1. Функциональная схема иерархической модели планирования и управления сложными системами

значительно повысить эффективность полученных решений. Полученный план проверяется на соответствие исходной агрегированной модели, при необходимости модель корректируется, после чего план выполнения заданий строится заново.

Построенные на втором уровне модели согласованные планы выполнения заданий по заданным критериям оптимальности передаются на блок принятия решений, в котором генерируется по различным критериям оптимальности в соответствии с поставленными требованиями целая серия возможных допустимых планов, отличающихся конкретным видом критерия, директивными сроками, весовыми коэффициентами, технологией реализации. В результате выбирается один план, передающийся на третий уровень системы для реализации. Если же план, удовлетворяющий поставленным требованиям, не получен, информация передается на первый уровень для коррекции: корректируются производственные ресурсы (мощности, материалы, рабочая сила), объемы выпускаемой продукции (состав портфеля заказов), меняются директивные сроки и т.д. Более подробное описание процедуры принятия решений приведено ниже.

Процедуры третьего уровня позволяют в соответствии с полученным планом для агрегированных работ построить распределение работ по ресурсам (так называемое точное планирование). На этом уровне решаются задачи по критериям оптимизации как для одного, так и для параллельных приборов [1].

Алгоритмизация процесса принятия решений

В блоке принятия решений процесс принятия решений разделен на два этапа. На этапе 1 формируются допустимые планы выполнения работ, на этапе 2 выбирается один из них, который и реализуется в течение планового периода.

Этап 1.

На основе результатов прогноза рыночной конъюнктуры, сформированного портфеля заказов, используя взаимосвязанные модели первого и второго уровня трехуровневой модели планирования, реализуются допустимые планы выполнения работ, каждый из которых может быть принят к реализации в

плановом периоде. При этом могут быть использованы следующие возможности алгоритмического обеспечения первого и второго уровня.

1) Допустимые планы выполнения работ генерируются с помощью взаимосвязанных моделей первого и второго уровня (планы выполнения агрегированных работ мультиресурсами) для того, чтобы на этапе 2 процесса принятия решений максимально облегчить эксперту получение оценки гарантированного выполнения в заданные сроки сравниваемых планов.

2) Критерий 1 [1] позволяет строить план выполнения работ, не привязываясь к конкретному заказчику, ориентируясь только на прогнозируемую рыночную конъюнктуру.

3) Критерий 3 [1] является смешанным: наряду с конкретными заказчиками может реализовываться продукция, не привязанная к конкретному потребителю.

4) На этапе 1 значения критериев 4, 5, 7 [1] интерпретируются как уменьшение прибыли, если предлагаемые заказчиками ограничения по срокам не могут быть выполнены производственной программой. После окончательного согласования с заказчиками ограничений по срокам (они могут быть новыми, и им соответствует меньшая прибыль), штрафы за их реальное невыполнение по плану выполнения работ, выбранному на этапе 2, могут быть существенно бóльшими.

5) Если генерируемые планы выполнения работ по критериям 2–7 [1] реализуют недостаточную прогнозируемую прибыль, либо не полностью удовлетворяют ограничениям (не все завершающие работы могут быть выполнены точно в срок), анализируется возможность увеличения ресурсов системы, возможны изменения сетевой технологии, и соответствующий план выполнения работ генерируется заново.

6) С точностью до процедуры изменения ресурсов и сетевой технологии процесс генерации допустимых планов выполнения работ является полностью детерминированным и однозначно задан взаимосвязанными моделями трехуровневой системы планирования.

Этап 2.

Из множества допустимых планов выполнения работ выбирается один, который и будет реализован в плановом периоде. Казалось бы, очевидное решение – выбирается тот план, которому соответствует максимальная ожидаемая прибыль, – тем не менее, является неверным.

Дело в том, что если плановый период является продолжительным (единица измерения не меньше месяца), то на величину реальной прибыли от реализации допустимого плана выполнения работ влияет такое количество неформализованных либо принципиально не измеряемых заранее факторов (часть из которых сформулирована ниже), что задача выбора лучшего плана выполнения работ из сгенерированного множества допустимых представляет собой проблему, сформулированную и эффективно решенную в [5, 6, 7, 8].

Действительно, учитывая, что планируемая прибыль от реализации любого допустимого плана выполнения работ является допустимой, необходимо выбрать тот из них, у которого риск от существенного уменьшения планируемой прибыли является минимальным. Сформулирована задача выбора наилучшей альтернативы из достаточно большого числа альтернатив (генерация большого количества допустимых планов выполнения работ не требует значительных ресурсов) в соответствии с неформализованной глобальной целью. Ее предлагается решать с помощью модифицированного метода анализа иерархий Саати. Основной сложностью, существенно влияю-

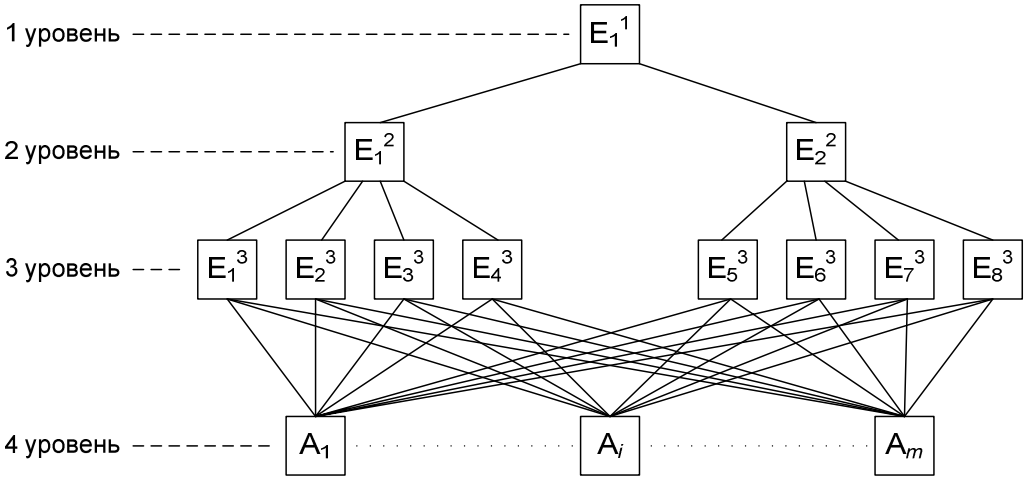


Рис. 2. Дерево иерархий

щей на эффективность использования метода анализа иерархий и его модификации является построение дерева иерархии, адекватного рассматриваемой задаче.

Очевидно, что каждой конкретной системе планирования соответствует свое собственное дерево иерархий. Однако можно предложить (см. рис 2) инвариантное дерево иерархий, которое может быть использовано для решения реальных задач либо быть хорошей основой для построения индивидуального дерева иерархий.

Локальные вклады критериев $E_1^3 \dots E_4^3$ в критерий E_2^2 равны нулю.

Локальные вклады критериев $E_5^3 \dots E_8^3$ в критерий E_1^2 равны нулю.

Таким образом, дерево иерархий соответствует формальному представлению (см. раздел 7.1). E_1^1 – глобальная цель – найти допустимый рабочий план с минимальным риском существенного уменьшения планируемой прибыли.

E_1^2 – риск существенного уменьшения планируемой прибыли для плана в случае его успешного выполнения.

E_2^2 – риск существенного уменьшения планируемой прибыли в случае срыва заданных ограничений по срокам.

E_1^3 – риск существенного уменьшения планируемой прибыли в силу ухудшения рыночной конъюнктуры, как следствие возможного локального (глобального) финансового кризиса.

E_2^3 – риск существенного уменьшения планируемой прибыли в силу возможного появления в планируемом периоде на рынке конкурентно эффективных альтернатив.

E_3^3 – риск существенного уменьшения планируемой прибыли в силу резкого увеличения затрат на выполнение допустимого плана, зависящих от возможной реализации ряда (определенных типом системы планирования) факторов, оценка которых не поддается детерминированному прогнозу.

E_4^3 – риск существенного уменьшения планируемой прибыли от возможного изменения в худшую сторону финансовой стабильности заказчика (заказчиков).

E_5^3 – риск существенного уменьшения планируемой прибыли от неправильной оценки ресурсных, трудовых, наукоемких затрат для выполнения допустимого плана. Фактор является существенным, когда допустимый план включает в себя выпуск качественно новой продукции.

E_6^3 – риск существенного уменьшения планируемой прибыли от возможного несоответствия качества исходной продукции (полуфабрикатов, комплектующих изделий и т. д.) стандартам, влияющих на значение показателей качества выпускаемой продукции, что может привести к увеличению сроков выпуска результирующей продукции.

E_7^3 – риск существенного уменьшения планируемой прибыли от возможного срыва запланированных сроков поставщиками.

E_8^3 – риск существенного уменьшения планируемой прибыли от возможного ухудшения на плановом периоде функционирования системы ее финансового состояния, что может привести к срыву допустимых сроков плана выполнения работ.

Элементы матриц парных сравнений на четвертом уровне иерархии $\gamma_{ij}, i, j = \overline{1,m} (i \neq j)$ интерпретируются следующим образом: во сколько раз степень риска существенного изменения прибыли у допустимого плана выполнения работ A_i выше, чем у допустимого плана выполнения работ A_j как следствие возможной реализации условий, определяющих критерий $E_e^3, e = \overline{1,8}$.

Элементы матриц парных сравнений на третьем уровне иерархий $\gamma_{ij}, i, j = \overline{1,4} (i \neq j)$ интерпретируются следующим образом: во сколько раз вклад критерия E_i^3 весомей вклада критерия E_j^3 в критерий E_1^2 (либо в критерий E_2^2) с точки зрения существенного уменьшения планируемой прибыли.

Нормированные локальные веса ω_1, ω_2 при значениях $E_i^2(A_i)$ и $E_2^2(A_i)$ (см. раздел 7.1), окончательно реализуют интегральный вес альтернативы A_i:

$E_1^1(A_i) = \omega_1 E_1^2(A_i) + \omega_2 E_2^2(A_i)$, $i = \overline{1,m}$, веса $\omega_1, \omega_2 \geq 0$ непосредственно определяются экспертным путем. Можно лишь указать очевидную тенденцию: чем стабильнее финансовое состояние самой системы и эконо-

мики в целом, тем больше ω_2 по сравнению ω_1 (в предположении, что фактором появления на рынке конкурентно эффективной продукции можно пренебречь) – если план выполнения работ реализован, то риск от существенного уменьшения планируемой прибыли тем меньше, чем больше финансовая стабильность.

Наименьший риск существенного уменьшения планируемой прибыли имеют те планы выполнения работ, у которых результирующий вес $E_1^1(A_i)$ является наименьшим.

Пусть достоверно (статистически достоверно) найдены все результирующие веса альтернатив $E_1^1(A_i)$, $i=\overline{1,m}$ (эмпирические матрицы парных сравнений хорошо обусловлены, если используется метод анализа иерархий либо выполняются условия получения эффективного решения при использовании модификаций метода анализа иерархий, изложенные в главе 7). Тогда процесс принятия решений реализуется следующим образом:

1) Если допустимый план выполнения работ с минимальным результирующим весом имеет максимальную плановую (прогнозируемую) прибыль, – он и выбирается к исполнению.

2) Если имеется совокупность допустимых планов выполнения работ, для каждого из которых результирующий вес допустимо не отличается от минимального, то выбирается тот план выполнения работ, которому соответствует максимальная планируемая прибыль.

Общий случай.

а) Учитывая, что у всех планов выполнения работ планируемая прибыль является допустимой, выбирается та, которой соответствует минимальный результирующий вес.

б) Получение результирующих весов существенно упрощает лицу, принимающему решение (ЛПР) найти компромисс между желанием получить максимальную прибыль и возможностью понести существенные финансовые потери.

Таким образом, подводя итоги, можно утверждать, что хотя использование метода анализа иерархий и его модификаций не является единственной возможностью обоснованного выбора плана выполнения работ из множества допустимых в условиях неопределенности, тем не менее, и его реализация является не только допустимым, но и эффективным способом решения поставленной задачи.

Список литературы

1 Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с.

2 Общая модель и методы иерархического планирования функционирования сложных организационно-производственных систем с ограниченными ресурсами / А.А. Павлов, Е.Б. Мисюра, О.В.Мельников, Сераджи Али Рухани // Системні дослідження та інформаційні технології.– 2005.– №4.– С.7-24

3 Конвей Р.В., Максвелл У.Л. Теория расписаний.– М.: Наука, 1975.– 359 с.

4 Згуровский М.З., Павлов А.А. Иерархическое планирование в системах, имеющих сетевое представление технологических процессов и ограниченные ресурсы, как задача принятия решений // Системні дослідження та інформаційні технології.– 2009.– №.49. (в печати)

5 Математические модели оптимизации для обоснования и нахождения весов объектов по неоднородным матрицам парных сравнений / А.А. Павлов, В.И. Кут // Системні дослідження та інформаційні технології.– 2007.– №3.– С.28-37

6 Павлов А.А., Лищук Е.И., Кут В.И. Многокритериальный выбор в задаче обработки данных матрицы парных сравнений // Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. К.: “ВЕК+”, 2007.– №46.– С.84-88

7 Математические модели оптимизации для обоснования и нахождения весов объектов в методе парных сравнений / А.А. Павлов, Е.И. Лищук, В.И. Кут // Системні дослідження та інформаційні технології.– 2007.– №2.– С.13-21

8 Павлов А.А., Иванова А.А., Зигура Р.А. Метод группового учёта аргументов и анализа иерархий (МГУАиАИ) в задачах принятия решений // Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. К.: “ВЕК+”, 2007.– №47.– С.205-214

Поступила в редакцию 11.12.2009

УДК 519.854.2

ЗГУРОВСКИЙ М.З.,
ПАВЛОВ А.А.,
МИСЮРА Е.Б.,
МЕЛЬНИКОВ О.В.

МЕТОДОЛОГИЯ ПОСТРОЕНИЯ ЭФФЕКТИВНОГО РЕШЕНИЯ МНОГОЭТАПНЫХ ЗАДАЧ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ НА ОСНОВЕ ПРИНЦИПА ИЕРАРХИИ И КОМПЛЕКСА ВЗАИМОСВЯЗАННЫХ МОДЕЛЕЙ И МЕТОДОВ

Рассмотрена методология иерархического планирования и управления в системах, имеющих сетевое представление технологических процессов и ограниченные ресурсы.

The methodology of hierarchical planning and control in systems with network representation of technological processes and limited resources is considered.

Общее описание систем, имеющих сетевое представление технологических процессов и ограниченные ресурсы

Постановкам задач планирования и управления сложными системами и методам их решения в последние десятилетия отводится важное внимание со стороны многих исследователей [1]. Несмотря на привлекательные перспективы, лишь некоторые из недавних результатов исследований введены в ежедневную практику. Хотя достижения в исследовании операций и искусственном интеллекте привели к разработке новых методов моделирования и решения задач, однако практическое применение часто требует большего со стороны исследователей – более полных моделей и более эффективных алгоритмов.

Наиболее хорошо изученная область планирования и управления сложными системами – производственное планирование и управление, подходы к которому и полученные результаты также имеют силу для других областей. Наиболее широкое распространение получили производственные системы с сетевым представлением технологических процессов и ограниченными ресурсами: их удельный вес представляет до 80% всех типов производств не только в Украине, но и во всем мире. Такие производства приближаются по сложности к гибким производственным системам [2, 3, 4]. Их особенности значительно усложняют процесс планирования и управления в современных условиях. Эффективное управление предприятиями требует применения современных концепций управления, быстрого реагирования на изменчивую ситуацию, что в свою очередь невозможно без точной и исчерпывающей информации о состоянии производственной, финансовой деятельности и ресурсах предприятия, без налаженных бизнесов-процессов и грамотного управленческого менеджмента. Основным средством эффективного планирования и управления является применение информационных технологий, удовлетворяющих современным требованиям [5].

Иерархическая модель планирования и управления сложными системами с сетевым представлением технологических процессов и ограниченными ресурсами

Общая постановка задачи планирования. Пусть задано множество n комплексов взаимосвязанных работ $J = \{J_1, J_2, \ldots, J_n\}$ (комплекс работ J_i, $i = \overline{1, n}$, в дальнейшем называется заданием). На каждом подмножестве J_i задан частичный порядок ориентированным ациклическим графом. Частичная упорядоченность очевидным образом определяется технологией выполнения комплекса работ. Каждая следующая работа может начаться только по завершению предыдущих работ. Вершины графа отвечают работам, связи указывают на отношения предшествования. Конечные вершины отвечают завершению выполнения заданий. Для каждой вершины j графа известна l_j – *детерминированная* продолжительность выполнения (интегрированный показатель, отображающий выделенные ресурсы – материальные, человеческие, производственные; длительность выполнения каждого задания определяется его критическим путем), для каждой работы $j^i \in I$ (I – множество конечных вершин) задан вес $\omega_i = \omega_j{}^i$; для отдельных заданий задан директивный срок окончания d_i. Величина

веса определяется потенциальной сложностью, важностью и неоднозначностью (для работ, связанных с необходимостью получения нового научного решения) выполнения тех работ, без которых в целом задание не может быть выполнено. Для выполнения работ применяется множество ограниченных ресурсов. Совокупность ресурсов и исполнителей разделено на отдельные, достаточно автономные модули – *мультиресурсы* (мультиресурс – устойчивая группа совместно работающих ресурсов – например, бригада, группа однотипного оборудования, однопрофильное подразделение). Мультиресурсы могут находиться как в одной, так и в разных организациях.

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

Главная цель планирования в условиях рынка – максимизация прибыли предприятия. Поэтому максимизация прибыли является общим критерием оптимальности для всех уровней модели планирования. Прибыль рассчитывается как планируемый доход от реализации всех изделий (выполнения всех заданий) минус затраты *З* (все издержки), очевидным образом рассчитываемые по оптимальному расписанию. В максимизацию прибыли оптимизация по *З* не входит.

В модели рассматриваются следующие критерии оптимальности и их комбинации.

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

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

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

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

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

отсюда критерий оптимальности:

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

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

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

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

ω_i – доход от выполнения i-го задания, если оно выполнено точно в срок. Критерий оптимальности: $\max\left\{\sum_{i=1}^{n}\omega_i U_i\right\}$ (*Критерий 2*);

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

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

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

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

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

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

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

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

Величина $\omega_i \max(0, C_i - d_i)$ – уменьшение прибыли P в случае выполнения задания i с запаздыванием $C_i - d_i$. Решение по выполнению или отказе от выполнения таких заданий принимается менеджером.

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

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

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

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

(*Критерий 5*). Величина $\omega_i'' \max(0, C_i - d_i)$ – уменьшение прибыли P в случае выполнения задания i с запаздыванием $C_i - d_i$. Решение по выполнению или отказе от выполнения таких заданий принимается менеджером.

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

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

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

$\max\left\{\sum_{i=1}^{n}\omega_i U_i\right\}$ (*Критерий 6*).

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

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

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

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

Величина $\omega_i |C_i - d_i|$ – уменьшение прибыли P в случае выполнения задания i с запаздыванием $C_i - d_i$. Решение по выполнению или отказе от выполнения таких заданий принимается менеджером.

Назовем *агрегированной работой* совокупность работ, выполняемых в одном мультиресурсе в рамках одного захода в мультиресурс по одному заданию.

Задачи по критериям 1–7 решаются при следующих ограничениях:

- длительность выполнения каждого задания, а также агрегированной работы, определяется его критическим путем;

общие агрегированные работы разных заданий лежат на их критических путях и выполняются в одном мультиресурсе;

агрегированная работа не передается в другие мультиресурсы до ее полного завершения.

Задачи календарного планирования по критериям 1–7 принадлежат к классу *NP*-трудных. Для решения задачи планирования и управления сложными системами по указанным критериям разработана общая методология, представленная ниже.

Описание модели. Иерархическая модель планирования и управления сложными системами, учитывающими сетевое представление технологических процессов и ограниченные ресурсы, состоит из трех уровней (рис. **Ошибка! Источник ссылки не найден.**): прогнозного, согласованного и точного планирования. В основу представленной ниже модели планирования положен созданный на основе новой конструктивной теории математический аппарат решения труднорешаемых задач календарного планирования за критериями максимизации прибыли предприятий [**Ошибка! Источник ссылки не найден.**, **Ошибка! Источник ссылки не найден.**].

Поставленная задача формирования согласованных планов выполнения работ с привязкой к ресурсам потребовала создать распределенную систему построения планов для каждого уровня управления. При этом обеспечивается четкая взаимосвязь решений, принятых на каждом уровне. Решение задачи МВМ на верхнем уровне управления является входной информацией для эффективного решения задач планирования по разным критериям оптимальности на низших уровнях иерархии. Это позволило создать систему взаимосвязанных алгоритмов, что сделало возможным решение проблемы планирования в комплексе.

В соответствии с трехуровневой моделью, построение распределения работ по ресурсам осуществляется в три этапа. Первый этап состоит в построении агрегированной модели. Если какие-либо работы выполняются в одном мультиресурсе в рамках одного захода в мультиресурс по одному заданию, то они агрегируются в одну агрегированную работу. Продолжительность выполнения агрегированной работы определяется ее критическим путем. Для каждого комплекса работ определяется критический путь выполнения агрегированных работ. На основе агрегированной информации строится граф на критических путях заданий. Вершины полученного графа – это агрегированные работы, дуги отображают связи между мультиресурсами, регламентирующие технологию выполнения заданий.

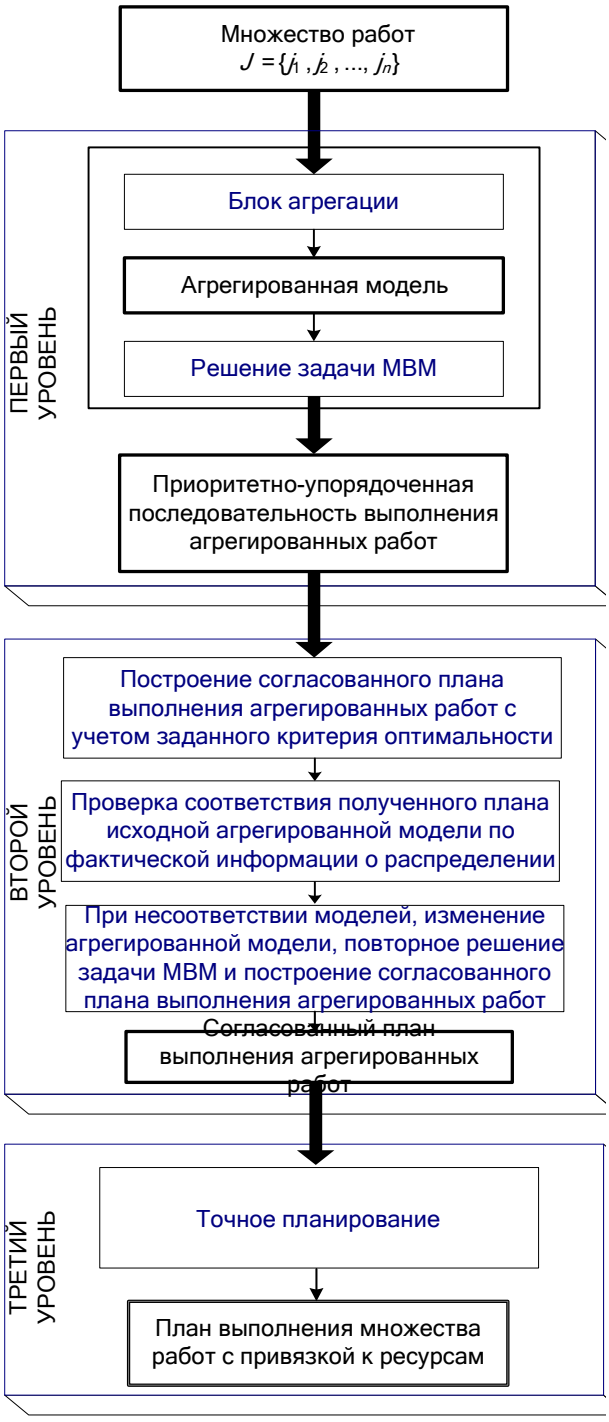


Рис. 1. Функциональная схема иерархической модели планирования и управления сложными системами

Некоторые работы, принадлежащие разным заданиям, требуют выполнения в специализированных уникальных мультиресурсах, желательно в рамках одного захода в мультиресурс. В этом случае при выполнении некоторых условий, описанных ниже, формируется объединенная агрегированная работа,что на графе связности отображено общими вершинами. Для определения приоритетов заданий при построении согласованного плана выполнения заданий соответственно критериям оптимальности, важным является решение на первом уровне задачи МВМ для случая, когда весовые коэффициенты всех вершин графа связности, кроме конечных, равны нулю (см. [**Ошибка! Источник ссылки не найден.**]). В результате решения этой задачи формируется приоритетно-упорядоченная последовательность, определяющая очередность запуска агрегированных работ на выполнение.

Примечание. Построение графа на критических путях заданий и решение задачи МВМ на первом уровне модели реализуется для каждого из семи сформулированных функционалов, подробно это описано ниже.

Второй этап заключается в построении согласованного плана выполнения заданий с учетом указанных выше критериев оптимальности. Полученные на первом уровне приоритеты агрегированных работ служат дополнительной информацией, позволяющей значительно повысить эффективность полученных решений. Полученный план проверяется на соответствие исходной агрегированной модели, при необходимости модель корректируется, после чего план выполнения заданий строится заново.

Процедуры третьего уровня позволяют в соответствии с полученным планом для агрегированных работ построить распределение работ по ресурсам (так называемое точное планирование). На этом уровне решаются задачи оптимизации по критериям 1–7 как для одного, так и для параллельных приборов [**Ошибка! Источник ссылки не найден.**].

Ключевые решения, принятые при разработке алгоритмического обеспечения планирования и управления иерархической модели планирования и управления сложными объектами

1. Длительность выполнения каждого задания определяется его критическим путем.

Это связано с одним из основных критериев составления календарных планов – минимизацией длительности производственного цикла выполнения заданий. Достоинство этого критерия – в том, что с сокращением календарного периода выполнения производственной программы происходит обычно:

а) уменьшение объема незавершенных работ;

б) сокращение длительности выполнения заданий;

в) уменьшение простоев и, тем самым, увеличение производственной мощности;

г) повышение рентабельности производства;

2. Объединение при построении графа на критических путях заданий агрегированных работ, требующих специализированного мультиресурса для их одновременного выполнения, если при таком объединении в зависимости от типа выполняемых работ сокращаются стоимость и время их выполнения, а в производственных системах в «общие» агрегированные работы также объединяются агрегированные работы, для которых требуется достаточно большое время на переналадку оборудования, что приводит, в свою очередь, к сокращению длительности производственного цикла. На графе на критических путях заданий это объединение отражено «общими вершинами».

Разработаны эвристические правила построения «общих вершин», учитывающие критерии оптимальности, приоритеты заданий, которым принадлежат объединяемые агрегируемые работы, директивные сроки и прогнозируемое время поступления агрегированных работ на выполнение в данный мультиресурс.

3. Построение приоритетно-упорядоченной последовательности агрегированных работ, определяющей очередность запуска их на выполнение по заданным критериям оптимальности (1–7).

Известно [8], что последовательность, оптимальная по критерию МВМ, является приоритетно-упорядоченной. Для определения очередности σ запуска агрегированных работ на выполнение в соответствии с приоритетами заданий на первом уровне модели решается задача МВМ для случая, когда весовые коэффициенты всех вершин, кроме конечных, равны нулю. Предложен эффективный эвристический алгоритм [**Ошибка! Источник ссылки не найден.**], основанный на точном

ПДС-алгоритме решения этой задачи в общем случае, когда все веса различны, приведенном в [8]. Разработанный эвристический алгоритм позволяет за приемлемое время решать реальные практические задачи с десятками тысяч переменных.

Приоритеты, определяющие очередность выполнения агрегированных работ, служат основной информацией при разработке алгоритмов планирования второго и третьего уровней модели.

4. Общая структура алгоритмического обеспечения иерархической модели планирования и управления.

Алгоритмы решения задач 1–7 имеют общую структуру, приведенную в [**Ошибка! Источник ссылки не найден.**]. Это позволило создать единый эффективный эвристический мега-алгоритм решения многоэтапных задач календарного планирования по семи различным критериям оптимальности.

5. Три алгоритма распределения для реализации согласованного планирования.

Для реализации согласованного планирования на втором уровне разработаны следующие алгоритмы распределения [**Ошибка! Источник ссылки не найден.**]:

а) построение компактных расписаний (алгоритм 1);

б) построение незадерживающих расписаний (алгоритм 2);

в) построение расписаний, обеспечивающих выполнение в заданные директивные сроки заданий с наивысшим приоритетом (алгоритм 3).

В компактных и незадерживающих календарных планах [**Ошибка! Источник ссылки не найден.**] минимизируются простои между выполняемыми агрегированными работами и, следовательно, минимизируется длительность прохождения заданий в системе. В зависимости от выбранного критерия оптимальности применяется один из разработанных алгоритмов либо их комбинация.

Основные эвристики, используемые при решении задач первого уровня иерархической модели планирования и управения

Постановки задач планирования, включенные в иерархическую модель, содержательно связаны между собой. Общая постановка и ограничения этих задач приведены выше. Рассмотрим агрегированные модели первого уровня и основные эвристики согласованного планирования на втором уровне для каждой из задач 1–7 [**Ошибка! Источник ссылки не найден.**].

Примечание 1. Как показано выше, для задачи 1 агрегированной моделью первого уровня является задача МВМ. Для задач 2–7 в качестве агрегированной модели первого уровня выбирается специальным образом построенная аппроксимированная задача МВМ. Ниже обосновывается для каждой из задач 2–7 целесообразность такой аппроксимации и рассматриваются особенности решения каждой задачи на первом и втором уровне.

Примечание 2. В задачах 1–7 действует общее ограничение, согласно которому длительность выполнения каждого задания и агрегированных работ определяется критическим путем. Ниже i и j_i обозначает номер задания в соответствии с индексацией, заданной функционалом; $j_{[g]}$ – номер работы, стоящей в допустимом расписании на позиции g.

Задача 1. Критерий оптимальности: максимизация суммарной прибыли предприятия в случае отсутствия директивных сроков:

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

где $\omega_i(T)$ – весовой коэффициент задания i, определенный экспериментальным путем; T – плановый период; C_i – момент окончания выполнения задания.

Построение графа на критических путях заданий (система планирования представляется в виде одного станка). Общие для критических путей вершины определяются для агрегированных работ, требующих выполнения в уникальных мультиресурсах и принадлежащих тем заданиям, приоритеты которых отличаются на заданную величину Δ. Объединение вершин выполняется, если разность длительностей пути от этих вершин до конечной вершины не более, чем на время переналадки в мультиресурсе (для производственных проектов) либо если при таком объединении сокращаются длительности прохождения объединяемых заданий в системе.

Задача 1 сводится к задаче МВМ, как показано выше на примере критерия F_1. Приоритетно-упорядоченная последовательность σ*, полученная в результате решения задачи

МВМ (алгоритм решения см. [**Ошибка! Источник ссылки не найден.**]).

Задача 2. Для всех заданий введены директивные сроки d_i, которые не могут быть нарушены (планирование «точно в срок»). Критерий оптимальности:

$$\max \sum_{i=1}^{n} \omega_i U_i, \text{ где } U_i = \begin{cases} 1, C_i = d_i \\ 0, C_i \neq d_i \end{cases},$$

ω_i – доход от выполнения i-го задания, если оно выполнено точно в срок.

При построении графа на критических путях заданий при определении общих вершин вводится дополнительное к задаче 1 условие: общие вершины формируются для заданий с равными директивными сроками или такими, что отличаются на заданную величину. В общие вершины объединяются агрегированные работы, директивные сроки которых (определенные как директивный срок задания минус длительности выполнения работ, следующих за назначаемым заданием по графу связности) также равны или отличаются на заданную величину.

Построение аппроксимирующей задачи МВМ для определения очередности запуска агрегированных работ на выполнение основано на следующих соображениях. Сама задача МВМ строится путем замены функционала $\max \sum_{i=1}^{n} \omega_i U_i$ на функционал $\min \sum_{i=1}^{n} \omega_i \cdot C_i$, где C_i – момент окончания выполнения i-го задания.

Обоснование. 1) Рассмотрим частный случай, когда задания являются независимыми. Имеет место следующее утверждение.

Утверждение 1 (Смит, [**Ошибка! Источник ссылки не найден.**]). Минимальное значение суммарной взвешенной длительности прохождения (взвешенной длительности производственного цикла – в наших терминах $\min \sum_{i=1}^{n} \omega_i \cdot C_i$) независимых заданий в системе с одним станком достигается при расписании, для которого выполняется $\omega_1/l_1 \geq \omega_2/l_2 \geq \ldots \geq \omega_n/l_n$, где ω_i и l_i – соответственно вес и длительность i-го задания в оптимальном расписании.

Таким образом, первым выполняется задание, у которого отношение стоимости (в терминах задачи 2) к длительности является максимальным, т.е. максимальной является удельная усредненная стоимость на единицу времени производственного цикла. Таким образом, в задаче 2 логично первым реализовать выполнение задания с максимальной удельной усредненной стоимостью на единицу производственного цикла ω_1/l_1 (максимально эффективное использование времени производственного цикла, затем задание с удельной усредненной прибылью ω_2/l_2 и т.д. Альтернативой этой эвристике является лишь полный перебор вариантов назначения заданий на выполнение.

2) Общий случай. Ограничение на выполнение заданий задано построенным ориентированным графом на критических путях. В этом случае оптимальным решением задачи МВМ является p-упорядоченное расписание (см. главу 2), в котором последовательность заданий разбита на множества максимальных приоритетов $G_i, i = \overline{1,k}$, $p(G_1) \geq p(G_2) \geq \ldots \geq p(G_k)$. Таким образом, если первыми выполняются работы оптимального расписания, соответствующие множеству G_1, этим гарантируется, что на начальном отрезке производственного цикла длительностью $\sum_{i \in G_1} l_i$ выполняются задания, для которых достигается максимальная усредненная стоимость на единицу времени производственного цикла: $p(G_1) = \sum_{i \in G_1} \omega_i \Big/ \sum_{i \in G_1} l_i$.

Обоснование использования оптимального расписания задачи МВМ в общем случае для задачи 2 аналогично рассмотренному выше частному случаю с учетом замены термина «максимальная удельная стоимость на единицу времени производственного цикла» на термин «максимальная усредненная удельная стоимость на единицу времени производственного цикла».

Примечание. Рассмотрим произвольное допустимое расписание, заданное на множествах максимального приоритета $G_1, G_2, \ldots, G_k$. Ему полностью соответствует приведенное выше обоснование для оптимальной последовательности. Однако на втором этапе системы планирования используется порядок выполнения заданий, соответствующий оптимальному расписанию, т.к. у него усредненная стоимость на единицу времени производственного цикла любой начальной под-

последовательности любого множества максимального приоритета не меньше, чем соответствующее значение усредненной стоимости на единицу времени производственного цикла для произвольного допустимого расписания, заданного на множествах максимального приоритета.

Задача 3. Для некоторых заданий $i \in \overline{1,k}$ заданы директивные сроки, которые не могут быть нарушены. Для остальных изделий $d_i = 0$. Критерий оптимизации:

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

где $U_i = \begin{cases} 1, C_i = d_i \\ 0, C_i \neq d_i \end{cases}$, ω_i – доход от выполнения i-го задания, если оно выполнено точно в срок; $\omega_i(T)$ – весовой коэффициент задания i (имеет тот же смысл, что и в задаче 1).

Аналогично выкладкам, приведенным для задач 1 и 2, очередность запуска агрегированных работ на выполнение определяется в результате решения следующей задачи МВМ:

$$\min\left(\sum_{i=1}^{k} \omega_i' \cdot C_i + \sum_{i=k+1}^{n} \omega_i'' \cdot C_i\right) = \min \sum_{i=1}^{n} \omega_i \cdot C_i,$$

где ω_i' имеет тот же смысл, что и в задаче 2, а ω_i'' – тот же смысл, что и в задаче 1. Очевидно, $\omega_i = \omega_i' \ \forall i = \overline{1,k}; \quad \omega_i = \omega_i'' \ \forall i = \overline{k+1,n}$.

Построение графа на критических путях заданий выполняется аналогично задаче 1.

Задача 4. Для всех заданий $i \in I$ введены директивные сроки d_i. Необходимо выполнить n заданий, минимизируя суммарное взвешенное запаздывание выполнения заданий относительно директивных сроков:

$$\min \sum_{i=1}^{n} \xi_i \max(0, C_i - d_i).$$

где ξ_i – штраф за запаздывание окончания выполнения i-го задания относительно директивного срока на единицу времени.

Построение графа на критических путях заданий выполняется аналогично задаче 1.

Аппроксимирующая задача МВМ имеет вид:

$\min \sum_{i=1}^{n} \omega_i \cdot C_i$, где $\omega_i = \xi_i$ – штраф за запаздывание момента C_i окончания выполнения i-го задания относительно директивного срока d_i на единицу времени. Ограничения на выполнение заданий одним прибором заданы ориентированным графом на критических путях (совпадает с построенным в задаче 1). Представим оптимальное расписание задачи МВМ в виде $\sigma^* = \{j_{l_1}, j_{l_2}, ..., j_{l_n}\}$.

Использование порядка выполнения заданий, определяемого оптимальной последовательностью σ^*, обосновывается аналогично приведенному в задаче 2. Необходимо только термин «усредненная стоимость задания на единицу времени производственного цикла» заменить на термин «усредненная величина штрафа на единицу времени производственного цикла» – в соответствии с содержательной трактовкой функционала задачи 4.

В связи с этим логично на втором этапе первым ставить на выполнение максимально близким к его директивному сроку задание j_{l_1} (точнее, представляющая задание j_{l_1} последовательность агрегированных работ, заданных на ориентированном подграфе). В этом случае штраф $\xi_{l_1} \max(0, C_{l_1} - d_{l_1})$ является нулевым либо минимально возможным.

Вторым максимально близким к своему директивному сроку ставится на выполнение задание j_{l_2} (представляющая задание j_{l_2} последовательность агрегированных работ, заданных на ориентированном подграфе). В этом случае можно ожидать, что штраф $\xi_{l_2} \max(0, C_{l_2} - d_{l_2})$ будет нулевым либо близким к минимально возможному с учетом того, что первым на выполнение было поставлено задание j_{l_1}, и т.д.

Задача 5. Постановка задачи соответствует постановке задачи 4. Введено дополнительное условие: для некоторых заданий $i \in \overline{1,k}$ директивные сроки не могут быть нарушены. Критерий оптимальности:

$$\max\Big(\sum_{i=1}^{k} \omega_i U_i - \sum_{i=k+1}^{n} \xi_i \max(0, C_i - d_i)\Big),$$

где $U_i = \begin{cases} 1, C_i = d_i \\ 0, C_i \neq d_i \end{cases}$, где ω_i – доход от выполнения i-го задания, если оно выполнено точно в срок; ξ_i – штраф за запаздывание окончания выполнения i-го задания относительно директивного срока на единицу времени.

Очередность запуска агрегированных работ на выполнение определяется решением следующей задачи МВМ:

$$\min\left(\sum_{i=1}^{k}\omega_i'\cdot C_i+\sum_{i=k+1}^{n}\omega_i''\cdot C_i\right)=\min\sum_{i=1}^{n}\omega_i\cdot C_i,$$

где ω_i' имеет тот же смысл, что и в задаче 2, а ω_i'' – тот же смысл, что и в задаче 4. Очевидно, $\omega_i=\omega_i'\ \forall i=\overline{1,k};\quad \omega_i=\omega_i''\ \forall i=\overline{k+1,n}$.

Построение графа на критических путях заданий выполняется аналогично задаче 1.

Примечание. Задания, у которых директивные сроки существенно больше других, необходимо отнести к категории заданий, для которых директивный срок не может быть нарушен.

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

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

где ω_i – доход от выполнения i-го задания, если оно выполнено без запаздывания относительно директивного срока.

Очередность запуска агрегированных работ на выполнение определяется в результате решения следующей задачи МВМ: $\min\sum_{i=1}^{n}\omega_i\cdot C_i$.

Обоснование этой процедуры аналогично задаче 2.

В отличие от задачи 1, при построении графа на критических путях заданий при определении общих вершин вводится дополнительное условие: общие вершины формируются для заданий с равными директивными сроками или такими, которые отличаются на заданную величину. В общие вершины объединяются работы, директивные сроки которых (определенные как директивный срок задания минус длительности выполнения работ, следующих за назначаемой работой по графу связности) также равны или отличаются на заданную величину.

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

$$\min\ \sum_{i=1}^{n}\ \xi_i\,|C_i - d_i|,$$

где ξ_i – штраф за отклонение момента окончания выполнения i-го задания от директивного срока на единицу времени.

Аналогично задаче 4, очередность запуска агрегированных работ на выполнение определяется решением задачи МВМ: $\min\sum_{i=1}^{n}\omega_i\cdot C_i$, где $\omega_i=\xi_i$ – вес задания i. Граф на критических путях строится аналогично задаче 6.

Примечание 1. На самом деле описанный здесь подход решает не семь, а тридцать одну ($\sum_{m=1}^{5}C_5^m$) задачу планирования – пять базовых функционалов (задачи 1, 2, 4, 6, 7), а также все возможные их комбинации. Такая возможность проиллюстрирована на задачах 3 и 5. Во всех случаях, когда, аналогично задачам 3 и 5, задания необходимо выполнить точно в срок, определяется разность между доходом от выполнения задания точно в срок и уменьшением значения функционала в случае исключения этого задания из расписания.

Примечание 2. Смысл аппроксимации задач планирования первого уровня соответствующими задачами МВМ обосновывается следующими причинами: 1) глубокая связь между этими задачами, проиллюстрированная выше; 2) создание эффективного приближенного алгоритма [**Ошибка! Источник ссылки не найден.**] для решения задачи МВМ первого уровня, базирующегося на идеях ПДС-алгоритма для задачи МВМ [8].

Эвристики, используемые при реализации второго уровня модели

На втором уровне модели строится согласованный план выполнения заданий с учетом выбранного критерия оптимальности. Назначение агрегированных работ осуществляется по следующим принципам.

Для задач без директивных сроков (критерий 1) задания назначаются по алгоритму 1 или 2 (построение компактных и незадерживающих календарных планов) в соответствии с приоритетами, полученными на первом уровне. Выбор алгоритма 1 или 2 осуществляется в зависимости от типа исходных данных:

если длительности заданий отличаются несущественно (разница не превышает длительности переналадок в мультиресурсах), выбирается алгоритм 2, в противном случае алгоритм 1. В таком случае вначале выполняются заданий с наивысшим приоритетом, что позволяет максимизировать прибыль предприятия по всем изделиям.

В задачах с директивными сроками (критерии 2, 4, 5, 6, 7) задания назначаются по алгоритму 3 в соответствии с приоритетом: наиболее высокоприоритетные задания встраиваются в свой директивный срок раньше. Это обеспечивает выполнение в заданный директивный срок в первую очередь заданий с максимальным приоритетом и, следовательно, позволяет максимизировать суммарную прибыль предприятия.

В задачах со смешанными критериями (3, 5) при требовании обязательного выполнения заданий с директивными сроками, они распределяются в первую очередь по алгоритму 3, остальные задания распределяются по алгоритму 1 или 2. В реальных условиях возможна ситуация, что задание, не выполненное в свой директивный срок, выгоднее исключить из выполнения, если при этом значение функционала улучшится. В этом случае эвристика модифицируется следующим образом. Анализируется полученный согласованный план и запоминается значение оптимизируемого функционала. Находим задание $j_{[q]}$ с минимальным директивным сроком. Проверяем условие: если доход от выполнения задания $j_{[q]}$ при выполнении в свой директивный срок больше, чем выигрыш от исключения задания $j_{[q]}$ по заданиям на интервале $\overline{q+1,t}$, для которых директивный срок не задан, то задание $j_{[q]}$ не исключается из выполнения, в противном случае повторно выполняются алгоритмы распределения без задания $j_{[q]}$. Полученное значение функционала сравнивается с запомненным. Аналогичные процедуры выполняются для всех заданий с директивными сроками, пока не будет рассмотрена вся последовательность.

При таком распределении без нарушения директивных сроков (или максимально близко к ним) выполняются наиболее выгодные задания. Для остальных заданий минимизируется отклонение от директивного срока, в результате максимизируется суммарная прибыль предприятия.

Реализация третьего уровня модели

На третьем уровне модели строится полный план выполнения работ с привязкой к ресурсам (точное планирование). На этом уровне решаются задачи как для одного, так и для параллельных станков в случае независимых или взаимосвязанных заданий по критериям 1–7.

При построении алгоритмов планирования используются приоритеты запуска заданий на выполнение, полученные при решении задачи МВМ на первом уровне модели, а также алгоритмы распределения, описанные выше. Эвристики, используемые при реализации алгоритмов планирования третьего уровня модели, не приводятся, т.к. их конструирование не представляется сложным на основании уже полученных результатов, и они не имеют дополнительной научной ценности.В алгоритмическое обеспечение третьего уровня включены ПДС-алгоритмы задач «Минимизация суммарного запаздывания при выполнении независимых заданий на одном приборе» [9], «Минимизация суммарного запаздывания при выполнении независимых заданий с равными весами и общим директивным сроком параллельными приборами» [10] (они являются частными случаями задачи 4) и «Минимизация суммарного штрафа как за опережение, так и за запаздывание относительно директивных сроков при выполнении независимых заданий одним прибором» [11] (частный случай задачи 7).

На основании приведенного в [11] приближенного алгоритма и методологии построения ПДС-алгоритмов, возможна разработка ПДС-алгоритма решения задачи МОЗ.

Выводы. В существующих системах планирования используется стандартная система базовых алгоритмов решения многих классов задач, что уже не отвечает требованиям эффективного планирования и управления в современных условиях. В статье показан комплекс последовательных взаимосвязанных математических моделей планирования в сложных организационно-производственных системах с сетевым представлением технологии и ограниченными ресурсами, система новых взаимосвязанных алгоритмов решения задач планирования по разным критериям оптимальности, позволяющая эффективно решать

задачи планирования сложных систем в комплексе. В отличие от существующих методов планирования, лучшие из которых содержат линейную или случайную комбинацию разных правил предпочтения, не гарантирующих качества полученных решений, в процессе решения задачи планирования определяется стратегия поиска глобального оптимума, что позволяет получить решения, близкие к оптимальным. В методологии учтена ограниченность мощностей и фактическая нагрузка при выполнении работ, что позволяет строить осуществимые детальные расписания, равномерно загружать ресурсы и минимизировать производственный цикл выполнения работ.

Созданный комплекс моделей и алгоритмов может быть использован для планирования в различных прикладных областях, включая различные типы производств, строительство и управление проектами.

Список литературы

1. 2. Kovacs A. Novel Models and Algorithms for Integrated Production Planning and Scheduling: Ph.D. Thesis. Computer and Automation Research Institute: Budapest,– 2005.
3. 4. Емельянов С.В. Управление гибкими производственными системами. Модели и алгоритмы.– Л.: Машиностроение; Берлин: Техник,– 1987.– 364 с.
5. 6. Многоуровневая система оперативного управления ГПС в машиностроении / С.А.Соколицын, В.А.Дуболазов, Ю.Н.Демченко; под общ.ред. С.А.Соколицына. – СПб: Политехника,– 1991. – 208 с.
7. 8. Петров В.А., Масленников А.Н., Осипов Л.А. Планирование гибких производственных систем.– Л.: Машиностроение, ЛО,– 1985.– 182 с.
9. 10. Требования к созданию систем производственного планирования и управления сложными объектами, имеющими сетевое представление технологических процессов и ограниченные ресурсы / А.А.Павлов, Е.Б.Мисюра, О.В.Мельников, О.В.Щербатенко, В.В.Михайлов // Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. К.: “ВЕК+”, 2007.– №46.– С.3-12
11. 12. Конвей Р.В., Максвелл У.Л. Теория расписаний.– М.: Наука, 1975.– 359 с.
13. 14. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с.
15. 16. Павлов А.А. Алгоритмическое обеспечение сложных систем управления / К.: Выща школа, 1989. – 162 с.
17. 18. Павлов А.А., Мисюра Е.Б. Эффективный точный ПДС-алгоритм решения задачи о суммарном запаздывании для одного прибора // Системні дослідження та інформаційні технології.– 2004.– №4.– С.30-59
19. 20. Павлов А.А., Мисюра Е.Б. Новый подход к решению задачи "Минимизация
0 суммарного взвешенного опоздания при выполнении независимых заданий с директивными сроками одним прибором" // Системні дослідження та інформаційні технології, – 2002.– №2.– С.3-32
21. 22. Павлов О.А., Місюра О.Б., Мельников О.В. Дослідження властивостей та
1 розв'язання задачі «Мінімізація сумарного штрафу як за випередження, так і за запізнення відносно директивних строків при виконанні незалежних завдань одним приладом» / Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. К.: “ВЕК+”, 2008.– №48.– С.3-6

Поступила в редакцию

УДК 62-82:658.512.011.56

БЕЗНОСИК О.Ю.,
КОТ Д.М.

ЗМЕНШЕННЯ НЕОДНОЗНАЧНОСТІ АЛГОРИТМУ СКОРОЧЕННЯ РОЗМІРНОСТІ RLC-СХЕМ

Пропонується модифікація алгоритму скорочення RLC-схем на основі Y-Δ перетворення, в якій усунуто недоліки, пов'язані з неоднозначністю та часом скорочення. Наведено результати досліджень, які підтверджують ефективність запропонованої модифікації.

Modification of the Y-Δ transformation based RCL-circuit reduction algorithm was proposed that eliminates disadvantages related to the reduction indeterminacy and time. Results of practical experiments that prove efficiency of the proposed modification were represented.

Вступ

Моделювання об'єктів складної фізичної природи (наприклад, об'єктів МЕМС – мікроелектромеханічних систем) може бути зведене до схемотехнічного моделювання завдяки використанню метода електромеханічних аналогій [1]. Нажаль, даний метод має суттєвий недолік: розмірність вихідної системи рівнянь є надзвичайно великою, що унеможливлює використання пакетів схемотехнічного моделювання без попереднього скорочення отриманої математичної моделі.

На сьогодні існує декілька підходів до вирішення проблеми зменшення розмірів математичних моделей МЕМС при мінімальних або таких, що контролюються, втратах точності. Для пакетів схемотехнічного проектування найбільш придатним є метод Y-Δ перетворення [2]. В роботі досліджено алгоритм скорочення математичних моделей RLC-схем на базі Y-Δ перетворення, який відрізняється від існуючих [3, 4] стратегією вибору послідовності виключення вузлів, яка забезпечує мінімальні втрати точності, більш широким набором формул для перерахунку та ідентифікації типів провідностей, які з'являються у процесі виключення вузлів, та методикою їх отримання [5].

Аналіз алгоритму

Для кожного вузла в схемі визначаються дві часові сталі $\tau_{RCi} = C_i / G_i$ та $\tau_{LCi} = \sqrt{C_i / B_i}$, де C_i – сума всіх ємностей, B_i – сума всіх величин, зворотних індуктивностям, G_i – сума всіх провідностей, підключених до вузла *i*.

Вузлова стала часу вузла *i* визначається як $\tau_i = \max(\tau_{RC}, \tau_{LC})$. Вузол вважається швидким, якщо його вузлова стала часу менше визначеного користувачем значення, яке залежить від максимальної частоти кола. Далі відбувається поступове скорочення швидких вузлів в схемі, починаючи з вузлів з найменшим значенням τ. При цьому створюються нові елементи між вузлами, їх кількість становить $k(k-1)/2$, де k – кількість вузлів, зв'язаних з вузлом, що видаляється (рис. 1).

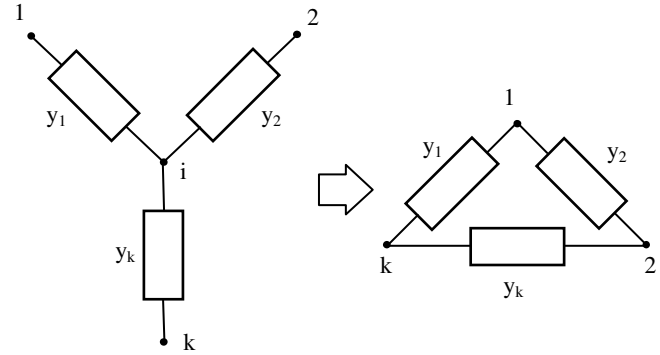


Рис. 1. Створення нових елементів при скороченні вузла i (Y-Δ перетворення)

Алгоритму скорочення на основі Y-Δ перетворення притаманні такі недоліки:

1. Похибка скорочення (рис. 2). В розглянутому алгоритмі похибка складається з методичної та інструментальної. Методична складова похибки, у загальному випадку, зумовлена недосконалістю та неточністю формул, які використовуються в алгоритмі, і виникає при обчисленні номіналів нових елементів, що утворюються при скороченні вузла. Інструментальна складова похибки обумовлена властивостями (або обмеженнями) ЕОМ, на яких виконується алгоритм. Її наявність спричинена тим, що обчислення на ЕОМ виконуються з кінцевою кількістю значущих цифр, обмеженою розрядною сіткою.

2. В даному алгоритмі інструментальна похибка виникає при розрахунку номіналів нових елементів, що додаються при скороченні вузла, та при об'єднанні елементів одного типу (у випадку появи паралельних/послідовних опорів, ємностей або індуктивностей).

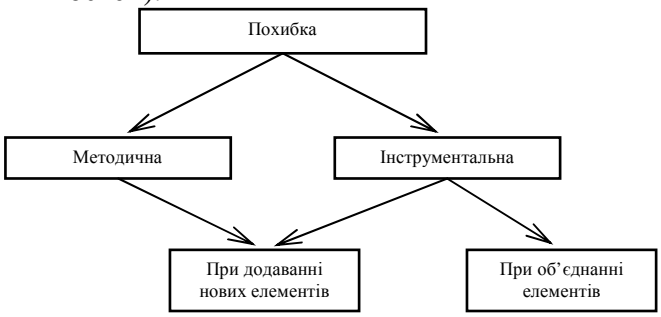


Рис. 2. Похибки базового алгоритму

3. Неоднозначність скорочення. Порядок скорочення вузлів в схемі заміщення визначається значенням часової сталої τ. Проте слід зазначити, що для еквівалентних схем заміщення МЕМС характерною є наявність досить значної кількості вузлів з однаковим значенням τ (на відміну від RLC-схем, отриманих в наслідок екстракції топології). В той же час, кількість під'єднаних до цих вузлів елементів може суттєво відрізнятися. Так, в схемі заміщення мікроакселерометра [6] кількість елементів, під'єднаних до різних вузлів з однаковим значенням τ, коливається від 500 до 972. При використанні базового алгоритму скорочення порядок виключення цих вузлів визначається випадково (порядком їх слідування в описі схеми та алгоритмом сортування, що використовується). Оскільки ж порядок скорочення вузлів впливає на кількість елементів, що створюються, цілком можливо отримати різні результати скорочення однієї і тієї ж схеми заміщення.

4. Велика кількість нових елементів, що створюються в схемі під час скорочення. Із зростом кількості нових елементів збільшується і кількість пов'язаних з їх утворенням арифметичних операцій. Для об'єктів з багатьма ступенями свободи та складною геометрією кількість зв'язків вузла може становити 100 та більше. Причому, під час скорочення щільність схеми і, відповідно, зв'язаність вузлів зростають. Для деяких схем заміщення навіть з невеликою кількістю вузлів ~10^4 та малою початковою середньою щільністю видалення окремих вузлів призводить до появи декількох мільйонів нових елементів, що значно збільшує час скорочення та може повністю нівелювати результат внаслідок зростання методичної та інструментальної похибок.

Модифікація алгоритму скорочення RLC-схем на основі Y-Δ перетворень

Для того, щоб зменшити неоднозначність скорочення і при цьому забезпечити якнайменшу кількість елементів, що додаються, необхідно визначити порядок скорочення вузлів з однаковим значенням часової сталої τ.

Слід взяти до уваги, що кількість елементів, які утворюються при скороченні, залежить не тільки від кількості вузлів, сусідніх до того, що видаляється, але й кількості елементів, з'єднаних з цим вузлом. Виходячи з цього, було запропоновано наступну модифікацію базового алгоритму:

- в першу чергу видаляються вузли із найменшим значенням сталої часу τ;
- якщо декілька вузлів мають однакові сталі часу τ, першим видаляється вузол з найменшою кількістю з'єднаних з ним елементів.

Запропонована модифікація дозволяє практично усунути проблему неоднозначності: порядок скорочення вузлів с однаковим значенням τ залишається випадковим тільки в разі рівності кількості з'єднаних елементів, але для схем заміщення це характеризує рівнозначність вузлів і не призводить до якихось істотних проблем під час скорочення. Крім того, із зменшенням кількості елементів, що утворюються, вочевидь, зменшується і вплив похибки обчислень та час пошуку в структурах даних.

Опис прикладу

Ефективність запропонованої модифікації перевіримо на моделі мікроакселерометра [6], який виконано у вигляді квадратної кремнієвої пластини (мембрани) 1 товщиною 1 мкм із стороною квадрата 50 мкм. Сторони пластини сперті на основу, а решта поверхні є вільною і може прогинатись. В центрі пластини закріплено металевий вантаж 2 у вигляді квадратної пластини із стороною 10 мкм (рис. 3).

Розрахунок власних частот та форм коливань мікроакселерометра виконано методом скінченних елементів за допомогою програмного комплексу ANSYS v10.0 [7]. Для дискретизації пластини використано скінченні

елементи типу SHELL93 прямокутної форми розміром 5×5 мкм. Такі ж елементи використані і для представлення центральної частини, але модуль пружності та густину встановлено рівними відповідним параметрам матеріалу пластини-вантажа. Таким чином, масою та жорсткістю кремнієвої пластини під вантажем було знехтувано. Отриману скінченно-елементну модель акселерометра представлено на рис. 4.

Наступним кроком було отримання повнорозмірних матриць, що описують стан системи, за допомогою програми mor4ansys [8]. Отримані з ANSYS за допомогою mor4ansys матриці були використані для створення еквівалентної електричної схеми заміщення на вхідній мові пакету ALLTED [9].

Отримана еквівалентна схема заміщення мікроакселерометра складається з 1883 вузлів та 62868 елементів.

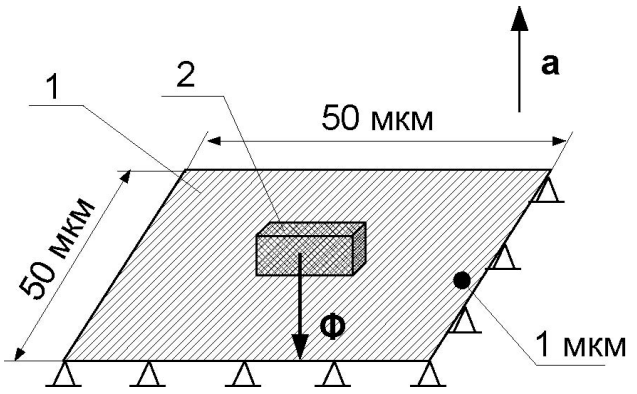


Рис. 3. Схема мікроакселерометра

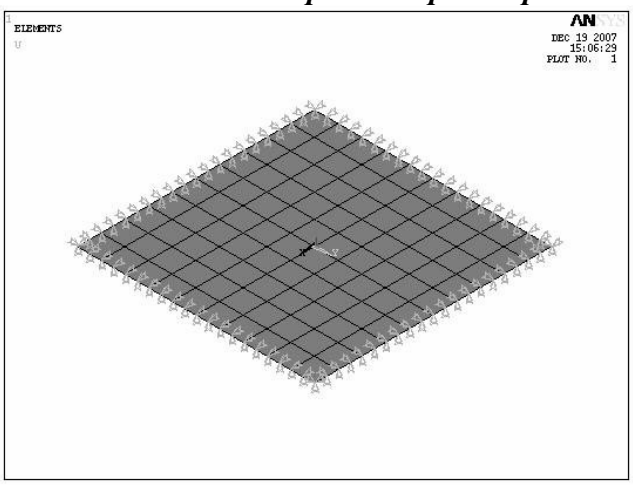


Рис. 4. Скінченноелементна модель мікроакселерометра

Перевірка ефективності запропонованої модифікації базового алгоритму

Ефективність розробленої модифікації в порівнянні з базовим алгоритмом можна оцінити за наступними критеріями:

- значення похибки (відхилення значень власних частот скороченої схеми від отриманих за допомогою ANSYS);
- час скорочення;
- кількість зафіксованих елементів в схемі в процесі скорочення та їх максимальне значення.

Для демонстрації ефективності були проведені такі скорочення схеми заміщення мікроакселерометра:

1. Скорочення базовим алгоритмом.
2. Найгірший випадок скорочення. В цьому випадку при наявності в схемі заміщення вузлів з однаковою часовою константою τ спочатку видаляються вузли з найбільшою кількістю елементів. Така ситуація цілком ймовірна в разі використання базового алгоритму при певному порядку елементів в описі схеми.
3. Скорочення модифікованим алгоритмом.

Скорочення схеми заміщення у всіх випадках виконувалося до 18 вузлів для більш коректного порівняння між собою.

На рис. 5 зображено кількість елементів в схемі на кожній ітерації при скороченні, результати скорочення наведені в табл. 1.

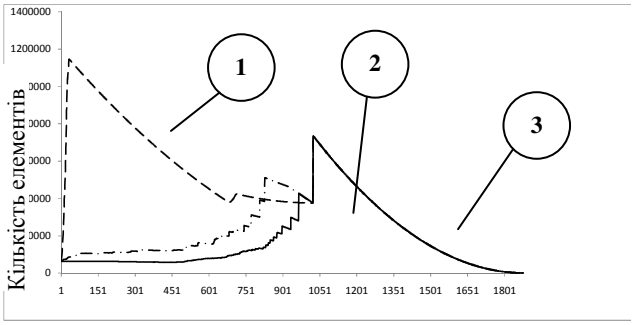


Рис. 5. Кількість елементів на кожній ітерації скорочення схеми заміщення мікроакселерометра:
1 - найгірший випадок скорочення;
2 - скорочення базовим алгоритмом;
3 - скорочення модифікованим алгоритмом

Висновки

За результатами експериментів можна зробити наступні висновки:

1) запропонована модифікація забезпечує зменшення кількості елементів, що утворюються при скороченні. При цьому, кількість елементів, що утворюються при скороченні вузлів з під'єднаними елементами од-

ного типу, на порядок менше, ніж в найгіршому випадку;

2) час скорочення модифікованим алгоритмом менший за час скорочення базовим алгоритмом і набагато менший, ніж час скорочення при найгіршому випадку;

3) зменшення кількості елементів і, як наслідок, зменшення кількості арифметичних операцій, зменшує інструментальну похибку обчислень. (Однакова похибка для всіх розглянутих випадків пояснюється доволі грубим обчисленням значень власних частот.);

4) найбільш ефективним є застосування модифікованого алгоритму для схем заміщення з великим розкидом кількості елементів, що під'єднані до вузлів з однаковим τ. Тому розроблену модифікацію рекомендується використовувати для скорочення схем, які отримані для об'єктів із складною геометрією та великою кількістю вузлів скінченно-елементної моделі.

Табл. 1. Результати скорочення схеми заміщення мікроакселерометра

	Початкове коло (ANSYS)	**Ущільнене коло**		
		Базовий алгоритм	**Найгірший випадок**	**Модифікований алгоритм**
τ_{min}, с	-	$2*10^{-5}$		
Кількість вузлів	1883	18	18	18
Кількість елементів	62826	306	306	306
1-а мода, кГц (похибка, %)	181,36	181,3 (0,03)	181,3 (0,03)	181,3 (0,03)
4-а мода, кГц (похибка, %)	3427,8	4210,85 (22,84)	4210,85 (22,84)	4210,85 (22,84)
Максимальна похибка, %	-	22,84	22,84	22,84
Час, с	-	4136	11178	3840
Максимальна кількість елементів	-	735306	1146774	735306
Кількість створених елементів *	-	105288301	105288301	105288301
Кількість створених елементів **	-	75799616	489783888	40111345

* при скороченні вузлів з під'єднаними елементами різних типів (вноситься методична та інструментальна похибка)

** при скороченні вузлів з під'єднаними елементами одного типу (вноситься тільки інструментальна похибка)

Список посилань

1. Hsu J.T. A rational formulation of thermal circuit models for electro-thermal simulation - Part I: Finite element method / Hsu J.T., Vu-Quoc L. // IEEE Transactions on Circuits and Systems I, Vol. 43. – 1996. – №9. – P. 721-732.
2. Ладогубец В.В. Методы макромоделирования МЭМС / Ладогубец В.В., Безносик А.Ю., Крамар А.В., Финогенов А.Д. // Электроника и связь: Тематический выпуск «Проблемы электроники». – 2008. – 4.1. – № 1-2. – С. 244-248.
3. Sheehan B.N. TICER: Realizable Reduction of Extracted RC Circuits // Digest of Technical Papers IEEE/ACM Proc. of ICCAD. – 1999. – P. 200-203.
4. Chirayu S.A. Realizable RLCK Circuit Crunching / Chirayu S.A., Masud H.C., Yehea I.I. // Proceedings of the 40th conference on Design automation 2003. - Anaheim, CA, USA. - 2003.
5. Ладогубец В.В. Алгоритм уменьшения размерности RLC цепей / Ладогубец В.В., Руденко Ю.А., Ладогубец А.В. // Электроника и связь. – 2004. – №21. – С. 72–74.
6. Петренко А.И. Расчет собственных частот мембранных структур с использованием пакетов схемотехнического проектирования / Петренко А.И., Ладогубец В.В., Безносик А.Ю., Финогенов А.Д., Чкалов А.В. // Радиоэлектроника: Известия вузов. – 2009. – №7. – С. 19-25.
7. Ansys Inc.: http://www.ansys.com/.
8. Rudnyi E. mor4ansys: Generating Compact Models Directly from ANSYS Models / Rudnyi E., Lienemann J., Greiner A., Korvink J.G. // Technical Proceedings of the 2004 Nanotechnology Conference and Trade Show (Nanotech 2004, March 7-11, 2004). – Boston, Massachusetts, USA, vol. 2. – P. 279-282.
9. Petrenko A. ALLTED – a computer-aided engineering system for electronic circuit design / Petrenko A., Ladogubets V., Tchkalov V., Pudlowski Z. – Melbourne: UICEE, 1997. – 205 p.

Поступила в редакцию 11.12.2009

УДК 62-82:658.512.011.56

ЛАДОГУБЕЦ В.В.,
КРАМАР А.В.,
ФИНОГЕНОВ А.Д.

РАСЧЕТ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПАРАЛЛЕЛЬНОГО МЕТОДА ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ НА ОСНОВЕ РАЗНОСТЕЙ ВЫСШИХ ПОРЯДКОВ

Рассмотрены способы определения времени выполнения параллельных методов на мультипроцессорных вычислительных системах. Получены основные соотношения, позволяющие априорно оценить время выполнения параллельного метода численного интегрирования для минимизации времени решения. Показана возможность использования полученных оценок для автоматического выбора метода.

The ways to define execution time of the parallel methods at the multiprocessor computing systems have been considered. The main relations have been obtained which allow estimating execution time of the numerical integration parallel method a priori to minimize solution time.

При разработке параллельных программ важную роль играет возможность предварительной оценки качественных и количественных характеристик реализуемого алгоритма. В теории параллельных вычислений принято выделять ряд основных количественных характеристик, таких как коэффициент ускорения параллельного алгоритма, коммуникационная трудоемкость, коэффициент масштабируемости и т.д. [1,2].

На практике для сложных вычислительных алгоритмов трудно оценить количество операций. Например, при решении задач численного интегрирования в пакетах схемотехнического проектирования, количество операций будет определяться как размерностью математической модели, так и моделями элементов схемы, видом задающих источников, наличием или отсутствием учета нестандартных ситуаций, связанных с демпфированием и т.д. При этом для одной и той же задачи при изменении начальных условий процесс решения может использовать различные ветви алгоритма. Эти факторы делают оценки, основанные на определении времени отдельных коммуникационных и вычислительных операций, применимыми лишь для задач, использующих «мелкозернистый» параллелизм. К таким задачам обычно относят решения систем линейных уравнений прямыми методами, операции умножения матриц и векторов и т.п.

Для задач, использующих «крупнозернистый» параллелизм, т.е. реализующих вычисления отдельных ветвей алгоритмов или использующих различные методы решения одной и той же задачи, целесообразно производить оценку по величинам, которые определяют тип передачи:

- $t_{1\text{-}1}(V)$ – время передачи данных объемом V по типу «один к одному»;
- $t_{p,1\text{-}\infty}(V)$ – время передачи данных объемом V по типу «один ко многим», где количество адресатов определяется значением p;
- $t_{p,\infty\text{-}1}(V)$ – время передачи данных объемом V по типу «многие к одному», где p – количество сообщений к одному адресату;
- $t_{p,\infty\text{-}\infty}(V)$ – время передачи данных объемом V по типу «многие ко многим», где p – количество принимающих/передающих адресатов.

Последнего типа передачи при создании параллельного алгоритма желательно избегать, т.к. подобные операции увеличивают нагрузку на средства коммуникации и усложняют контроль выполнения алгоритма.

Значения времени для различных типов передачи и различных объемов передаваемых данных могут быть определены до выполнения алгоритма, и использоваться для предварительной оценки качества реализации на конкретной мультипроцессорной вычислительной системе (МВС). Если объем передаваемых данных каким-либо образом зависит от размерности задачи (что чаще всего встречается на практике), целесообразно построить таблицу зависимости времени передачи от передаваемого объема данных. Наибольшее внимание необходимо уделять точности определения времени передачи небольших объемов данных, т.к. при их пере-

даче значительными оказываются влияния особенностей работы сетевых протоколов, различных типов оперативной памяти и т.д. С увеличением объемов данных их влияние становится менее заметно и для практических оценок ими можно пренебречь.

Оценки времени, основанные на типе передачи данных, не только могут дать информацию о качестве реализации алгоритма, но и позволить модифицировать схемы обмена данными, выбрать оптимальные маршруты передачи, количество процессоров, которые целесообразно использовать для ускорения работы алгоритма или, при фиксированном количестве процессоров, выбрать метод решения и т.д.

Рассмотрим расчет времени выполнения параллельного метода численного интегрирования на основе разностей высших порядков с использованием выборки всех доступных порядков и набора размеров шагов на каждом шаге интегрирования [3], реализованного в составе пакета схемотехнического САПР ALLTED [4] на кластере НТУУ «КПИ» [5]:

$$T_p = T_{p,comp} + T_{p,comm.}, \quad (1)$$

где T_p – общее время реализации параллельного алгоритма на параллельной архитектуре;

$T_{p,comp}$ – время решения задачи заданного размера с использованием параллельного алгоритма на компьютере из p-процессоров без учета операций на обмен (время реализации вычислений); $T_{p,comm.}$ – коммуникационная трудоемкость или время выполнения межпроцессорных операций обмена.

Время решения задачи без учета операций обмена $T_{p,comp}$ включает в себя этапы:

- трансляцию файла задания и формирования мат. модели $T_{мат.модели}$;
- численного интегрирования $T_{интегр.}$;
- обработки результатов и генерации выходных файлов $T_{рез.}$.

$$T_{p,comp} = T_{мат.модели} + T_{интегр.} + T_{рез.} \quad (2)$$

Основная идея параллельного метода заключается в использовании на каждом шаге выборки $[K \times h]$ шагов и порядков и выборе лучших в качестве новых текущих значений. В этом случае, время, затрачиваемое параллельным алгоритмом на процесс численного интегрирования, зависит от времени выполнения следующих операций:

- генерации выборки шагов и порядков $T_{ген}$;
- формирования системы линейных уравнений $T_{фор}$;
- выполнения итераций Ньютона $T_{ин}$;
- оценки результатов и определения следующего базового размера шага $T_{оц}$.

Количество этапов формирования линейных систем, выборки порядков и размеров шага, а также количества оценок лучшего зависит от количества шагов интегрирования. Время выполнения итераций Ньютона зависит от количества итераций на шаге, т.е. качества определения прогнозированного значения.

$$T_{интегр.} = H(T_{фор} + T_{ген} + T_{оц}) + NT_{ин}, \quad (3)$$

где H – количество временных шагов, N – количество итераций Ньютона.

С ростом размера выборки время оценки $T_{оц}$ увеличивается незначительно, т.к. варианты могут быть отброшены еще на первом этапе сравнения с лучшим результатом. Время генерации выборки $T_{ген}$ так же незначительно, а время формирования $T_{мат.модели}$ и время генерации файлов результата $T_{рез.}$ является для конкретной задачи константой.

Таким образом, можно определить время, которое практически не влияет на суммарное время решения или является постоянным, т.е. не зависит от размерности выборки:

$$T_{p,const} = T_{мат.модели} + T_{рез.} + H(T_{ген} + T_{оц}) \quad (4)$$

С учетом (3,4), время реализации параллельного алгоритма $T_{p,comp}$ (2) будет определяться как:

$$T_{p,comp} = T_{p,const} + HT_{фор} + NT_{ин} \quad (5)$$

Т.к. в случае параллельного алгоритма значение N для каждого компонента выборки может лежать в пределах от 1 до $MAXIT$ (максимально допустимое количество итераций Ньютона на шаге), то в качестве N – необходимо учитывать максимальное количество итераций Ньютона, которое было получено на компонентах выборки. Пусть $isi1_n$ – количество итераций Ньютона, полученных на i-й компоненте выборки $[K \times h]$ на n-м шаге интегрирования, при этом размерность выборки равна p – количеству доступных процессоров. В этом случае, суммарное количество итераций Ньютона, выполняемых не одновременно, будет равно:

$$N_{max} = \sum max_n(isi1_i),\ i=1(1)p,\ n=1(1)H \quad (6)$$

С учетом (6), выражение (5) примет вид:

$$T_{p,comp} = T_{p,const} + HT_{фор} + N_{max}T_{ин} \quad (7)$$

Для определения коммуникационной сложности алгоритма $T_{p,comm}$, рассмотрим данные, которые необходимо передавать при

параллельной реализации. Для этого необходимо выделить три типа данных, которые могут использоваться при обмене.

1) *Входные* или *константные данные*, т.е. данные, описывающие параметры решаемой системы и не изменяющиеся на протяжении всей работы алгоритма.

2) *Данные процесса решения*. К ним относятся данные, которые в пределах параллельных блоков не изменяются, но изменяются при анализе лучшего варианта и в дальнейшем необходимы для пересчета ряда параметров.

3) В*ыходные* или *результирующие данные*, к которым относятся результаты работы параллельных блоков и использующиеся для анализа лучшего варианта и расчета данных на следующем временном шаге.

В соответствии с выделенными типами, представим объем этих данных как:

- $V_{ВД}$ – объем входных данных;
- $V_{ДПР}$ – объем данных процесса решения;
- $V_{РД}$ – объем результирующих или выходных данных.

Очевидно, что входные данные нужны для начала работы алгоритма, однако в последствии их передача не требуется, т.е. обмен этими данными производится один раз перед первым шагом работы алгоритма. Соответственно, данные процесса решения требуют только передачи на каждом временном шаге, т.к. их значения изменяются вне параллельных блоков. Для результирующих данных должен происходить только процесс приема на каждом шаге.

Рассчитаем объем данных, которые необходимо будет передавать при решении задачи численного интегрирования. На первом шаге интегрирования параллельным блокам необходимо передать входные данные и данные процесса решения.

$$V_{m,1} = V_{ВД} + V_{ДПР} \quad (8)$$

Начиная со второго и для последующих шагов, объем передаваемых данных в параллельные блоки будет составлять:

$$V_{m,2-\infty} = (H-1) \cdot V_{ДПР} \quad (9)$$

Исключение составляет последний шаг. В случае достижения предела интегрирования, необходима передача сигнала о завершении работы параллельных блоков. Однако, в этом случае, головной программе нет необходимости ожидания, поэтому данная передача практически не будет влиять на время выполнения. Объем возвращаемых данных будет всегда одинаков:

$$V_s = HV_{РД} \quad (10)$$

В данном случае не учитывается неполная нагрузка на процессоры на начальном этапе. Очевидно, что полная выборка $[K \times h]$ будет сформирована по достижению значения порядка K_{max}, однако на практике это чаще всего происходит уже на первых десяти шагах и практически не влияет на оценку времени выполнения или коэффициента эффективности в виду большого (10^3 и более) количества шагов для решения.

В качестве архитектуры параллельной программы удобно выбрать архитектуру «ведущий (master) –подчиненный (slave)», где на slave возлагаются этапы формирования линейной системы уравнений $T_{фор}$ и проведение итераций Ньютона $T_{ин}$. Функции master включают в себя трансляцию входного файла задания и формирование математической модели $T_{мат.модели}$, генерацию выборки $T_{ген}$, оценку результатов для выбора лучшего $T_{оц}$ и формирование выходных файлов результата $T_{рез.}$.

Для оценки времени решения, рассмотрим схему обмена данными, которая не учитывает особенности архитектуры (рис. 1), и проведем ее модификацию для систем с распределенной памятью.

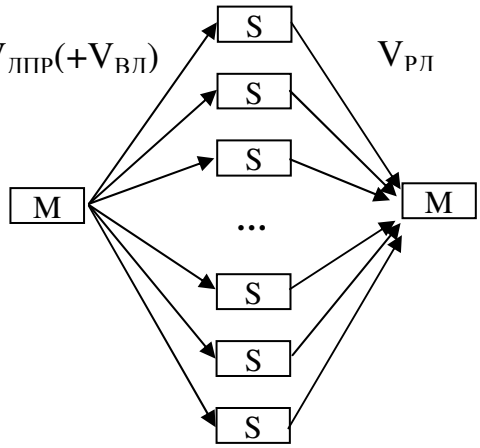


Рис. 1. Схема обмена данными

Для данной схемы обмена данными необходимо определить время выполнения такого типа обмена, как «один к одному», т.е. многократную пересылку данных процесса решения от master к slave и тип передачи «многие к одному», т.е. сбор результирующих данных.

В этом случае $T_{p,comm}$ с учетом (8-10) будет определятся как:

$$T_{p,comm} = pt_{1-1}(V_{m,1}) + p(H-1)t_{1-1}(V_{m,2-\infty}) + H\, t_{p,\infty-1}(V_s)$$

или

$$T_{p,comm} = p(Ht_{1\text{-}1}(V_{ДПР}) + t_{1\text{-}1}(V_{ВД})) + H\, t_{p,\infty\text{-}1}(V_{РД}) \quad (11)$$

Недостатком такой схемы взаимодействия (рис. 1) является многократная пересылка практически одинаковых данных $V_{ДПР}$ каждому параллельному блоку (за исключением значений компонентов выборки $[K \times h]$). Поэтому на практике на этапе запуска slave-программ целесообразно определить уникальные идентификаторы каждой и в дальнейшем передавать весь массив значений выборки $[K \times h]$. Идентификатор slave-программы будет соответствовать номеру компоненты выборки и сделает весь объем передаваемых данных полностью идентичным для всех slave. В этом случае становится возможным использование на каждом шаге не множественной операции пересылки по типу «один к одному», а однократной пересылки «один ко всем».

Передача всего объема результирующих данных от всех slave также имеет ряд недостатков: с ростом размерности задач увеличивается объем требуемой памяти для хранения данных, большая часть из которых (за исключением данных, полученных на компоненте выборки признанной лучшей) в дальнейшем не используется. Кроме того, будет происходить рост объема передаваемых данных, что в свою очередь повлечет увеличение времени на их передачу. В связи с этим представляет интерес использование следующей схемы возврата значений расчетов slave: на первом этапе передаются только данные, которые необходимы для анализа лучшего варианта. После проведения анализа, master инициирует обмен всеми данными только с одним из slave, результаты которого признаны лучшими. Такая схема взаимодействия резко снижает объем передаваемых данных и объем требуемой памяти, но увеличивает количество требуемых обменов. Оценка эффективности по времени такой схемы обмена данными будет зависеть от соотношения времени передачи единицы данных и латентности сети.

Для большинства существующих систем соотношение латентности сети и времени передачи одного байта данных обычно лежит в пределах 10^2-10^3. Компенсация времени дополнительного обмена возможна при уменьшении объема предаваемых данных на 200-2000 байт, что возможно даже для небольших схем. Например, в зависимости от используемых моделей транзисторов, такое уменьшение будет наблюдаться для схем, состоящих из 4-5 транзисторов. В этом случае схема обмена данными примет вид, представленный на рис. 2.

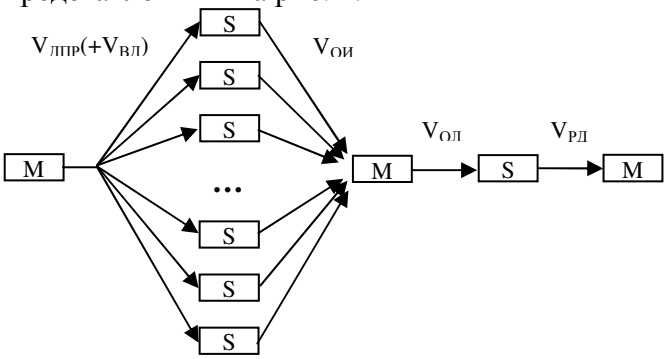


Рис. 2. Схема обмена данными для систем с распределенной памятью

Для такой схемы обмена необходимо определить следующие типы данных:

-$V_{ОИ}$ – объем информации для оценки лучшего (обычно не болем 100 байт);

-$V_{ОЛ}$ – признак того, что получивший данную информацию slave, должен передать все результирующие данные (1 байт).

В этом случае необходимо определение времени выполнение типов передачи по типу $t_{p,1\text{-}\infty}$ – «один ко многим» и $t_{p,\infty\text{-}1}$ – «многие к одному» для p slave, и $t_{1\text{-}1}$ – время выполнения операции передачи данных типа «один к одному».

Очевидно, что время выполнения этих операций будет зависеть от конфигурации, технических характеристик, а также от размещения соответствующих slave-процессов на архитектуре МВС. Для предложенной схемы обмена данными, коммуникационная сложность алгоритма (11) будет определяться следующим образом:

$$T_{p,comm.} = T^1 + (H\text{-}1)T^2 \quad (12)$$

где T^1 – время, затрачиваемое на обмен на первом шаге; T^2 – время, затрачиваемое на обмен на втором и последующих шагах, которые определяются как:

$$T^1 = t_{p,1\text{-}\infty}(V_{ВД} + V_{ДПР}) + t_{p,\infty\text{-}1}(V_{ОИ}) + t_{1\text{-}1}(V_{ОЛ}) + t_{1\text{-}1}(V_{РД});$$

$$T^2 = t_{p,1\text{-}\infty}(V_{ДПР}) + t_{p,\infty\text{-}1}(V_{ОИ}) + t_{1\text{-}1}(V_{ОЛ}) + t_{1\text{-}1}(V_{РД}).$$

В случае использования систем с общей памятью, время решения будет в большей мере зависеть от времени выполнения итераций Ньютона, т.к. отсутствует необходимость в передаче данных. Основными затра-

тами на организацию параллельных вычислений в этом случае будут выступать затраты на копирование экземпляров данных в памяти, т.к. каждый параллельный блок производит вычисления на своей паре *(K,h)*, и, соответственно, не может использовать единый набор данных процесса решения. Входные же данные наоборот остаются неизменными в процессе всего решения и могут использоваться всеми параллельными блоками совместно.

Процесс выбора лучшего в этом случае сводится к определению экземпляра результирующих данных, который необходимо скопировать остальным параллельным блокам (рис. 3).

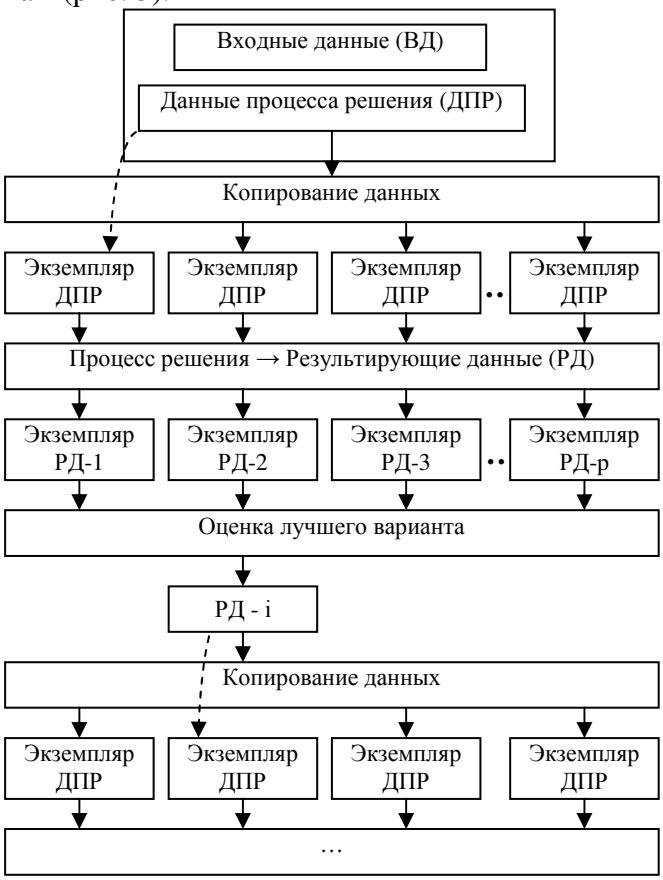


Рис. 3. Схема работы алгоритма с использованием общей памяти

Коммуникационная сложность данной схемы будет в основном определяться временем копирования экземпляров данных процесса решения для *p* параллельных блоков после определения лучшего, т.к. затратами времени на синхронизацию, при решении сложных задач большой размерности можно пренебречь:

$$T_{p,comm} = (H+1) \cdot t_{p,k}(V_{ДПР}) \qquad (13)$$

где $t_{p,k}$ – время копирования экземпляров данных $V_{ДПР}$, а дополнительное копирование связано с начальным заполнением массивов данных.

Оценку полученных соотношений проведем для двух методов, использующих на каждом шаге интегрирования выборку из всех доступных порядков и набор размеров шагов с уменьшением: *h={h,0.95h,0.9h,0.8h,0.7h}*. «Метод 1» для выбора лучшего использует оценку, основанную на максимизации прогнозированного времени $max(T_n+h_{n+1})$ и является примером «задачи на максимум», т.е. для оценки необходимо получение результатов работы всех параллельных блоков. «Метод 2» для выбора лучшего использует оценку, основанную на минимальном количестве итераций Ньютона *min(isi1)* и является примером «задачи на минимум», т.к. для определения лучшего необходимы только данные от одного(нескольких) успешных блоков, закончивших работу с минимальным количеством итераций, что позволяет использовать в качестве количества неодновременно выполняемых итераций Ньютона значение *N*. Для проверки полученных соотношений времени использовались временные характеристики, полученные на кластере НТУУ «КПИ».

В качестве теста использовалась схема FADD32 из тестового набора, предложенного институтом Беркли [6], с параметрами, приведенными в табл. 1.

Табл. 1. Параметры схемы FADD32

Параметры	**FADD32**		
	Базовый	**Метод 1**	**Метод 2**
Количество временных шагов, *S*	17407	9413	11925
Количество итераций Ньютона, *N*	31092	17571	14466
Количество отказанных шагов, *R*	1019	522	399
Максимальное число итераций Ньютона, N_{max}	31092	22961	27743
Объем входных данных, байт	-	271100	
Объем данных процесса решения, байт	-	66080	
Объем результирующих данных, байт	-	66816	-
Объем данных для оценки лучшего, байт	-	32	-
Объем данных обмена с лучшим, байт	-	1	-
Среднее время выполнения одной итерации Ньютона, с		$1077*10^{-6}$	
Среднее время формирования, с		$100*10^{-6}$	

Для выборки размером $p = 35$ в качестве МВС с распределенной памятью использовались пять 8-ми ядерных узлов кластера с равномерным распределением параллельных блоков в узлах. Время выполнения операции «один к одному» производилось с помощью отправки сообщения объемом V байт и ожидания ответа от slave. Для определения времени операции «один ко всем» использовалась аналогичная система отправки сообщений различного объема к slave с ожиданием получения ответа. В отличие от операции «один к одному» в данном случае интерес представляет непосредственно среднее время обмена, т.к. при малых объемах возвращаемых данных (менее 1кб) время фактически не меняется. Таким образом, полученное значение времени можно использовать для определения сразу суммы времен: времени передачи данных процесса решения и времени возврата данных для оценки лучшего:

$$t_{ДПР+ОИ} = t_{p,1-\infty}(V_{ДПР}) + t_{p,\infty-1}(V_{ОИ}) \qquad (14)$$

Для определения времени копирования 35 экземпляров данных процесса решения использовался запуск 35 потоков на одном узле кластера (табл. 2).

Для 8-ми ядер кластера коммуникационная сложность алгоритма на основании (13) рассчитывается как:

$$T_{p,comm} = L(H+1) \cdot t_{p,k}(V_{ДПР}) \qquad (15)$$

где L – коэффициент, округленный к большему целому отношения p/C – размерности выборки p к числу задействованных ядер C.

Соответственно, время выполнения параллельного метода (1) с учетом выражений (4), (13) и (15) для «Метода 2» примет вид:

$$T_p = T_{p,const} + L\,(HT_{фор} + NT_{ин} + (H+1)t_{p,k}(V_{ДПР})) \qquad (16)$$

где $H=S+R$.

Погрешность определения времени выполнения с использованием соотношений (7), (12) и (14) для систем с распределенной памятью и (16) для систем с общей памятью не превышает 20%, что делает приведенные соотношения применимыми для практической оценки времени выполнения параллельных алгоритмов (табл. 3).

Табл. 2. Время выполнения передачи данных

Время трансляции файла задания, генерации мат. модели и т.д., с	$T_{p,const} = 0.51$
Среднее время передачи данных для оценки лучшего, с	$t_{1-1}(V_{ОЛ})=36.735\cdot10^{-6}$
Среднее время передачи результирующих данных, с	$t_{1-1}(V_{РД})=0.48\cdot10^{-3}$
Среднее время передачи входных данных и возврат данных для оценки лучшего (первый шаг), с	$t_{ДПР+ОИ}(V_{ДПР}+V_{ВД})=19.1\cdot10^{-3}$
Среднее время передачи входных данных и возврат данных для оценки лучшего (кроме первого шага), с	$t_{ДПР+ОИ}(V_{ДПР})=4.029\cdot10^{-3}$
Время копирования данных процесса решения, с	$t_{35,k}(V_{ДПР})=0.47\cdot10^{-3}$

Табл. 3. Ожидаемое и реальное время выполнения (p=35)

Схема	Тип памяти	Распределенная память		Общая память		
	Метод	$t_{ожидаемое}$ C=35, с	$t_{реальное}$ C=35, с	$t_{ожидаемое}$ C=8, с	$t_{ожидаемое}$ C=35, с	$t_{реальное}$ C=8, с
Fadd32	Базовый	35.3	36.35	35.3	35.3	36.35
	Метод 1	70.9	88.6	152.46	36.47	182.37
	Метод 2	87.09	107.34	113.52	26.19	130.96

Полученные оценки времени выполнения параллельных алгоритмов численного интегрирования позволяют выбрать наиболее эффективный алгоритм решения. Базовые параметры решаемой задачи (табл. 2), необходимые для расчета времени выполнения, можно оценить как на основании априорного анализа математической модели задачи, так и в ходе работы алгоритма при видах анализа, использующих многократный расчет, например мультивариантном анализе или оптимизации параметров схемы во временной области.

Список литературы

1. Гергель В.П. Теория и практика параллельных вычислений / Гергель В.П. – М. : Бином. Лаборатория знаний, 2007. – 423 с.
2. Молчанов И.Н. Введение в алгоритмы параллельных вычислений / Молчанов И.Н. – К. Наукова думка, 1990. – 128 с.
3. Фіногенов О.Д Паралельний метод чисельного інтегрування жорстких систем диференційних рівнянь з визначенням кращого порядку на кроці / Фіногенов О.Д. // Комп'ютерні технології друкарства: Зб. наук. праць. – Львів : МОНУ Українська академія друкарства, 2008. – С. 108–113.
4. Petrenko A. ALLTED – a computer-aided engineering system for electronic circuit design / Petrenko A., Ladogubets V., Tchkalov V., Pudlowski Z – Melbourne: UICEE, 1997, 205 p.
5. Центр суперкомпьютерных вычислений НТУУ «КПИ». – Режим доступа: http://hpcc.org.ua/index.php
6. Набор тестовых схем института Беркли. CircuitSim90 Benchmark Information. – Режим доступа: http://www.cbl.ncsu.edu/CBL_Docs/csim90.htm

Поступила в редакцию 11.12.2009

УДК 62-82:658.512.011.56

ПЕТРЕНКО А.И.,
МОЛЯВКО А.С.

ПРОГРАММА МОДЕЛИРОВАНИЯ ПРОЦЕССА ЭКСПОНИРОВАНИЯ РЕЗИСТА В ЭЛЕКТРОННО-ЛУЧЕВОЙ ЛИТОГРАФИИ

Описана программа, позволяющая моделировать процесс экспонирования резиста в электронно-лучевой литографии и использующая в качестве входных данных геометрическое описание маски и параметры электронно-лучевой системы. Приведен пример результатов моделирования и описаны возможности программы.

A program used to simulate resist exposure process in electron-beam lithography is described. As input data the program uses layout geometry and parameters of electron-beam system. An example of simulation results is given, and capabilities and limitations of simulation program are described.

Вступление

Непрерывное уменьшение геометрических размеров элементов интегральных схем делает все более сложным формирование этих элементов путем экспонирования резистов электромагнитным излучением (ультрафиолетовым светом с различными длинами волн). Хотя возникающие при этом сложности могут быть преодолены с помощью специальных методов экспонирования (иммерсионная литография, двойное экспонирование со сдвигом и другие), ведутся интенсивные исследования альтернативных технологий литографии, более подходящих для экспонирования структур нанометровых размеров – с использованием рентгеновского излучения (рентгеновская литография), электронных потоков (электронно-лучевая литография) [1] или пучков ионов (ионно-лучевая литография). Эти технологии к данному моменту носят экспериментальный характер и используются только в исследовательских целях или для изготовления уникальных объектов. Процессы экспонирования резиста в этих технологиях литографии относительно слабо изучены.

Постановка задачи

Электронно-лучевая литография в качестве заменителя традиционной оптической имеет как ряд преимуществ (отсутствие масок, высокая разрешающая способность), так и недостатки («последовательный» характер экспонирования резиста и вызванная этим малая производительность, а также специфические эффекты, обусловленные рассеиванием электронов в резисте). Поскольку оборудование для электронно-лучевой литографии дорогостоящее и малодоступное, возникла необходимость в программе, которая позволила бы моделировать процесс экспонирования резиста с учетом основных эффектов, оказывающих влияние на получаемое «изображение».

Описание программы

Для исследования этих эффектов и их влияния на процесс экспонирования была создана программа, позволяющая проводить моделирование для элементов произвольной геометрической формы при различных параметрах электронно-лучевых систем. Входными данными для программы служат параметры электронно-лучевой системы (диаметр электронного луча, его интенсивность, шаг линий растра и параметры распределения интенсивности электронного потока в луче) и тестовая геометрия, экспонирование которой моделируется. При этом подразумевается, что все элементы геометрии, которые необходимо экспонировать, должны получать одинаковые удельные дозы экспозиции в пересчете на единицу их площади. На данный момент программа поддерживает только моделирование экспозиции в режиме с последовательным движением луча по параллельным линиям растра с постоянным шагом между линиями, хотя для реализации поддержки движения луча по произвольной траектории достаточно ввести возможность описания этой траектории в файлы с заданиями на моделирование. Для вычисления уровня экспонирования используется модификация модели, предложенной в 1975 году Чангом, представляющей рассеивание электронов в резисте с помощью свертки распре-

деления доз с функцией близости. Функция близости описывает распределение полученной дозы на заданной глубине под поверхностью резиста для точечного электронного луча. В работе Чанга [2] функция близости является суммой двух гауссовых функций:

$$f_r(r) = \frac{1}{\pi(1+\eta)}\left(\frac{1}{\alpha^2}e^{-\frac{r^2}{\alpha^2}} + \frac{\eta}{\beta^2}e^{-\frac{r^2}{\beta^2}}\right) \quad (1)$$

В формуле (1) α отвечает за прямое рассеивание электронов луча, а коэффициент β – за обратное рассеивание при взаимодействии луча с подложкой. Коэффициент η определяет соотношение α и β.

В работе [6] описаны результаты моделирования движения електронов в резисте по методу Монте-Карло. У этой роботе показано, что значение параметра α из формулы (1), отвечающего за прямое рассеивание электронов, возрастает с увеличением толщины слоя резиста и падает с увеличением ускоряющего напряжения в электронно-лучевой системе. Значение параметра β, отвечающего за обратное рассеивание, увеличивается з увеличением ускоряющего напряжения и мало зависит от толщины слоя резиста. На величину параметра η в основном влияет толщина подложки.

Формула (1) дает приемлемые с точки зрения точности результаты вплоть до размеров элементов в единицы микрон. Однако, как показано в работе Pay, МакКлеллана и Драбика [3], для размеров элементов порядка сотен и десятков нанометров такое описание распределения електронов уже становится недостаточно точным. Авторы [3] предлагают ввести в формулу (1) дополнительные нормальные распределения, чтобы учесть дополнительные эффекты, влияющие на распределение электронов – при этом формула приобретет такой вид:

$$F(x,y) = \frac{1}{\pi\sum_{i=1}^{n}\eta_i}\left(\sum_{i=1}^{n}\frac{\eta_i}{\sigma^2}e^{-\frac{x^2+y^2}{\sigma_i^2}}\right) \quad (2)$$

При этом общая сумма распределений

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}F(x,y)dxdy = 1 \quad (3)$$

Рассматриваемая в данной статье программа моделирования использует при моделировании распределения электронов в луче вариант формулы (2), в котором максимальное значение n равно 10.

Поверхность резиста в программе считается плоской и моделируется в виде прямоугольной сетки с постоянным шагом по двум координатам. Для узлов этой сетки и рассчитываются полученные дозы экспозиции.

Программа не учитывает ряд сложных для моделирования эффектов, например, возможных неровностей на поверхности резиста, падения луча на эту поверхность под непрямым углом, а также возможности накопления в резисте остаточного заряда, который сделает уровень экспонирования нелинейно зависящим от полученного заряда.

Моделирование процесса экспонирования проводится в несколько этапов. На первом этапе соответственно параметрам электронно-лучевой системы рассчитывается траектория, по которой будет двигаться луч по поверхности резиста. На втором этапе по рассчитанной траектории выбирается последовательность точек, которые луч будет проходить через равные промежутки времени. На третьем этапе вычисляются уровни экспонирования точек на поверхности резиста. Эти уровни равны сумме вкладов в экспозицию от каждой из точек, выбранных на втором этапе. Программа позволяет сохранять полученные результаты моделирования в файлах, а также визуализировать их в виде плоских изображений, в которых относительные уровни экспонирования изображаются в виде цветовых градиентов.

Пример результатов моделирования

На рис. 1 показан пример исходной геометрии, экспонирование которой моделируется в программе.

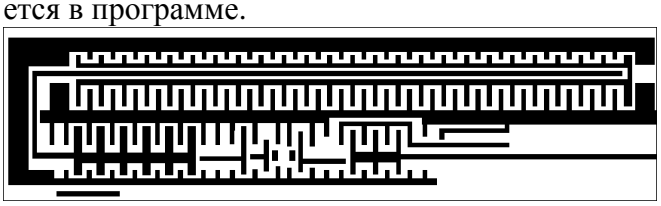


Рис. 1. Пример исходной геометрии

На рис. 2 ниже показан пример результатов моделирования. Заметно уменьшение ширины тонких линий и элементов малых размеров, далеко отстоящих от других элементов.

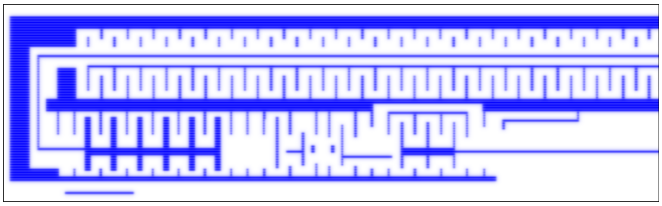


Рис. 2. Экспонирование без коррекции

Для случаев, когда заданные параметры электронно-лучевой системы не позволяют провести экспонирование заданной геометрии с приемлемым качеством (обычно вследствие недостаточной разрешающей способности), на результатах моделирования становятся видны дефекты (разрывы тонких линий, исчезновение или слияние мелких элементов геометрии).

Программа позволяет моделировать коррекцию уровней экспонирования путем управления интенсивностью луча в равномерно размещенных на его траектории точках. Целью коррекции является как можно более точное воспроизведение заданной геометрии и устранение возможных дефектов (невоспроизведение мелких элементов или слияние близко расположенных элементов).

На рис. 3 показан результат работы алгоритма коррекции процесса экспонирования за счет управления интенсивностью электронного луча – тонкие линии менее размыты, геометрические размеры мелких элементов соблюдены точнее, однако узкие промежутки между элементами становятся размытыми, что чревато возникновением перемычек между близко расположенными элементами геометрии.

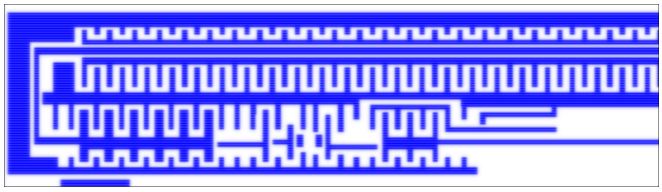


Рис. 3. Результат работы алгоритма коррекции

При коррекции путем управления интенсивностью луча эта интенсивность уменьшается при экспонировании больших отдельно отстоящих элементов, а при экспонировании мелких или близко расположенных элементов – уменьшается. Поскольку достичь идеально точного воспроизведения геометрии элементов невозможно, перед выполнением коррекции задаются критерии, по которым эта коррекция выполняется. Программа позволяет выполнять коррекцию по следующим критериям:

- максимальный допустимый уровень экспонирования фона (результирующий уровень экспонирования точек за пределами подлежащих экспонированию элементов геометрии не должен превышать заданного);
- минимальный допустимый уровень экспонирования элементов (уровень экспонирования точек в пределах элементов геометрии не должен быть меньше заданного).

Коррекция выполняется итеративно, и процесс коррекции останавливается, когда заданный критерий выполняется, или в случае, если две последовательные итерации не дают изменения результата, большего, чем заданная погрешность.

Альтернативой методу коррекции с помощью управления интенсивностью луча может выступать метод, основанный на модификации исходной геометрии, но его применение связано с рядом сложностей в случае, когда размеры элементов геометрии приближаются к пределу разрешающей способности электронно-лучевой системы, используемой для экспонирования этой геометрии. Поэтому в описываемой программе методы коррекции путем изменения геометрии на данный момент не реализованы.

Выводы

Созданная программа может с определенными ограничениями использоваться для моделирования процесса экспонирования резиста в электронно-лучевой литографии, а также для проверки работы методов коррекции, используемых для улучшения качества изображения элементов топологии на поверхности резиста.

В перспективе в программу будут добавлены поддержка методов коррекции с изменением геометрии элементов топологии, а также поддержка произвольных траекторий луча при экспонировании.

Список литературы

1. Tseng, A.A. Electron beam lithography in nanoscale fabrication: recent development / Tseng A.A., Kuan Chen, Chen C.D., Ma K.J. // IEEE Transactions on Electronics Packaging Manufacturing. - 2003, April. - Vol. 26, Issue 2. – P. 791-799.
2. Chang T. H. P. Proximity effect in electron-beam lithography / Chang T. H. P. // J. Vac. Sci. Technol. – 1975. – Issue 1271. – P. 208-219.

3. Rau Richard. Proximity effect correction for nanolithography / Rau Richard, James H. McClellan, Timothy J. Drabik. – 1996. – 71 p.
4. P. Rai-Chodhury. / Handbook of Microlithography, Micromachining and Microfabrication / P. Rai-Chodhury. – 2006. – 802 p.
5. J.A.Liddle. Resist requirements and Limitations for Nanoscale Electron-Beam Patterning / J.A.Liddle et al. // Mat. Res. Soc. Symp. Proceedings. – № 739 (19). – P. 19-30.
6. Parikh M. Calculation of changes in pattern dimensions to compensate for proximity effects in electron lithography / Parikh M. // J. Appl. Phys. – 1980. – № 51. – P. 705.

Поступила в редакцию 11.12.2009

УДК 004.02

ЖДАНОВА О.Г.
ПАДАЛКО О.Б.

ПРО ПРОБЛЕМУ ГЕНЕРАЦІЇ ПЛАНІВ ТЕСТУВАННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ВЕРИФІКАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

Робота присвячена задачі автоматизації тестування – побудові планів тестування програмних продуктів за їх діаграмами станів. Обговорюються проблеми, що виникають при вирішенні задачі, та шляхи їх подолання. Дається теоретичне обґрунтування розроблених алгоритмів. Подається опис можливостей програмного забезпечення, створеного згідно матеріалу роботи.

Работа посвящена задаче автоматизации тестирования – созданию планов тестирования программных продуктов по их диаграммам состояний. Обсуждаются проблемы, которые возникают при решении задачи, и пути их преодоления. Дается теоретическое обоснование разработанных алгоритмов. Описываются возможности программного обеспечения, созданного в соответствии с материалами работы.

Research is dedicated to the software testing automation task – application testing plan generation based on their statechart diagram. Theoretical background is included. Problems that appear while solving the task are discussed along with the way to overcome them. It includes feature list of created application software which was developed based on the research material.

Вступ

Створення програмної системи – складний та багатогранний процес. Для підвищення якості та надійності програмних продуктів потрібно значну увагу приділяти тестуванню. Тестування програмного продукту являє собою перевірку адекватності результатів його роботи початковим вимогам, тобто відповідності створеної функціональності вимогам технічного завдання. Для великих програмних продуктів практично не можливо виконати повне тестування без застосування засобів автоматизації цього процесу. Отже, постає проблема розробки плану тестування з метою покращення якості тестування.

Переважна більшість сучасних розробок орієнтована на перевірку правильності програмного коду [1, 2]. В даній роботі розглядається підхід до тестування програмної системи за відповідною їй діаграмою станів, яка створюється на етапі проектування. Така діаграма має один вхід (початкова точка програми) та один або декілька виходів, при чому можлива наявність циклів та петель. Такий підхід до тестування є зручним та надійним, оскільки на діаграмі відображені усі можливі переходи між станами системи, що може бути враховано при побудові плану. Далі діаграму станів будемо представляти у вигляді орієнтованого графу (рис. 1).

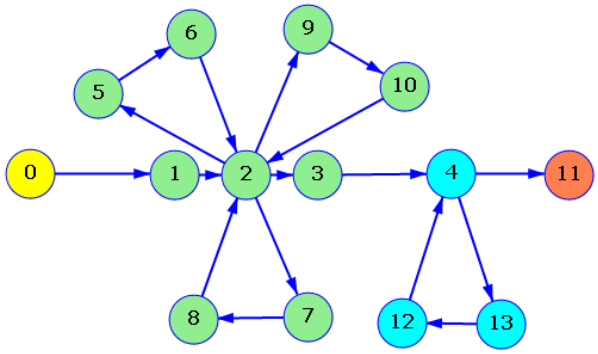


Рис. 1

Кожному тесту ставиться у відповідність шлях, який починається в початковій вершині і завершується в одній з кінцевих. Під планом тестування розуміється сукупність тестів, яка задовольняє певним умовам. В залежності від цих умов можливі декілька підходів до побудови плану тестування по діаграмі (графу) станів системи. Поширеними є покриття гілками (Branch Coverage або Edge Coverage), покриття шляхами - Path Coverage (яке має багато різновидностей, тому надалі під Path Coverage будемо мати на увазі Edge-Pair Coverage [2]). З перерахованих план тестування за покриттям типу Path Coverage дає найбільш високу якість.

При побудові планів тестування може бути застосовано декілька стратегій. З точки зору тестувальника (реалізації процесу тестування) виділяються такі стратегії: „довга”, „коротка” та стратегія, що враховує вагу дуг.

Під „довгою” стратегією розуміється знаходження плану тестування, який включає в себе найбільш довгі тести. „Коротка” стратегія вимагає знаходження плану тестування, який складався б з більш коротких тестів.

1 Постановка задачі

Дано орієнтований граф з однією початковою і декількаами кінцевими вершинами. Побудувати покриття типу Path Coverage, тобто знайти таку (мінімальну за потужністю) множину шляхів, яка покриває усі комбінації пар дуг вхід–вихід для кожної вершини графу (далі цю умову будемо позначати як умову УКПДВВ) і відповідає заданій стратегії.

2 Загальна схема розв’язання

Якщо граф системи не містить циклів, то шуканим планом тестування буде сукупність усіх шляхів з початкової точки в кінцеві. Цей окремий випадок є тривіальним і не є для нас цікавим. Наявність циклів різко збільшує кількість можливих планів тестування.

Для розв’язання поставленої задачі пропонується така схема:

Етап 1. Формування множини шляхів з початкової точки в кінцеві.

Етап 2. Формування множини тріад.

Етап 3.Розв’язання задачі.

3 Формування множини шляхів з початкової точки в кінцеві

Цей етап складається з трьох кроків:

Крок 1. Знаходження множини простих шляхів та множини циклів.

Крок 2. Формування послідовностей циклів.

Крок 3. Комбінування множин шляхів і послідовностей циклів.

3.1 Знаходження множини простих шляхів та множини циклів

На цьому кроці алгоритмом обходу графу в глибину знаходиться множина W_0 простих шляхів (шляхів, що не містять циклів), множина C циклів та точки входу в цикли.

Точка входу в цикл – вершина шляху, що є спільною для цього шляху і деякого циклу. Результатом роботи цього кроку є дерево простих шляхів та циклів (рисунок 2). (Зауважимо, що цей крок є спільним для побудови усіх видів покриттів).

3.2 Формування послідовностей циклів та множині розширених шляхів

Після визначення множин простих шляхів та циклів необхідно виконати їх комбінування (вбудування циклів в шляхи). У випадку, коли мають місце цикли, задача забезпечення основної вимоги Path Coverage має ряд нетривіальних особливостей. Розглянемо ці особливості на прикладі графу, що наведений на рисунку 1 (відповідне дерево шляхів та циклів показано на рис. 2).

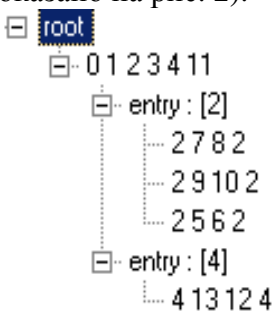


Рис. 2

Вершини 2 та 4 мають по декілька вхідних та вихідних дуг. Для забезпечення УКПДВВ у результуючий множині шляхів:

– через вершину 2 (вершину, яка має чотири входи та чотири виходи) необхідно забезпечити проходження шістнадцятьма підшляхами 1-2-5, 1-2-9, 1-2-3, 1-2-7, 6-2-5, 6-2-9, 6-2-3, 6-2-7, 10-2-5, 10-2-9, 10-2-3, 10-2-7, 8-2-5, 8-2-9, 8-2-3, 8-2-7;

– через вершину 4 (у неї два входи та два виходи) необхідно забезпечити проходження чотирма підшляхами 3-4-11, 3-4-13, 12-4-11, 12-4-13.

Далі такі підшляхи будемо називати ***тріадами*** (Edge-Pair [2])

3.2.1 Послідовності циклів

Якщо на деяку вершину i «навішано» один або декілька циклів, то для виконання умови УКПДВВ необхідно сформувати ***послідовності циклів*** (ПЦ), які б забезпечили покриття усіх тріад, що містять у собі цю вершину i. Проілюструємо сказане на прикладі графу з рисунку 1. Цей граф має один простий шлях 0–1–2-3-11 та чотири незалежних цикли (цикли називаються ***незалежними***, якщо вони не мають спільних дуг):

цикл1: 2-7-8-2;

цикл2: 2-9-10-2;

цикл3: 2-5-6-2;

цикл4: 4-13-12-4.

По-перше: немає необхідності формувати шляхи, що включають в себе тільки один

прохід по циклу. Це означає, що немає сенсу формувати шлях, що проходить по циклу 1 тільки один раз:

0–1–цикл1-3-11 (0-1-2-7-8-2-3-11)

оскільки при цьому не буде покрита тріада 8-2-7 (кінцева дуга циклу 1 + початкова дуга циклу 1). Отже, для покриття такого типу тріад необхідно сформувати такі послідовності циклів (ПЦ), що є дублями циклів:

по точці входу 2:

цикл1–цикл1, цикл2–цикл2, цикл3–цикл3;

по точці входу 4:

цикл4–цикл4.

По-друге, якщо на деяку вершину i «навішано» декілька циклів (в нашому випадку це вершина 2), то для неї необхідно сформувати змішані ПЦ, які б забезпечили покриття усіх тріад, що є перехідними з одного циклу в другий. Тобто, необхідно, щоб деякий шлях (деякі шляхи) включали в себе такі ПЦ:

цикл1–цикл2, цикл1–цикл3,

цикл2–цикл1, цикл2–цикл3,

цикл3–цикл1, цикл3–цикл2.

Слід зауважити, що для забезпечення умови УКПВВ можна сформувати ПЦ, що включає у себе усі можливі комбінації циклів:

цикл1–цикл2–цикл1–цикл3–цикл2–цикл3–цикл1.

Отже, якщо цикли, навішані та точку входу, є незалежними, то процедура повинна передбачати комбінацію усіх можливих пар циклів.

Алгоритм генерації послідовності циклів

Нехай на деяку вершину i «навішано» декілька циклів Послідовність циклів повинна покривати усі можливі переходи між ними. Алгоритм, що досягає цього є простим. Наведемо результат його роботи для кількості незалежних циклів $k = 2...8$ (таблиця 1). При цьому, при формуванні ПЦ для кількості циклів k $(k > 2)$ в якості початкової використовується ПЦ, що отримана для $k-1$.

Табл. 1

k	Послідовності пар циклів	Результуюча послідовність
2	21 12	212
3	…23 31 13 32	…3132
4	…24 41 14 43 34 42	…414342
5	…25 51 15 53 35 54 45 52	…51535452
6	…26 61 16 63 36 64 46 65 56 62	…6163646562
7	…	…717374757672
8	…	…81838485868782

Для більшого контролю над результуючою множиною шляхів можна впливати на довжину отримуваної послідовності – довгі ПЦ розбивати на декілька підпослідовностей. В результаті варіюється максимальна та середня довжини шляхів у результуючий множині, що є додатковим інструментом збільшення відповідності шуканих покриттів обраній стратегії.

3.2.2 Наявність дочірніх циклів

Розглянемо випадок, в якому один з циклів має так званий дочірній цикл (цикл, який не має спільних вершин з будь-яким простим шляхом). Так у графі, що наведений на рисунку 3, дочірнім є цикл c_3.

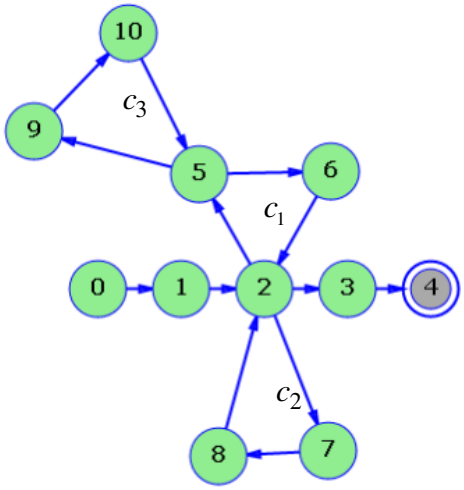


Рис. 3

Необхідно побудувати ПЦ, яка б забезпечила виконання УКПВВ для точки входу у цей цикл. Для цього батьківський цикл розглядається як простий шлях і для нього будується ПЦ за правилами, описаними в п. 3.2.1.

Для того, щоб уникнути дублювання точок входу (і необґрунтованого збільшення довжин шляхів), у ПЦ батьківський цикл включається один раз.

Так для графу з рисунку 3 маємо:

– множина простих шляхів: W_0={0-1-2-3-4};

– множина циклів: $C = \{ c_1 = $ 2-5-6-2, $c_2 =$ 2-7-8-2, $c_3 =$ 5-9-10-5$\}$;

– множина послідовностей циклів для простого шляху: $\{ c_1 - c_1,\ c_2 - c_2,\ c_1 - c_2 - c_1 \}$;

– послідовність, у якій в батьківський цикл c_1 двічі вбудовано дочірній цикл c_3: 2-c_3-c_3-6-2.

3.2.3 Формування послідовностей циклів, серед яких є залежні

Залежними будемо називати цикли, які мають спільні дуги.

При аналізі результатів формування ПЦ для реальних задач ми зіткнулися з наступною проблемою. Якщо деяка вершина є точкою входу у декілька циклів (позначимо цю кількість через k), то ПЦ, у якій передбачені усі можливі комбінації циклів, містить у собі (k^2-k+1) циклів. З урахуванням того, що кожен з циклів включає в себе не менш ніж дві дуги, то кількість вершин отриманої ПЦ може бути дуже великою.

Якщо цикли, навішані на точку входу, є залежними, то після застосування вищевказаної процедури (яка передбачає комбінацію усіх можливих сполучень пар циклів) отримана ПЦ буде мати підпослідовності переходів, що багатократно повторюються. Це призводить до значного (і необгрунтованого) збільшення кількості вершин сформованої ПЦ.

Отже постала задача знаходження шляхів подолання цієї проблеми.

Розглянемо частину дерева шляхів та циклів (рисунок 4) деякого графу.

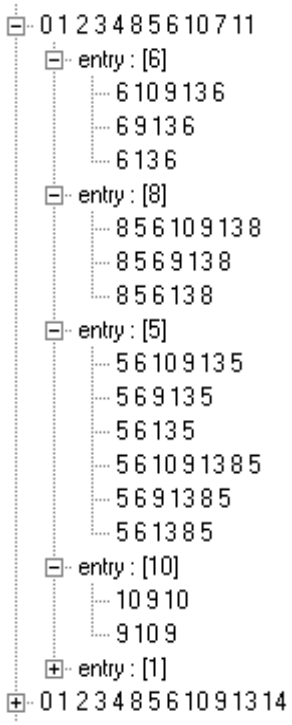


Рис. 4

Проаналізуємо особливості груп циклів, навішаних на вершини 6, 8, 5 та 10.

Особливість 1. В групі циклів 8 початкові та кінцеві дуги усіх циклів однакові. Відповідна тріада (13-8-5) покривається будь-яким продубльованим циклом. Крім того, достатньо продублювати тільки один з циклів, а з кожного іншого створити по послідовності, що включає їх по одному разу.

Особливість 2. Розглянемо групу циклів 5. Ця група розбивається на дві підгрупи і кожна з них володіє особливістю 1. Тому достатньо продублювати по одному циклу з кожної підгрупи.

Особливість 3. В групі циклів 6 усі (три) цикли мають різні початкові дуги. Отже, при побудові підпослідовності циклів будуть використані усі ці цикли.

Для визначення мінімальної множини циклів (яку достатньо буде продублювати) сформулюємо допоміжну оптимізаційну задача. Суть цієї задачі полягає в наступному. Для аналізованої групи циклів вершини відомі (визначаються) множина початкових дуг і множина кінцевих дуг. Необхідно визначити мінімальний набір циклів, які в б сукупності покривали (включали) всі задані початкові і кінцеві дуги. Тим самим, згенерована підпослідовність, безнадлишково покриє всі можливі тріади.

Математична постановка цієї задачі така. Нехай C_v – множина циклів, вершини v, $n=|C_v|$; B_v – множина початкових дуг циклів з C_v; $m=|B_v|$; F_v – множина кінцевих дуг циклів з C_v; $k=|F_v|$;

$$b_{ij}=\begin{cases}1, \text{ якщо початк. дуга } i \text{ входить до циклу } j\\ 0, \text{ якщо початк. дуга } i \text{ не входить до циклу } j\end{cases},\ i=\overline{1,m},\ j=\overline{1,n};$$

$$f_{ij}=\begin{cases}1, \text{ якщо кінц. дуга } i \text{ входить до циклу } j\\ 0, \text{ якщо кінц. дуга } i \text{ не входить до циклу } j\end{cases},\ i=\overline{1,k},\ j=\overline{1,n}.$$

Змінні моделі:

$$x_j=\begin{cases}1, \text{ якщо обираємо цикл } j,\\ 0, \text{ якщо не обираємо цикл } j,\end{cases}\quad j=\overline{1,n}$$

Цільова функція (кількість обраних циклів):

$$\sum_{j=1}^{n} x_j \to \min.$$

Обмеження:

початкова дуга i повинна входити хоча б в один обраний цикл:

$$\sum_{j=1}^{n} b_{ij} x_j \geq 1; \quad i = \overline{1,m};$$

кінцева дуга i повинна входити хоча б в один обраний цикл:

$$\sum_{j=1}^{n} f_{ij} x_j \geq 1, \quad i = \overline{1,k}.$$

3.2.4 Схема формування множини послідовностей циклів для заданої вершини v

Нехай S_v – результуюча множина послідовностей для вершини v, що є точкою входу в цикли (початково $S_v = \varnothing$).

1 Визначити множину B_v початкових и множину F_v кінцевих дуг множини C_v циклів.

2 ВИРІШИТИ допоміжну оптимізаційну задачу (на її вхід подаються C_v, B_v, F_v), отримати множину C_{gen} циклів, що приймають участь у генерації.

3 ЗГЕНЕРУВАТИ послідовність циклів $s = gen(C_{gen})$; Додати її до множини S_v

4 ЦИКЛ по циклах c множини C_{gen}

СФОРМУВАТИ підпослідовність – дубль $s = c\text{-}c$, ДОДАТИ її до множини S_v

КІНЕЦЬ ЦИКЛУ 4

5 ЦИКЛ по циклам множини $C_v \setminus C_{gen}$

СФОРМУВАТИ підпослідовність $s = c$, ДОДАТИ її до множини S_v

КІНЕЦЬ ЦИКЛУ 5

Таким чином, множина S_v включає в себе три типи послідовностей:

- послідовність $gen(C_{gen})$;
- послідовності – дублі (їх в загальному випадку декілька);
- послідовності, що включають у себе одинарні цикли, які не приймали участь в генерації послідовності $gen(C_{gen})$ і не є батьківськими (таких послідовностей в загальному випадку також може бути декілька):

У наведеному вигляді цей алгоритм застосовується до вершин (точок входу), що входять до простих шляхів. У тому разі, коли вершина v належить одному з циклів, то знайдені для неї послідовності з дочірніх циклів (що будуються за цим же алгоритмом) вбудовуються в цей цикл і додаються до результуючої множини ПЦ.

3.2.5 Комбінування множин шляхів і послідовностей циклів

На цьому кроці виконується вбудовування отриманих на кроці 2 послідовностей циклів в шляхи.

Для кожної точки входу простого шляху відома множина послідовностей циклів (див. п. 3.2.4). Можливі декілька варіантів комбінування послідовностей зі шляхом.

Варіант 1. В шлях додається тільки одна послідовність з множини S_v і отриманий шлях включається у результуючу множину шляхів. Ця процедура повторюється для усіх послідовностей, що мають відношення до цього шляху. В результаті отримуємо $\sum_{v \in V} |S_v|$ нових шляхів, де V – множина точок входу шляху, $|S_v|$ – кількість послідовностей для точки входу v.

Варіант 2. У шлях вбудовуються усі можливі для нього послідовності і після цього отриманий шлях включається до результуючої множини. В результаті отримуємо один шлях з усіма можливими послідовностями.

Варіант 3. Є проміжним між першими двома варіантами – ПЦ вбудовуються в деякі точки входу шляху.

Вибір варіанту залежить від виду прийнятої стратегії побудови планів тестів.

4 Формування множини тріад. Формулювання задачі покриття множини як задачі булевого програмування

За заданим графом формується множина тріад. Після цього кожній тріаді ставиться у відповідність підмножина шляхів, що її покривають. Далі формулюється задача знаходження покриття.

4.1 Математична постановка задачі

Задача знаходження покрить зводиться до класичної задачі мінімального покриття множини [3]. Початковими даними цієї задами є скінчена множина T тріад і множина W шляхів (кожен з яких визначається множиною покритих тріад, тобто шлях можна розглядати як підмножину T). Покриттям $\overline{W} \subseteq W$ називають множину найменшої потужності (найменшої вартості) підмножин, об'єднанням яких є T.

Ця задача відноситься до класу NP-складних і може бути сформульована як за-

дача булевого програмування. Нехай граф має m тріад (множина T) і в ньому можливі n шляхів від початкової до термінальних вершин (множина W). Для кожного шляху відомі тріади, що входять до нього, тобто для $i=\overline{1,m}$ та $j=\overline{1,n}$ задані

$$a_{ij}=\begin{cases}1, \text{ якщо тріада } i \text{ входить у шлях } j,\\ 0, \text{ якщо тріада } i \text{ не входить у шлях } j.\end{cases}$$

Вимагається знайти таку підмножину шляхів $\overline{W}\subseteq W$, щоб кожна тріада входила хоча б у один шлях і ця множина була б найкращою згідно обраної стратегії. За умови, що

$$x_j=\begin{cases}1, \text{ якщо } j-й \text{ шлях входить до множ. } \overline{W},\\ 0, \text{ якщо } j-й \text{ шлях не входить до множ. } \overline{W},\end{cases}$$

вказане обмеження формулюється наступним чином:

$$\sum_j a_{ij}x_j \ge 1,\ i=\overline{1,m}\ .$$

Щодо цільової функції. Оскільки "довга" стратегія передбачає знаходження множини планів тестування, які включають в себе найбільш довгі тести, то з цього виходить, що такий план буде включати в себе найменш можливу кількість шляхів, тобто $\sum_{j=1}^{n}x_j\to\min$. При цьому автоматично буде гарантуватися ненадлишковість знайденого покриття. (Покриття є ***ненадлишковим***, коли виключення з нього хоча б одного шляху приведе до недопустимості розв'язку). „Коротка" стратегія вимагає знаходження множини планів тестування, які включають в себе найбільш короткі тести. Тобто такий план буде включати в себе найбільш можливу кількість коротких шляхів (за відсутності надлишковості розв'язку). Для цього випадку цільова функція така: $\sum_{j=1}^{n}x_j\to\max$ (але у цьому випадку наведені обмеження не гарантують ненадлишковості розв'язку, для цього необхідно формулювати додаткові обмеження). У разі врахування вартості шляхів (c_j, $j=\overline{1,n}$) критерій ефективності можна записати так: $\sum_{j=1}^{n}c_jx_j\to\min$.

5 Знаходження розв'язку

Цей етап складається з двох кроків:

Крок 1. Оптимізація системи рівнянь (опціонально).

Крок 2. Розв'язання задачі.

В залежності від розмірності задачі та наявних обчислювальних ресурсів пропонується один з наступних методів:

– отримання повної множини покрить методом бектрекінгу;

– розв'язання задачі із застосуванням жадібного алгоритму;

– розв'язання задачі із застосуванням генетичного алгоритму.

5.1 Отримання повної множини покрить

Алгоритмом, в основу якого покладений метод бектрекінгу [4], знаходиться повна множина покриттів. Алгоритм полягає у послідовному виклику для кожного (починаючи з першого) елементу рівнянь рекурсивної процедури, яка додає до цього елементу елемент з наступного рівняння і так далі, поки не буде досягнуто останнє рівняння, що свідчитиме про отримання деякого допустимого розв'язку. При цьому забезпечується ненадлишковість розв'язків, що отримуються.

Знаходження повної множини покрить дозволяє в подальшому при виборі найкращого покриття застосовувати декілька критеріїв.

Для задачі побудови множини покрить типу Path Coverage час розв'язання зростає експоненційно з ростом кількості шляхів.

5.2 Шляхи прискорення обчислень

Природа NP-складних задач вимагає приділення особливої уваги проблемі зростання часу пошуку розв'язку у випадку росту розмірності задачі. Прискорення процесу обчислень було досягнуто такими шляхами:

– застосуванням критеріїв відсікання "безперспективних" підмножин, що в результаті дає деяку групу покриттів, що задовольняють певному критерію (згідно обраної стратегії);

– зменшенням вхідної множини шляхів: якщо, є два шляхи, таких, що $T_1\subseteq T_2$, де T_1, T_2 –множини тріад, які покриваються цими шляхами, то виключається шлях, якому відповідає множина T_1;

– оптимізацією системи рівнянь: на цьому етапі для збільшення швидкості подальших обчислень виконується оптимізація системи рівнянь (видалення рівнянь, які містять усі шляхи; видалення рівнянь, які містять тільки один шлях з подальшим включенням цього шляху до кожного знайденого

розв'язку; видалення рівнянь у випадку, якщо воно повністю включає в себе інше рівняння). Цей процес дозволяє значно зменшити розмірність задачі (у деяких випадках виграш становив десятки разів).

Поряд з алгоритмічною оптимізацією суттєвого збільшення швидкості обчислень можна досягти використанням технології відомої як GPGPU (general-purpose GPU) – перенесенням обчислень з центрального процесору на графічний чип. Найпоширенішими реалізаціями технології GPGPU є NVidia CUDA та AMD FireStream. Графічний процесор має більші обчислювальні потужності завдяки великій кількості процесорів (512 і більше) та швидкої пам'яті. Після перенесення обчислень на графічну карту вдалося збільшити швидкість роботи алгоритму в декілька разів. Недоліком цього підходу є висока залежність програмного забезпечення від апаратних засобів.

5.3 Розв'язання задачі із застосуванням жадібного алгоритму

Досить часто для реальних практичних NP-складних задач знайти точний розв'язок за прийнятний час не можливо. В такому випадку можна взяти ліпший розв'язок, який вдалося знайти за певний встановлений ліміт часу роботи точного алгоритму, або використовувати наближені або евристичні алгоритми. Жадібний алгоритм є одним з найпоширеніших зразків евристичних алгоритмів. Перевага цього алгоритму полягає в тому, що час знаходження рішення поліноміально зростає з ростом розмірності. Для задачі знаходження покриття графу шляхами розроблено та реалізовано наступний жадібний алгоритм. Спочатку маємо початкове покриття, яке не включає в себе жодного шляху. Доки це покриття не задовольняє умові УКПВВ додаємо до нього новий шлях, який покриває найбільш можливу кількість ще не покритих тріад. Цілком можливо, що отриманий розв'язок буде надлишковим. Надлишковості можна позбутися використовуючи алгоритм оптимізації, описаний у п. 5.1. Вихідна система рівнянь будується на основі шляхів, що увійшли у знайдений розв'язок. Після оптимізації можливе отримання декількох розв'язків, що дуже схожі між собою.

З урахуванням сутності дій алгоритму побудовані розв'язки будуть в більшості випадків задовольняти вимогам довгої стратегії.

5.4 Розв'язання задачі із застосуванням генетичного алгоритму

Структура множини допустимих розв'язків задачі знаходження покриття графу шляхами (кожний розв'язок є булевим вектором розмірності n) дозволяє розробити для цієї задачі генетичний алгоритм (ГА), що відноситься до так званих метаевристичних алгоритмів [5].

ГА включає в себе наступні основні оператори:

- генерування початкової популяції;
- схрещування обраних за певним критерієм особин;
- мутація окремих особин;
- локальне покращення особини.

Те, яким чином реалізовані ці оператори, впливає на якість отриманого результату. Розглянемо деякі особливості розроблених операторів. Оператори генерування та мутації базуються на випадковому додаванні\вилученні елементів з розв'язку із одночасним збереженням допустимості. Схрещування є "змішуванням" батьківських розв'язків (особин) у певних пропорціях – якась частина нової особини унаслідується від "матері", якась - від "батька". В операторі локального покращення виключається "найслабкіша ланка" – елемент розв'язку, що у найбільшій мірі не задовольняє обраному критерію оптимальності. Замість виключеного елементу знаходиться новий (нові), який максимально покращить значення критерію оптимальності для розв'язку не порушуючи допустимості. У всіх операторів є одна спільна риса – після створення нового розв'язку (особини) цілком можливим є те, що він є надлишковим. Щоб гарантувати ненадлишковість застосовується алгоритм, що описаний у п 5.1.

Звичайно, час роботи ГА більший порівняно із жадібним, але результати, отримані за його допомогою, є значно кращими. Експерименти показали, що у більшості випадків для задач, для яких можна за прийнятний час отримати оптимальний розв'язок, цей же розв'язок був отриманий і ГА. Але самою сильною стороною генетичного алгоритму є можливість пошуку рішень за практично довільними критеріями – значно складнішими, за критерії довгої та короткої стратегій. При чому застосування нелінійних та комплексних критеріїв практично не позначається на швидкодії алгоритму.

5.5 Програмний продукт

За матеріалами роботи створений програмний продукт “Тріада”.

На рисунку 5 наведено головне вікно розробленого програмного продукту.

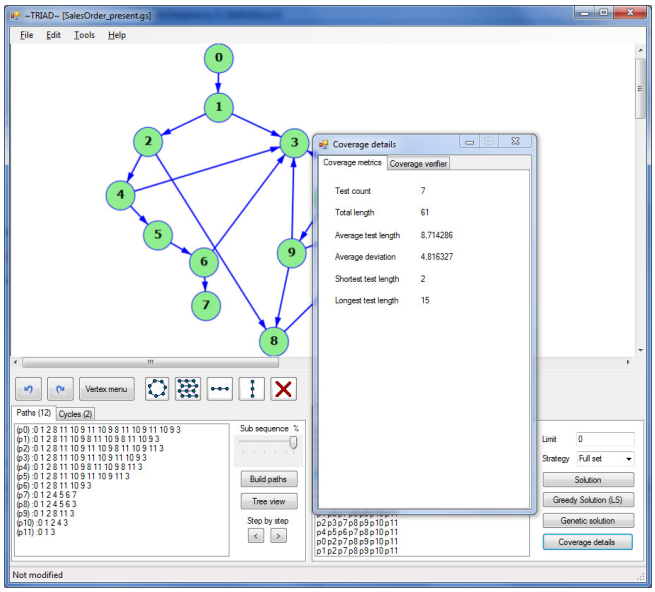


Рис. 5

В ньому реалізовано:

– WYSIWYG- редактор для створення графу з можливістю додавання, видалення та редагування дуг та вершин графу;

пошук усіх шляхів з початкової вершини у термінальні;

– пошук усіх циклів графу:

– побудова множини покрить шляхами з урахуванням обраної стратегії;

– пошук покриття жадібним алгоритмом;

– пошук покриття за допомогою ГА;

– додаткові інструменти: метрики отриманих покриттів, верифікатор покриття, дерево шляхів та циклів та ін.

Використовуючи систему плагінів, можна транслювати отриманий план у іншу предметну область, наприклад створити динамічну бібліотеку, яка буде викликати методи програми у порядку, що вказаний у плані тестування.

Висновки

В роботі досліджені особливості задачі побудови покриття типу Path Coverage. Розроблені точний та евристичні алгоритми розв’язання задачі. Запропоновані шляхи оптимізації процесу та зменшення часу обчислень.

Дослідження проблеми побудови покриття типу Path Coverage було тісно пов’язане з розробкою програмного продукту “Тріада”. Деякі особливості задачі вдалося дослідити саме завдяки інструментарію програмного продукту “Тріада”.

Розроблені та реалізовані такі методи розв’язання задачі, що дозволяють менеджеру (керівнику групи тестувальників) застосовувати їх в реальних умовах з можливістю впливу на параметри результуючого плану тестування в залежності від обраної стратегії.

Перелік посилань

1. Kolawa A., Huizinga D. (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press. 254 p.
2. Ammann P., Offutt J.. Introduction to software testing. Cambridge University press - New York. 2008, – 322 p
3. Кристофидес Н. Теория графов. Алгоритмический подход. М.: Мир, 1978, – 432 с.
4. Кнут Д. Искусство программирования. В 3-х т. 3-е изд. – М.: «Вильямс», 2006. Т. 1– 720 с.
5. Сергиенко И.В., Шило В.П. Задачи дискретной оптимизации: проблемы, методы решения, исследования. – К. Наук. думка, 2003. – 264 с.

Поступила в редакцию 3.12.2009

УДК 683.519

БЕРЕЗОВСКИЙ А.Н.
КОРОЧКИН А.В

МОДЕЛИ ВЫЧИСЛЕНИЙ ДЛЯ МАСШТАБИРУЕМЫХ КЛАСТЕРНЫХ СИСТЕМ

В работе рассматривается подход к разработке программных компонент для масштабируемых кластерных систем, основанный на использовании моделей вычислений. Предложены модели вычислений, основанные на применении теории последовательных взаимодействующих процессов Ч.Хоара. Показана возможность использования моделей для повышения качества программ и сокращения времени их разработки.

In paper the problem of creation of effective program components for scalable cluster systems based on models of computation is considered. Models of computation based on C.Hoare theory of communicating sequential processes are presented. Use models of computation for improving software and reduction design time is showed.

Кластерная архитектура сегодня рассматривается как одна из основных при построении высокопроизводительных компьютерных систем, оптимальных по критерию производительность/стоимость и включает набор вычислительных узлов, соединенных быстродействующей системой связей узлов (ССУ). Узлы кластерной системы (КС) обеспечивают обработку и хранение данных, ССУ обеспечивает быструю передачу больших объемов данных между узлами КС.

Современные КС характеризуются увеличением количества узлов, а также использованием в узлах многоядерных процессоров [1]. Кластерная система с многоядерной архитектурой (КСМА) представляет собой сложную систему, разработка программного обеспечения для которой представляет сложную задачу, связанную с организацией вычислений на двух уровнях: между узлами и внутри узлов. При этом необходимо обеспечить с помощью механизма процессов (потоков) поведение каждого узла, а также обеспечить оптимальное по времени и объемам передаваемых данных взаимодействие процессов.

Современная практика разработки приложений для сложных компьютерных систем основывается на использовании моделей вычислений, которые выступают в качестве интерфейса между архитектурой системы и программой, реализующей выбранную модель параллельного (распределенного) программирования (рис.1). Назначение модели вычислений:

- описать архитектуру КС;
- описать поведение процессов;
- описать взаимодействие процессов.

На основе модели выполнятся анализ корректности поведения процессов и их взаимодействия еще на этапе проектирования программного продукта. При этом математический аппарат модели позволяет выявить возможные причины возникновения тупиковых ситуаций. Кроме того, удачно выбранная модель может упростить процесс дальнейшего создания написания программы.

Рис.1. Модель вычислений

Кластерные системы являются хорошо масштабируемыми системами с точки зрения увеличения производительности. Кластерная архитектура допускает практически неограниченное наращивание числа узлов. Поэтому проблема разработки программного обеспечения для масштабируемых кластерных систем (МКС) является важной задачей и требует разработки методов и средств, позволяющих повысить качество приложений для МКС.

Анализ последних исследований и публикаций

Рассматриваемая тематика широко представлена в публикациях. В основном, они сводятся к выбору и обоснованию подхода к построению модели параллельных вычислений [2, 4, 5, 6, 9].

Модели вычислений могут базироваться на сетях Петри [9], теории конечных автоматов [5], CSP-теории [2, 3, 6].

Большое число разнообразных моделей вычислений не гарантирует их адекватность множеству практических задач, с которыми приходится сталкиваться при разработке распределенных приложений. Причина заключается в противоречивости требований к этим моделям и важности представления тех или иных особенностей функционирования вычислительных систем.

Нерешенные проблемы

В работе [2] предложено использование теории Ч.Хоара для построения моделей вычислений для кластерных систем. Показано, что математический аппарат CSP теории позволяет описать архитектуру кластерной системы, а также поведение и взаимодействие процессов. Однако в работе не рассмотрены вопросы построения, реализации и использования моделей вычислений на основе CSP теории для масштабируемых систем. Учитывая, что масштабируемость является важной составляющей компьютерной системы, решение проблем, связанных с созданием программного обеспечения для МКС, является актуальной задачей.

Постановка задачи

В работе [4] выделены две особенности распределенных вычислений в МКС, которые определяют требования к их разработке. Это *недетерминизм,* означающий произвольные последовательности формирования компонентов структурированных данных и *неоднородность,* которая связана с произвольной структурированностью данных, которыми обмениваются взаимодействующие процессы.

Недетерминизм связан с обеспечением однозначности результатов вычислений, при которой одному входному набору данных соответствует один выходной набор результатов. Неоднородность сообщений связана с тем, что процессы могут принимать (передавать) данные произвольного размера, зависящего от количества узлов в системе. Выбор средств для построения моделей вычислений должен обеспечить решение указанных проблем.

Целью настоящей работы является развитие подхода, предложенного в работе [2], для создания и использования моделей вычислений при разработке программных компонент для масштабируемых кластерных систем с многоядерной архитектурой.

Модели вычислений

При разработке масштабированных распределенных программ необходимо обеспечить возможность статического (динамического) создания (порождения) множества процессов, описание их поведения, запуск на выполнение и организацию взаимодействия.

Существуют различные средства описания процессов в современных языках и библиотеках параллельного программирования [7]. Языки программирования Java, Ада позволяют описать процесс с помощью специальных программных модулей (классов). В языке C# и библиотеке Win32 для этого используются потоковые функции. OpenMP позволяет выделять параллельные участки программы с помощью операторов `parallel/end parallel`. Библиотека MPI создает параллельные процессы как копии основной программы.

Программа для масштабируемой архитектуры должна базироваться на возможности создания (как статически, так и динамически) любого множества процессов. Это возможно только с помощью тиражирования базового процесса, параметризация которого позволит описать все различия между создаваемыми копиями этого процесса. В языках, поддерживающих объектно-ориентированную парадигму, это можно сделать с помощью классов и конструкторов, в языка Ада для этого используется задачный тип с дискриминантом, в MPI – идентификатор задачи, автоматически порождаемый при создании копии процесса.

Организация взаимодействия процессов включает синхронизацию процессов и обмен данными [6]. Модели взаимодействия при этом базируются на общих переменных или на посылке сообщений.

При использовании общих переменных возникают задачи взаимного исключения и

синхронизации, решение которых надо обеспечить для произвольного числа процессов. Для задачи взаимного исключения это не представляет сложности, так как примитивы типа семафоры, мютексы, критические секции, также мониторы обеспечивают контроль общих ресурсов для любого числа процессов, которые их используют. Проблемы могут возникнуть при решении задачи синхронизации, так как здесь имеют место сложные формы синхронизации вида: *процесс→группа процессов, группа процессов→процесс, группа процессов→ группа процессов.* При использовании низкоуровневых примитивов потребуется динамическое создание и использование объектов синхронизации. Эта проблема решается путем использования мониторных объектов синхронизации, которые позволяют эффективно реализовать указанные формы групповой синхронизации.

При использовании механизма посылки сообщений в МКС основная проблема связана с формированием объема передаваемых данных, который может меняться в зависимости от количества динамически создаваемых процессов. Здесь важной является возможность организация обмена данными произвольной размерности. В MPI это обеспечивается функциями `MPI_Gatherv`, `MPI_Scat-terv`, в языке Ада – использованием типов данных с уточняемыми границами, что позволяет описать вход задачи для передачи любого объема данных.

Кроме того, эффективность взаимодействия увеличивается, если язык (библиотека) поддерживают коллективные формы взаимодействия, как при передаче сообщений, так и при синхронизации. В этом плане представляют интерес коллективные операции MPI, которые реализованы в виде операций групповой рассылки (сборки) данных `MPI_Bcast`, `MPI_Gather`, `MPI_Scatter` и операций групповой синхронизации `MPI_Barrier`.

Таким образом, можно сформулировать основные требования к моделям вычислений для масштабируемых систем: поддержка недетерминизма для неограниченного числа динамически порождаемых процессов, взаимодействующих путем посылки сообщений произвольного объема и через групповую синхронизацию.

Рассмотрим возможности теории Ч.Хоара с точки зрения реализации сформулированных требований. Теория Ч.Хоара позволяет описать поведение последовательных недетерминированных процессов с произвольным независимым чередованием процессов. Это позволяет описать особенности работы МСК. Поддерживается динамическое порождение процессов без ограничения их количества. Обеспечивается синхронная однонаправленная без буферизации посылка сообщений. При этом рассматривается одна форма, когда взаимодействую только две задачи, то есть не используются групповые операции обмена. Не рассматривает общие переменные как средства взаимодействия процессов, тем более для групповой синхронизации и взаимного исключения.

Таким образом, теория Ч.Хоара может быть использована для построения моделей вычислений для МКС при ее расширении (доработке) в плане реализации двунаправленных механизмов посылки сообщений произвольной размерности, а также мониторподобных средств для взаимного исключения и синхронизации при описании группового взаимодействия процессов.

Модели вычислений для МКС

Рассмотрим МКС как сложную динамическую систему, структура которой может меняться как путем изменения числа узлов и связей между узлами. В общем случае, любая кластерная система является полносвязной, так как любые два узла связаны между собой и допускаю прямую передачу данных между собой. В этом случае масштабирование реализуется простым добавлением новых узлов. Если архитектура МКС построена путем объединения узлов в группы (подкластеры), то взаимодействие осуществляется между подкластерами и внутри каждого подкластера. При этом возможно использование различных коммуникационных систем для взаимодействия между подкластерами и внутри каждого подкластера.

Рассмотрим использование теории Ч.Хоара для построения моделей вычислений для масштабируемых кластерных систем с многоядерной архитектурой, в узлах которых используются многоядерных процессоры. Масштабированность КС на структурном уровне связана с возможностью изменения количества узлов и количества ядер

в многоядерных процессорах в узле системы. Теория Ч.Хоара позволяет поведение любой системы рассматривать в терминах объект, абстрагируясь при этом от аппаратной или программной сущности системы построении модели вычислений. Это позволяет упростить переход от описания аппаратной части к программной реализации модели.

При разработке модели вычислений будем исходить из того, что она должна быть двухуровневой и соответствовать двум уровням структуры МКС. Модель *МКС1* описывает поведение системы, связанное с взаимодействием узлов, а модель *МКС2* описывает поведением системы внутри каждого узла.

Модель первого уровня МКС1. Для модели *МКС1* используем модель клиент-сервер, которая включает один объект клиент и P объектов серверов. Параметр P может меняться, тем самым обеспечивая масшабируемость на множестве узлов.

Набор событий, связанных с объектом клиент K можно представить в виде:

$$\alpha K = \{K.\text{ввод}, K.\text{вывод}, K.\text{принять}.C, K.\text{передать}.C\},$$

где события: *К.ввод* – ввод данных на клиенте, *К.вывод* – вывод результат на клиенте, *К.принять.С* – групповой прием данных от всех P серверов C_i, ($i = 1..P$), *К.передать.С* – групповая передача данных всем серверам C_i. Масштабируемость здесь связана с групповыми передачами и приемами данных между клиентом и всеми серверами. Для реализации групповых операций *К.принять.С* и *К.передать.С* их необходимо параметризировать через параметр P. Кроме того, эти операции должны позволять либо передачу любого объема данных, либо допускать настройку на конкретный объем передаваемых данных. В MPI при передаче данных используются функции `Send/Receive`, в которых указывается объем передаваемых данных. В языке Ада - типы с уточняемыми границами (`type Vector is array(integer range <>) of float`), позволяющие передавать объемы данных любой размерности [8].

Набор событий, связанных с i-м объектом сервер (C_i, $i = 1 .. P$) можно представить в виде

$$\alpha C_i = \{ C_i.\text{принять}.K, C_i.\text{передать}.K, C_i.\text{счет} \}$$

где события: C_i.*принять.К* – прием данных от клиента, C_i.*передать.К* - передача данных клиенту, C_i.*счет* - счет на сервере. Масштабируемость на серверной стороне не фиксируется.

Поведение объекта клиент, используя алфавит αK, можно описать как процесс

$$K = (K.\text{ввод} \rightarrow K.\text{передать}.C \rightarrow K.\text{принять}.C \rightarrow K.\text{вывод}).$$

Для множества объектов клиентов C_i их поведение описывается следующим процессом

$$C_i = (C_i.\text{принять}.K \rightarrow C_i.\text{счет} \rightarrow C_i.\text{передать}.K).$$

Модель *МКС1* можно описать как параллельную комбинацию поведения процессов K и C_i :

$$\text{СЕРВЕРЫ} = (C_1 \parallel C_2 \parallel \ldots \parallel C_p),$$
$$\text{КЛИЕНТ} = (K),$$
$$\text{МКС1} = (\text{КЛИЕНТ} \parallel \text{СЕРВЕРЫ}).$$

Взаимодействие процессов в модели МКС1. Взаимодействие процессов в модели *МКС1* осуществляется на основе посылки сообщений. Для описания взаимодействия процессов используем объект коммуникации (*ОК*), предложенный в работе [2]. Взаимодействие через *ОК* реализуется как обращение к объекту (вызов *ОК* и прием вызова *ОК*) :

$$\alpha OK = \{ OK.\text{передать}, OK.\text{принять} \}.$$

Процесс, который описывает поведение объекта коммуникации, задается рекурсивным соотношением:

$$OK = ((OK.\text{принять} \mid OK.\text{передать}) \rightarrow OK).$$

ОК является параметризированным объектом *ОК(Д, Н, В, Б)*, где параметр *Д* определяет тип и объем передаваемых данных, параметр *Н* - направление передаваемых данных, параметр *В* - синхронное или асинхронное взаимодействие, параметр *Б* – буферизированное или не буферизованное взаимодействие. Для реализации масштабированности параметр *Д* должен допускать возможность указания объема передаваемых данных (*Д.Х*), зависящего от P.

В теории Ч.Хоара поведение объекта *ОК* описывается процессом, который является подчиненным для процессов, которые через него взаимодействуют. Корректность такого взаимодействия определяется следующими правилами:

Правило 1 определяет необходимое условие взаимодействия процессов через *ОК* относительно алфавитов процессов: “Если A и

B процессы, *Z* - объект коммуникации, то необходимым условием взаимодействия *A* и *B* через *Z* есть выполнение соотношения $\alpha Z \subseteq (\alpha A \cup \alpha B)$ ".

Правило 2 определяет необходимое условие взаимодействия процессов через *ОК* относительно поведения процессов: "Если *A* и *B* процессы, *Z* - объект коммуникации, то необходимым условием взаимодействия *A* и *B* через *Z* есть следующие действия в процессах $Z = ((Z.принять \mid Z.передать) \rightarrow Z)$, $A = ((Z.передать) \rightarrow A)$, $B = ((Z.принять) \rightarrow B)$, $(Z \;//\; (A \parallel B))$. При этом $A.Z.Д.X = B.Z.Д.X$".

Реализация объекта коммуникации для взаимодействия узлов МКС можно выполнить различными способами. С учетом сетевой технологии, которая определяет аппаратное взаимодействие узлов, это могут быть сокеты, а также удаленные методы [6]. При использовании удаленных методов для операции умножения вектора на матрицу $A = B*MC$ в МКС, включающей *P* серверов, каждый сервер содержит удаленную процедуру `Счет()`, реализующую вычисления части результата $A_н=B*MC_н$, где H = N/P.

Описание удаленной процедуры на стороне сервера:

```
type Vector is array(integer
                  range    <>)    of
float;
type Matrix is array(integer
          range    <>)    of    Vec-
tor(1..N);
procedure Счет(B: in Vector; MC: in
             Matrix;     A:     out
Vector);
```

Удаленный вызов на стороне клиента для *i*-го сервера:

```
Сервер.Счет(B(1..N),MC(h*(i-1)+1..
             i*H),A(h*(i-
1)+1..i*H);
```

Модель второго уровня МКС2. Для модели *МКС2,* описывающей поведение узла, используем модель мастер-рабочий. Объект мастер отвечает за связь с клиентским узлом МКС и здесь связываются первый и второй уровни системы. Объект мастер (*B*) принимает исходные данные от клиента, распределяет их в узле по всем *K* объектам рабочий, собирает результаты счета в узле и отсылает результат клиенту. Объект рабочий (*W*) принимает данные от объекта мастер, выполняет счет и возвращает результат мастеру.

Набор событий, связанных с объектом мастер *i*-го узла (B_i), можно представить в виде

$\alpha B_i = \{B_i.принять.W_{ij}, B_i.передать.W_{ij}, B_i.счет\}$,

где события: $B_i.передать.W_{ij}$ – передать данные в узле объекту рабочий W_{ij}, $B_i.счет$ – счет, $B_i.принять.W_{ij}$ – принять данные в узле от объекта рабочий W_{ij}, ($j = 1..K$, K – количество ядер в многоядерном процессоре).

Набор событий, связанных с объектом *j*-й рабочий *i*-го узла (W_{ij}) можно представить в виде

$\alpha W_{ij} = \{ W_{ij}.принять.B_i, W_{ij}.передать.B_i, W_{ij}.принять.W_{iк}, W_{ij}.передать.W_{iк}\}$,

где события: $W_{ij}.принять.B_i$ – прием данных от мастер объекта, $W_{ij}.передат.B_i$ – передача данных мастер объекту, $W_{ij}.счет$ – счет в объекте рабочий, $W_{ij}.принять.W_{iк}$ – прием данных от объекта рабочий $W_{iк}$, $W_{ij}.передать.W_{iк}$ – передача данных объекту рабочий $W_{iк}$.

Описание поведения объектов мастер и рабочий зависит от системы связей ядер в многоядерном процессоре, которая будет определять форму взаимодействия объектов мастер и рабочий. В общем случае для полносвязной архитектуры узла, когда все ядра связаны между собой, поведение объектов в узле на основе модели посылки сообщений можно представить следующим образом.

Поведение объекта мастер B_i (процесс мастера) используя алфавит αB_i, можно описать как

$B_i = (B_i.передать.W_{ij} \rightarrow B_i.счет \rightarrow B_i.принять.W_{ij})$.

Поведение объекта рабочий W_{ij}:

$W_{ij} = (W_{ij}.принять.B_i \rightarrow W_{ij}.счет \rightarrow W_{ij}.передать.B_i)$.

Модель второго уровня *МКС2* для *i*-го клиентского узла можно описать как параллельную комбинацию поведения процессов B_i и W_{ij}:

$$МАСТЕР_i = (B_i), \quad РАБОЧИЕ_i = (W_i),$$
$$УЗЕЛ_i = (B_i \parallel РАБОЧИЕ_i).$$

Взаимодействие процессов в модели МКС2. При рассмотрении вопросов масштабирования внутри узлов МКС, построенных с использованием многоядерных процессоров, взаимодействие процессов осуществля-

ется на основе общей памяти и модели общих переменных. Проблема масштабирования в узле МКС характерна для SMP – систем и связана с наличием общей памяти и когерентностью кэш-памяти.

В работе [2] введено понятие объекта синхронизации (*ОС*) для организации взаимодействия через общие переменные. Для реализации групповой синхронизации будем использовать объект синхронизации типа М, который реализует концепцию мониторов. При этом необходимо дополнить *ОС(М)* средствами, обеспечивающими эффективную реализацию групповой синхронизации.

Набор событий, связанных с параметризированным объектом *ОС*(*М)* для синхронизации можно представить в виде

$\alpha M(к)$ = { *М.ждать*(*к*), *М.сигнал*, *М.несигнал*, *М.установить(к)* },

где события: *М.ждать(к)* – ожидание сигнала от *к* процессов, *М.сигнал*– сигнал о событии, *М.несигнал* – установка *ОС(М)* в несигнальное состояние, *М.установить* – настроить объект на синхронизацию с *к* процессами.

В задаче групповой синхронизации, когда один процесс (А) ждет сигналов от *s* процессов (E_1 .. E_s), необходимо поведение *ОС(М)* описать как *М* = ((*М.несигнал*) → *М.установить(s)* →((*М.ждать(s)* | *М.сигнал*) → *М*)), ожидание сигналов в процесс *А* описать как *М.ждать(s),* а посылку сигналов в процессах E_1 .. E_s реализовать событием *М.сигнал.*

Использование моделей вычислений

После построения модели вычислений она может быть использована для анализа корректности взаимодействия процессов, который выполняется на основе проверки предложенных правил. Это позволит выявить и предотвратить тупиковые ситуации, оптимизировать объемы передаваемой информации, выбрать более эффективные средства коммуникации и синхронизации.

Второй важной стороной моделей вычислений является возможность их применения при разработке распределенных приложений. Использование моделей позволяет упростить построение алгоритмов процессов и их последующую программную реализацию. Построение алгоритмов процессов осуществляется в терминах событий, которые были определены и использовались для объектов моделей. Сами алгоритмы полностью соответствуют поведению объектов в модели вычислений. При реализации модели следует учитывать тот факт, что большинство современных языков параллельного программирования реализуют концепцию взаимодействующих последовательных процессов, составляющих основу теории Ч.Хоара. Это упрощает реализацию процессов для объектов с помощью процессов (потоков) в программе.

Выбор и реализация объектов коммуникации и синхронизации также упрощается, так как они разработаны с учетом и анализом различных существующих механизмов взаимодействия процессов, таких как семафоры, мютексы, критические секции, замки, мониторы, рандеву [7].

Авторами были проведены экспериментальные исследования эффективности использования предложенных моделей вычислений, построенных на основе теории Ч.Хоара, для разработки приложений для реальной масштабируемой кластерной системы, оснащенной четырехядерными процессорами. Модели вычислений позволили описать особенности архитектуры КСМА, включающей переменное число узлов. Использование моделей вычислений для разработки распределенных приложений позволило сократить время их программирования и отладки. Использовались языки программирования C, Java, C#, Ада, библиотеки Win32, MPI. Разработанный пакет программ эффективно выполнялся в МКС с любым числом узлов, обеспечивая высокоскоростную передачу данных между узлами и высокий коэффициент загрузки многоядерных процессоров в узлах.

Выводы

Предложен подход к построению моделей вычислений для масштабируемых кластерных систем с многоядерной архитектурой, основанный на расширении теории взаимодействующих последовательных процессов путем введения параметризованных объектов синхронизации и коммуникации. Модели позволяют описать архитектуру кластерной системы, поведение процессов и их взаимодействие с учетом особенностей масштабированности системы. Математический аппарат CSP теории позволил сформулировать ряд требований для оптимального взаимо-

действия процессов, выполнить анализ корректности взаимодействия процессов на обоих уровнях кластерной системы при использовании модели, основанной на общих переменных, и модели, основанной посылке сообщений.

Использование предложенных моделей при проектировании программных компонент МКС позволит повысить качество программ и сократить время их разработки.

Список литературы

1. TOP500 Supercomputing sites. http://www.top500.org
2. Садег Растгу. Модели вычислительных процессов в кластерных системах с многоядерной архитектурой // Вісник НТТУ “КПІ”.Сер. Інформатика, управління та обчислювальна техніка. № 48. – 2008. С.58-62.
3. Хоар Ч.Взаимодействующие последовательные процессы: Пер. с англ. – М.: Мир, 1989 .– 264 с.
4. Топорков В.В. Модели распределенных вычислений. – М.: ФИЗМАТЛИТ, 2004. – 320 с.
5. Любченко В.С. Автоматная модель параллельных вычислений / В.С.Любченко // Высокопроизводительные параллельные вычисления на кластерных системах : 3-я межд. конф. 2-4 окт. 2006 г. : труды конф. – М. - 2006. – С. 1359 – 1374.
6. Березовский А. Организация вычислений в масштабируемых кластерных системах // Комп’ютерні системи та мережеві технології: друга міжнар.наук.- техн.конф., CSNT’2009, 10-12 черв. 2009 р.: збірник наукових праць. – 2009, вип. 2(24).
7. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення. Навч. посібник. – К.: Корнійчук, 2005. – 226 с.
8. Корочкин А.В. Ада95. Введение в программирование. – К.: Корнійчук, 1998. – 240 с.
9. Каляев И.А. Реконфигурируемые мультиконвейерные вычислительные структуры / И.А.Каляев, И.И.Левин, Е.А.Семерников, В.И.Шмойлов. – М.: ид-во ЮНЦ РАН, 2008. – 320 с.

Поступила в редакцию 14.12.2009

УДК 004.75

ЗАЙЧЕНКО Ю. П.,
ЛАБЖИНСКИЙ В. А.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ДВУХ МЕТОДОВ ПРОГНОЗИРОВАНИЯ ЗАГРЯЗНЕНИЯ ВОЗДУШНОГО БАССЕЙНА

В данной работе рассмотрена задача краткосрочного прогнозирования концентраций загрязняющих веществ в атмосфере с помощью двух статистических методов: Бокса-Дженкинса и метода группового учета аргументов. Приведено описание методик построения прогнозирующих моделей для обоих методов, а также сравнительная характеристика результатов прогноза.

In the given work short-term forecasting of pollutants' concentration in atmosphere have been considered with two statistical methods: Box-Jenkins and group method of data handling. The description of forecasting models building procedures and comparative characteristic of predictions' results has been adduced for both methods.

Введение

Резкий рост уровня антропогенного воздействия на окружающую среду в сочетании с низкой эффективностью и разобщенностью природоохранных мероприятий привел за последние десятилетия к значительному ухудшению экологической обстановки в различных регионах Украины. Наиболее уязвимыми оказались промышленно-развитые районы с высокой концентрацией промышленных производств топливно-энергетического, металлургического, химического и др. профилей, приведших к неконтролируемому превышению уровней допустимой экологической нагрузки на окружающую среду, возникновению потенциально опасных чрезвычайных ситуаций [1].

В связи с этим проблемы снижения эколого-экономического ущерба и управления экологической ситуацией приобретают в последнее время все возрастающее значение. Наряду с оценкой и контролем концентраций вредных примесей в воздухе в районе источников выбросов по данным наблюдений также необходимо осуществлять краткосрочные прогнозы загрязнения воздуха и использовать их для регулирования промышленных выбросов. Приоритетность регулирования выбросов вредных веществ в атмосферу обусловлена доминирующей ролью этой проблемы в качестве отправной точки при создании технологий, повышающих эффективность и действенность экологических нормативных ограничений загрязнения в целом.

Процесс загрязнения воздуха, как и многие другие атмосферные процессы, отличается большой сложностью. Методы исследования и прогноза, базирующиеся на использовании уравнений математической физики (теории диффузии и переноса примесей), учет метеофакторов (скорости и направления ветра, условий температурной стратификации), особенностей подстилающей поверхности, химического превращения примесей и т. п., обеспечивают получение результатов с высокой степенью точности [2–4]. Однако алгоритмы, реализующие эти методы, громоздки, требуют больших затрат машинного времени и ресурсов памяти ЭВМ. Указанные методы целесообразно применять при решении задач долгосрочного прогнозирования, исследования тенденций загрязнения воздушной среды за длительные промежутки времени (месяц, сезон и т. п.).

В свою очередь, для краткосрочного прогнозирования загрязнения воздуха в городах и промышленных районах, обусловленного действием многих источников, целесообразно использовать статистические методы, основанные на анализе материалов наблюдений. Здесь предполагается, что за период, к которому относится исследуемый материал, а также за срок прогноза, выбросы и расположение источников выбросов практически не изменяются. Поэтому рост среднего и суммарного загрязнения воздуха в городе связывается главным образом с изменением метеорологических условий или синоптической ситуации [5].

Разработка методов прогноза начинается в первую очередь с выявления периодов наибольшего загрязнения атмосферы. Затем устанавливаются корреляционные зависимости между наблюдавшимися в эти периоды степенью загрязнения воздуха и некоторыми метеорологическими величинами или их определенным сочетанием, рассматриваемыми в качестве предикторов. Таким путем вырабатывают различные прогностические правила. Также используют методы статистической экстраполяции во времени режима изменения загрязнения воздуха с учетом выявленных автокорреляционных зависимостей и инерционных факторов [2].

Постановка задачи

Рассмотрим один из вариантов постановки задачи моделирования загрязнения окружающей среды, а именно: моделирование загрязнения выполняется исключительно по данным измерений нескольких контрольно-измерительных станций (КИС) [6].

В этой задаче уравнение диффузии не задается априори и находится по экспериментальным данным по принципу самоорганизации при помощи перебора шаблонов и их нелинейностей. Оценки коэффициентов определяются с помощью метода наименьших квадратов (МНК). Решение находится при помощи пошагового интегрирования конечно-разностного уравнения. Знать это решение в аналитической форме не требуется.

Исходной информацией для прогнозирования концентрации загрязняющих веществ являются выборочные данные, представляющие собой замеры концентрации загрязняющих веществ через фиксированные интервалы времени в местах установки КИС. Таким образом, выборка представляет собой многомерный временной ряд.

Такая постановка задачи моделирования поля имеет целью построение поля загрязнения не только в области интерполяции (охваченной КИС), но и на значительном расстоянии за пределами этой области, т.е. построение экстраполяции поля и прогнозов его развития во времени. При этом предполагается, что сбросы загрязнения сравнительно мало изменяются во времени, и поэтому информация о них непосредственно не учитывается. Косвенно она содержится в данных измерений КИС. Поэтому в этой постановке множество аргументов включает только данные КИС.

Метод Бокса-Дженкинса. Модель ARIMA (AutoRegressive Integrated Moving Average – модель авторегрессии – проинтегрированного скользящего среднего)

Модель ARIMA относится к классу линейных моделей, которая может хорошо описывать поведение как стационарных, так и нестационарных временных рядов. Для прогнозирования модель ARIMA использует информацию, содержащуюся в исходном ряде. Модель ARIMA опирается, в основном, на автокорреляционную структуру данных временного ряда.

Внутренняя структура временного ряда, то есть зависимость его уровня y_t от предыдущих значений $y_{t-1}, y_{t-2}, ..., y_{t-p}$ может быть описана авторегрессионной функцией

$$y_t = \mu + a_1 \cdot y_{t-1} + a_2 \cdot y_{t-2} + ... + a_p \cdot y_{t-p} + \varepsilon_t,$$

где μ – константа, p – порядок авторегрессии, $a_1, ..., a_p$ – коэффициенты авторегрессии, ε_t – белый шум. Другими словами, каждое значение временного ряда есть сумма, содержащая в своем составе линейную комбинацию p его предыдущих значений и случайную ошибку ε_t.

Другим типом модели, описывающей поведение временного ряда, является модель скользящего среднего, в которой уровень ряда y_t линейно зависит от q предыдущих значений белого шума ε:

$$y_t = \mu + \varepsilon_t - \theta_1 \cdot \varepsilon_{t-1} - \theta_2 \cdot \varepsilon_{t-2} - ... - \theta_q \cdot \varepsilon_{t-q},$$

где символы $-\theta_1, -\theta_2, ..., -\theta_q$ используются для обозначения конечного набора весовых параметров. Эта модель называется процессом скользящего среднего порядка q. Другими словами, каждое значение временного ряда есть сумма, содержащая в своем составе линейную комбинацию q предыдущих значений белого шума и случайную ошибку ε_t.

Для достижения большей гибкости при построении модели исследуемых процессов полезно включать в нее как выражение для скользящего среднего, так и выражение для авторегрессии. Это приводит к смешанной модели ARMA (p, q):

$$y_t = \mu + a_1 \cdot y_{t-1} + a_2 \cdot y_{t-2} + ... + a_p \cdot y_{t-p} + \\ + \varepsilon_t - \theta_1 \cdot \varepsilon_{t-1} - \theta_2 \cdot \varepsilon_{t-2} - ... - \theta_q \cdot \varepsilon_{t-q}.$$

Если в модели необходимо учитывать наличие тренда, то это делается введением в ряд, описываемый моделью ARMA, разностей *d*-го порядка вместо значений y_t. В частности, $\nabla^1(y_t) = y_t - y_{t-1}$ (линейный тренд), $\nabla^2(y_t) = y_t - y_{t-2}$ (квадратичный тренд) и т. д. Разности ∇^d должны быть стационарными. В результате получаем модель ARIMA порядка (*p*, *d*, *q*) [7]:

$$\nabla^d(y_t) = \mu + a_1 \cdot \nabla^d(y_{t-1}) + a_2 \cdot \nabla^d(y_{t-2}) + ... \\ + a_p \cdot \nabla^d(y_{t-p}) + \varepsilon_t - \theta_1 \cdot \varepsilon_{t-1} - \theta_2 \cdot \varepsilon_{t-2} - ... - \theta_q \cdot \varepsilon_{t-q}.$$

Вид модели ARIMA, ее адекватность реальному процессу и прогнозные свойства зависят от порядка авторегрессии *p* и порядка скользящего среднего *q*. Поэтому ключевым моментом моделирования является процедура идентификации – обоснования выбора модели того или иного вида. Выбор начальной модели ARIMA основывается на изучении графиков автокорреляций и частных автокорреляций между данными исходного временного ряда. Выбранная модель сопоставляется с исходными данными, чтобы проверить, насколько точно она описывает временной ряд. Модель считается приемлемой, если остатки малы и распределены случайно [8].

Метод группового учета аргументов (МГУА)

В основу МГУА [9, 10] положен принцип самоорганизации, и алгоритмы МГУА воспроизводят схему массовой селекции. В алгоритмах МГУА особым образом синтезируются и отбираются члены полинома, который называют обобщенным полиномом Колмогорова-Габора:

$$Y = a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{j=1}^{N} \sum_{i \le j} a_{ij} x_i x_j + \sum_{i=1}^{N} \sum_{j \le i} \sum_{k \le j} a_{ijk} x_i x_j x_k + ...$$

где *N* – количество независимых переменных.

Этот синтез и отбор производится с нарастающим усложнением, и заранее нельзя предугадать, какой окончательный вид будет иметь обобщенный полином. Сначала обычно рассматривают простые попарные комбинации исходных переменных, из которых составляются уравнения решающих функций, как правило, не выше второго порядка. Каждое уравнение анализируется как самостоятельная решающая функция, и по обучающей выборке тем или иным способом находятся значения параметров составленных уравнений. Затем из полученного набора решающих функций отбирается часть в некотором смысле лучших. Проверка качества отдельных решающих функций осуществляется на контрольной (проверочной) выборке, что иногда называют принципом внешнего дополнения. Отобранные частные решающие функции рассматриваются далее как промежуточные переменные, служащие исходными аргументами для аналогичного синтеза новых решающих функций и т. д. Процесс такого иерархического синтеза продолжается до тех пор, пока не будет достигнут экстремум критерия качества решающей функции, что на практике проявляется в ухудшении этого критерия при попытках дальнейшего увеличения порядка членов полинома относительно исходных переменных [9].

Рассмотрим процесс синтеза модели оптимальной сложности более подробно. Представим функцию, аппроксимирующую набор исходных данных, в общем виде: $y = F(x_1, x_2, ..., x_N)$. Выше упоминалось, что такой функцией может быть полином Колмогорова-Габора, с помощью которого можно добиться весьма точной аппроксимации любой дифференцируемой функции. Заменим эту сложную зависимость множеством частных описаний, т.е. простых функций, аргументами которых являются все возможные пары исходных аргументов:

$y_1 = f(x_1, x_2), y_2 = f(x_1, x_3), ..., y_s = f(x_{N-1}, x_N)$,

где $s = C_N^2$, причем вид функции *f* одинаков для всех пар в течение всего процесса обучения. Очень часто в качестве функции *f* выбираются простые зависимости

$$y(x_i, x_j) = a_0 + a_1 \cdot x_i + a_2 \cdot x_j + a_3 \cdot x_i \cdot x_j$$

или

$$y(x_i, x_j) = a_0 + a_1 \cdot x_i + a_2 \cdot x_j + a_3 \cdot x_i \cdot x_j + \\ + a_4 \cdot x_i^2 + a_5 \cdot x_j^2.$$

Предварительно вся выборка разделяется на две части: обучающую и проверочную. Тем самым порождается внешнее дополнение (проверочная выборка), которая играет роль сита, отсеивающего худшие частные модели. Коэффициенты $a_0 - a_5$ частных опи-

саний определяются с помощью МНК по данным обучающей выборки.

В результате комбинирования всех возможных пар из N исходных аргументов получается множество решений, поскольку частное уравнение каждой пары рассматривается как некоторая упрощенная модель восстанавливаемой функции. Из полученного набора упрощенных моделей первого ряда отбирается часть, например, t в некотором смысле наилучших, показавших хорошие результаты на проверочной выборке, не участвовавшей в определении коэффициентов уравнений (т.е. на внешнем дополнении).

Далее вступает в действие принцип неокончательности решений: ни одна из полученных на первом этапе моделей не принимается за истину и наращивание сложности модели продолжается. Отобранные частные описания формируют множество новых переменных, которые являются исходными аргументами для частных описаний второго ряда селекции [10]:

$$z_1 = f\left(y_i, y_j\right), z_2 = f\left(y_i, y_k\right), ..., z_u = f\left(y_{t-1}, y_t\right),$$

где $u = C_t^2$.

Коэффициенты новых моделей находятся по МНК на точках той же обучающей последовательности. Новые модели проверяются на точках проверочной последовательности, и среди них выбирается v наилучших, которые используются в качестве аргументов следующего третьего ряда и т. д.

Сложность общей модели возрастает от ряда к ряду. Так, например, во втором ряду могут появиться нелинейные члены вида $\left(x_1 \cdot x_2 \cdot x_3\right), \left(x_1^2 \cdot x_3\right), \left(x_1^2 \cdot x_2 \cdot x_3\right)$ и т. д. Алгоритм останавливается сразу же по достижении минимума внешнего критерия (обычно среднеквадратичного отклонения или несмещенности), полученных на проверочной выборке. На практике усложнение модели прекращают, когда дальнейшее улучшение критерия селекции, например, средний квадрат ошибки прогноза, не будет превышать некоторого числа ε (параметр алгоритма). Тем самым выбирается модель оптимальной сложности, устанавливающая компромисс между сложностью и опасностью “переобучения” [10].

Существуют различные разновидности МГУА: однорядные, многорядные, гибридные. Для решения поставленной задачи будем использовать идею многорядного алгоритма МГУА.

Сравнение результатов прогнозирования с использованием двух методов

Выбор именно этих двух методов для сравнительного анализа точности получаемого прогноза объясняется следующими причинами. Метод Бокса-Дженкинса является классическим статистическим методом прогнозирования, широко применяемым, в частности, для получения краткосрочного прогноза состояния воздушного бассейна в автоматизированных системах контроля загрязнения воздуха [3]. Метод МГУА, в свою очередь, в этих целях почти не применяется [1]. Поэтому интересно сравнить результаты прогнозирования, полученные с помощью этих двух методов. Исходные данные по загрязнению воздуха оксидом углерода (CO), диоксидом азота (NO_2) и диоксидом серы (SO_2) одного из районов Лондона (Bexley) за май 2009 г. были взяты с сайта Национального архива загрязнения воздуха Великобритании [11]. Показания снимались в конце каждого часа. Объем выборки составил 744 наблюдения. Выборка была разделена на две части в соотношении 2:1 на обучающую и проверочную соответственно. Для каждого загрязнителя ниже представлены по две лучшие (каждая в своем классе) прогнозные модели.

Перед моделированием данные выборки были нормализованы с использованием формулы

$$\tilde{y}_t = 2 \cdot \frac{y_t - y_t^{(\min)}}{y_t^{(\max)} - y_t^{(\min)}} - 1$$

для приведения к диапазону изменения [-1; 1]. Оптимальный порядок моделей АРПСС выбирался на основе информационного критерия Акаике [12]. Оптимальный порядок моделей МГУА выбирался на основе критерия PSE (predicted squared error – квадрат ошибки прогноза) [13].

Лучшей моделью для описания концентрации CO (рис. 1), полученной с помощью метода Бокса-Дженкинса, оказалась модель АРПСС(1,0,0):

$$\tilde{y}_t = -0{,}8605 + 0{,}8041 \cdot \tilde{y}_{t-1} + \varepsilon_t .$$

Лучшей моделью для описания концентрации CO, полученной с помощью метода МГУА, оказалась следующая модель:

$$\tilde{y}_t = -0,15 + 1,4 \cdot \tilde{y}_{t-1} - 1,4 \cdot \tilde{y}_{t-3} + 0,65 \cdot \tilde{y}_{t-4} -$$
$$-0,37 \cdot \tilde{y}_{t-1}^3 + 0,4 \cdot \tilde{y}_{t-3}^3 - 0,96 \cdot \tilde{y}_{t-1} \cdot \tilde{y}_{t-3} + \quad .$$
$$+0,53 \cdot \tilde{y}_{t-1} \cdot \tilde{y}_{t-4} - 0,26 \cdot \tilde{y}_{t-1} \cdot \tilde{y}_{t-3} \cdot \tilde{y}_{t-4}$$

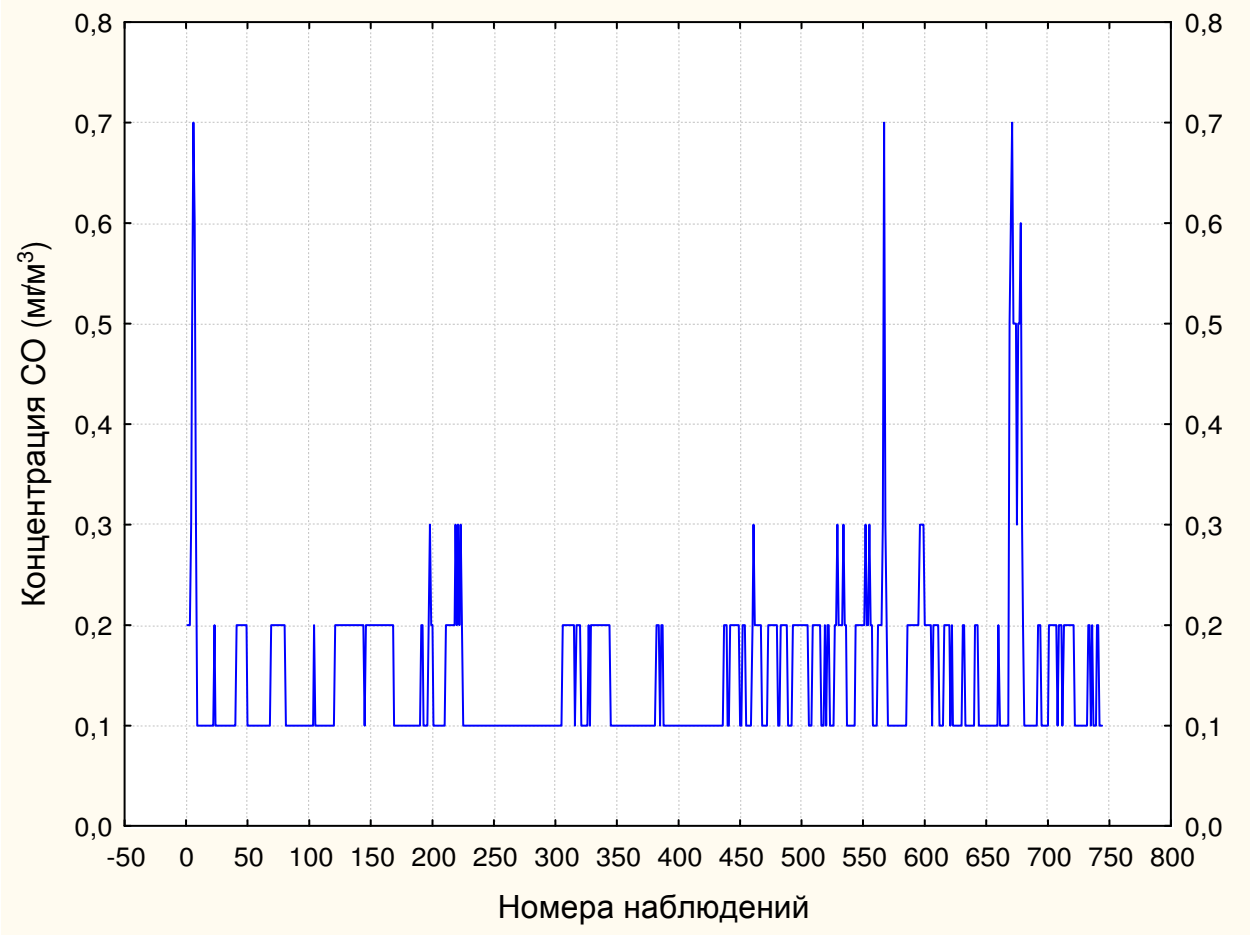


Рис. 1. Зависимость концентрации CO от времени

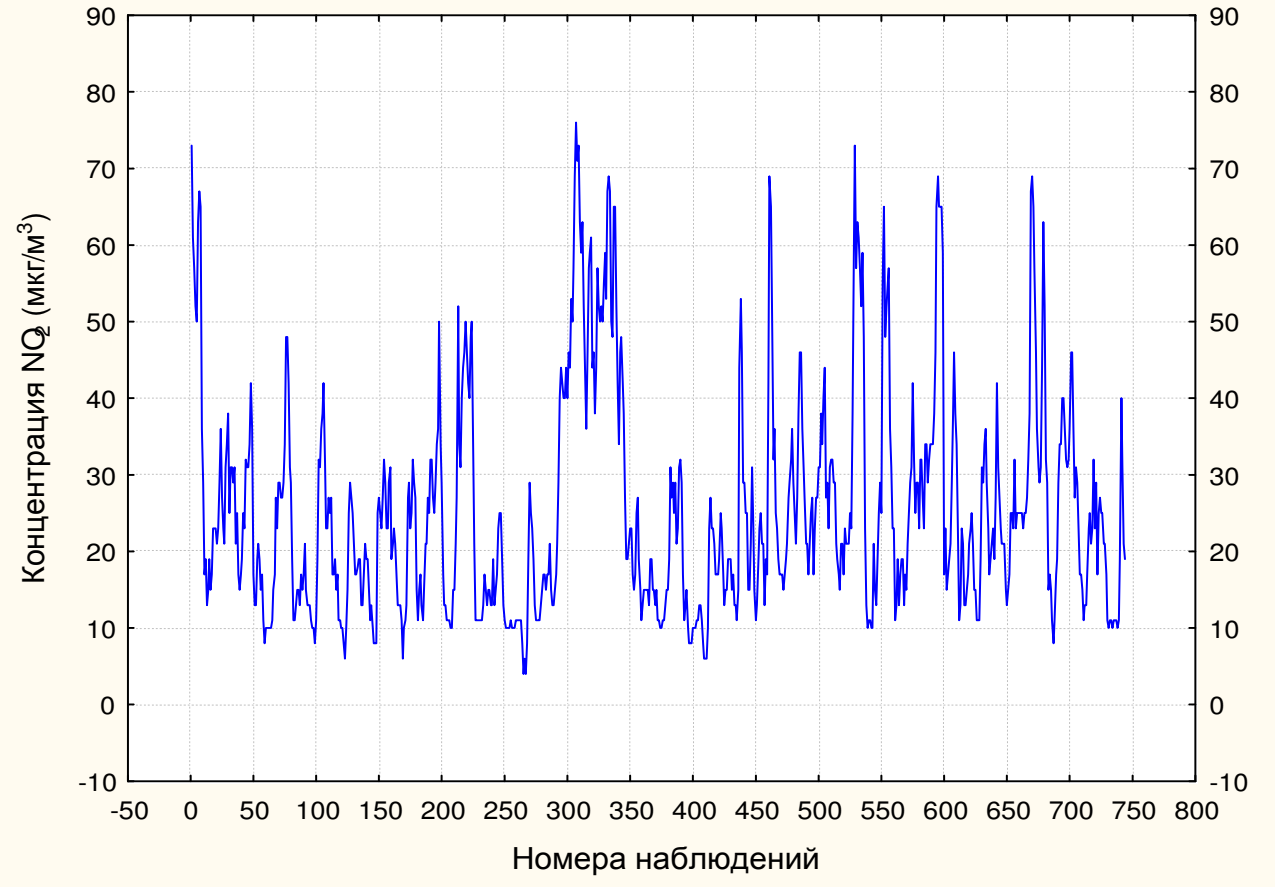


Рис. 2. Зависимость концентрации NO_2 от времени

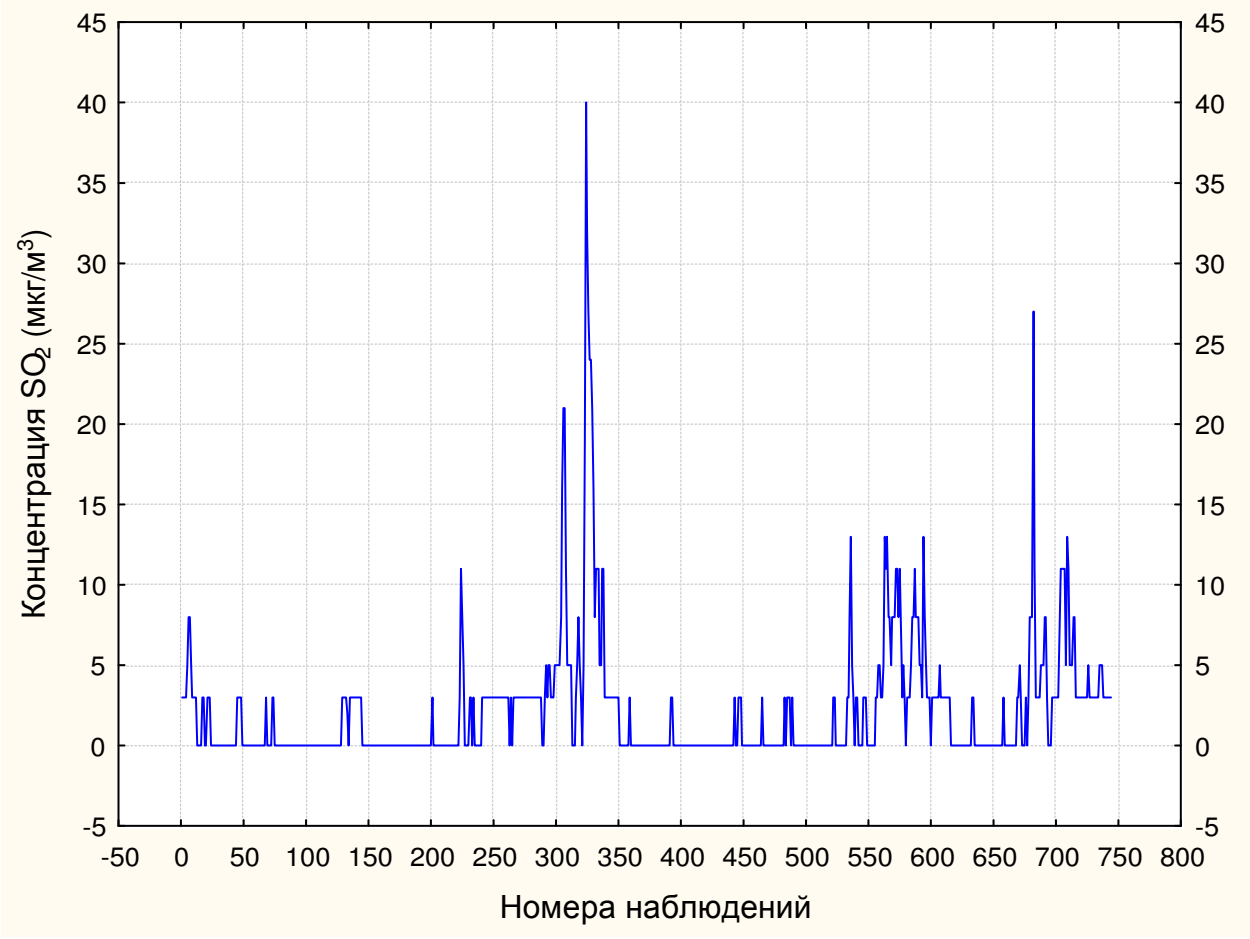


Рис. 3. Зависимость концентрации SO_2 от времени

Лучшей моделью для описания концентрации NO_2 (рис. 2), полученной с помощью метода Бокса-Дженкинса, оказалась модель АРПСС(1,0,1):

$$\tilde{y}_t = 0,9356 + 0,9915 \cdot \tilde{y}_{t-1} + \varepsilon_t + 0,1955 \cdot \varepsilon_{t-1}$$

Лучшей моделью для описания концентрации NO_2, полученной с помощью метода МГУА, оказалась следующая модель:

$$\tilde{y}_t = -0,045 + 1,1 \cdot \tilde{y}_{t-1} - 0,3 \cdot \tilde{y}_{t-2} + 0,13 \cdot \tilde{y}_{t-4} - -0,16 \cdot \tilde{y}_{t-1} \cdot \tilde{y}_{t-2} + 0,21 \cdot \tilde{y}_{t-2} \cdot \tilde{y}_{t-4}.$$

Лучшей моделью для описания концентрации SO_2 (рис. 3), полученной с помощью метода Бокса-Дженкинса, оказалась модель АРПСС(1,0,1):

$$\tilde{y}_t = 0,9928 \cdot \tilde{y}_{t-1} + \varepsilon_t + 0,1780 \cdot \varepsilon_{t-1}.$$

Лучшей моделью для описания концентрации SO_2, полученной с помощью метода МГУА, оказалась следующая модель:

$$\tilde{y}_t = -0,22 + 1,3 \cdot \tilde{y}_{t-1} - 0,77 \cdot \tilde{y}_{t-2} + 0,29 \cdot \tilde{y}_{t-4} - -0,59 \cdot \tilde{y}_{t-1}^2 + 1,9 \cdot \tilde{y}_{t-2}^2 - 0,15 \cdot \tilde{y}_{t-4}^2 - 0,76 \cdot \tilde{y}_{t-1}^3 + +0,88 \cdot \tilde{y}_{t-2}^3 - 2,7 \cdot \tilde{y}_{t-1} \cdot \tilde{y}_{t-2} + 0,44 \cdot \tilde{y}_{t-2} \cdot \tilde{y}_{t-4}$$

Результаты прогнозирования сведены в табл. 1.

Табл. 1. Средний квадрат ошибки прогнозирования загрязнения воздуха

Загрязняющее вещество	*Метод прогнозирования*			
	Бокса-Дженкинса		МГУА	
	Выборка			
	обучающая	проверочная	обучающая	проверочная
оксид углерода (CO)	0,0155	0,0509	0,0119	0,0709
диоксид азота (NO_2)	0,0281	0,0534	0,0249	0,0436
диоксид серы (SO_2)	0,0106	0,0229	0,0052	0,0393

Выводы

Подход Бокса-Дженкинса к анализу временных рядов является весьма мощным инструментом для построения точных краткосрочных прогнозов. Модели ARIMA достаточно гибкие и могут описывать многие временные ряды, встречающиеся на практике. Формальная процедура проверки модели на адекватность проста и доступна. Однако использование моделей ARIMA имеет несколько недостатков:

1. Необходимо относительно большое количество исходных данных. При использовании модели ARIMA для несезонных данных необходимо наличие значительного количества наблюдений (не менее 50), которые не всегда возможно осуществить.
2. Построение удовлетворительной модели ARIMA зачастую требует больших затрат временных и вычислительных ресурсов.

Кроме того, перед построением модели Бокса-Дженкинса необходимо провести анализ данных на однородность. В ряде случаев следует усилить однородность путем преобразования части исходных данных. Здесь к каждому ряду необходимо применять индивидуальный подход, перед построением модели проводить тщательный качественный анализ исходного ряда.

МГУА является лучшим методом для решения задач идентификации и краткосрочного прогноза. Математическая теория МГУА показала, что регрессионный анализ является частным случаем МГУА. Наиболее целесообразно функциональные предикторы использовать в краткосрочном прогнозировании. Надежность таких прогнозов может быть достаточно высокой.

МГУА по сравнению с другими статистическими методами имеет ряд преимуществ:

1. Основное преимущество – позволяет найти как структуру модели, так и ее коэффициенты.
2. Имеется возможность работы с короткими выборками.
3. Имеется возможность работы с зашумленными данными.
4. Имеется возможность решения задач большой размерности, какой является задача прогнозирования полей концентрации загрязняющих веществ.

К недостаткам МГУА можно отнести большое время вычислений, так как с увеличением числа аргументов N и/или степени обобщенного полинома p число возможных вариантов экспоненциально возрастает.

Точность прогнозов, полученных на основе двух рассмотренных выше статистических моделей, вполне удовлетворительна при установившемся состоянии атмосферных процессов и малом (до 8 часов) времени прогноза.

Список литературы

1. Ковальчук П. І. Моделювання і прогнозування стану навколишнього середовища: Навч. посібник. – К.: Либідь, 2003. – 208 с.
2. Берлянд М. Е. Прогноз и регулирование загрязнения атмосферы. – Л.: Гидрометеоиздат, 1985. – 272 с.
3. Примак А. В., Щербань А. Н., Сорока А. С. Автоматизированные системы защиты воздушного бассейна от загрязнения. – К.: Техніка, 1988. – 166 с.
4. Бретшнайдер Б., Курфюрст И. Охрана воздушного бассейна от загрязнений: технология и контроль: Пер. с англ. – Л.: Химия, 1989. – 288 с.
5. Каменева І. П., Яцишин А. В., Полішко Д. О., Попов О. О. Комплексний аналіз екологічної безпеки міста на основі сучасних ГІС-технологій // Екологія довкілля та безпека життєдіяльності. – 2008. – № 5. – С. 41–46.
6. Головін В. В. Інформаційно-логічна структура регіональної системи моніторингу довкілля // Екологія довкілля та безпека життєдіяльності. – 2004. – № 5. – С. 73–79.
7. Бокс Дж., Дженкинс Г. Анализ временных рядов. Прогноз и управление. Вып. 1: Пер. с англ. – М.: Мир, 1974. – 408 с.
8. Лукашин Ю. П. Адаптивные методы краткосрочного прогнозирования временных рядов. – М.: Финансы и статистика, 2003. – 416 с.
9. Ивахненко А. Г., Юрачковский Ю. П. Моделирование сложных систем по экспериментальным данным. - М.: Радио и связь, 1987. – 120 с.
10. Ивахненко А. Г., Степашко В. С. Помехоустойчивость моделирования. – К.: Наукова думка, 1985. – 216 с.
11. UK National Air Quality Archive. [Электронный ресурс]. (http://www.airquality.co.uk/data_selector.php). Проверено 12.10.2009.
12. Современные методы идентификации систем: Пер. с англ. / Под ред. П. Эйкхоффа. – М.: Мир, 1983. – 400 с.
13. Barron A. R. Predicted squared error: a criterion for automatic model selection // Self-organizing methods in modeling / Edited by S. J. Farlow. – New York: Marcel Dekker, 1984. – P. 87-103.

Поступила в редакцию 21.12.2009

УДК 007.51: 519.712.3

МОЛЧАНОВСЬКИЙ О.І.,
СТЕПАНОВ І.О.,
КОВТУН Д.І.

ПОРІВНЯННЯ АЛГОРИТМУ ЖАДІБНОГО РЯДКОВОГО ЗАМОЩЕННЯ І МЕТОДУ ВІДБИТКІВ ПРИ ВИЯВЛЕННІ ПЛАГІАТУ У ВИХІДНИХ КОДАХ ПРОГРАМ

У роботі наводиться порівняльний аналіз алгоритмів жадібного рядкового замощення і методу відбитків для попарного порівняння вихідних кодів програм для пошуку повних або часткових співпадінь у них. Наводяться експериментальні дані, що дозволяють порівняти ефективність і часові характеристики цих алгоритмів для різних вхідних даних і параметрів. На основі проведених експериментів обрано алгоритм для подальшої розробки системи пошуку плагіату у вихідних кодах програм.

In this research comparative analysis of the algorithms for pair comparison of the source codes is given. The received experimental data that gives capability to compare the efficiency and time characteristics of these algorithms are given for different input data and parameters. The optimal algorithm for future researches is selected based on taken experiments.

Вступ

В наш час інтенсивного розвитку інформаційних технологій інтелектуальна власність стає все більш цінною, тому виникла необхідність її захисту. Це означає, що потрібні потужні засоби для боротьби з плагіатом і, насамперед, виявлення його.

Задача пошуку плагіату, як і поняття плагіату, досить широка і нечітка, тому спробуємо для початку привести визначення цього поняття.

Плагіат – це вид порушення авторських прав, який являє собою протизаконне використання під своїм ім'ям чужого витвору (наукового, літературного, музичного), чи винаходу, раціоналізаторської ідеї (повністю або частково) не вказуючи джерело запозичення. Вимушення до співавторства теж є плагіатом.

Задачею системи пошуку плагіату є автоматичне виявлення (за заданими критеріями) того, чи була використана у програмі деяка чужа ідея. На практиці певним чином задаються деяка функція близькості і поріг, за якими можна визначити наскільки ймовірно, що деяка частина програмного коду була запозичена [1].

Для вирішення цієї задачі найчастіше використовують структурні методи аналізу, що передбачає порівняння програмних кодів з урахуванням їх структури. Такі методи досліджують структуру програми не ізольовано, а як би в контексті, встановлюють взаємозв'язок різних характеристик, їх спільну поведінку.

Недоліком структурних методів є їх складність та важкість обчислень. Класичним прикладом структурного підходу є побудова дерева програми з наступним порівнянням дерев для різних програм. Складність такого методу кубічна, що не дає можливості скільки-небудь ефективно його застосовувати для великої кількості досить довгих програм. Крім того, структурні методи зазвичай спираються на синтаксис конкретної мови програмування. Адаптація методу для іншої мови потребує значних зусиль. Складність реалізації алгоритмів, що порівнюють структуру програм, є платою за їх точність.

Тому для запланованої системи необхідно обрати максимально швидкий алгоритм, що можна визначити лише порівнянням існуючих алгоритмів між собою. Досліджуючи існуючі системи і створені алгоритми варто виділити два досить прості алгоритми – алгоритм жадібного рядкового замощення і метод відбитків.

Єдиним ефективним шляхом їх дослідження є реалізація цих алгоритмів і збір значної кількості даних на великій виборці програних кодів. Це дозволить порівняти отримані дані за допомогою таких загальновідомих продуктів як Microsoft Excel, використовуючи його зручні засоби для обробки

табличних даних. На основі цієї обробки і будуть зроблені певні висновки про реалізовані алгоритми і їх ефективність у контексті запланованої системи.

Опис алгоритмів, що порівнюються

Для початку розглянемо евристичний алгоритм отримання жадібного рядкового замощення (жадібного спільного покриття рядків). Він отримує на вході два рядки символів певного алфавіту, що відповідають двом програмним кодам. Для представлення програми у вигляді такого рядка застосовують процедуру токенізації [2], яка полягає у виділенні з програмного коду усіх значущих операторів і ключових слів (наприклад: *if, else, while, for, try* та інших) та побудові рядка із символів, відповідних вибраним словам. Такі символи називають токенами, а рядок – рядком токенів, чи токенізованим представленням программного коду. Результатом застосування жадібного алгоритму для таких рядків буде набір їх спільних підрядків, що не перетинаються. Причому можна сказати, що такий набір буде близьким до оптимального. Підрядок, що входить у цей набір, ми будемо називати тайлом.

В структурі цього алгоритму можна виділити 4 цикли:

- зовнішній цикл. Його результатом є додавання у набір чергового тайлу, причому тайли додаються у порядку зменшення, починаючи із найбільшого;
- внутрішній цикл по усім символам першого рядку;
- внутрішній цикл по усім символам другого рядку;
- цикл по співпадаючим символам, починаючи від першого співпадання поточних (відносно другого і третього циклів) символів даних рядків.

Слід зазначити, що цей алгоритм використовує дві евристики:

- спочатку ми знаходимо найбільші спільні підрядки, що складаються з непомічених символів (помічаються такі символи, які вже належать певному токену, були вибрані раніше), тому вважаємо, що довгі співпадання для нашої задачі найбільш значущі. Зрозуміло, що внаслідок цієї евристики ми іноді можемо не знайти найбільш оптимальне (в плані кількості покриваючих підрядків) спільне часткове покриття, але це було б для нас абсолютно несуттєво без наступної евристики;
- якщо в алгоритмі ми будемо дозволяти враховувати надто малі найбільші спільні префікси, то випадкові співпадання невеликої кількості токенів будуть впливати на визначення подібності програм, тому, щоб цього уникнути, ми вводимо константу мінімальної довжини тайлу, яка є вхідним параметром алгоритму.

Псевдокод алгоритму:

```
Greedy-String-Tiling(String P, String T, int
MinimumMatchLength) {
    tiles = {};
    do {
      maxmatch = MinimumMatchLength;
      matches = {};
      Forall unmarked tokens P_p in P {
        Forall unmarked tokens T_t in T {
        j = 0;
        while (P_{p+j} == T_{t+j}) &&
unmarked(P_{p+j}) && unmarked(T_{t+j})
        j++;
        if (j == maxmatch) {
          matches = matches U match(p, t, j);
          } else if (j > maxmatch) {
            matches = {match(p,t, j)}:
            maxmatch = j;
            }
         }
      }
    Forall match(p,t. maxmatch) ∈ matches {
      if (not occluded) {
        for j = 0 ... (maxmatch — 1) {
          mark(P_{p+j});
          mark(T_{t+j});
        }
        tiles = tiles U matches(a, b,
maxmatch);
      }
    }
    } while (maxmatch >
MinimumMatchLength);
      return tiles;
  }
```

Табл. 1. Коментарі до псевдокоду

String P	перший рядок токенів, вхідний параметр алгоритму
String T	другий рядок токенів, вхідний параметр алгоритму
int MinimumMatchLength	довжина мінімального тайлу, вхідний параметр алгоритму
tiles	множина знайдених тайлів
maxmatch	довжина найбільшого спільного підрядку (не тайлу) на поточній ітерації зовнішнього циклу (першого з чотирьох описаних вище)
unmarked(*токен)*	певна функція, що повертає *true*, якщо відповідний токен непомічений (не належить жодному із вже знайдених тайлів), чи *false* у протилежному випадку
matches	множина найбільших (рівної довжини) підрядків, знайдених на поточній ітерації зовнішнього циклу (першого з чотирьох описаних вище)
match(p, t, j)	поточні індекси кожного із двох рядків (параметри другого і третього з описаних вище циклів), що порівнюються і знайдена довжина спільного підрядку (параметр четвертого циклу)
not occluded	жоден із токенів, які належать підрядку не був помічений на попередніх ітераціях
mark(*токен)*	помітити відповідний токен

Зрозуміло, що чим більша нормалізована сума довжин отриманих тайлів (знайдених за описаними вище правилами), тим більше схожість даних підрядків. Тому функцію подібності для цього алгоритму можна представити у вигляді:

$$sim(P,T) = \frac{2 \cdot |tiles|}{|P| + |T|}, \qquad (1)$$

де $|tiles|$ - потужність (за кількістю символів) знайденого набору тайлів, а $|P|$, $|T|$ - відповідні потужності рядків, що порівнюються.

Метод відбитків [3], або метод ідентифікаційних меток передбачає представлення токенізованого рядку у вигляді набору хешзначень (відбитків) і порівняння в подальшому цих наборів. Для цього даний рядок розбивають на $(m-(k-1))$ підрядків (де m – довжина вхідного рядку) довжини k. Мається на увазі що беруть k символів починаючи з першого, k символів починаючи з другого, k символів починаючи з третього… і так далі до кінця рядку. Такі підрядки називатимемо k -грамами.

Наприклад маємо на вході рядок:

afriendtoallisafriendtonone

Він складається з 27 символів, тобто m = 27. Припустимо будемо розбивати рядок на k-грами довжиною 5 символів (k = 5):

afrie, frien, riend, iendt, endto, ndtoa, dtoal, toall, oalli,
allis, llisa, lisaf, isafr, safri, afrie, frien, riend,
iendt, endto, ndton, dtono, tonon, onone

Тобто маємо наступну кількість k-грамів:

$$n = (m-(k-1)) = (27-(5-1)) = 23. \qquad (2)$$

До кожного з k-грамів застосовуємо певну хеш-функцію, отримуючи в результаті набір ідентифікаційних міток. Для порівнювання таких наборів міток, отриманих для різних програм, використовують алгоритм просіювання. Під час пошуку спільного підрядка за допомогою цього алгоритму, використовуються наступні умови:

Якщо довжина підрядка більша чи рівна гарантованій довжині t, то співпадіння буде виявлено.

Спів падіння, коротші за шумовий поріг k, ігноруються (цей пункт забезпечується виділенням з тексту k-грамів).

Суть алгоритму в тому, що ми просуваємо вікно розміру w вздовж нашої послідовності міток, вибираючи на кожному кроці мінімальне значення, причому значення ширини вікна знаходимо за формулою:

$$w = (t-(k-1)). \qquad (3)$$

Покажемо принцип роботи алгоритму на прикладі. Нехай для даного рядка:

afriendtoallisafriendtonone

побудовано набір ідентифікаційних міток за k-грамами довжини k = 5:

30, 12, 7, 41, 55, 3, 37, 24, 62, 15, 43, 22, 59, 6, 30, 12, 7, 41, 55, 29, 17, 49, 57.

Припустимо нас цікавлять спів падання довжини 7 і більше, тоді:

$$w = (t-(k-1)) = (7-(5-1)) = 3, \qquad (3a)$$

і отримуємо наступну послідовність вікон:

*(30, 12, **7**), (12, 7, 41), (7, 41, 55),*
*(41, 55, **3**), (55, 3, 37), (3, 37, 24),(37, **24**, 62),*
*(24, 62, **15**), (62, 15, 43), (15, 43, 22),*
*(43, **22**, 59), (22, 59, **6**),(59, 6, 30), (6, 30, 12),*
*(30, 12, **7**), (12, 7, 41), (7, 41, 55),*
*(41, 55, **29**),(55, 29, **17**), (29, 17, 49),*
(17, 49, 57),

причому жирним виділено значення, що назначено мітками. Кінцевий набір міток буде наступним:

7, 3, 24, 15, 22, 6, 7, 29, 17.

Далі цей набір порівнюється з аналогічним набором іншого програмного коду для пошуку схожості між ними.

Проведення експерименту
Засоби розробки

Для реалізації поставленої задачі обрано мову програмування Java, оскільки вона досить проста для розуміння, безкоштовна і надає необхідні для більш простої реалізації інструменти (значна бібліотека класів, зручні засоби створення інтерфейсу, незалежність від платформи). Також для розробки використовується середовище програмування NetBeans 6.5, оскільки її розробники найбільш тісно пов'язані з розробниками Java, що робить її дуже зручною і, крім того, вона безкоштовна. Для обробки отриманих даних використовується Microsoft Excel, загальновизнаний стандарт у цій сфері.

Вимоги до апаратного забезпечення

Для тестування використовувались двоядерний процесор Intel Core 2 Duo 2 ГГц, оперативна пам'ять 2 Гб, Unix-подібна операційна система Mac OS X 10.5 Leopard. Але, оскільки програма є платформонезалежною (реалізована на Java), то вибір операційної системи залежить лише від уподобань і можливостей користувача.

Результати експерименту

У процесі розробки і тестуванні алгоритмів на реальних даних, було досягнуто їх стабільної роботи, що дозволило оцінювати їх ефективність і швидкодію.

Було обрано 2 групи файлів: першу – на 82 файли для оцінки швидкодії алгоритмів, другу – на 10 (найбільш цікавих) програмних кодів для оцінки ефективності пошуку плагіату. Причому останні 10 були попарно оцінені людиною на наявність в них плагіату. В результаті отримали наступну кількість пар:

$$\frac{n \cdot n - n}{2} = \frac{10 \cdot 10 - 10}{2} = 45. \qquad (4)$$

Шкала оцінювання лежить на відрізку $[0,1]$, проте людські оцінки знаходяться лише у межах кількох значень на цьому відрізку, оскільки важко досить точно визначити наскільки схожі різні програми. Тому взято 5 можливих оцінок – дві для мало схожих файлів, дві для дуже схожих і одна "середня".

Табл. 2. Шкала оцінювання

0.00	Якщо подібність є, то дуже незначна, щоб приймати до уваги - повністю ігноруємо.
0.10	Це не плагіат, файли зовсім не схожі за виглядом, є певна структурна схожість, але файли досить великі.
0.50	Це не плагіат, але це різні програмні коди однією людини, тому є досить велика схожість.
0.90	Це плагіат, файли дуже схожі. Були змінені коментарі, назви деяких змінних, але все одно це очевидне використання чужого коду.
0.95	Це плагіат, файли практично ідентичні, використані різні вхідні дані.

Оскільки при порівнянні порядок файлів неважливий (порівняння першого файлу з другим, а потім навпаки другого файлу з першим дасть одну й ту саму оцінку) і порівняння двох однакових файлів не потрібно, то отримаємо наступну трикутну матрицю оцінок:

Табл. 3. Попарні оцінки файлів вихідних кодів

Номера файлів, що порівнюються	1	2	3	4	5	6	7	8	9	10
1	–	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00
2	–	–	0.10	0.00	0.90	0.10	0.10	0.10	0.10	0.10
3	–	–	–	0.00	0.10	0.10	0.95	0.10	0.10	0.10
4	–	–	–	–	0.00	0.00	0.00	0.00	0.00	0.00
5	–	–	–	–	–	0.10	0.10	0.10	0.10	0.10
6	–	–	–	–	–	–	0.10	0.10	0.50	0.10
7	–	–	–	–	–	–	–	0.10	0.10	0.10
8	–	–	–	–	–	–	–	–	0.10	0.90
9	–	–	–	–	–	–	–	–	–	0.10
10	–	–	–	–	–	–	–	–	–	–

Потім ту ж саму роботу необхідно було виконати за допомогою реалізованих алгоритмів при різних значеннях вхідних параметрів. Для цього, перш за все, необхідно було отримати токенізовану форму і відбит-

ки кожного з файлів, причому було взято 3 набори відбитків для трьох значень параметра k (див. опис алгоритму) – $k = 3$, $k = 4$, $k = 5$.

Значення взято саме такі, для того щоб забезпечити «відсіювання» надто малих співпадінь (< 3 токенів), але в той же час не втратити у точності роботи алгоритму.

Слід зазначити, що процес побудови відбитків виконується після токенізації, що веде за собою певний програш у часі методу ідентифікаційних міток перед алгоритмом жадібного рядкового замощення. Наведемо для порівняння діаграму середнього витраченого часу (у наносекундах) на обробку 82 файлів (для більшої достовірності результатів для кожного з алгоритмів викинуто максимальне і мінімальне значення з вибірки):

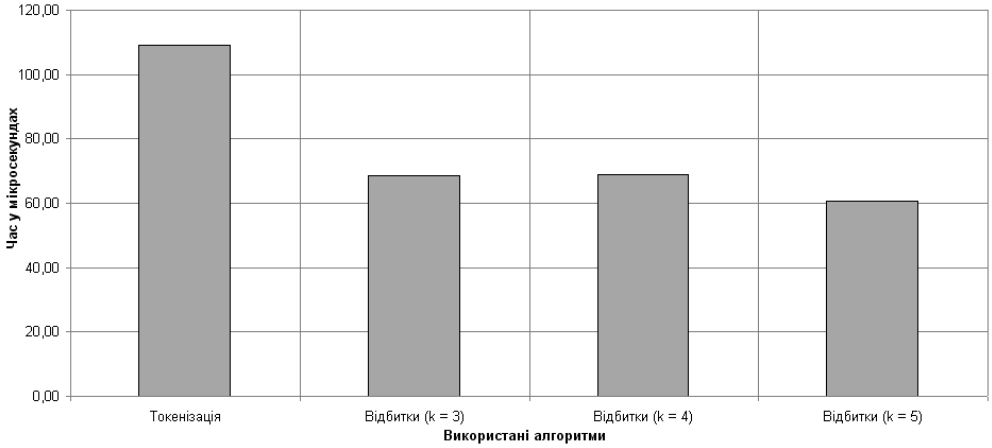


Рис. 1. Порівняння середнього часу обробки програмних кодів алгоритмів

Оскільки цей програш у часі одноразовий і, як видно з діаграми, досить невеликий, то він не є суттєвим для оцінки швидкодії цих алгоритмів і надалі ми не будемо до цього повертатись.

Далі оброблені 10 програмних кодів були порівняні між собою використовуючи реалізовані алгоритми, в результаті чого отримали досить значну вибірку даних.

Спочатку наведемо графік порівняння людських оцінок і оцінок базового жадібного алгоритму:

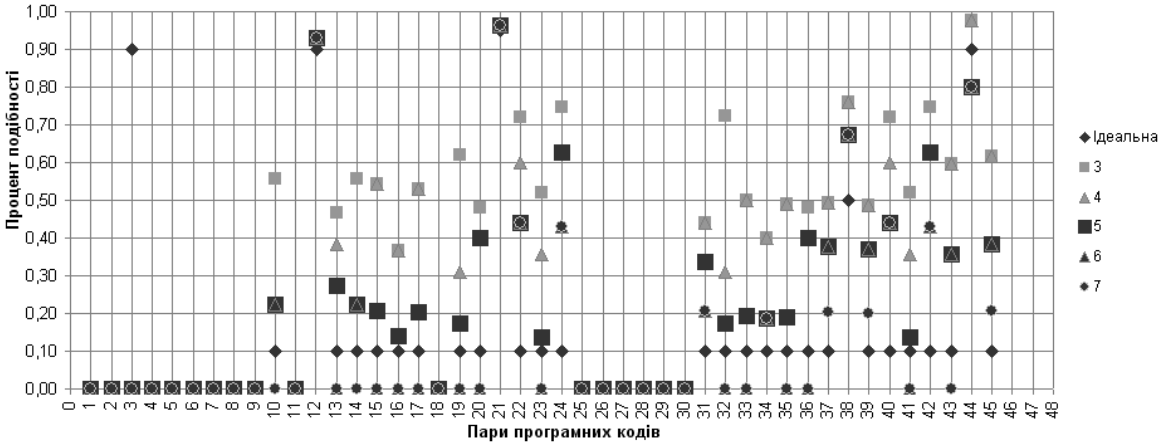


Рис. 2. Порівняння роботи базового Жадібного алгоритму при різних мінімальних довжинах найбільшого префіксу

Людські оцінки зазначено як «ідеальні», а всі інші (при різних значення мінімальної довжини максимального покриття – цими значеннями і помічені графіки у легенді). Перший «пік» ідеальної оцінки не помітила жодна з реалізацій (навіть з подальших) – це пояснюється дуже малою величиною кожного з цієї пари програмних кодів (токенізоване представлення складало 2 і 3 символи відповідно), тому ці співпадіння відсіювались як

випадкові і не приймались до уваги взагалі. Всі інші випадки плагіату були добре помічені усіма реалізаціями, навіть той випадок коли два файли вихідних кодів різних програм писала одна й та сама людина, внаслідок чого у коді присутній деякий свій виразний стиль, який програма розпізнала плагіат. Але, оскільки плагіат в межах однією людини таким не є, то з цією неточністю варто боротись не шляхом зміни алгоритму, а шляхом перевірки хто є номінальним автором програмного коду іншими шляхами.

З графіку очевидно, що найближчим до ідеального є графік 5, оскільки він близький до нього не тільки у випадках явного плагіату, але й на усій вибірці також. Це означає, що при токенізації для отримання вірогідних результатів варто вибирати саме таку мінімальну довжину максимального покриття.

Тепер поглянемо наскільки впливає на результат оцінювання введення коефіцієнту помилки (розглянемо для обраної мінімальної довжини максимального покриття рівній 5):

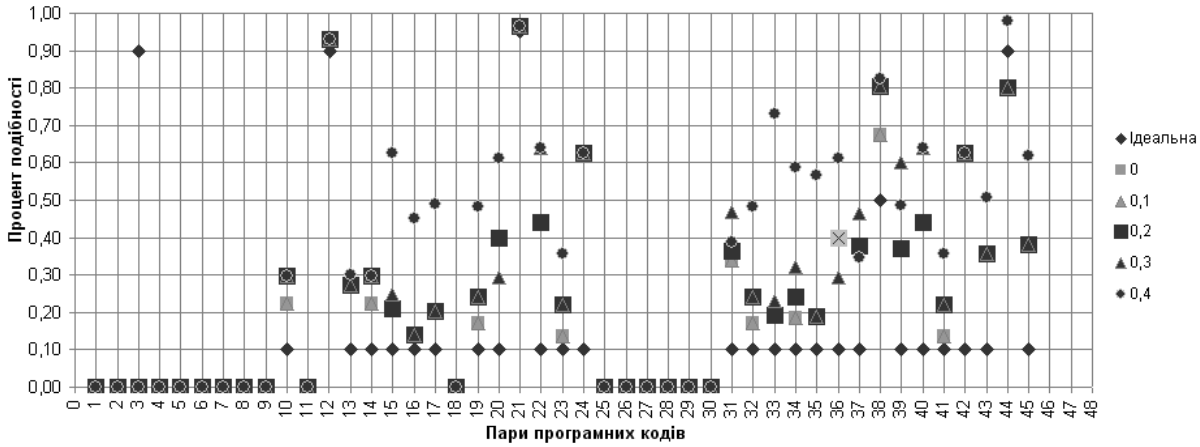


Рис. 3. Порівняння роботи Жадібного алгоритму при мінімальній довжині найбільшого префіксу = 5 і різних відносних похибках

Порівнюючи побудовані графіки з ідеальним, можна сказати що в межах від 0 до 0,3 цей коефіцієнт слабко впливає на результат, а при більшому вже з'являються «паразитні» випадки плагіату, що є негативним ефектом для оцінювання.

Отже з усіх перевірених варіантів застосування жадібного алгоритму слід зупинитися на його базовій реалізації з мінімальною довжиною максимального покриття рівній 5.

Тепер перевіримо ефективність методу ідентифікаційних міток. Побудуємо три графіки, розділивши всі варіанти використання на три групи за коефіцієнтом k:

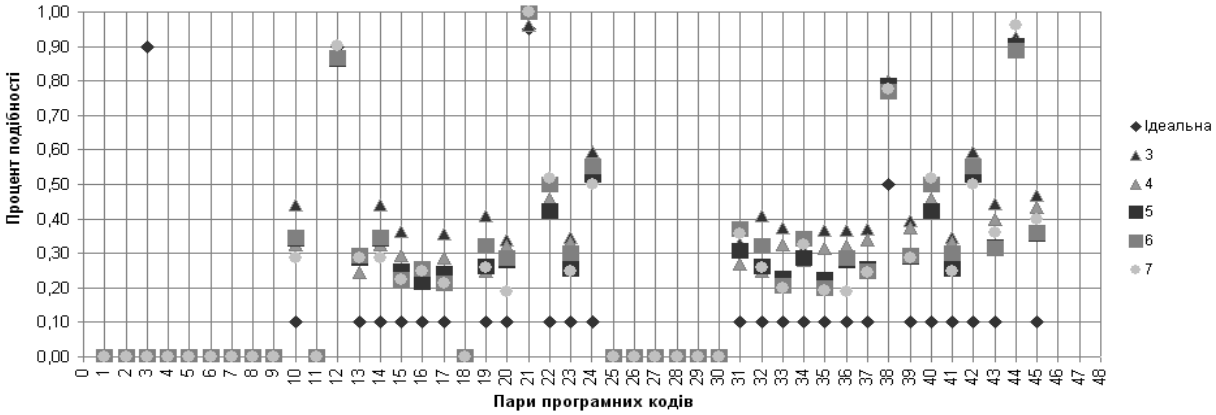


Рис. 4. Порівняння роботи методу відбитків при сталому значенні довжини одного k-граму (k = 3) і різних значеннях ширини вікна

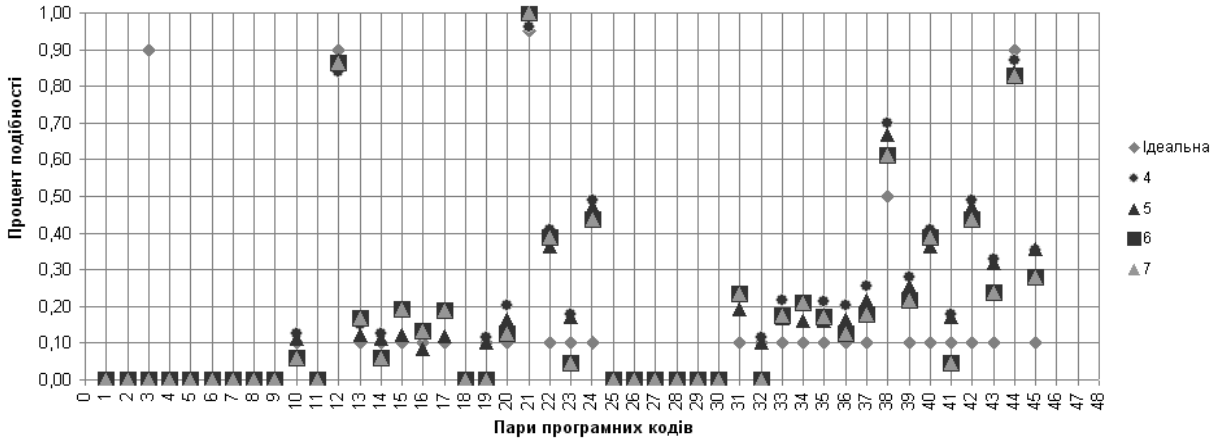


Рис. 5. Порівняння роботи методу відбитків при сталому значенні довжини одного k-граму (k = 4) і різних значеннях ширини вікна

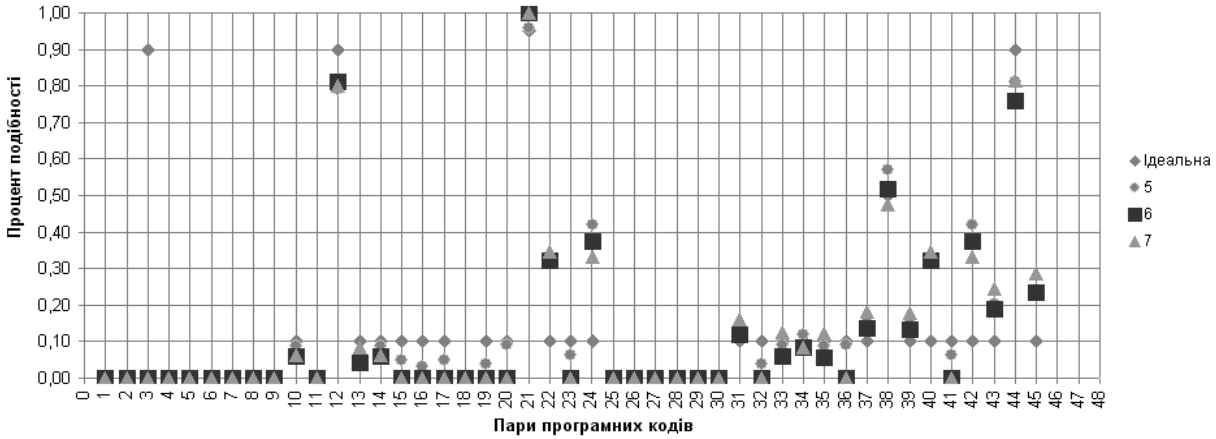


Рис. 6. Порівняння роботи методу відбитків при сталому значенні довжини одного k-граму (k = 5) і різних значеннях ширини вікна

Можна побачити, що для значення $k = 3$ оцінки імовірності плагіату між піками дуже великі, що призводить до помилки другого роду (знаходимо плагіат там, де його насправді немає), а для значення $k = 5$ навпаки маємо дуже малу ймовірність плагіату, що може призвести до помилки першого роду (плагіат не буде знайдено, хоча насправді він має місце). Тому досить розумним буде обрати середній варіант, особливо для значень коефіцієнтів $k = 4$, $t = 5$, оскільки вони найбільш близькі до ідеального у піках. Проте цей вибір ще не є остаточним.

Для порівняння точності роботи алгоритмів (відносно людської оцінки подібності програмних кодів) використаємо метод середньоквадратичної похибки:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (sa_i - sh_i)^2}{n}}, \qquad (5)$$

де *sa* – коефіцієнт подібності, знайдений поточним алгоритмом (з поточними параметрами); *sh* – заданий людиною коефіцієнт подібності (ідеальний); *n* - кількість пар програмних кодів, що порівнюються.

Знайдемо і порівняємо середньоквадратичні похибки для всіх наведених алгоритмів:

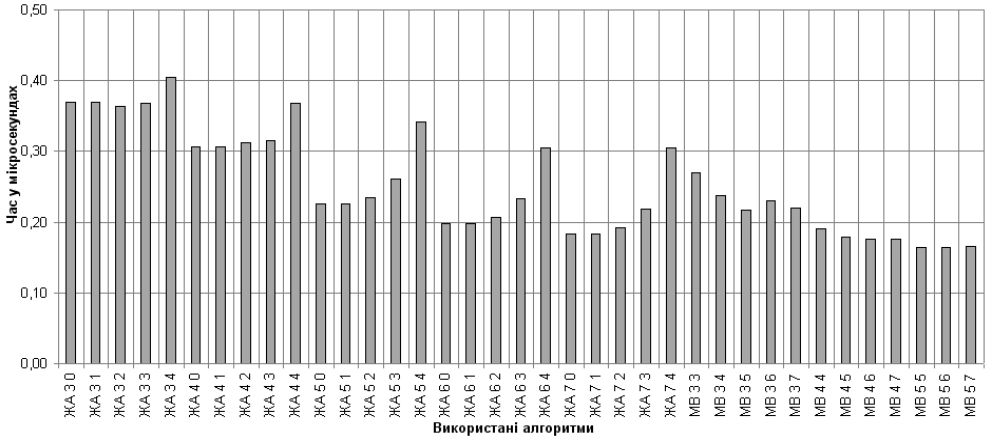


Рис. 7. Порівняння середньоквадратичної похибки алгоритмів

Тепер порівняємо час роботи алгоритмів для різних параметрів на виборці у 82 файли програмних коди. З графіку очевидно, що при будь-яких параметрах метод відбитків працює значно швидше (приблизно вдвічі) ніж алгоритм жадібного рядкового замощення.

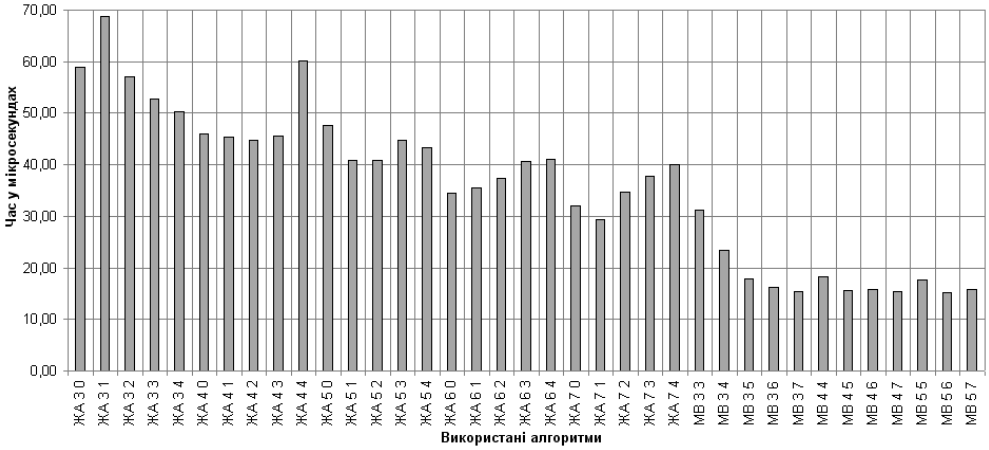


Рис. 8. Порівняння середнього часу роботи алгоритмів

Висновки

У цій роботі ми порівняли два алгоритми попарного порівняння програмних кодів програм для пошуку плагіату в них – це алгоритм жадібного рядкового замощення і метод відбитків (ідентифікаційних міток).

Алгоритм жадібного рядкового замощення у якості вхідних данних використовує токенізоване представлення програмного коду, а метод відбитків використовує набір числових значень (відбитків), що будується з токенізованого представлення. Очевидно, що на цьому етапі другий метод має певний програш у часі, але він є незначним, оскільки отримані один раз мітки можна зберігати для повторного використання при наступних порівняннях.

Порівняння алгоритмів проводилося при різних значеннях основних параметрів, тому важливо було не тільки обрати оптимальний алгоритм, але й оптимальні значення цих параметрів. Для дослідження було використано попарно оцінений людиною набір програмних кодів і всі результати роботи алгоритмів

порівнювалися перш за все з оцінками людини.

Дослідивши час роботи і ефективність реалізованих алгоритмів, ми зробили висновок, що метод ідентифікаційних міток є більш перспективним для використання у потужній системі ніж алгоритм жадібного рядкового замощення, оскільки він працює майже у два рази швидше, не поступаючись при цьому жадібному алгоритму (а часто і випереджаючи його) за якістю знаходження плагіату у програмних кодах. Серед можливих значень параметрів для цього алгоритму варто виділити алгоритм з параметрами k = 4, або k = 5 та t = 5, оскільки при цих значеннях спостерігається найкраще співвідношення швидкодії та ефективності. Але якщо взяти k і t рівні, то це дозволить не використовувати алгоритм просіювання, а одразу шукати множину спільних міток, тим самим трохи скоротивши процес порівняння кожної з пар кодів.

Результати цих досліджень доводять значну ефективність пошуку плагіату цими алгоритмами, показуючи при цьому очевидну перевагу у швидкодії одного алгоритму над іншим. Це дає змогу використовувати у системі пошуку плагіату найбільш оптимальний алгоритм, що критично для великої бази вихідних кодів, яка лежить в основі такої системи.

Список літератури

1. Огляд автоматичних детекторів плагіату в програмах, 2006 http://detector.spb.su/pub/Sandbox/ReviewAlgorithms/survey.pdf

2. Michael J. Wise. String similarity via greedy string tiling and running Karp-Rabin matching. Dept. of CS, University of Sydney, ftp://ftp.cs.su.oz.au/michaelw/doc/RKR.

GST.ps, December 1993

3. Aiken A., Schleimer S., Wikerson D., Winnowing: local algorithms for document fingerprinting. In *Proc 2003 ACMSIGMOD Int. Conf. on Management of Data,* San Diego, CA, June 9-12, pp. 76-85. ACM Press, New York, USA, 2003.

Поступила в редакцию 2.12.2009

УДК 004.415.2.045 (076.5)

ДИШЛЕВИЙ О.П.

ЗАСТОСУВАННЯ ПРЕДМЕТНО-ОРІЄНТОВАНОГО МЕТОДУ ПОБУДОВИ ЗАЛЕЖНОСТЕЙ МІЖ МЕТРИКАМИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Стаття присвячена описанню застосування предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення для визначення залежностей між прямими та непрямими метриками. Розглянута побудова залежності непрямої метрики «розуміння», яка відповідає аналогічній властивості програмного забезпечення, від прямих метрик. Проводиться розрахунок ступеня впливу прямої метрики на метрику «розуміння», будується функція залежностей між найбільш впливовою прямою метрикою та непрямою метрикою.

The article is dedicated to describing the application of subject-oriented method of building dependences between software metrics for determine the dependencies between direct and indirect metrics. It was considered building depends indirect metrics "understanding", which corresponds to similar property of software from a number of direct metrics. Level of impact on the metric "understanding" is calculated for each direct metric, there is determined the most influential metric and there is built functions of dependencies between the most powerful direct and indirect metric.

Вступ

Все більшої актуальності набуваються питання дослідження програмного забезпечення. Вирішенням цих питань займається емпірична інженерія програмного забезпечення [1].

Ключовим методом дослідження програмного забезпечення є вимірювання. Головний інструмент, який при цьому застосовується,– метрики. За допомогою метрик оцінюють властивості складових розробки програмного забезпечення (продуктів, процесів). Властивості програмного забезпечення дають його розуміння для аналізу та подальшого використання.

Існують прямі та непрямі метрики [2]. Прямі метрики можна отримати при вимірюванні. На основі прямих метрик важко проводити оцінку властивостей програмного забезпечення, так як вони ніяк не характеризують ці властивості. Для цього використовують непрямі метрики. Непрямі метрики дозволяють отимати числові характеристики программного забезпечення, які допомагають в розумінні программного забезпечення. Їх не можна виміряти, але можна отримати аналітичним чи емпіричним шляхом на основі прямих метрик. Головна задача, яка при цьому постає,– як визначати прямі метрики, від яких залежать непрямі, та яка залежність буде існувати між ними [3].

Огляд останніх досліджень

Для побудови залежностей використовують два підходи – статистичний аналіз [4] та нейронні мережі [5]. Вирішення задачі по визначенню залежностей між метриками за допомогою нейромереж потребує знати не тільки значення метрик (вхідних величин), а повинне бути проведене навчання нейромережі [5]. Навчання нейромережі проводиться на уже отриманих раніше даних. Так як питаннями визначення залежностей між метриками займалися тільки в реальному контексті (досліджувалися деякі конкретні метрики), а визначенням залежностей між різними прямим та непрямими метриками вцілому не займалися, сформулювати правила для навчання нейромережі неможливо. Для вирішення задачі по визначенню залежностей між метриками за допомогою статистичного аналізу достатньо знати тільки значення метрик, а залежності будуються статистичними методами [4]. Тому в даному випадку необхідно застосовувати статистичний аналіз.

Статистичний аналіз проводиться за допомогою відповідних математичних програмних середовищ, до яких належать MatLab, MathCad, Maple, Mathematica, MS Excel. Крім них можна використати статистичні програмні середовища загального призначення Statistica, SPSS, SAS, Systat, Minitab, Statgraphics чи програмні середовища спеціального призначення SYSTAT, S-plus, STATA, PRISM, STADIA, Олимп, Класс-Мастер,

Статистик-Консультант. Досвід їх використання свідчить, що математичні програмні середовища та статистичні програмні середовища загального призначення для вирішення поставленої задачі потребують додаткового програмування з використанням статистичних алгоритмів. Середовища для емпіричних досліджень в програмному забезпеченні немає.

Отже, для побудови залежностей між прямими та непрямими метриками програмного забезпечення потрібне середовище та метод, який буде використовуватися в середовищі.

Для вирішення головної задачі був розроблений предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення. На основі методу був реалізований засіб [6].

У статті розглядається застосування предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення на прикладі побудови залежності між непрямою метрикою «розуміння» та рядом прямих метрик.

Ціль статті

Основною ціллю статті є дослідження практичного застосування розробленого методу побудови залежностей між метриками через побудову залежності непрямої метрики «розуміння» від прямих метрик та визначення прямих метрик, від яких залежить дана непряма метрика.

Застосування предметно-орієнтованого методу

Сутність предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення полягає в тому, що побудова залежностей відбувається статистичними методами з врахуванням високої точності вимірювань програмного забезпечення без похибок та гіперболічною спадною залежністю між значеннями метрик та кількістю виміряних програм.

Загальна схема методу представлена на рис. 1.

На основі данного методу будуються залежності між непрямою метрикою «розуміння» та рядом прямих метрик. Метою дослідження є визначення прямих метрик, від яких залежить досліджувана непряма метрика, та яким чином.

Непряма метрика «розуміння» відповідає аналогічній властивості програмного забезпечення. Вона показує зрозумілість компонента програмного забезпечення, яка достатня для його аналізу та використання при розробці нового програмного забезпечення. Вона характеризує зусилля, які потрібно затратити для розуміння логічної структури компонента. В процесах проектування та повторної використання цю метрику слід застосовувати при визначення доцільності аналізу компонента програмного забезпечення. При малому значенні розуміння доцільніше створити новий компонент, ніж аналізувати та застосовувати готовий. Максимально допустиме значення метрики «розуміння» визначається для кожного компонента індивідуально згідно потреб.

Використання непрямої метрики «розуміння» дозволяє спростити аналіз готового програмного забезпечення та прийняття рішень про можливості його подальшого застосування.

Отримання метрик для дослідження. Прямі метрики для досліджень були отримані шляхом вимірювань програмного забезпечення. Вимірювання прямих метрик програмного забезпечення проходило під час експерименту по визначенню законів розподілів метрик [6]. Було виміряно 50 прямих метрик.

Непряма метрика «розуміння» є деякою функцією від однієї чи декількох прямих метрик. Оскільки для непрямої метрики «розуміння» поки не визначено функції, на основі якої можна було б розрахувати метрику, вона була отримана шляхом експертного оцінювання програмного забезпечення [7]. Оцінку експерт міг вибритати із запропонованих: «погано», «задовільно», «добре», «відмінно». Експертне оцінювання проходило тільки для того програмного забезпечення, яке було до цього виміряне.

Об’єм вибірки. Важливим критерієм при отриманні метрик виступає їх кількість. Необхідний об’єм метрик напряму впливає на результати побудови залежностей. Мінімальна кількість метрик, які необхідні для побудови залежностей між метриками, була визначена експериментальним шляхом.

Для цього було проведене дослідження статистичних характеристик ряду метрик при збільшенні вибірки. Дослідження прово-

дилося в зворотному порядку. З усього об'єму були розраховані статистичні характеристики для декількох вибірок. Ці вибірки були отримані з загальної кількості метрик шляхом зменшення вибірки на 10 %. Наприклад, для метрики WMC було отримано 5000 значень. Для цієї вибірки були розраховані статистичні характеристики. Далі вибірку було зменшено на 10 %, і статистичні характеристики були пораховані для 4500 значень метрики. Далі вибірку було зменшено ще на 10 %, і статистичні характеристики були пораховані для цієї вибірки. Таким чином вибірка зменшувалася до половини початкової вибірки.

Виявилося, що для визначення кількості значень метрик достатньо провести декілька контрольних розрахунків статистичних характеристик, щоб виявити, чи потрібно збільшувати вибірку. Крім цього, було визначено, що найсуттєвішим показником є медіана. Це пов'язано з відсутністю нормального розподілу серед метрик. А також стало зрозуміло, що більше 4-х вибірок брати із запропонованої недоцільно, так як вони в повній мірі відображають загальні закономірності.

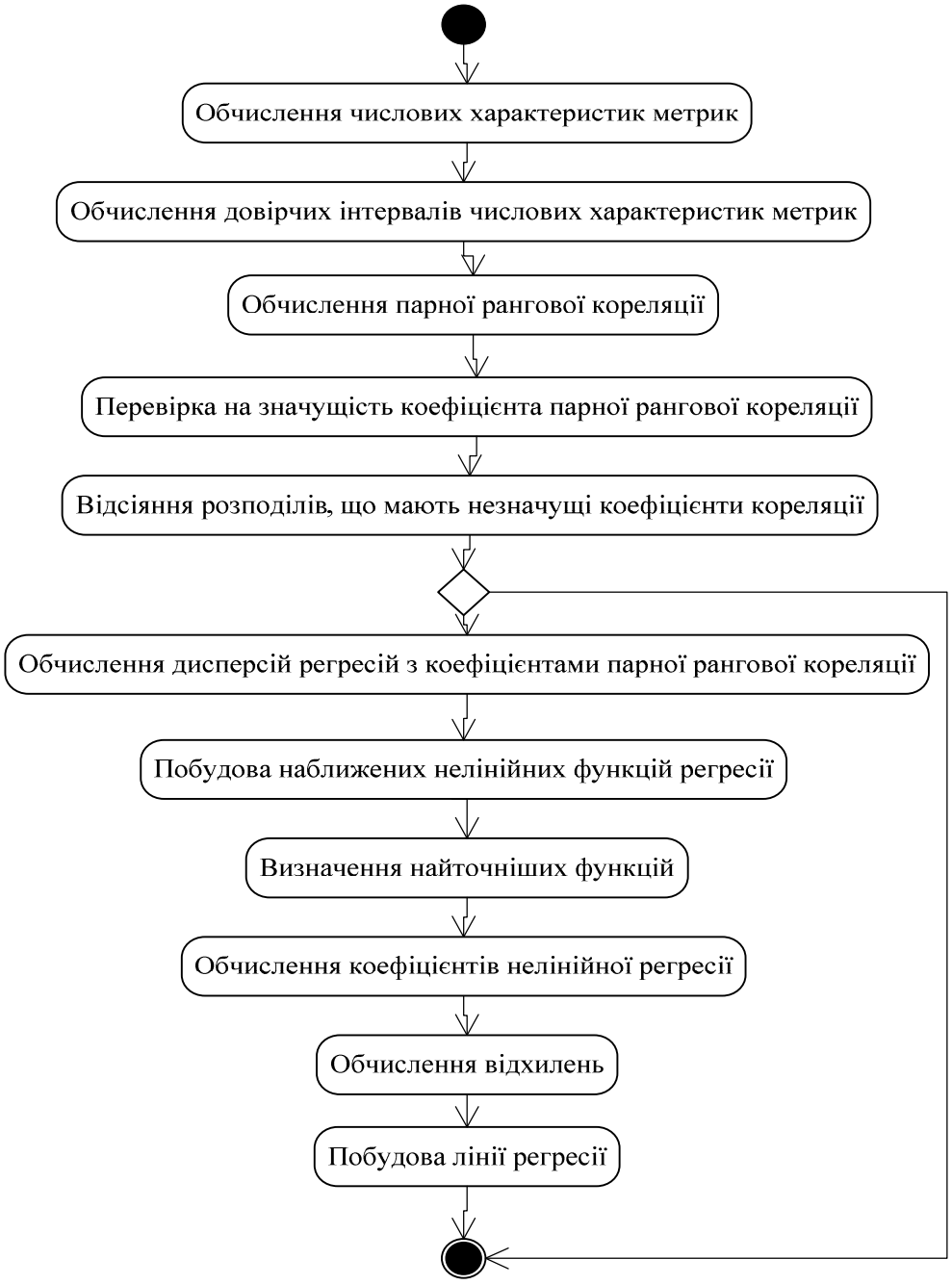


Рис. 1. Предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення

При цьому ключовим показником тут є точність. Для різної заданої точності вибірка

може змінюватися. Тому цей параметр повинен задавати сам дослідник.

Якщо співвідношення між сусідніми замірами не менше, ніж задана точність, то дану вибірку можна вважати повною для побудови залежностей. Тобто збільшувати кількість вимірів непотрібно. Описане співвідношення має вид:

$$\gamma \le \frac{\tilde{x}_1}{\tilde{x}_2} \approx \frac{\tilde{x}_2}{\tilde{x}_3} \approx \frac{\tilde{x}_3}{\tilde{x}_4} \quad (1)$$

де γ – задана точність, x_i- медіани кожної з вибірок.

Відповідно, якщо медіани відрізняються для кожної з вибірок на задану точність, виміряний об'єм метрик приймається для подальших розрахунків. Якщо ні – пропонується збільшити об'єм вибірки.

Підтвердженням отриманих результатів слугують статистичні характеристики деяких виміряних й досліджених метрик [8].

Обчислення числових характеристик метрик. Відповідно до розробленого методу, спочатку були отримані числові характеристики прямих метрик та непрямої «розуміння» (RA): математичне сподівання, середнє квадратичне відхилення, коефіцієнти ексцесу та асиметрії. Деякі значення приведені в табл.1.

Табл. 1. Початковий та центральний моменти метрик

Познач. метрики	Мат. сподівання	Середнє квадрат. відхилення	Коеф. Асиметрії	Коеф. ексцесу	Пояснення метрики
AVRPF	58.8417	17.9308	-0.429	2.4917	Середнє значення виключень в модулі
TM	2.5140	2.4356	1.7780	6.0558	Загальна кількість функцій (модулів компонента)
TLOC	170.9636	142.7548	1.2357	3.8945	Загальне число непорожніх рядків, включаючи коментарі та рядки заголовків
TNCSS	121.5091	111.8934	1.3023	4.0540	Загальне число вихідних конструкцій. Не включає порожні рядки, рядки коментарю, чи заготовочні рядки
TVG	16.1000	19.7032	2.1608	7.5764	Загальна цикломатична складність у всіх модулях (кількість ключових слів розгалужень в кожному модулі).
TMV	2004.0275	2450.4168	2.0724	7.2958	Загальний об'єм по Хольстеду, обчислений для модулів [9]
AHV	668.0450	577.4597	0.8903	2.9307	Середній об'єм по Хольстеду, обчислений для модулів [9]
THE	53018.407	72147.877	2.2935	9.3088	Загальні зусилля по Хольстеду [9]
MHE	34660.263	43164.184	0.8843	6.7029	Максимальне значення зусиль по Хольстеду [9]
AVGHE	19865.306	25139.999	1.9345	6.6292	Середнє значення зусиль по Хольстеду [9]
MFO	8.9912	9.3785	1.5897	5.7260	Максимальне число викликів інших функцій в модулі
MFI	1.0000	1.7008	2.4493	10.2992	Максимальне число викликів модуля
AVGFI	0.3838	0.5715	1.5512	5.0540	Загальне число викликів інших функцій в модулях
'RA'	2.9279	1.3823	-0.2189	0.6029	«розуміння»

Обчислення кореляції та відсіяння незалежних метрик. Обчислення парної рангової кореляції відбувається шляхом розрахунків коефіцієнтів Спірмена та Кендала [9]. Деякі розраховані коефіцієнт приведені в табл. 2. Крім цього, в таблиці подана значущість отриманих коефіцієнтів, яка говорить про наявність чи відсутність залежності.

Як видно з табл. 2, непряма метрика «розуміння» залежить від більшості прямих метрик, про що свідчить значущість. З загальної кількості 50 прямих метрик було відсіяно 5. Для інших 45 метрик будувалися регерсії.

Побудова регресій. Побудова регресії проводилася для кожної прямої метрики окремо. Далі буде приведений приклад побудови регресії для непрямої метрики «розуміння» та прямої метрики «кількість непорожніх рядків» (TLOC).

Табл. 2. Коефіцієнти кореляції для непрямої метрики «розуміння» та прямих метрик

Назва метрики	Коефіцієнт кореляції Кендала	Коефіцієнт кореляції Спірмена	Значущість
AVRPF	0.0912	0.1368	+
TM	0.3986	0.5979	+
TLOC	0.0693	0.1040	+
TNCSS	0.0263	0.0395	-
TVG	-0.2482	-0.3724	+
TMV	0.1616	0.2424	+
AHV	0.0926	0.1389	+
THE	-0.1294	-0.1940	+
MHE	0.1991	0.2987	+
AVGHE	0.0992	0.1489	+
MFO	0.0161	0.0241	-
MFI	0.1941	0.2912	+
AVGFI	0.1327	0.1991	+

Спочатку проводиться розрахунок дисперсій. Дисперсії представлені у графічному

виді на рис. 2. Як видно з нього, значних відхилень по кожному з вимірювань немає, тому наступним етапом буде побудова регресії. Перед побудовою регресії було визначене кореляційне поле для метрик (рис. 3). Кореляційне поле має складну конфігурацію, що свідчить про нелінійну функціональну залежність між метриками.

Далі, як сказано в [10], проводилася побудова декількох наближених регресій, розраховувалися їх коефіцієнти та вибиралася найближча.

Побудовані типові функції регресії та їх відхилення приведені в табл. 3. У функціях: x – математичне сподівання, умовні математичні сподівання експертних оцінок, a, b, c – коефіцієнти.

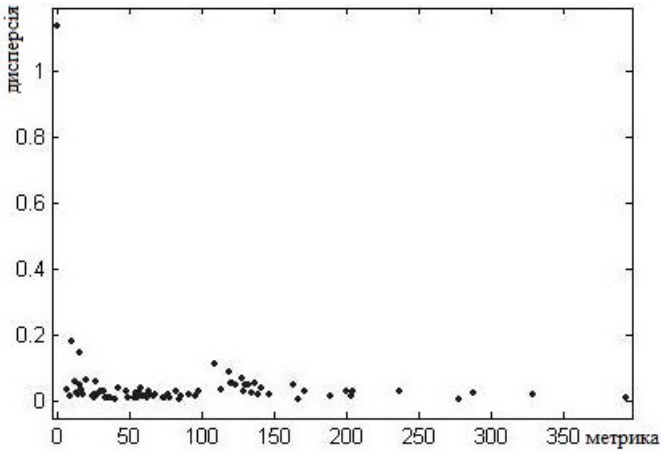


Рис. 2. Значення дисперсій

Табл. 3. Коефіцієнти та відхилення функцій регресії

№ п/п	Функція	Коефіцієнт *a*	Коефіцієнт *b*	Коефіцієнт *c*	Відхилення
1	$y = ax+b$	0.001161	2.83	0	0.2917
2	$y=1/(ax+b)$	-0.0001219	0.352	0	0.2913
3	$y=x/(ax+b)$	0.329	0.4844	0	0.3142
4	$y = a\ln x+b$	1.252	1.127	0	5.7184
5	$y=b\exp(ax)$	0.0003761	2.836	0	0.2914
6	$y = ax^3+b$	0.996e-009	2.918	0	0.2912
7	$y=c+b\exp(ax)$	-11.4	-0.07609	2.938	0.2909
8	$y=ax^2+bx+c$	-6.025e-006	0.002988	2.75	0.3016
9	$y=1/(ax^2+bx+c)$	-0.006006	0.5697	0.376	2.8139
10	$y=x/(ax^2+bx+c)$	-5.971e-005	0.3377	0.3557	0.3210
11	$y=a/(1/x+b)$	2.064	0.6792	0	0.3142
12	$y=1/(a\exp(-x)+b)$	0.009149	0.3403	0	0.2906

За основу для вибору оптимальної функції слугує відхилення.

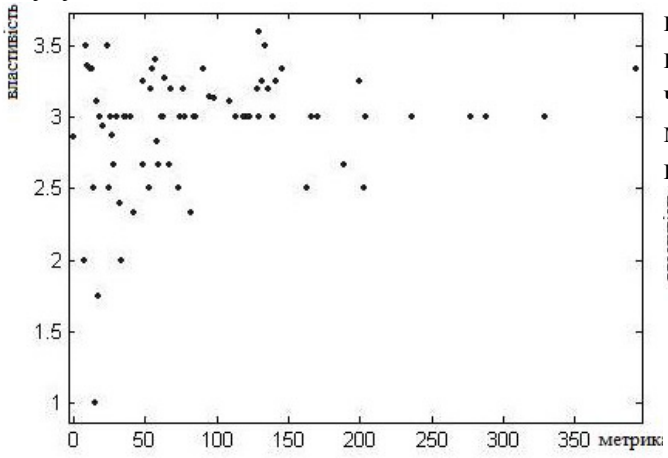


Рис. 3. Кореляційне поле метрик

Проаналізувавши значення відхилень можна зробити висновок, що найменше – 0.2906 для функції $y=1/(a\exp(-x)+b)$ (рис. 4). Тобто це функція, яка найточніше виражає залежність між метрикою «кількість непорожніх рядків» та «розуміння».

Висновки

В результаті проведеної роботи була виведена закономірність при визначенні об'єму вибірки метрики. Виявилося, що дану величину можна розрахувати на основі перевірки математичного сподівання декількох менших вибірок.

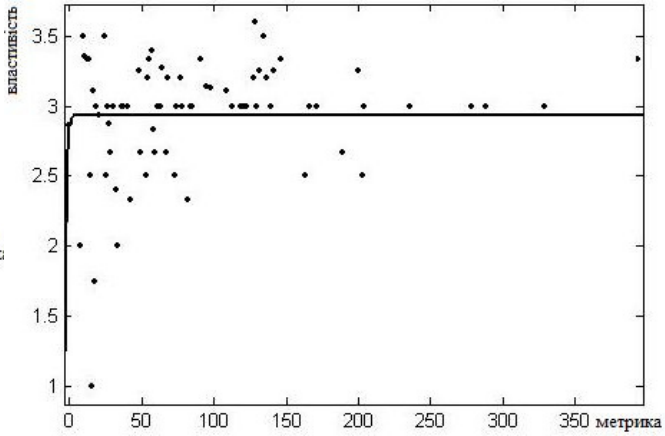


Рис. 4. Функція регресії між метриками

Було визначено, що більшість метрик не мають нормального закону розподілу. Це пов'язано з особливостями даних програм-

ного забезпечення, так як існує великий об'єм програмного коду з невеликими значеннями метрик. Підтвердження цьому є у [8].

Проведені дослідження свідчать про те, що на непряму метрику «розуміння» впливає багато прямих метрик. Серед усіх прямих метрик була вибрана одна, для якої й визначалася функція залежності. Як виявилося, залежності між прямими метриками та непрямою метрикою «розуміння» мають нелінійних характер, про що свідчать коефіцієнти кореляції та побудована регресія між прямою метрикою «кількість непорожніх рядків» та непрямою «розуміння». Для виведення залежності непрямої метрики «розуміння» від прямих метрик, останні отримуються шляхом вимірювання програмного забезпечення, а непряма метрика «розуміння» отримується шляхом експертного оцінювання. В подальшому значення непрямої метрики пропонується отримувати шляхом розрахунків на основі отриманої функції залежності від прямих метрик. Результати досліджень для ряду об'єктно-орієнтованих метрик підтверджуються в [8].

Предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення, приклад застосування якого приведений у даній статті, дозволяє будувати залежності між тими прямими та непрямими метриками, для яких цих залежностей ще не було виведено. В першу чергу це стосується тих непрямих метрик, які безпосередньо характеризують властивості програмного забезпечення та прямими метриками. Це дасть можливість аналізувати властивості програмного забезпечення на основі показників непрямих метрик та приймати управлінські рішення по проекту. Характеризувати властивості програмного забезпечення досить просто при наявності функціональних зв'язків із прямими метриками. Виведені залежності допомагають пов'язати прямі метрики, які можна виміряти, із властивостями програмного забезпечення, які використовуються при прийнятті рішень.

Подальша робота буде направлена на визначення закономірностей між ключовими прямими та непрямими метриками.

Список літератури

1. Koji Torii, Ken-ichi Matsumoto, Kumiyo Nakakoji, Yoshiro Takada, Kaduyuki Shima, Ginger 2: An Enviroment for Computer-Aided Empirical Software Engineering // IEEE Transactions on Software Engineering, vol 25. No 4, July/August 1999, P 475-486.
2. Norman E. Fenton, Shari Lawrence Pfleeger Software Metrics: A Rigorous and Practical Approach.- Cambridge University Press,1996.-638p.
3. Forrest Shull, Janice Singer, Dag I.K. Sjoberg Guide to Advanced Empirical Software Engineering. – Springer-Verlag London Limited 2008.-394p.
4. Вентцель Е.С. Теория вероятностей: Учеб. для вузов. – 7-е изд. стер. – М.: Высш. шк., 2001. – 575 с.: ил.
5. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. - Пер. с англ. – М.: Мир, 1992. - 118 с.
6. Дишлевий О.П. Пакет статистичного аналізу для емпіричної інженерії програмного забезпечення / О.П. Дишлевий // Наука і молодь. Прикладна серія. Збірник наукових праць. – 2009. – № 9. – С. 104–108.
7. Хоменко В.А., Дышлевый А.П. Метод экспертного оценивания свойств повторно используемых компонентов программного обеспечения // Том 1: Матеріали VIII Міжнародної науково-технічної конференції «Авіа-2007».- Т.1.- К.:НАУ, 2007, с. 13.169-13.172.
8. Michele Lanza, Radu Marinescu. Obect-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems. – Springer- Verlag Berlin Limited 2006.-205p.
9. Software Measurement and Estimation: a practical approach / Linda M. Laird, M. Carol Brennan. John Wiley & Sons, Inc., Hoboken, New Jersey 2006.- 257 p.
10. Бабак В.П., Білецький А.Я., Приставка О.П., Приставка П.О. Статистична обробка даних/ Монографія. – Київ: «МІВВЦ», 2001. – 388 с.

Поступила в редакцию 15.12.2009

УДК 681.322

СЕРГІЄНКО А.М.,
ЛЕПЕХА В. Л.

ДЕЯКІ ОСОБЛИВОСТІ ПРОЕКТУВАННЯ МІКРОКОНТРОЛЕРІВ ДЛЯ СНК

Рассмотрены особенности проектирования ядра микроконтроллера для системы на кристалле (СНК). Показан пример разработки ядра микроконтроллера с архитектурой i8051 с высокими характеристиками при реализации СНК в программируемой логической интегральной схеме (ПЛИС).

Details of the microcontoller IP core design for the modern SOC are considered. An example of the i8051 microcontroller IP core design is shown which has good characteristics by configuring in FPGA-based SOC.

Вступ

Одним з основних принципів розробки систем на кристалі (СНК) є застосування готових бібліотечних обчислювальних модулів (IP cores), які повинні бути надійно повторювані в ряді проектів СНК. Технологія використання готових модулів поширилась з кінця дев′яностих років, завдяки чому зменшились трудовитрати і строки проектування СНК. Але їхнє впровадження при ускладненні технології СНК стає рідшим, що пов′язано з їхньою втратою властивості повторюваності [1].

Мікроконтролер i8051 набув поширення у формі повторюваного мікропроцесорного ядра. Ядра клонів архітектури i8051 виконують команди за 1–6 тактів, тобто на порядок швидше, ніж стандартний мікроконтролер i8051. За даними сайту www.design-reuse.com, зараз більше трьох десятків фірм пропонують ці модулі споживачам. Причому більшість з них, як рекламний хід, декларують, що мікроконтролер виконує 1 команду за такт.

В статті розглядаються можливості збільшення повторюваності мікропроцесорних ядер на прикладі ядра з архітектурою i8051.

Деякі особливості технології СНК

При впровадженні глибокої субмікронної технології (DSM) стали діяти нові фізичні явища, які перешкоджають використанню готових модулів. Так, при нормах 65 нм відношення середньої затримки в лініях зв′язку і затримки вентилів складає 9:1. Кожні чотири роки сумарна довжина ліній зв′язку кристалу подвоюється і зараз дорівнює близько 2000 м/см^2. В лініях зв′язку розсіюється все більша частка споживаної потужності.

Як висновок, при DSM-технології швидкодія, енергоспоживання, площа кристалу в більшому ступені залежать від системи ліній зв′язку, ніж від множини вентилів СНК. Тому оптимізація ліній зв′язку повинна виконуватись на початкових стадіях проектування СНК [1]. Те саме стосується розробки IP cores.

Виконання СНК на базі програмованих логічних інтегральних схем (ПЛІС) дає змогу розробляти складні СНК у стислі строки і з мінімальними витратами. Але в ПЛІС одержуються довгі лінії зв′язку, затримка в яких складає 20–90% загальної затримки критичного шляху зконфігурованого модуля. Тому проблема оптимізації ліній зв′язку стосується СНК на ПЛІС ще більше, ніж СНК на ЗВІС [2].

Архітектура ядра i8051

В мікроконтролері i8051 команди зберігаються в пам′яті команд і вибираються з неї за послідовними адресами за винятком команд переходу і переривання. Тобто він має нейманівську архітектуру. Уточненням архітектури є те, що пам′ять команд відокремлена від пам′яті даних і виконана як постійний запам′ятовуючий пристрій (ПЗП).

Нейманівська архітектура передбачає атомарне виконання команд, тобто виконання наступної команди можна почати лише після закінчення виконання попередньої команди. Те саме стосується архітектури i8051. Більш того, коректне переривання в цій архітектурі неможливе без строгого виконання цієї вимоги. Також всі компілятори для цієї архітектури розраховані на атомарність виконання команд.

Особливості ядер з архітектурою i8051

Критичний шлях в СНК представляє собою маршрут з найбільшою затримкою, який проходить сигнал, починаючи з виходу одного з тригерів і кінчаючи входом іншого тригера. Довжина критичного шляху, вимірювана в наносекундах, визначає мінімальний період тактової частоти F_{max} СНК , тобто її швидкодію.

Більшість ядер i8051, що пропонуються фірмами, мають частину команд, які виконуються за 1 такт. Це означає, що критичний шлях починається і кінчається в лічильнику команд, і проходить через ПЗП програм (ПЗПП), дешифратор команд, а також арифметико-логічний пристрій (АЛП). Альтернативний критичний шлях починається і кінчається в пам′яті даних і проходить через АЛП.

Вимога виконання команди за 1 такт призводить до ряду суворих обмежень на реалізацію ядра i8051. По-перше, програма повинна зберігатись тільки в безрегістровому ПЗП. По-друге, дані, що обробляються командою і результати повинні зберігатись тільки в регістрах або локальній пам′яті даних.

ПЗП великого об′єму без проміжних регістрів при реалізації в ПЛІС має великі апаратні витрати і затримку. Тому в ПЛІС велику ПЗП, як правило, виконують на основі оперативного запам′ятовуючого пристрою (ОЗП), який має додаткові регістри адреси і зчитаного даного, тобто це – синхронний ОЗП, звертання до якого додає такт до періоду виконання команд.

Як результат, в ядрах мікроконтролерів об′єм ПЗП програм має невеликий об′єм. А для звертання до ПЗП і ОЗП великого об′єму використовують 1–8 тактів затримки командного періоду (wait states).

Крім того, команди переходу, виклику підпрограм, читання-модифікації-запису, команди з непрямою адресацією, в тому числі звертання до зовнішнього ОЗП даних, принципово не можуть бути виконані за 1 такт.

Таким чином, хоча в мікропроцесорних ядрах і задекларовано, що команди виконуються за 1 такт, насправді, при вирішенні практичних задач для їхнього виконання потрібно 2 і більше тактів. При цьому необхідне збільшення кількості тактів виконується штучно і в додаткові такти ядро практично простоює.

Проектування ядра FS8051

При проектуванні власного ядра мікропроцесора FS8051 на кафедрі обчислювальної техніки НТУУ "КПІ" були визначені наступні умови. Вважаючи на показану вище безпідставність виконання однієї команди за 1 такт, домагатись застосування 2,3 і 4-тактних команд. Задля підвищення можливості повторюваності ядра, знайти таке рішення, яке має як мінімізовані апаратні витрати, так і оптимізовану систему ліній зв′язку.

Таким чином, при виконанні команди за 2 такти командний цикл можна розділити на 2 частини, перша з яких включає читання команди з ПЗПП і запис її в регістр команди (РК), а друга – власне виконання команди і визначення адреси наступної команди в лічильнику команд (ЛК).

Слід відмітити, що при розробці СНК широко використовується метод ресинхронізації, який полягає в пересуванні регістрів вздовж шляхів розповсюдження сигналів, чим досягається балансування затримок в критичних шляхах. Причому ресинхронізацію можна виконувати як вручну, так і автоматично. Після ресинхронізації схеми мікроконтролера РК і ЛК можуть стати відсутніми як такі, так як, наприклад, регістр РК може переміститися на вихід дешифратора команд. Слід також відзначити, що регістри в ПЗПП і ОЗП даних (ОЗПД) не переміщуються при ресинхронізаціїї, так як вони є складовою частиною відповідних блоків синхронної пам′яті.

Як правило, структура мікроконтролера має дві чітко виражні частини: перша – це керуючий пристрій, що включає ПЗПП, ЛК і пристрій керування (ПК) у вигляді окремого мікропрограмного автомата, а друга – це операційний пристрій, який включає АЛП, блок пам'яті даних (ОЗПД), блок керування перериваннями (БП), блок спеціальних регістрів і портів (БСР). Таким чином, керування мікроконтролера зосереджено в ПК, ЛК і ПЗПП .

Раніше при розробці ПК мікропроцесорів віддавалась перевага мікропрограмному автомату, завдяки простоті його розробки і модифікації. Але при його реалізації в СНК, як апаратні витрати, так і критична затримка схем можуть виявитись надмірно великими.

Сучасні засоби логічного синтезу значно зпрощують розробку ПК на основі автомата Мілі, мінімізючи при цьому апаратні витрати. Але при локалізації керування процесора в одному автоматі виникають довгі затримки логічних схем і великі апаратні витрати, які викликані наступними причинами.

По-перше, складний ПК передбачає використання логічних рівнянь з великим числом аргументів. Такі рівняння частіше всього погано мінімізуються і потребують для своєї реалізації схему з довгими логічними ланцюжками. По-друге, як було згадано вище, затримка в лініях з'вязку може бути більшою, ніж сумарна затримка в логічних схемах і її можна оцінити як таку, що пропорційна їхній довжині. Якщо все керування зконцентрувати в одному блоці, то його багаточисельні входи і виходи необхідно підключити до решти блоків мікропроцесора через довгі маршрути з великими затримками. Крім того, сумарне навантаження на один вихід блоку виявляється великим, що також збільшує затримку.

Тому для мінімізації як затримки в критичних шляхах, так і апаратних витрат пропонується структура мікроконтролера з розподіленим керуванням, яка показана на рис. 1.

Чотири блоки: ЛК, ОЗПД, АЛП і БП мають локальні регістри команди РК і блоки керування, які дешифрують тільки ті команди і видають такі керуючі сигнали, які мають відношення тільки до даного блоку. Причому, кожен з блоків керування має в декілька разів менші апаратні витрати і більш високу швидкодію, ніж загальний ПК.

Завдяки такій структурі, блоки мікропроцесора стають локалізованими кластерами, які зв'язані між собою малою кількістю зв'язків.

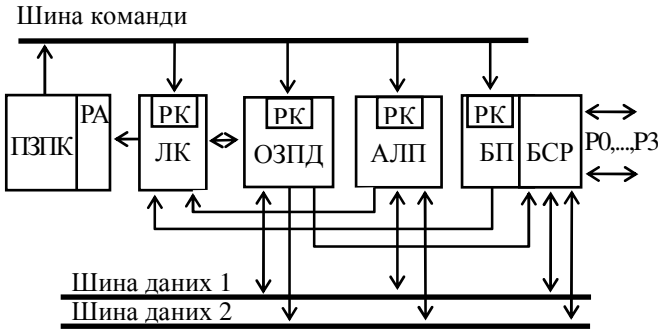


Рис.1. Структура мікроконтролера FS8051

Така кластеризованість проекту забезпечує одержання схеми з мінімальними критичними затримками при автоматичному розміщенні і розводці проекту у відповідній САПР ПЛІС.

Розділення ресурсів вважається ефективним способом мінімізації апаратних витрат. При його реалізації до входів розділених ресурсів, таких як АЛП, лічильники, блоки пам'яті, підключаються багатовходові мультиплексори, а до розділених загальних шин приєднують велике число тристабільних буферів.

Якщо ці розділені ресурси реалізувати в ПЛІС, то і апаратні витрати, і часові затримки будуть збільшеними. Це роз'яснюється тим, що великі мультиплексори реалізуються в ПЛІС як декілька рівнів логічних схем. Також багатовходові мультиплексори передбачають наявність великого числа міжз'єднань. Автоматичне розведення таких міжз'єднань в ПЛІС може привести до надмірних затримок в міжз'єднаннях.

З іншого боку, в ПЛІС складність сумматора порівняно невелика і співпадає зі складністю двохвходового мультиплексора, а його затримка значно менша, ніж у складного мультиплексора. Причому в ядрі мікроконтролера ефективніше збільшити кількість таких ресурсів, як суматор, лічильник, регістр, ніж мінімізувати їхнє число за рахунок додавання великої кількості багатовходових мультиплексорів. Наприклад, додавання, множення, ділення, бітові операції і порівняння в АЛП можна виконувати в окремих вузлах. Структура такого АЛП показана на рис. 2. Цей АЛП складається з арифметічного пристрою (АП), блоків бітових операцій (ББО), множення (БМ), ділення (БД) і десяткової корекції (БДК). Таким чином, відмова від розподілених ресурсів забезпечує мінімізацію критичних шляхів поширення як даних, так і керуючих сигналів.

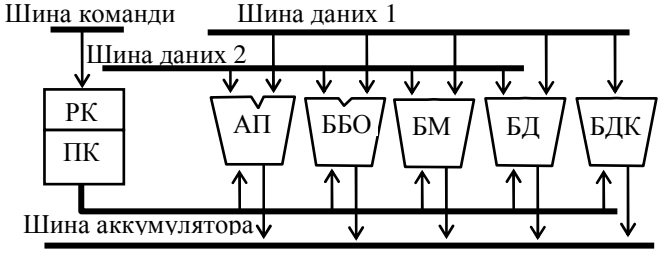


Рис.2. Структура АЛП

Крім того, кожний з пристроїв (БМ, БД та ін.) може бути видалений з АЛП під час синтезу ядра, якщо відповідні йому команди не

використовуються в програмі. При цьому, крім мінімізації апаратних витрат АЛП, спрощується ПК, який не дешифрує ці команди.

Аналогічно оптимізуються структури БП і БСР. В них під час синтезу можна вилучати незадіяні периферійні пристрої і регістри, а також входи переривань.

Результати реалізації ядра FS8051

Розроблене ядро мікроконтролера FS8051 було зконфігуроване в ряді ПЛІС фірми Xilinx. Одержані параметри ядра показані в табл.1.

Табл.1. Параметри ядер мікроконтролерів

Параметр	R8051	DP80390	E8051	FS8051
$T_{ком}$, тактів	2,8-3,5	3,4	3,3	2,9
Q_{V4}, CLB slices	1706	1175	1424	1126
Q_{V5}, CLB slices	779	–	710	615
F_{maxV4}, МГц	79	107	130	198
F_{maxV5}, МГц	73	–	185	229
$T_{лог}$, %	–	–	19-51	46-73
Посилання	[3]	[4]	[5]	

В табл.1. $T_{ком}$ – це середня тривалість виконання однієї команди. Вона розраховувалась з урахуванням вірогідності виконання тих чи інших команд в типових програмах, які виконуються в мікроконтролерах і8051. Для аналогів в $T_{ком}$ враховано, що команди читаються з синхронного ПЗП, доступ до якої збільшується на 1 такт. Q_{V4} і Q_{V5} – це апаратні витрати в еквівалентних конфігурованих логічних блоках ПЛІС Virtex-4 і Virtex-5, відповідно. F_{maxV4} і F_{maxV5} – це максимальні тактові частоти ядер в цих ПЛІС. $T_{лог}$ – це частка затримки в логічних схемах від загальної затримки в критичному шляху, яка взята зі звітів САПР розведення ядер в ПЛІС.

Аналіз таблиці показує, що ядро FS8051 має більшість переваг в порівнянні з кращими аналогічними ядрами мікроконтролерів з такою самою архітектурою.

Висновки

Сучасні СНК мають такі характерні особливості, як висока затримка міжз′єднань, ускладнення повторюваності готових обчислювальних модулів. Це вимагає переглянути сталі підходи до розробки проектів СНК в бік більшої уваги до міжз′єднань, навіть за рахунок збільшених апаратних витрат. Новий підхід, який полягає в проектуванні розподілених керування і АЛП, формування кластерної структури, був перевірений при розробці ядра мікроконтролера FS8051, результати якого показали наступне.

Розподілені керування і операційні блоки дали змогу, завдяки локалізації зв′язків і зменшенню частки затримок в міжз′єднаннях, збільшити тактову частоту в 1,5–2,5 рази в порівнянні з аналогами. Відмова від однотактного виконання команд і використання синхронної пам′яті для зберігання програми дали змогу збільшити продуктивність ядра в 1,14–1,2 рази. Як результат, було одержано ядро мікроконтролера, яке має продуктивність вдвічі вищу, ніж у кращих світових зразків при таких самих або зменшених до 1,5 разів апаратних витратах.

Перелік посилань

1. Pasricha S., Dutt N. Trends in Emerging On-Chip Interconnect Technologies // IPSJ Trans. On Syst. LSI Design Methology. –2008. –V1. –p.2-17.
2. Ball J. Designing Soft-Core Processors for FPGAs. In "Processor Design. System-on-Chip Computing for ASICs and FPGAs" /J. Nurmi Ed. – Springer. –2007. – P.229-256.
3. R8051XC2 High Performance, Configurable 8-bit Microcontroller Core. //CAST inc. –2009. –4p. /available at http://www.cast-inc.com
4. DP80390. Pipelined High Performance Microcontroller Instructions set details. ver 3.10.// DCD –2008. –84p. /available at http://www.dcd.pl
5. Avaliable at http://www.e8051.com

Поступила в редакцию 16.12.2009

УДК 681.322

СЕРГИЕНКО А.М.

ПРИМЕНЕНИЕ РАЦИОНАЛЬНЫХ ДРОБЕЙ В СПЕЦИАЛИЗИРОВАННЫХ ВЫЧИСЛИТЕЛЯХ

В статье рассмотрено нетрадиционное представление данных в виде рациональных дробей, которое позволяет, не применяя чисел с плавающей запятой, выполнять вычисления с повышенной точностью.

An untraditional rational factor data representation in the application specific processors is considered. This data representation provides high computational precision and helps to do without floating point numbers.

Введение

Традиционно, вычисления с повышенной точностью выполняются с плавающей запятой. Однако, большинство микроконтроллеров, а также программируемые логические интегральные схемы (ПЛИС) вычисляют, как правило, с фиксированной запятой, так как в них плавающая запятая реализуется неэффективно. Поэтому актуален поиск способов реализации вычислений с повышенной точностью, основанный на представлении целых чисел.

Для минимизации погрешностей, задержек и аппаратурных затрат при реализации вычислений в специализированных устройствах предлагается использовать нетрадиционное представление данных в виде дробей. Такое представление, например, приведено в [1] для целочисленного решения задач с полиномами.

Рациональные дроби

Рациональная дробь – это числовой объект, состоящий из числителя и знаменателя. Эта дробь представляет рациональное число, т.е. число, полученное как решение системы целочисленных линейных уравнений или как частное от деления целочисленных полиномов. Рациональное число n_x/d_x может эффективно аппроксимировать заданное иррациональное или трансцендентное число x. Если нецелое число x представлено $2n$ разрядами с погрешностью ξ_1, то оно может быть представлено дробью $x = n_x/d_x$ с погрешностью $\xi_2 \approx \xi_1$, причем числа n_x и d_x будут иметь не более чем n разрядов в своем представлении [2].

Представление чисел рациональными дробями имеет ряд преимуществ. Во-первых, используемые в ЭВМ двоичные дроби приближенно представляют вещественные числа. Так, например, дробь $1/9 = 1/1001_2$ – это точное рациональное число в любой системе счисления, но представляется десятичной дробью 0.1111 или двоичной дробью 0.111000111000011_2 с погрешностью, равной отбрасываемым разрядам бесконечной периодической дроби.

Во-вторых, рациональные дроби упрощают аппроксимацию иррациональных или трансцендентных чисел. Элементарные функции эффективно вычисляются по формулам рациональной аппроксимации [3].

И в-третьих, рациональные дроби имеют простой аппарат арифметических действий. При этом основные операции выглядят так:

$$x \cdot y = n_x\, n_y / (d_x\, d_y)\,; \qquad x/y = n_x\, d_y / (d_x\, n_y)\,; \quad (1)$$
$$x+y = (n_x\, d_y + n_y\, d_x)/(d_x\, d_y).$$

При этом собственно деление не выполняется. Для сравнения чисел вычисляется

$$n_x\, d_y - n_y\, d_x\,.$$

При сопоставлении сложности операций нужно учесть, что разрядность числителя и знаменателя, по крайней мере, вдвое меньше, чем у целых чисел при той же точности представления. Поэтому сложность аппаратных сумматоров дробей приближается к сложности аппаратных умножителей целых чисел, а сложность умножителей дробей оказывается вдвое меньшей, чем для целых чисел.

Недостаток вычислений с рациональными дробями состоит в росте разрядности числителя и знаменателя при выполнении серии точных последовательных действий. Для компенсации этого явления приходится искать в дробях наибольшее общее кратное и делить на него числитель и знаменатель [4].

При вычислениях с рациональными дробями, если не требуются точные результаты, следует использовать округление и нормализацию числителя и знаменателя. При этом точность вычислений приближается к точности плавающей запятой при малых значениях ее экспоненты, что будет показано ниже. Но сложность вычислений при этом существенно меньше, так как не нужно выполнение операций с порядками и выравнивание порядков при сложении [5].

Однако остается открытым вопрос преобразования данных в традиционном представлении в рациональные дроби и обратно. Преобразование целого в дробь – тривиально, число подставляется в числитель, а в знаменатель подставляется общий масштабный множитель. Обратный процесс затруднен тем, что для него требуется полноразрядное деление с результатом двойной разрядности.

Вычисления с повышенной точностью – это, как правило, сложные, многоитерационные вычисления. Поэтому все вычисления на первых N–1 итерациях, например, решения системы уравнений, предлагается выполнять с данными в виде рациональных дробей. И только в конце последней итерации алгоритма числители дробей делятся на знаменатели обычным делением с получением результатов.

Поэтому представление данных рациональными дробями обеспечивает как малые погрешности вычислений, так и расширенный динамический диапазон в сравнении с арифметикой чисел с фиксированной запятой, а также простоту реализации в ПЛИС и высокое быстродействие в сравнении с устройствами с плавающей запятой. При большой разрядности дробей (более 32) точность вычислений может превышать точность плавающей запятой.

Следует также отметить, что представление данных рациональными дробями естественно для задач линейной алгебры (ЛА). Если исходные данные представляют обусловленную задачу ЛА, то при достаточной разрядности дробей получаются точные результаты. Эффективность рациональных дробей покажем на примере разработки спецпроцессора для разложения Холецкого.

Процессор для разложения Холецкого с арифметикой рациональных дробей

Разложение Холецкого обычно применяется для LL^{T}– разложения матрицы A таким образом, что

$$A = L \cdot L^T ,$$

где $L_{(\mathrm{N,N})}$ – нижняя треугольная матрица. Этот алгоритм может быть представлен так:

```
do i = 1, N
    l(i, i) = SQRT(a(i, i));
    do j = i+1 , N
        l(j, i)= a(j, i)/l(i, i);
        do k = i+1 , j
            a(j, k) = a(j, k) – a(j, i)*a(k, i);
        end do
    end do
end do.
```

У этого алгоритма в 2 раза меньше объем вычислений, чем у известного алгоритма Гаусса. Алгоритм стабилен и не требует выбора ведущего элемента, благодаря чему он удачно подходит для реализации в специализированных вычислительных системах.

Критический участок алгоритма – это циклическое вычисление умножения с вычитанием. Также повторяющейся операцией является деление. Наконец, необходимо N раз вычислить квадратный корень. Эта функция в арифметике дробей эффективно вычисляется по формуле Ньютона[3]:

$$t^{k+1} = (t^k + p / t^k) / 2 , \qquad (2)$$

где $k = 1,\ldots,K$. Моделирование вычислений рациональных дробей показало, что при начальном приближении $t^1 = 1$ итерационный процесс (2) сходится с погрешностью представления исходного данного p за $K = 5$ итерации для разрядности дроби менее 64.

Для уменьшения числа итераций до $K = 3$ предлагается $t^1 = t_n / t_d$, где t_n, t_d, равны константе a=1 или 3, умноженной на $2^{-]n/2[}$ или $2^{-]d/2[}$ (т.е. это a, сдвинутое на $]n/2[$ или $]d/2[$ разрядов вправо), где n и d – число нулей перед старшей значащей единицей в числителе и знаменателе числа p, причем a= 3, если n или d – нечетное.

Арифметическое устройство (АУ) спецпроцесора имеет структуру, показанную на рис.1. В ней выполняется базовая операция $P=AX\pm Y$, в которой действия выполняются согласно (1). Для деления A/X операнд Y=0, числитель и знаменатель X меняются местами. Блоки нормализации БН1, БН2 сдвигают влево числитель и знаменатель на одинаковое число разрядов, которое не превосходит половины их разрядности.

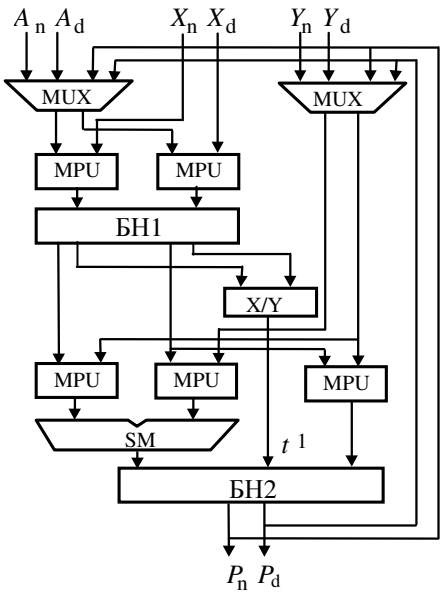


Рис.1 Структура АУ для вычисления алгоритма Холецкого

Обратные связи в структуре АУ предназначены для реализации аккумулятора парных произведений и вычислений квадратного корня по (2). При этом коды разрядов n и d для получения приближения t^1 в схеме X/Y поступают из блока нормализации БН1. АУ выполняет вычисления в конвейерном режиме с глубиной конвейера 9, причем вычисления с накоплением $P=P\pm AX$ имеют период 4 такта, а время вычисления квадратного корня составляет 36 тактов.

Характеристики полученного АУ при его реализации на ПЛИС разных серий представлены в табл. В ней аппартная сложность выражена в количестве эквивалентных конфигурированных логических блоков (ЭКЛБ) и блоков 18-разрядных умножителей DSP48. Для сравнения там же показаны характеристики АУ, составленного из блоков операций с плавающей запятой одинарной точности (разрядность мантиссы – 23), которые обеспечивают такую же функциональность и доступны в САПР Xilinx Coregen [6].

Таблица. Характеристики АУ для реализации алгоритма Холецкого.

Тип ПЛИС и АУ	**XC 4VSX25-12**		**XC 5VLX50-3**
	Xilinx Coregen	**Предлагаемое**	**Предлагаемое**
Апп. затраты, ЭКЛБ	1450 (14%),	1089 (10%),	729 (15%)
Апп. затраты, DSP48	8 (6%),	15 (11%),	15(46%)
Макс. тактовая частота	165 МГц	195 МГц	225 МГц

Как видим, предлагаемое АУ имеет на 30% меньшие затраты в числе ЭКЛБ и большее быстродействие. Хотя количество умножителей увеличено почти вдвое, но аппаратные затраты оказываются сбалансированными, т.е. при полной загрузке кристалла число незадействованных как умножителей, так и ЭКЛБ будет минимальным.

Погрешности алгоритма Холецкого

Был выполнен ряд вычислений разложения Холецкого с различной разрядностью дробей и для сравнения – с плавающей запятой одинарной точности. На рис. 2 показаны зависимости погрешностей, помноженных на 10^7, от разрядности операндов и размерности задачи.

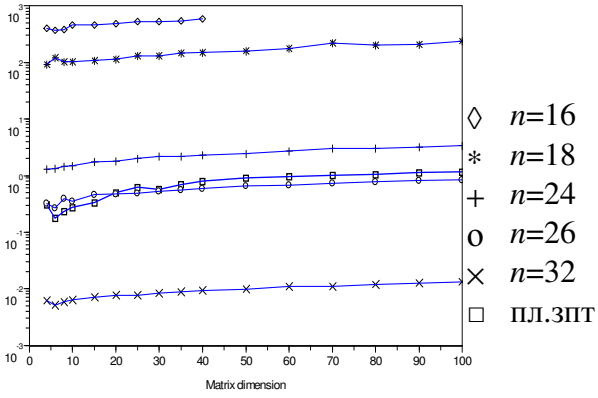


Рис. 2. Погрешности разложения Холецкого

Анализ графиков показывает, что арифметика рациональных дробей с разрядностью 26 обеспечивает такую же погрешность вычислений, как и арифметика плавающей запятой одинарной точности; увеличение разрядности на единицу уменьшает погрешность вдвое.

Эксперименты с разложением Холецкого с показали, что при изменении масштаба входной матрицы (умножении ее на коэффициент M) существенно меняется ошибка выч-лений. Графики зависимости ошибки вычислений от масштаба M показаны на рис. 3.

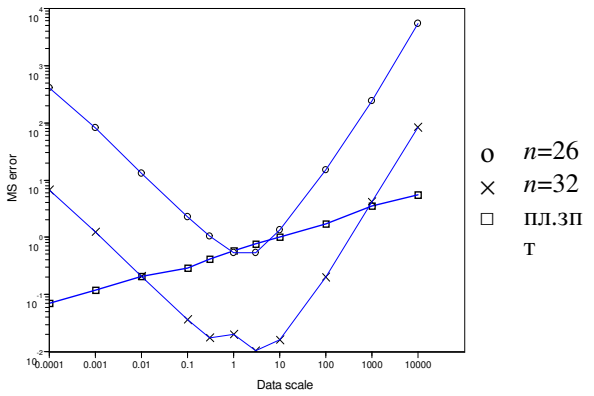


Рис. 3. Зависимость погрешности от масштаба входных данных

Как видим, минимальная погрешность вычислений достигается при M=1, т.е. при матожидании данных, равном 1. Это объясняется тем, что при отклонении M от единицы или в числителе, или в знаменателе дробей число значащих цифр уменьшается на $\log_2 M$. Критическим становится масштаб $M=2^{n/2}$, когда при выполнении операций может получиться нулевой или максимальный результат.

Также замечено, что при увеличении размерности задачи в 10 раз ошибки вычислений растут в $2{,}3 \approx \sqrt[3]{10}$ раза. При вычислении с плавающей запятой это отношение несколько больше – 4,3 [7].

Выводы

Таким образом, показана высокая эффективность арифметики рациональных дробей на примере спецпроцессора для разложения Холецкого. Погрешность разложения Холецкого при применении n-разрядных дробей оценивается как $2^{-n} MN^{1/3}$. Кроме того, если размерность матрицы не превосходит N=100, то разложение Холецкого дает среднеквадратическую ошибку около $2 \cdot 10^{-5} \cdot M$ при вычислении с 18-разрядными дробями. Такой точности достаточно для решения многих задач цифровой обработки сигналов, например в адаптивных фильтрах. При этом эффективно используются ресурсы современных ПЛИС.

Эффективность применения рациональных дробей также была проверена при решении систем уравнений с теплицевой матрицей [8], реализации метода сопряженных градиентов [5].

Представление и обработка данных в виде рациональных дробей может также получить эффективную программную реализацию, например, в сигнальных микропроцессорах.

Список литературы

1. Кнут Д. Искусство программирования. Т.2. –М.:Мир. –1979. –556с.
2. Хинчин А.Ю. Цепные дроби. – М.: Наука, 3-е изд., –1978. –112с.
3. Попов Б.А., Теслер Г.С. Вычисление функций на ЭВМ. –Киев: Наукова думка. –1984. –599с.
4. Irvin M.J., Smith D.R. A rational arithmetic processor // Proc. 5-th Symp. Comput. Arithmetic. –1981. –P.241-244.
5. Сергиенко А.М. Применение арифметики рациональных дробей для реализации метода сопряжения градиентов. //Электрон. моделирование. –2006. –Т.28. –№ 1. – С. 33–41.
6. Sergyienko A., Maslennikov O. Ratushniak P. Implementation of Linear Algebra Algorithms in FPGA-based Fractional Arithmetic Units. //Proc. 9-th Int. Conf. “The Experience of Designing and Application of CAD Systems in Microelectronics”, CADSM’2007. –Lviv-Polyana –20–24 Feb. –2007. – p.228–234.
7. Sergiyenko A., Maslennikow O., Lepekha V., Tomas A., Wyrzykowski R. Parallel Implementation of Cholesky LL^T Algorithm in FPGA–Based Processor //Lecture Notes in Computer Science. —Berlin: Springer. —2008. –V. —p. 137 —147.
8. Maslennikov O., Shevtshenko Ju., Sergyienko A. Configurable Microprocessor Array for DSP Applications.// Lecture Notes in Computer Science. –V.3019. –2004. –P.36–41.

Поступила в редакцию 27.11.2009

УДК 681.322

ЛЕПЕХА В.Л.,
ЛЕСИК Т.М.,
ФРАНКО Р.А.

РЕАЛИЗАЦИЯ ДЕКОДЕРА РИДА-СОЛОМОНА В ПЛИС

Рассматриваются вопросы реализации декодера Рида-Соломона с учетом архитектурных особенностей программируемых логических интегральных схем (ПЛИС). Разработанный с учетом этих особенностей быстродействующий декодер Рида-Соломона может найти применение в телекоммуникациях и системах хранения данных.

Questions of the Reed-Solomon decoder implementation taking into account the FPGA architecture properties are considered. The designed high-speed Reed-Solomon decoder can be utilized in telecommunications and data storing arrays.

Введение

Современные технологии хранения и передачи данных невозможны без адекватных средств их защиты от потерь. Одним из наиболее эффективных методов помехоустойчивого кодирования данных является метод Рида-Соломона [1]. При этом декодирование кодов Рида-Соломона представляет собой решение сложной неоднородной математической задачи с применением нестандартной арифметики полей Галуа. В результате этого, программная реализация декодера Рида-Соломона, как правило, имеет низкую скорость реализации. Поэтому в подавляющем числе случаев такой декодер реализован аппаратно в виде спецвычислителя.

Применение программируемых логических интегральных схем (ПЛИС) дает возможность в сжатые сроки с минимальными затратами разработать сложную систему на кристалле (СНК). Реализация декодера Рида-Соломона в ПЛИС дает возможность разрабатывать на его основе новые СНК для таких приложений, как сохранение баз данных большого объема, прием и передача спутниковых, мультимедийных и телевизионных сигналов и многие другие.

Такая реализация СНК в ПЛИС дает возможность перенастраивать устройство на различные стандарты кодирования информации, его модернизировать без изменения схемы, изменять длину посылок, количество исправляемых ошибок и т.п.

При этом элементный базис ПЛИС существенно отличается от базиса заказной СБИС. Это приводит к тому, что блоки, разработанные для СБИС, будут реализованы в ПЛИС неэффективно.

В статье исследуются вопросы реализации алгоритма Рида-Соломона с учетом элементного базиса современных ПЛИС, возможностей перенастройки алгоритма. В качестве изменяемых параметров декодера выступает длина кодированного слова, количество исправляемых ошибок, характеристический полином поля Галуа.

Алгоритм декодирования Рида-Соломона

Декодирование кодов Рида-Соломона представляет собой довольно сложную задачу. Типовая схема декодирования, получившая название авторегрессионого спектрального метода декодирования, состоит из следующих шагов [2, 3].

– прием закодированного кадра

– вычисления синдрома ошибки (синдромный декодер);

– построения полинома ошибок, осуществляемое либо с помощью высокоэффективного, но сложного алгоритма Берлекэмпа-Месси, либо посредством простого, но медленного Евклидового алгоритма;

– нахождения корней полинома ошибок, которое обычно выполняется перебором (алгоритм Ченя);

– определения характера ошибки, сводящееся к построению маски, вычисляемой на основе обращения алгоритма Форни или другого алгоритма обращения матрицы;

– исправления ошибочных символов путем наложения маски на информационное

слово и последовательного инвертирования всех искаженных бит операцией XOR.

Структурная схема такого алгоритма представлена на рис. 1.

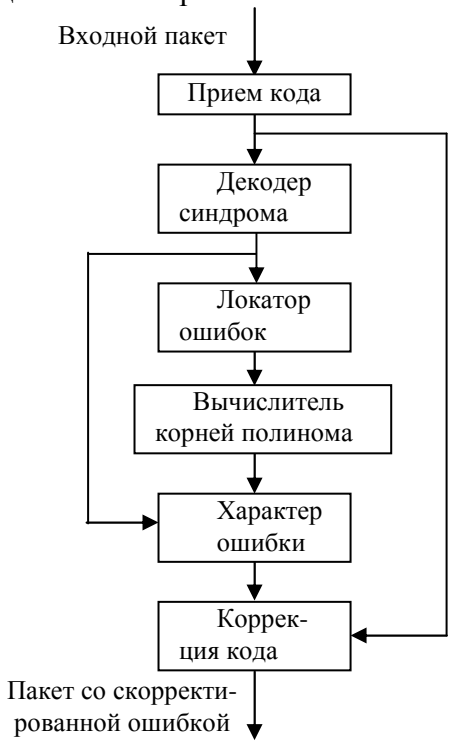


Рис.1. Алгоритм авторегрессионого спектрального декодера Рида-Соломона

Выбор элементной базы

Анализ вышеприведенного алгоритма показывает, что основными операциями являются векторное умножение, деление и возведение в степень в поле Галуа. Это довольно трудоемкие операции, особенно возведение в степень и деление, так как они выполняются в нестандартной системе счисления. Отметим, что в большинстве стандартных декодеров, как и в примерах ниже, вычисления выполняются в поле Галуа GF(2^8).

Ниже будет приведено сравнение реализации декодера на обычном микропроцессоре и ПЛИС. Базовая операция умножения A = B*C может быть выполнена на специализированном комбинационном умножителе чисел Галуа, который имеет сравнительно большие аппаратные затраты. Кроме того, для возведения в m-ю степень необходимо более $\log_2 m$ умножений, а для деления – 16 умножений [1]. Если выполнять вычисления с помощью таблиц, то как умножение, так и другие операции можно выполнять существенно быстрее. Тогда вычисление операции умножения имеет следующий вид при записи на языке VHDL:

```
B0 = tabl0(B) -- выбор операнда из таблицы
C0 = tabl0(C) -- выбор операнда из таблицы
A0 = B0 + C0 -- суммирование в G2^8
A = tabl1(A0) -- выбор операнда из таблицы
```

Здесь tabl0 и tabl1 – таблицы логарифмов и антилогарифмов чисел в поле Галуа размером 256 входов. Как видим, операция умножения будет происходить за 4 макрокоманды, а с учетом организации микропроцессора – за 10 – 15 команд. Учитывая, что, например, для исправления 8 ошибок (часто встречаемый случай) требуется умножение векторов с 16 элементами, такая векторная операция может занять 160 – 240 тактов работы микропроцессора.

При реализации на ПЛИС эти операции можно распараллелить. Современные ПЛИС имеют довольно большое количество независимых блоков памяти, что позволяет сделать такое векторное умножение за 1 – 3 такта. Кроме того, возможны и другие способы ускорения вычислений, которые, в совокупности, дают возможность значительно ускорить реализацию алгоритма.

Исходя из этих рассуждений, сделан вывод о целесообразности реализации декодера Рида-Соломона на ПЛИС с арифметикой, реализованной на таблицах.

Реализация декодера

Как было указано выше, основная операция данного алгоритма – векторное умножение в поле Галуа. Реализация умножителя выполняющего операцию умножение (деления) в G8 за 2 такта показана ниже:

```
d0<=conv_std_logic_vector(rom1(conv_integer(a)),8);
d1<=conv_std_logic_vector(rom1(conv_integer(b)),8
);
sm1 <=ext(d0,9) + ext(d1,9) + 1
     when m_d = '0' else ext(d0,9) - ext(d1,9) -
0;
b1 <= not (m_d xor sm1(8));
sm2 <= sm1 - ext(('0' & b1),9);

process(clk,rst)
begin
if rst = '1' then
sm <= (others => '0');
elsif clk = '1' and clk'event then
          sm <= sm2;
          end if;
end process;

a2 <= sm(7 downto 0);
d2<=conv_std_logic_vector(rom0(conv_integer(a2)),8
);
process(clk,rst)
```

```
begin
if rst = '1' then
z0 <= '0';
elsif clk = '1' and clk'event then
        if a = x"00" or b = "00" then
                z0 <= '1';
        else
                z0 <= '0';
        end if;
        end if;
        end process;
res <=  x"00" when (z0 = '1')  else  d2;
```

Здесь rom1 – двухпортовое ПЗУ таблицы логарифмов, а rom0 – ПЗУ антилогарифмов.

Параметризированный векторный умножитель имеет вид:

```
U_md: for i in 0 to G_range - 1 generate
mul : mul_g8
 port map(
        clk => clk,
rst => rst,
        m_d => m_d,
         a => a(i),
 b => b(i),
         res => c(i)
            );
 end generate;
```

Структурно декодер разбит на три блока, которые выполняют (см. рис.1):

– прием кадра и нахождение синдрома ошибки.

– определение характера ошибки и формирование корректирующей маски;

– коррекцию ошибки.

Прием кадра длиной *N* слов осуществляется в сдвиговый регистр. В дальнейшем эти данные участвуют в формировании синдрома ошибок, который вычисляется последовательно на одном умножителе и накапливающем сумматоре. Результатом является синдром ошибок, на основании которого находится полином ошибки.

Полином ошибки вычисляется с помощью алгоритма Берлекэмпа-Месси. Для этого используется описанный выше векторный умножитель. После этого на том же векторном умножителе реализуются алгоритм Ченя – для поиска корней данного полинома и алгоритм Форни – для формирования корректирующей маски.

В оконечном блоке принятый кадр пословно корректируется путем суммирования с корректирующей маской.

Все модули работают в старт-стопном режиме (последующий модуль начинает работать по сигналу окончания работы предыдущего модуля). Такая реализация позволяет организовать конвейерную работу декодера, что может улучшить временную характеристику системы.

Модуль параметризирован для операций с байтовыми последовательностями (длина слова – 8 бит). В качестве параметров выступает количество исправляемых ошибок *t* (2 – 8 ошибок), длина кадра *N* (до 255 слов), порождающий полином.

Технические характеристики декодера

Основной характеристикой декодера является время выполнения операции декодирования. Поскольку аналитически представление довольно сложное, покажем некоторые данные полученные экспериментальным путем.

В табл.1 указано количество тактов, необходимых для выполнения декодирования, начиная от сигнала начала последовательности, заканчивая, сигналом окончания работы, т.е. время выполнения операции декодирования при различных параметрах. При этом возможность конвейерных вычислений не учитывается.

Табл. 1. Число тактов декодирования

t	*Длина кадра* **N,** *слов*			
	32	**64**	**128**	**255**
2	472	632	1012	1647
3	618	842	1290	2179
4	704	992	1568	2711
5	790	1142	1846	3243
6	876	1292	2124	3775
7	962	1442	2402	4307
8	1048	1592	2680	4839

Существенный рост времени декодирования при увеличении количества исправляемых ошибок связан с тем, что при последовательном нахождении синдрома ошибки здесь используется только один умножитель. Можно ускорить эту операцию, используя векторный умножитель, за счет пропорционального роста аппаратных затрат.

В табл. 2 указаны аппаратные затраты на декодер, параметризованный для различного числа исправляемых слов и длины пакета при его конфигурировании в ПЛИС Xilinx серий Virtex-4 и Spartan-3E. При этом в ячейках указано количество блоков памяти (BRAM) и количество конфигурируемых логических блоков (CLB slices). Данные получены при синтезе проекта декодера в среде ISE 9.2.

Табл. 2. Аппаратные затраты декодера

t	*Длина кадра* **N**, *слов*			
	32	**64**	**128**	**255**
2	8 1324	8 1399	8 1417	8 1567
3	11 1879	11 1882	11 1888	11 1998
4	14 2389	14 2546	14 2638	14 2755
5	17 3052	17 3086	17 3127	17 3196
6	20 3667	20 3714	20 3741	20 3941
7	23 4163	23 4204	23 4252	23 4327
8	26 4425	26 4663	26 4697	26 4712

Максимальная тактовая частота, с которой работает декодер, определяет как его пропускную способность, так и возможности его внедрения в различных проектах СНК. В табл. 3 показана максимальная тактовая частота работы в различных ПЛИС фирмы Virtex.

Табл. 3. Максимальная тактовая частота декодера, МГц

Серия ПЛИС	***N* = 32, *t*=2**	***N* =255, *t* = 8**
Spartan 3 xc3s1500-5	74	66
Spartan 3E xc3s1200e-5	91	72
Spartan 3A xc3sd1800a-4	80	67
Virtex4 xc4vlx40-12	149	126
Virtex5 xc5vlx50-3	184	143

Как по тактовой частоте, так и по аппаратным затратам синтезированный декодер соответствует лучшим мировым образцам декодеров Рида-Соломона, реализованным в ПЛИС.

Вывод

Разработанный декодер Рида-Соломона, благодаря учету структурных свойств современных ПЛИС, реализуется в них с высокой тактовой частотой при умеренных аппаратных затратах, которые сравнимы с параметрами лучших мировых образцов.

Вместе с тем, есть возможности существенного улучшения декодера по увеличению быстродействия. Для этого нужно распараллелить процесс нахождения синдрома ошибки, применяя векторный умножитель в поле Галуа. Также целесообразно увеличить тактовую частоту работы декодера при реализации декодера в сериях ПЛИС Xilinx Spartan 3 и Altera Cyclone, которые .отличаются высоким отношением эффективность-цена.

Список литературы

1. Вернер М. Основы кодирования. –М.: Техносфера. –2004. –288с.
2. Касперски К. Могущество кодов Рида-Соломона или информация, воскресшая из пепла // Системный администратор. –2004. –с.88-94.
3. Блейхут Р. Теория и практика кодов, контролирующих ошибки. –М.:Мир. –1986. –566с.

Поступила в редакцию 17.12.2009

УДК 004.724.4(045)

ДАВИДЕНКО І. М.
КРАВЧЕНКО С. О.
БАБІЙ К. Ю.

ДЕЦЕНТРАЛІЗОВАНА МОДЕЛЬ КЕРУВАННЯ GRID-СИСТЕМОЮ

In this work the method of decentralized *GRID* model management by using the access method which based on *P2P* (*Peer-to-Peer*) technology for represents a promising platform for scalable demanding tasks is offered.

У роботі розглянуто спосіб організації децентралізованої моделі управління *GRID*-системою за рахунок використання методу доступу до ресурсів на основі технології *P2P*, яка представляє собою перспективну платформу для використання крупномасштабованих ресурсномістких задач.

Вступ

Сучасний розвиток обчислювальної техніки знаходиться в переломній стадії, коли окремі обчислювальні вузли через технологічні обмеження вже не в змозі обробляти поставлені завдання в необхідному обсязі із заданим рівнем якості обслуговування *QoS* (*Quality of Service*).

В таких умовах єдиною альтернативою нарощування потужності системи є об'єднання множини обчислювальних вузлів у розподілені *GRID*-системи, які включають не тільки інструменти організації обчислень, але й інструменти управління ресурсами зберігання даних, забезпечення безпеки доступу до даних, моніторингу використання та переміщення даних із необхідним рівнем *QoS* [1]. Важливо зауважити, що основною перевагою таких систем є можливість не тільки паралельної віддаленої обробки даних, але й можливість одночасного доступу до цих ресурсів віддаленим користувачам із заданим рівнем *QoS*.

Традиційно пошук ресурсів в *GRID*-системах в основному базується на централізованій моделі керування. Ця модель керування складно масштабується, що може стати потенційною проблемою продуктивності та безпеки *GRID*-систем. Таким чином, головною метою *GRID*-систем є можливість забезпечення користувачів обчислювальною потужністю, необхідною для обробки завдань за допомогою географічно розподілених ресурсів з заданим рівнем *QoS*.

1.Огляд та аналіз існуючих рішень

Під *GRID*-системою розуміється географічно розподілена інфраструктура, яка об’єднує множину ресурсів різних типів, доступ до яких користувач може отримати із будь-якої точки, незалежно від місця їх розташування [2]. Основною метою *GRID*-систем є об’єднання гетерогенних, широкомасштабних та багатоцільових ресурсів для надання прозорого, надійного та скоординованого доступу до обчислюваних ресурсів, які знаходяться на великій відстані.

У зв'язку з тим, що *GRID*-система являє собою логічне налаштування над розподіленою обчислювальною системою, однією з найважливіших її частин є система управління ресурсами, що виконує завдання виявлення, розміщення, відстеження та управління мережевими ресурсами для забезпечення різних вузлових і глобальних параметрів *QoS* [3]. Крім того зауважимо, що ресурси *GRID*-системи належать різним класам, які визначаються низкою атрибутів та характеристик [4]. У кожного ресурсу є певне значення для кожного атрибуту, визначеного відповідним класом ресурсу. Вимоги до системи управління ресурсами *GRID*-системи, роблять децентралізоване управління складним і не маючим однозначного рішення.

Найбільш розповсюдженим методом децентралізованого управляння ресурсами є технологія *P2P*, яка стає перспективною платформою для виконання великомасштабних ресурсномістких задач. Підхід *P2P* має ряд особливостей, які дають можливість забезпечити управляння великою кількістю вузлів із заданим рівнем *QoS*, дії яких основані на локальній інформації, що зберігається в кожному вузлі. Крім того системи, засновані на такому підході, легко масштабуються.

Для управління ресурсами та самоорганізації великомасштабованих і розподілених *GRID*-систем, найчастіше використовують

оптимізований мурашковий алгоритм *ACO* (*Ant Colony Optimization*). В роботі [5] запропоновано метод подібний до алгоритму самоорганізації мурашок, який дозволяє рівномірно розподіляти роботу серед доступних ресурсів.

Для гетерогенного обчислювального середовища запропоновано та спроектовано гібридний *ACO*, який балансує навантаження на ресурси *GRID*-системи. У зв'язку з тим, що ресурси *GRID*-системи належать різним класам (інформаційний, обчислювальний та ресурс зберігання даних), вони вимагають великої кількості параметрів, які характеризують клас саме цього ресурсу.

2.Постановка задачі

У зв'язку з тим, що більшість існуючих систем управління *GRID* архітектурою мають централізоване управління із складним процесом налаштування та обслуговування, актуальною є проблема розробки нових децентралізованих підходів для вирішення завдань надання доступу до ресурсів *GRID*-системи, для груп віддалених користувачів із заданим рівнем *QoS*.

Тому необхідно запропонувати спосіб організації децентралізованої моделі управління *GRID*-системами, який гарантує надійну доставку інформації. Проте, щоб гарантувати доступ до даних з вимогами *QoS*, які пред'являються до спільних обчислень, необхідна більш проста структура, яка буде використовувати метод доступу до даних заснований на концепції *P2P*-систем. В даний час існує ряд рішень цієї задачі, але залишаються відкритими ще такі питання, як: швидке та динамічне виявлення та резервування обчислювальних ресурсів; визначення і пошук вільних ресурсів; зменшення часу аналізу і планування ресурсів (затримок); динамічне виявлення та аналіз ресурсів; вирішення задач для зменшення затримок з урахуванням балансування завантаження; розподіл мережевих ресурсів між задачами.

3.Рішення поставленої задачі

У роботі запропоновано підхід організації децентралізованої моделі управляння *GRID*-системою за рахунок використання методу доступу до ресурсів за технологією *P2P*. Однак, для забезпечення гарантованого доступу до ресурсів із заданим *QoS* необхідна більш проста структура. Для цього у роботі запропоновано підхід організації системи моніторингу та виявлення *MDS* (*Monitoring and Discovery System*) ресурсів *GRID*-системи. Розроблена *MDS* працює на основі архітектури *P2P*, на відміну від традиційної ієрархічної, за рахунок використання розподіленої системи мурашкових колоній *DACS* (*Distributed Ant Colony System*) [6] для виявлення необхідних ресурсів, зареєстрованих у *MDS*. У основу алгоритму покладено поведінку мурашкових колоній, де агент *MDS* керує множиною запитів на ресурс із маркування найбільш вдалих шляхів більшим значенням вагового коефіцієнту.

Нехай *GRID*-система представляє собою граф *G(V,E)*, який складається з *n* вузлів та *m* ліній зв'язку *e(i, j)*, кожна з яких з'єднує два вузли *i, j* (рис. 1).

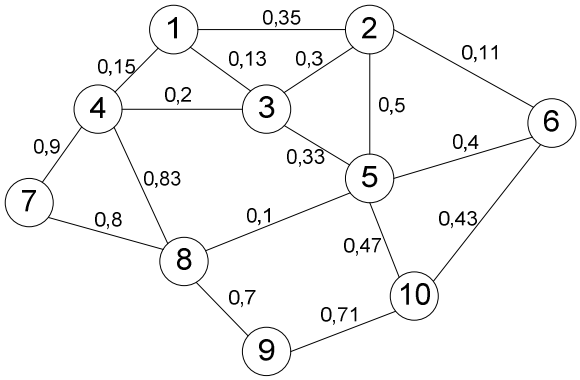


Рис.1. Приклад GRID-системи із заданими ваговими коефіцієнтами пошуку k-го ресурсу

На першому етапі у вершинах графа розміщуються запити на ресурс, після чого починається їх направлене розповсюдження по графу згідно алгоритму, наведеному нижче.

Мітка запиту на ресурс розташована у вершині V_i і визначає тип необхідних ресурсів *k*, і в разі їх виявлення відправляє вузлу-відправнику. Якщо ж ресурс не виявлено, мітка запиту на ресурс передається до наступної вершини V_n,, при цьому для переходу обчислюється ймовірність переходу на кожну з сусідніх вершин з використанням вагового коефіцієнта кожної лінії зв'язку *e(i,j)* для *k*-го типу ресурсу. Ця ймовірність обчислюється за формулою:

$$P_{ij}^{k} = \frac{\tau_{ij}^{k}}{\sum_{q} \tau_{iq}^{k}}, \quad q \in J_i, \qquad (1)$$

де J_i – представляє сусідів вершини, τ_{iq}^{k} – значення вагового коефіцієнта відповідного зв'язку з сусідньою вершиною.

Коли на деякому переході мітки запиту буде виявлено потрібний ресурс, мітка повертається до вершини-відправника запиту. Наступним кроком є оновлення значення вагового коефіцієнта для кожного зв'язку на знайденому шляху, що визначається за функцією:

$$\tau_{ij}^{k} = (1-\rho)\tau_{ij}^{k} + \rho\Delta_{ij}^{best},\ \rho \in (0,1],$$
$$\Delta_{ij}^{best} = \frac{1}{L_{best}}, \qquad (2)$$

де ρ – коефіцієнт зміни значення вагового коефіцієнта, Δ_{ij}^{best} – ваговий коефіцієнт, що встановлюється по шляху проходження кращою міткою запиту, L_{best} – довжина найкращого шляху проходження для заданої позначки.

Нехай мітка запиту на ресурс розташована у вершині V_5, а ресурс k-го типу, що потребується згідно запиту, знаходиться у вершині V_7. За формулою (1) проведемо розрахунки ймовірностей переходів. Оскільки ми розраховуємо ймовірності переходів, то шлях може бути обрано випадковим чином із поступовим збільшенням ймовірності знаходження найоптимальнішого шляху. В даному прикладі прокладемо шлях переходами з найбільшими значеннями ймовірності. Отримані результати наведено у табл. 1.

Табл. 1. Шлях мітки запиту зі значеннями ймовірності вибору наступного переходу

V_5	P_{ij}^k	V_2	P_{ij}^k	V_1	P_{ij}^k	V_4	P_{ij}^k
V_3	0.18	***V₁***	**0.46**	V_2	0	V_1	0
V₂	**0.28**	V_3	0.4	V_3	0.46	V_3	0.1
V_6	0.22	V_5	0	***V₄***	**0.54**	***V₇***	**0.47**
V_8	0.06	V_6	0.14			V_8	0.43
V_{10}	0.26						

Зверніть увагу, що ймовірність переходу на вершину, з якої прийшла мітка, встановлюється рівною 0, оскільки в цій вершині мітка вже була і розшукуваний ресурс там знайдено не було.

Після того, як мітка запиту, що згенеровано, повернеться до відправника запиту V_5, на основі довжини найліпшого шляху L_{best} та ймовірності вибору мітки запиту P_{ij}^k по формулі (2) визначаються оновлені значення вагового коефіцієнта для кожного зв'язку на знайденому шляху.

У даному випадку, це будуть зв'язки за маршрутом, який проходить через вершини: V_5 -V_2 -V_1 - V_4 - V_7.

У роботі запропоновано рішення задачі виявлення ресурсів у *GRID*-системі, перевагою якого є можливість децентралізованого пошуку ресурсів в *GRID*-системі і простоті реалізації.

4.Моделювання

В рамках роботи проведено моделювання запропонованого алгоритму *MY*, *DACS* і алгоритму *AS*, який є одним з існуючих рішень, що також заснований на алгоритмі мурашиних колоній (рис.2).

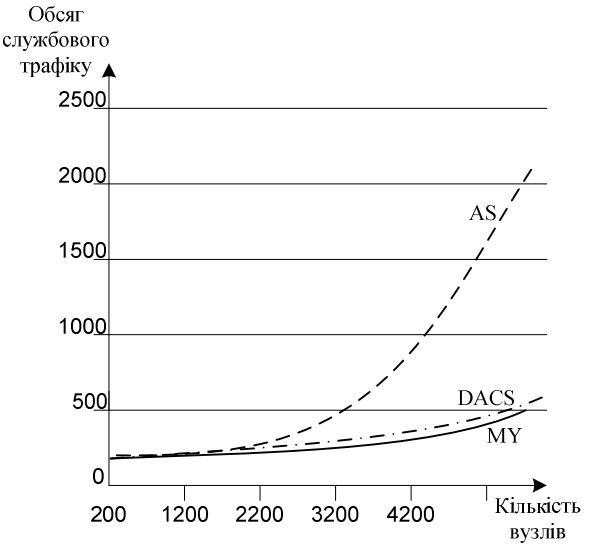


Рис. 2. Порівняння часових характеристик запропонованого алгоритму MY із DACS та AS.

За критерієм обсягу службового трафіку запропонований алгоритм *MY* показав найкращі результати, але не значним зменшення службового трафіку по відношенню до алгоритмів *DACS* та *AS*. Істотне зменшення стає при загальній кількості вершин більше 3200.

Висновки

Розроблена *MDS* для розв'язання задач пошуку ресурсів в *GRID*-системі має ряд переваг, основними серед яких є можливість децентралізованого управлення системою. Розроблений спосіб організації управління *GRID*-системою у повній мірі відповідає всім необхідним вимогам динамічного середовища *GRID* та має властивості децентралізованого виконання, що збільшує швидкість його роботи та надійність, зменшує обсяг службового трафіку та надає можливість функціонування у динамічному середовищі.

Список літератури

1. J. Coomer, C.Chaubal, Introduction to the Cluster Grid. – Grid Computing, Sun BluePrints, 2002. – P. 416.

2. I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. – International Journal Supercomputer Applications, 2001. – Vol.15(3). – P.315 – 327.

3. J. M. Pullen, R. Brunton, D. Brutzman, D. Drake, M. Hieb, K. Morse, A. Tolk, Using web services to integrate heterogeneous simulations in a grid environment. – Future Generation Computer Systems, 2005. – Vol. 21. – P.97–106.

4. J. Luo, P. Ji, X. Wang, Y. Zhu, F. Li, T. Ma, X. Wang, Resource management and task scheduling in grid computing. – Computer Supported Cooperative Work in Design, 2004. – Vol. 2. –P. 431–436.

5. C. Blum, Beam-ACO – Hybridizing ant colony optimization with beam search: an application to open shop scheduling. – Comput. Oper. – 2005. – Vol. 32. – P.1565–1591.

6. S. Fattahi, N. Charkari, Distributed ACS Algorithm for Resource Discovery in Grid, 2009. – P.37. –45.

УДК 004.3

БЛІНОВА Т.О.

ДІЛЕННЯ В КОДАХ ЗОЛОТОЇ ПРОПОРЦІЇ В ПОРОЗРЯДНОМУ РЕЖИМІ, ПОЧИНАЮЧИ ЗІ СТАРШИХ РОЗРЯДІВ

Розроблено алгоритм виконання ділення для чисел, представлених у нестандартній мінімізованій формі (MN-формі) коду класичної золотої пропорції. Використання MN-форми дозволяє здійснювати порозрядні обчислення, починаючи з найбільш значущого розряду, та контролювати їх у динамічному режимі. Наведені приклади сфер використання кодів Фібоначчі та золотої пропорції, зокрема, у системах, що працюють у середовищах із великими індустріальними перешкодами.

The division algorithm is developed for numbers, presented in the nonstandard minimized form (MN-form) of the classic golden ratio base number code. The MN-form usage allows to carry out on-line computations, beginning from the most meaning digit, and to control them in the dynamic regime. Example of the Fibonacci and golden ratio base number code usage is shown, e. g. in the systems which work in environments with large industrial obstacles.

Із 1980-х років на кафедрі обчислювальної техніки НТУУ "КПІ" проводяться дослідження по використанню кодів Фібоначчі та золотої пропорції з метою здійснення обробки інформації та її контролю у динамічному режимі, не чекаючи завершення конкретної обчислювальної операції. Ці дослідження були впроваджені при виконанні декількох науково-дослідних робіт, зокрема, роботі "Розробка структури локальної обчислювальної мережі MAP/TOP і її контролеру на основі кодів Фібоначчі". Питанням побудови пристроїв, що працюють у коді Фібоначчі та золотої пропорції (золотого перетину) присвячені публікації та авторські свідоцтва, наведені в [1]. На відміну від цих досліджень автором статті були розроблені пристрої, що виконують порозрядне обчислення починаючи із найбільш значущого розряду (MSDF-режим). Такий режим обробки інформації дає можливість починати виконання чергової обчислювальної операції як тільки будуть отримані перші найбільш значущі розряди операндів, ще до закінчення попередньої операції. Для цього автором статті була розроблена мінімізованої форма представлення чисел у класичному коді Фібоначчі та золотої пропорції.

У роботі [2] наведені алгоритми виконання усіх, за винятком операції ділення, обчислювальних операцій, що здійснюються над кодами класичної золотої пропорції в MSDF-режимі.

Мета даної статті – описати розроблений автором алгоритм виконання операції ділення над числами, що поступають порозрядно, починаючи з найбільш значущих розрядів, і представлені в коді золотої 1-пропорції у вигляді мінімізованої форми.

Правило виконання ділення без відновлення остатку в прямому коді може бути сформульоване наступним чином.

1. Відняти від діленого дільник. Якщо результат від'ємний, то старша цифра (з вагою α_1^0) частки дорівнює нулю; в іншому – одиниці. Отримана різниця є першим залишком.
2. Збільшити попередній залишок у α_1 разів.
3. Якщо попередня цифра частки дорівнює нулю, то додати дільник. При цьому утворюється черговий залишок, знак котрого визначає чергову цифру частки за правилами, п 1.
4. Повторювати пункти 2 та 3 доти, доки не будуть отримані всі цифри частки.

Особливості виконання цієї операції в MSDF-режимі наступні:

- обидва операнди представляються в мінімізованій формі;
- у найбільш значущому розряді (із вагою α_1^{-1}) дільника повинна бути одиниця, тобто дільник має бути нормалізованим;
- операнди поступають порозрядно, починаючи з найбільш значущих розрядів;
- підсумовування та віднімання виконуються порозрядно (відповідно до алгоритмів, наведених у [2]);

- при визначенні чергової цифри частки аналізуються обидва знакових розряди;
- після отримання знакових розрядів чергового залишку може починатися черговий такт підсумовування чи віднімання;
- частку треба округлити та привести до мінімізованої форми, причому виконання цих операцій може бути суміщено.

Остання вимога обумовлена тим, що частка може бути отримана з недостачею та в немінімізованій формі. Це буде показано нижче.

Віднімання має сенс здійснювати за допомогою представлення у доповняльному коді.

У [2,3] було показано, що при зсуві у старші розряди від'ємного залишку, представленого в доповняльному коді, необхідно додати найбільш значущий знаковий розряд, що має бути відкинутим, до розряду з вагою α_1^0. Корекцію можна виконувати одночасно з операцією додавання дільника та залишку, збільшеного в α_1 разів.

Виконання операції ділення в прямому коді без відновлення залишку може здійснюватися у відповідності до наступного алгоритму.

Нехай виконується ділення *D* та *B*. Позначимо доповняльний код дільника як *B'*, розряди перенесення як *e*, проміжних сум як *u*, і результату як *ü*. Тоді алгоритм цієї операції можна записати наступним чином.

1: $a_{0,-1} = a_{1,-1}=0$; $ü_{-1}=1$;
$a_{2,-1}= d_1$; $a_{3,-1}= d_2$;…; $a_{i+1,-1}= d_i$;…; $a_{m+1,-1}= d_{im}$;
$i = -1$; $j = 0$;

2: $e_{0,j}= a_{-1,\,j-1}$; $o_{i,j}=b_{i-1}\bar{ü}_{j-1} \vee b'_i ü_j$;

3: $e_{i+2,j} = E(a_{i+1,j-1},o_{i,j},e_{i,j},u_{i-1,j}^{\,i-1})= o_i e_i \bar{u}_{i-1}^{\,i-1} \vee a_{i+1,j-1} e_i \bar{u}_{i-1}^{\,i-1} \vee a_{i+1,j-1} o_i \bar{u}_{i-1}^{\,i-1} \vee a_{i+1,j-1} o_i e_i$;
$u_{i,j}^{\,i} = U_0(a_{i+1,j-1},o_{i,j},e_{i,j},u_{i-1,j}^{\,i-1})= \bar{a}_{i+1,j-1} \bar{o}_{i,j} e_{i,j} \bar{u}_{i-1,j}^{\,i-1} \vee a_{i+1,j-1} \bar{o}_{i,j} \bar{e}_{i,j} \bar{u}_{i-1,j}^{\,i-1} \vee \bar{a}_{i+1,j-1} o_{i,j}\ \bar{e}_{i,j}\ u_{i-1,j}^{\,i-1} \vee a_{i+1,j-1} \bar{o}_{i,j}\ e_{i,j}\ u_{i-1,j}^{\,i-1} \vee a_{i+1,j-1} o_{i,j}\ \bar{e}_{i,j}\ u_{i-1,j}^{\,i-1}$;
$u_{i-1}^{\,i} = U_1(a_{i+1,j-1},o_{i,j},e_{i,j},u_{i-1,j}^{\,i-1})= \bar{a}_{i+1,j-1} o_{i,j}\ e_{i,j}\ \bar{u}_{i-1,j}^{\,i-1} \vee a_{i+1,j-1} \bar{o}_{i,j}\ e_{i,j}\ \bar{u}_{i-1,j}^{\,i-1} \vee a_{i+1,j-1} o_{i,j}\ \bar{e}_{i,j}\ \bar{u}_{i-1,j}^{\,i-1} \vee a_{i+1,j-1} o_{i,j}\ e_{i,j}\ u_{i-1,j}^{\,i-1}$;
$u_{i-2}^{\,i} = U_2(a_{i+1,j-1},o_{i,j},e_{i,j},u_{i-1,j}^{\,i-1},\ u_{i-2,j}^{\,i-1})= u_{i-2,j}^{\,i-1} \vee a_{i+1,j-1} o_i e \vee \bar{a}_{i+1,j-1} e_{i,j}\ u_{i-1,j}^{\,i-1} \vee a_{i+1,j-1} \bar{o}_{i,j}\ u_{i-1,j}^{\,i-1} \vee o_{i,j}\ e_{i,j}\ u_{i-1,j}^{\,i-1}$;

4: $u\grave{}_{i-2,j} = u_{i-2,j}\ \bar{u}\grave{}_{i-3,j-1}$;
$u\grave{}\grave{}_{i-3,j} = \bar{u}\grave{}_{i-2,j}\ u_{i-3,j-1}$;
$y_{i-4,j} = u\grave{}\grave{}_{i-4,j-1} \vee u_{i-2,j-1}\ \bar{u}\grave{}_{i-3,j}$;

5: Якщо $i < m+8+j$, тоді $i = i+1$ і на 2;

6: $ü_j = a_{-i,j}\ a_{0,j} \vee \bar{a}_{-1,j} \bar{a}_{0,j}$;

7: Якщо $j < m+1$, тоді $j = j+1$ і на 2;

5: Кінець.

Слід зазначити, що умова прирощення змінної за п.5 враховує можливість виходу одиниці переносу з молодшого розряду за межі *m*-розрядного залишку.

Знакові розряди $a_{-1,j}$ та $a_{-1,j}$ чергового *j*-го залишку остаточно формуються, коли відомі тільки перші чотири розряди $a_{i+1,j-1}$ та $o_{i,j}$ (i = -1,0,1,2,3,4) доданків. При діленні завжди додаються числа із протилежними знаками (залишок та дільник). Отже, якщо додатний доданок $B{\cdot}\alpha_1$ за абсолютною величиною більший, ніж поточне значення від'ємного залишку, знак тільки що сформованого залишку повинен змінитися на "+". Однак може статися, що знак не зміниться. Це обумовлено тим, що при його формуванні враховуються тільки старші розряди операндів. Отже при виконанні ділення в MSDF-режимі на якомусь *j*-му кроці частка може бути обчислена з недостачею. Однак, завдяки надлишковості коду золотої пропорції, тому що ваги сусідніх розрядів у кодовому слові зв'язані співвідношенням $\alpha_1^{\,j}=\alpha_1^{\,j-1}+\alpha_1^{\,j-2}$, цифру частки $ü_j$=1 можна замінити цифрами $ü_j'$=0, $ü'_{j-1}$=1, $ü'_{j-2}$=1, при умові, що $ü_{ij-1}$=0. Це означає, що для продовження ділення повинна виконуватися умова

$$A_{j+1} < B+B, \qquad (1)$$

де A_{j+1} – залишок, отриманий на(j +1)-м кроці. Нехай знак попереднього залишку A_j повинен був змінитися, але залишився від'ємним.

У відповідності до розробленого алгоритму при від'ємному знаку залишку A_j

$$A_{j+1} = A_j\ \alpha_1^{\,0} + B\alpha_1^{\,-1}.$$

Після підстановки у (1) значення A_{j+1} отримаємо наступну умову

$$A_j{\cdot}\alpha_1 < B(\alpha_1^{\,0} + \alpha_1^{\,-2}). \qquad (2)$$

Максимальне значення від'ємного залишку A_j, знак котрого залишився від'ємним, не перевищує числа 10,110110110… (це додатне число, і після проведення згортки можна отримати інший його запис, а саме 00,001001001…).

Ураховуючи, що у відповідності із правилами здійснення зсуву мінімізованих форм чисел

$$(10,110110110\ldots)\alpha_1^{\,0} = 01,10110110\ldots + \\ +01.000\ldots = 00,01010101\ldots$$

можна зробити висновок, що для виконання умови (2) досить виконання умови

$$00,010101\ldots < B(\alpha_1^{\,0} + \alpha_1^{\,-2}). \qquad (3)$$

Однак, мінімальне значення нормалізованого дільника дорівнює $\alpha_1^{\,-1}$, отже для виконання умови (3) достатньо, щоб число 00.01001001… було строго менше, ніж $\alpha_1^{\,-1}(\alpha_1^{\,0}+\alpha_1^{\,-2})=\alpha_1^{\,-1}+\alpha_1^{\,-3}$. Виконання останньої умови очевидно, позаяк число $00,01001001\ldots = \alpha_1^{\,-2}+\alpha_1^{\,-5}+\alpha_1^{\,-8}+\ldots$ напевно менше за число $\alpha_1^{\,-1}+\alpha_1^{\,-3}$.

Отже величина залишку A_j при запропонованому алгоритмі ділення не перевищує величини, визначеної виразом (1).

Таким чином, якщо на j-кроці алгоритму чергова цифра частки визначена з недостачею, операція ділення може продовжуватися, позаяк значення j-й цифри частки корегується при формуванні чергових цифр частки з номерами (j+1) та (j+2). Із цього також випливає, що частка підлягає частковій згортці для переведення в мінімізовану форму.

Підкреслимо, що виконання операції ділення зводиться до виконання додавання, яке може контролюватися на кожному кроці, як показано у [2]. Представлення інформації в мінімізованій формі дозволяє також контролювати передачу інформації між окремими обчислювальними пристроями системи шляхом виявлення заборонених кодових наборів.

Таким чином, показана можливість виконання та контролю всіх логічних та арифметичних операцій над кодами золотої пропорції у MSDF-режимі. Основна ідея MSDF-режиму та подібних до нього – якщо якась інформація у цифровому коді надходить послідовно і якщо можливо розпочати обробку інформації, не чекаючи надання її повного об'єму, то таке суміщення у часі може прискорити швидкодію цифрової системи.

Слід зазначити, що перетворення в MSDF-режимі інформації, що представлена в кодах із ірраціональною основою, знайшли широке застосування у теорії quasicrystal [5].

Традиційно порозрядні обчислення у MSDF-режимі використовуються в сигнальних процесорах, що здійснюють DCT, FFT, CORDIC, фільтрацію та обробку матриць. Перспективним може виявитися застосування перетворень у MSDF-режимі інформації, що представлена в кодах Фібоначчі та золотої пропорції, у теорії fuzzy sets.

У [3] відзначена ефективність використання системи АЦП(ЦАП)-оптичний канал-операційні пристрої, яка працює цілком у класичному коді Фібоначчі та золотої 1-пропорції, у середовищах із великими індустріальними перешкодами. Система містить АЦП(ЦАП), що самокоректуються, цифровий оптичний канал, який нечутливий до електромагнітних перешкод, правильність передачі інформації по котрому може контролюватися за рахунок надмірності кодів, і операційні пристрої, що працюють у коді Фібоначчі та золотої пропорції, для того, щоб уникнути зайвих операцій по перекодуванню сигналів у(із) двійковий код та використати додаткові можливості по динамічному контролю інформації.

Слід відзначити ефективність застосування такої системи і у випадках, коли необхідно за певний проміжок часу обробити зняту з АЦП інформацію, що швидко змінюється. Іншими словами, доцільно дослідити можливість використання кодів Фібоначчі та золотої пропорції при побудові систем реального часу.

Список літератури

1. Институт золотого сечения. http://www.trinitas.ru/
2. Блінова Т.О. Додавання та множення в кодах золотої пропорції в порозрядному режимі, починаючи зі старших розрядів //Вісник НТУУ "КПІ". Сер. Інформатика, управління та обчислювальна техніка. .№ 48. – 2008.– С.54-57.
3. Блинова Т.А. Представление отрицательных чисел в кодах золотой пропорции //Вестник КПИ. Сер. Автоматика и электроприборостроение. – N24. – 1987. – С. 125-127.

4. Блінова Т.О. Використання кодів Фібоначчі та золотої пропорції в цифрових оптичних каналах передачі інформації //Вісник НТУУ ”КПІ”. Сер. Інформатика, управління та обчислювальна техніка. № 47. – 2007. – С.255-261.
5. Ch. Frougny. On-line digit set conversion in real base. Theoret. Comp. Sci. – 2003, pp. 221-235. http://citeseer.ist.psu.edu/frougny00line.html.

Поступила в редакцию 15.12.2009

УДК 004.056

ЗЮЗЯ А.А.

СПОСОБ ПРОТИВОДЕЙСТВИЯ РЕКОНСТРУКЦИИ КЛЮЧЕЙ БЛОКОВЫХ АЛГОРИТМОВ ЗАЩИТЫ ИНФОРМАЦИИ АНАЛИЗОМ ДИНАМИКИ ПОТРЕБЛЯЕМОЙ МОЩНОСТИ

В статье рассмотрены методы реконструкции ключей криптоалгоритмов с помощью анализа динамики потребляемой мощности. Проведен анализ существующих программных контрмер для таких видов атак. Предложен новый способ противодействия анализу динамики потребляемой мощности, основанный на организации псевдопараллельной обработки блоков в режиме разделения времени. Способ отличается стохастическим переключением между обрабатываемыми блоками для обеспечения случайной временной локализации этапов обработки, что значительно снижает эффективность реконструкции ключа с помощью анализа динамики потребляемой мощности. Предложенный способ реализован для алгоритма ГОСТ 28.147-89. При этом падение производительности меньше, чем при использовании других способов противодействия анализу динамики потребляемой мощности.

The article studies power analysis key reconstruction methods for cryptographic algorithms. The analysis of existing software countermeasures for such types of attacks was done. A new power analysis obstruction method was proposed. It is based on pseudo parallel data processing in time sharing mode. The method is characterized by a stochastic switching between proceeding blocks of data for random time location of block processing stages. It significantly reduces effectiveness of the power analysis key reconstruction methods. Proposed method was implemented in software implementation of GOST 28.147-89 cryptographic algorithm. Decrease in performance of such modified implementation is less then via using other power analysis obstruction methods.

Введение

Рост интеграции информационных ресурсов на основе компьютерных технологий к началу 21-го века стал одним из решающих факторов поступательного развития всех сфер деятельности современного общества. Оборотной стороной расширения информационной интеграции является увеличение числа угроз нарушения конфиденциальности и безопасности данных. Для нейтрализации этих угроз широко используются системы защиты информации, в основе большинства из которых лежат криптографические алгоритмы. Современные алгоритмы криптографической защиты являются полностью открытыми (закрытым элементом является только ключ) и базируются на аналитически неразрешимых математических задачах. Соответственно, в большинстве случаев, для нарушения защиты данных необходимо узнать ключ.

Традиционно, для нарушения защиты – то есть реконструкции ключа используется перебор возможных его значений, при этом, для уменьшения объема перебора используются алгебраические свойства криптографических алгоритмов. Такие технологии нарушения защиты анализируют лишь математические аспекты алгоритма, абстрагируясь от вычислительной платформы, на которой реализуется система защиты данных.

В последнее десятилетие появились и активно развиваются новые технологии получения несанкционированного доступа к ключевой информации криптографических алгоритмов. Эти технологии базируются на получении информации о ключе криптографического алгоритма посредством анализа параметров технической реализации на конкретной вычислительной платформе. В литературе [1] такие технологии получения незаконного доступа к ключу алгоритмов защиты информации получили название атак по сторонним каналам. В отличии от традиционных методов, атаки по сторонним каналам учитывают особенности конкретной аппаратной реализации криптосистемы, а не только математического алгоритма, используемого в ней. Такое сужение специализации позволяет таким атакам достигать значительно более высокой эффективности при работе с конкретной криптографической системой [1].

Наиболее распространенной технологией реконструкции ключа алгоритма по параметрам его технической реализации является измерение и анализ динамики

потребления мощности (АДПМ) вычислительным устройством, на котором выполняется алгоритм.

Для эффективного применения АДПМ важным является то, чтобы на устройстве реализовался только один вычислительный процесс. Это условие выполняется для микроконтроллеров, встроенных микропроцессоров и смарт-карт. Именно эти вычислительные устройства являются основными объектами для применения методов реконструкции ключей криптографических алгоритмов с помощью АДПМ.

Учитывая, расширяющееся использование указанного класса вычислительных устройств в современных информационных технологиях, задача создания эффективных средств противодействия широкое несанкционированного доступа к ключам защиты посредством АДПМ является актуальной и важной.

Технологии реконструкции ключей на основе АДПМ и анализ методов противодействия

Впервые метод реконструкции ключей криптографических алгоритмов с использованием АДПМ был предложен Полом Кочером [2] и получил широкое распространение благодаря комбинации таких факторов, как высокая эффективность, низкая стоимость и невозможность обнаружения системой защиты информации того, что она стала объектом атаки (ввиду пассивности и неинвазивности). АДПМ базируется на различии в потребляемой мощности устройством при выполнении одних команд на различных данных. В основе большинства известных технологий реконструкции ключей криптографических алгоритмов с использованием АДПМ лежит модель утечки, называемая ”моделью Хеммингового веса” (Hamming weight model) [1]. Согласно этой модели, потребляемая мощность S (сигнал утечки) в момент обработки кода D определяется следующим образом:

$$S = \delta + \beta \cdot H(D) + \eta \qquad (1)$$

где δ – постоянное смещение, β - действительное число, $H(X)$ – Хеммингов вес кода X, η – шум с математическим ожиданием 0 и среднеквадратичным отклонением σ.

Согласно этой модели потребление мощности пропорционально числу единиц в обрабатываемых данных (Хэммингов вес).

Приведенная модель - наиболее простая. существуют и другие модели описания зависимости мгновенного значения потребляемой мощности от обрабатываемого кода. Так на практике часто используется “модель дифференциала Хеммингова веса”, согласно которой мгновенное значение потребляемой мощности пропорционально числу изменяемых при обработке разрядов кода. Ясно, что степень адекватности той или иной модели существенным образом зависит от используемой при изготовлении микроконтроллера интегральной технологии. Поэтому оправданным представляется создание модели утечки для каждого конкретного типа микроконтроллера.

На сегодняшний день существует ряд методов использования АДПМ для реконструкции ключей криптографических алгоритмов. Эти методы разделяются на две группы [3]:

- простой АДПМ (SPA- Simple Power Analysis) и его развитие - профильный АДПМ (Profiling Analysis) основаны не непосредственном сопоставлении динамики потребления мощности с командами обрабатываемого криптографического алгоритма; в большинстве случаев простой АПМД позволяет установить последовательность выполнения команд и если она зависит от битов ключевого кода – восстановить его.

- дифференциальный АДПМ или Д-АДПМ (DPA - Differential Power Analysis) и его развитие Д-АДПМ высоких порядков (HO-DPA - High Order Differential Power Analysis), основанные на статистическом анализе данных о динамике потребления мощности для ряда вычислений алгоритма при меняющихся данных и постоянном ключе.

Для противодействия методам реконструкции ключей криптографических алгоритмов с помощью АДПМ используют программные и аппаратные способы. Суть аппаратных способов заключается в том, чтобы сделать неотличимой разницу в потребляемой мощности для различных данных путем использования специальных схемных решений, таких как балансные схемы или шумящие схемные компоненты [2].

Существующие к настоящему времени программные средства противодействия ре-

конструкции ключей АДПМ-технологиями основаны на использовании случайных и псевдослучайных элементов при реализации алгоритмов криптографической защиты. Базовыми критериями эффективности средств противодействия АДПМ-технологиями являются достигаемый ими уровень защищенности и объем дополнительно затрачиваемых вычислительных ресурсов. Последнее особенно важно с учетом того, что вычисления производятся на малоразрядных микроконтроллерах и смарт-картах, обладающих низкой производительностью и ограниченными ресурсами памяти. Для оценки достигаемого уровня защищенности от АДПМ-технологий в большинстве случаев используют упомянутые выше модели зависимости потребляемой мощности от обрабатываемого кода.

Основными направлениями при создании программных средств противодействия АДПМ-технологиям являются:

- маскирование – наложение на обрабатываемые данные и ключи случайных компонент перед обработкой и выполнение специальных вычислений для получения истинного результата – “снятие маски” после обработки. Маскирование применяют, преимущественно, для защиты ключей симметричных блоковых алгоритмов, таких как DES, Rijndael, ГОСТ 28.147-89 [5], в которых используются относительно простые операции. Существенным недостатком маскирования является возможность исключения влияния случайных масок путем специальной статистической обработки;
- случайное изменение порядка выполнения независимых по данным команд алгоритма (полиморфная реализация), что затрудняет сопоставление участков программы диаграмме потребляемой мощности. Полиморфная реализация алгоритма является эффективным средством противодействия простому АДПМ и наиболее часто используется для защиты ключей алгоритмов на основе мультипликативных операций модулярной арифметики – RSA, DSA, El-Gamal. Недостатком полиморфной реализации алгоритмов является необходимость значительных вычислительных ресурсах, требующихся для анализа зависимости по данным команд и реализации случайного их выполнения. Это существенно замедляет реализацию алгоритмов криптографической защиты данных на микроконтроллерах;
- разделение ключей, используемых в алгоритмах на несколько частей случайным образом с последующей раздельной их обработкой и компоновкой результата. Подход эффективен только для алгоритмов типа DES, IDEA, в которых используются только логические операции. Недостатком является также необходимость не менее, чем двукратном увеличении времени реализации криптографического алгоритма.
- случайное перемешивание выполнения операций над реальными и фиктивными данными; действенность подхода зависит от эффективности средств генерации фиктивных команд и данных ими обрабатываемых;
- использование сбалансированных кодов с постоянным числом единиц.

Указанные подходы могут использоваться как отдельно, так и комбинированно.

Эффективность этих способов в значительной степени зависит от вида АДПМ в используемом методе реконструкции ключей. Так, например, маскирование неэффективно против Д-АДПМ высоких порядков, если порядок анализа выше числа масок. Для обхода перемешивания используются интегральные модификации различных видов АДПМ. Общим недостатком рассматриваемых программных способов противодействия АДПМ-методам является снижение производительности криптографического алгоритма пропорционально числу масок при маскировании, либо фиктивных переменных при перемешивании.

Проведенный анализ показал, что наиболее действенным, с точки зрения достигаемого уровня защищенности от АДПМ, является динамичная модификация программной реализации криптографического алгоритма во время выполнения (полиморфизм кода). При этом последовательность команд для реализации криптографического алгоритма каждый раз будет различной, как и динамика потребления мощности. Это позволяет противодействовать как профильному АДПМ, так и различным Д-АДПМ (в том числе высоких порядков). При этом, эффективность применения такого механизма определяется общим числом возможных модификаций ко-

да и величиной разницы временной локализации этапов обработки данных для различных модификаций.

Эффективность полиморфной реализации существенно ограничивается зависимостью команд по данных в большинстве криптографических алгоритмов, а также от степени ветвления алгоритма. Очевидно, что наиболее эффективно использование полиморфизма для реализации маловетвящихся алгоритмов, к числу которых относятся блоковые алгоритмы шифрования, такие как ГОСТ 28.147-89, DES, IDEA, Rijndael.

Целью исследований является повышение эффективности защиты от АДПМ-технологий в рамках полиморфной реалиизации криптографических алгоритмов блочного типа за счет увеличения временного диапазона выполнения его операций.

Способ организации псевдопараллельной обработки блоков в криптографических алгоритмах

В качестве способа организации полиморфного исполнения для блочных криптографических алгоритмов предлагается использовать одновременную псевдопараллельную, в режиме разделения времени, обработку нескольких блоков данных.

Блочные алгоритмы состоят из k циклов обработки данных $R_i(D,K_i)$, $i=1..k$, в каждом из которых производится некоторое преобразование блока D данных с участием ключа K_i. В зависимости от типа блочного шифра цикл обработки может рассматриваться как некоторая макрооперация для сети Фейстеля либо последовательность макроопераций для SP-сети. В любом случае для преобразования блока данных выполняется конечная последовательность P таких макроопераций, длина K которой равна (сеть Фейстеля) либо кратна (SP-сеть) числу k циклов обработки данных.

Для противодействия АДПМ необходимо реализовать полиморфное исполнение алгоритма, при котором указанная последовательность макроопераций будет меняться при каждом выполнении алгоритма. Предлагается использовать помирфизм на двух уровнях: на уровне макроопераций и на уровне цикла обработки данных (псевдопараллельная обработка алгоритмом нескольких блоков данных).

Полиморфизм макроопераций блоковых алгоритмов шифрования основан на том, что длина блока существенно превышает разрядность обрабатываемого фрагмента. При этом, каждый из фрагментов может обрабатываться независимо. Так, в большинстве блочных алгоритмов нелинейное преобразование выполняется с использованием ряда табличных преобразователей (S-блоков), каждый из который преобразует относительно небольшое число разрядов кода. Так в DES используется 16 S-блоков, каждый из которых независимо обрабатывает 4 разряда. В силу наличия независимо обрабатываемых фрагментов существует возможность менять очередность их обработки, то есть организовать полиморфную обработку фрагментов. Выбор очередности обработки фрагментов может выполняться статически или динамически. В первом случае предварительно формируются и сохраняются в памяти ряд вариантов программной реализации цикла преобразования, отличающихся последовательностью обработки фрагментов. Выбор варианта выполняется случайно в начале каждого выполнения цикла криптографического преобразования. При динамическом полиморфизме программные фрагменты-заготовки имеют меньшую длину (вплоть до команды) и их выбор осуществляется непосредственно во время выполнения цикла преобразования

Полиморфная, на уровне макроопераций, реализация криптографического алгоритма потенциально уязвима для интегральной разновидности АДПМ в которой интервал интегрирования равен длине макрооперации. Следовательно для нейтрализации такого вида анализа необходимо обеспечить различную временную локализацию макроопераций из P на разных циклах обработки блоков данных — полиморфизм уровня цикла обработки данных. Для этого предлагается использовать псевдопараллельную организацию обработки нескольких блоков данных в криптографическом алгоритме.

Возможность такой организации выполнения блоковых алгоритмов основана на том, что обработка блоков шифруемого (дешифруемого) сообщения выполняется независимо. Соответственно, сущность полиморфизма на уровне блоков сообщения состоит в том, что организуется одновременно-поочередная обработка нескольких блоков

сообщения, причем выбор длина программного фрагмента обработки блока выбирается случайным образом. Псевдопараллельная организация подразумевает одновременную обработку m блоков данных на меньшем числе вычислительных устройств (ВУ) - чаще всего на одном в режиме разделения времени. При условии, что ВУ только одно, такая организация вычислений не приводит к повышению производительности (традиционная цель распараллеливания), однако позволит смешивать выполнение m последовательностей P_j, $j=1..m$ обработки различных блоков данных D_j, $j=1..m$.

При псевдопараллельном, в режиме разделения времени, исполнении последовательностей P_j, $j=1..m$ происходит их разделение на q подпоследовательностей $P_{j,l}$, $l=1..q$ различной длины, которые могут чередоваться различными способами с сохранением порядка расположения в P_j. Пример такого псевдопараллельного выполнения блоков P_1, P_2 и P_3 графически показан на рисунке 1. При полиморфном выполнении криптографического алгоритма разбиение на подпоследовательности и их чередование осуществляется стохастически.

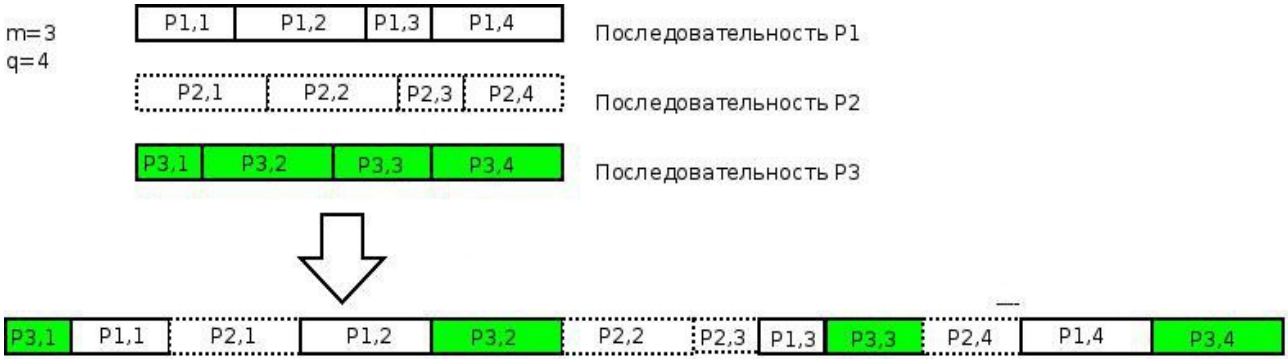


Рис.1. Пример чередования при выполнении трех последовательностей

Полиморфизм псевдопараллельной реализации обработки m блоков данных обеспечивается на этапах:

- случайного выбора длин q подпоследовательностей $P_{j,l}$ $l=1..q$ - при этом по случайному закону выбираются (q-1) длин подпоследовательностей, а последняя длина является дополнением суммы остальных длин до K;
- стохастического планирования выполнения подпоследовательностей $P_{j,l}j=1..m,l=1..q$ - во время работы алгоритма. Из m готовых к выполнению подпоследовательностей случайно выбирается одна, по окончании её исполнения ситуация повторяется; в конце обработки число готовых $P_{j,l}$ сокращается за счет завершения обработки все большего числа блоков данных; признаком окончания обработки всех m блоков данных является отсутствие готовых к выполнению подпоследовательностей.

Следует отметить, что оба этапа могут быть как статическими (выполняться перед обработкой данных) так и динамическими (во время обработки данных).

Для реализации псевдопараллельной обработки с разделением по времени необходимо обеспечить механизм сохранения/восстановления состояния ВУ при переключении между P_j. Реализация такого механизма требует дополнительных ресурсов, что снижает производительность алгоритма на величину пропорциональную числу переключений. Значительных временных затрат требует также программная генерация псевдослучайного кода. Исходя из этого, общее время t^m_{par} псевдопараллельной обработки m блоков данных определяется формулой :

$$t^m_{par} = m \cdot t^l + (m \cdot q - 1) \cdot (t_{state} + t_g) \qquad (2)$$

где t^l - время последовательной обработки одного блока данных, t_{state} – время, затрачиваемое на сохранение/восстановление состояния при переключении, t_g -время программной генерации псевдослучайного числа.

Таким образом, для уменьшения негативного влияния предложенной псевдопараллельной реализации на производительность необходимо использовать малые значения q и минимизировать объем данных при сохранении состояния. Это достигается за счет рационального выбора размеров макроопераций и их атомарного выполнения — переключение возможно только между макрооперациями.

Вполне очевидно, что максимальное значение вариации Δt_m времени выполнения команды определяется формулой:

$$\Delta t_m = (m-1)\cdot t_{par}^{m-1} + \frac{(q-k)\cdot t^1}{q\cdot k} \qquad (3)$$

Среднее значение времени Δt_a вариации момента выполнения команды вычисляется следующим образом:

$$\Delta t_a = \frac{m\cdot t^1}{k}\cdot(\frac{q}{k}+1) \qquad (4)$$

Анализ формулы (3) показывает, что среднее время вариации момента выполнения команды при использовании предлагаемого способа выполнения криптографических алгоритмов увеличивается в m раз.

Важным фактором производительности полиморфной реализации алгоритмов криптографической защиты данных является способ получения случайных чисел. Учитывая технологические сложности измерения физически случайных величин, для большинства применений терминальных микроконтроллеров в качестве источника случайных чисел можно использовать встроенный таймер вместо случайных чисел применять превдослучайные числа.

Полиморфная реализация алгоритма ГОСТ 28.147-89

Предложенный способ организации псевдопараллельной обработки конкретизирован для модификации реализации сертифицированного на Украине алгоритма ГОСТ 28.147-89. В качестве базовой использована реализация указанного алгоритма блочного шифрования для 32-х битных процессоров архитектуры Intel x86, разработанная Винокуровым А.Ю.[4].

Объектом модификации является макрооперация — цикл обработки данных алгоритма ГОСТ 28.147-89, реализованный в процедуре _gost32. В этой процедуре элементами наиболее уязвимыми для анализа динамики потребления мощности являются 4 операции табличного преобразования (xlat) после наложения ключа на данные. В этих операциях данные, содержащие часть ключевой информации выставляются на шину, для обращения к памяти. Остальные операции (кроме наложения ключа) регистровые и потребление мощности при их выполнении существенно меньше, чем при работе с памятью. Следовательно, основным элементом при получении множества эквивалентных реализаций процедуры _gost32 должны быть именно эти операции. Поскольку операции табличного преобразования независимы по данным, порядок их выполнения может быть произвольным, что позволило получить 24 варианта реализации процедуры _gost32 на каждом из 32-х циклов.

Для псевдопараллельной обработки в режиме разделения времени использовались следующие параметры m=2, q=4·32=128. При этом вместо случайного планирования выполнения подпоследовательностей реализовано циклическое переключение со случайным выбором стартовой подпоследовательности. В процедуре _gost32 единственный свободный регистр EDI использован для хранения длин подпоследовательностей $P_{j,l}$ $l=1..q=4,j=1..2$. Поскольку общее число подпоследовательностей равно 8, то на хранение длины отводится 4 бита, что позволяет хранить значение до 15 (из 32 возможных). Объем данных которые необходимо сохранить при переключении последовательностей достаточно мал — текущий блок данных (EDX:EAX) и указатель на текущее смещение в развернутом ключе (ESI). Ввиду отсутствия свободных регистров сохранение ведется в стек. Таким образом, для сохранения и восстановления состояния при переключении достаточно 4 и 6 команд соответственно. Значение t_{state} для произведенной реализации равно:

$$t_{state} = 3\cdot t_{PUSH} + 3\cdot t_{POP} + t_{SUB} + t_{MOV} + \\ + t_{AND} + t_{JMP} = 10\cdot t_C \qquad (5)$$

где t_{PUSH} , t_{POP} , t_{SUB}, t_{MOV} , t_{AND} , t_{JMP} - время выполнения команд PUSH, POP, SUB, MOV, AND и JMP соответственно, а t_C - время выполнения цикла процессора Intel Pentium.

Для реализации псевдослучайного генератора случайных чисел, используемого для выбора обрабатываемого цикла и подпоследовательности в нем, наиболее целесообразным с точки зрения времени реализации представляется использование сдвигового регистра с линейной функцией обратной связи (LFSR-Linear Feedback Shift Register), выбираемой случайно из некоторого наперед заданного набора. Для генерации псевдослучайного числа требуется: вычислить значение линейной функции обратной связи (команда AND-для наложения маски обратной

связи, 3 команды MOV и команда сдвига для копирования флага четности в флаг переноса) и сдвинуть регистр (одна команда ROL) – всего 6 команд.

При этом полный цикл обработки блока данных в реализации процедуры _gost32 равен: $t^1 = 1408t_C$. Численное значение времени t_{par}^2 выполнения 2-х блоков, вычисленное по формуле (2) составляет: $t_{par}^2 = 2 \cdot 1408t_C + 127 \cdot (10+6) \cdot t_C = = 4848 \cdot t_C$.

Таким образом, при псевдопараллельной обработке блоков в алгоритме ГОСТ 28.147-89 соотношение времени выполнения пары блоков - t_{par}^2 к времени $2 \cdot t^1$ их последовательного выполнения равно: $t_{par}^2/(2 \cdot t^1) = 4848/2816 = 1.721$.

Полученное значение подтверждается результатами экспериментальных исследований для m=2, q=128. При этом, определяющим фактором уменьшения производительности реализации криптографического алгоритма является величина q.

Выводы

Методы реконструкции секретных ключей криптографических алгоритмов посредством анализа динамики потребляемой мощности (АДПМ) представляют на сегодняшний день одну из наиболее серьезных угроз для средств защиты информации с одним исполняющим блоком (смарт-карты, встраиваемые системы и др.). Аппаратные способы противодействия АДПМ-технологиям требуют дорогостоящей специализированных средств. Исполь-зование традиционных программных средств противодействия имеет следствием деградацию производительности алгоритмов и малоэффективно против некоторых видов атак (например Д-АДПМ высоких порядков).

В качестве эффективной контрмеры АДПМ-технологиям несанкционированного доступа к ключам криптографических алгоритмов предложен способ их полиморфной реализации. Основной эффект противодействия достигается за счет того, что каждый раз криптографическое преобразование производится с помощью различной последовательности команд. Это препятствует сбору корректных образцов-профилей для профильного АДПМ и использованию статистических методов усреднения в различных Д-АДПМ. Отличительной особенностью предложенного способа является организация пседопараллельной криптографической обработки m блоков данных, что обеспечивает в m раз увеличение среднего значения вариации момента времени выполнения каждой команды алгоритма. Это значительно снижает эффективность существующих технологий реконструкции ключа с помощью анализа динамики потребляемой мощности.

Способ может быть эффективно использован для защиты ключей алгоритмов как симметричного шифрования: DES, Rijndael, ГОСТ 28.147-89, так и несимметричного типа RSA.

Предложенный способ конкретизирован в виде программной реализации стандартизированного в Украине алгоритма шифрования ГОСТ 28.147-89. Экспериментально доказано увеличение роста вариации момента времени выполнения команд алгоритма при снижении производительности в 1.7 раз, что меньше деградации производительности при использовании таких альтернативных способов противодействия АДПМ-технологиям, как маскирование и чередование [6].

Предложенный способ и его реализация для алгоритма ГОСТ 21147-89 могут быть использованы для повышения эффективности защиты информации на смарт-картах и встроенных терминальных устройствах компьютерных сетей.

Список литературы

1. Rivain M., Prouff E., Doget J. Higher-order masking and shuffling for software implementations of block ciphers // Cryptographic Hardware and Embedded Systems - CHES 2009. of Lecture Notes in Computer Science. - Springer, Heidelberg.- 2009.- Vol. 5747. - P. 171-188.
2. Kocher P., Jafft J., Jub B. Differential power analysis // Advances in Cryptology - CRYPTO '99 Lecture Notes in Computer Science. - Springer, 1999. - Vol. 1666 . - P. 388—397.
3. Chari S., Jutla C., Rao J., Rohatgi P. Towards Sound Approaches to Counteract Power-Analysis Attacks // Advances in Cryptology – CRYPTO ’99 Lecture Notes in Computer Science. - Springer, 1999. - Vol. 1666 . - P. 398–412.
4. Винокуров А. Алгоритм шифрования ГОСТ 28.147-89, его использование и реализация для компьютеров платформы Intel x86 [Электронный ресурс]. - Работа на правах рукописи, режим доступа: http://www.enlight.ru/crypto/articles/vinokurov/gost_i.htm
5. Марковский А.П., Абабне О.А., Ияд Мохд Маджид Ахмад Шахрури. Способ защиты ключей алгоритма ГОСТ 28.147-89 от реконструкции анализом динамики потребляемой мощности // Вісник національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. – 2007.- № 46.- С.128-138.

Поступила в редакцию 15.10.2009

УДК 004.724.4(045)

КУЛАКОВ А.Ю.
КЛИМЕНКО И.А.

СПОСОБ ФОРМИРОВАНИЯ СТРУКТУРЫ ВИРТУАЛЬНОЙ GRID СИСТЕМЫ

В статье рассмотрены вопросы формирования структуры виртуальной динамической GRID системы. Виртуальная GRID система строиться на основе минимального покрывающего дерева. Проведен сравнительный анализ предложенного алгоритма формирования минимального покрывающего дерева с известными алгоритмами. Предложен алгоритм построения покрывающего дерева для виртуальной динамической GRID.

The article deals with issues of structure formation of dynamic virtual GRID system. Virtual GRID system is built on the basis of the minimum spanning tree. A comparative analysis of the proposed algorithm for generating the minimum spanning tree with the known algorithms is shown. An algorithm for constructing spanning tree for the virtual dynamic GRID is offered.

Введение

GRID-система [1] представляет собой аппаратно-программную инфраструктуру, которая обеспечивает надежный, устойчивый, повсеместный и недорогой доступ к высокопроизводительным компьютерным ресурсам. В качестве вычислительной среды *GRID*-системы (*CFG – Computing Fabrics GRID*) используется совокупность ресурсов различного типа вычислительных систем и компьютерных сетей. *CFG* представляет собой виртуальную среду с возможностью динамической реконфигурации, что позволяет при необходимости обеспечить максимальную вычислительную мощность для решения наиболее сложных и ресурсоёмких задач [2].

Как правило, в рамках *Grid-* систем объединяются между собой ресурсы различных компьютеров, которые зачастую входят в состав разных компьютерных сетей и управляются разными администраторами. Использование различных средств защиты, таких как брандмауэр, частный IP адрес, безопасная оболочка SSH требует специальной настройки и эксплуатации таких систем, которая осуществляется с участием администратора. Например, чтобы добраться до хоста за брандмауэром, сначала требуется войти на шлюз, а за тем уже на узел назначения. Другой пример, получая доступ к DHCP, клиент должен иметь некоторую базу данных для хранения его адреса. Ситуация может быть запутанной, когда эти адреса являются локальными IP адресами. В этом случае указание работ часто становится громоздким процессом. При небольшом количестве компьютеров эта работа может быть осуществлена вручную администратором. Пользователь запоминает промежуточные шлюзы к каждому хосту и хранит некоторую специализированную базу для отслеживания DHCP адресов клиентов. Это, в свою очередь, существенно усложняет процесс организации структуры *CFG* и ее реконфигурацию в процессе функционирования *GRID*-системы. Для решения данной проблемы используют технологию построения виртуальных частных сетей (*VPN*) [3], на основе которой строятся виртуальные частные *Grid-* системы (*VPG*) [4,5].

1.Постановка задачи

На структурном уровне *CFG* может быть представлена в виде неориентированного нагруженного графа $G(V,E,W)$, где V – множество узлов *CFG*; E – множество ребер графа $G(V,E,W)$, которые соответствуют каналам связи *CFG*, $W= \{w_{i,j} \mid i,j=1,2\ldots n\}$ – множество весов ребер графа. Вес ребра характеризует параметры передачи информации по каналам связи. При этом *VPG* представляет собой подграф G_s (V_s, E_s, W_s) графа $G(V, E, W)$, где: $V_s \subseteq V$; $E_s \subseteq E$; $W_s \subseteq W$.

В общем случае $V_s = V$. Как правило, при организации *VPG* исходное множество E ребер графа $G(V,E,W)$ является избыточным. В этом случае задача формирования множества E_s ребер подграфа $G_s(V, E_s, W_s)$ сводится к задаче построения покрывающего дерева, отвечающего заданным критериям оптимальности и обладающего требуемыми показателями отказоустойчивости при изменении параметров и структуры *CFG*. Это связано с

тем, что *CFG*, как правило, состоит из большого количества узлов с динамически изменяемой структурой связей. Большое количество территориально удаленных узлов *CFG* определяет необходимость использования децентрализованного алгоритма построения покрывающего дерева. В свою очередь, динамически изменяемая структура связей *CFG* предопределяет использования так называемых самостабилизирующихся алгоритмов формирования покрывающих деревьев [6,7,8].

Большинство известных алгоритмов построения покрывающего дерева являются различными модификациями алгоритмов Прима или Крускала [9].

Одним из наиболее распространенных алгоритмов формирования покрывающего дерева является алгоритм Прима. Алгоритм состоит из $N-1$ итерации, на каждой из которых к дереву добавляется только одна вершина, которая не нарушает свойств дерева, то есть один конец ребра принадлежит формируемому дереву, а другой – добавляемой вершине, при этом его вес является минимальным. Следует отметить, что алгоритм Прима отличается постоянной сложностью, то есть отсутствием чувствительности к степени связности графа.

Отличительной особенностью алгоритма Крускала является возможность строить дерево одновременно для нескольких поддеревьев, рассматривая исходный граф в виде леса. В процессе работы алгоритма отдельные поддеревья объединяются в одно покрывающее дерево. Полный граф задается списком ребер. В начале построения дерева список ребер сортируется по возрастанию веса. На каждом шаге просматривается список ребер, начиная с ребра, которое еще не обрабатывалось. К формирующейся компоненте связности присоединяется ребро, которое не образует цикла с ребрами, уже включенными в решение. Перед началом работы алгоритма количество компонент связности равно количеству вершин в графе и содержит соответственно по одной вершине. После завершения работы алгоритма остается только одна компонента связности, а искомое покрывающее дерево определяется теми ребрами которые использовались для объединения всех компонент связности в одну.

Способность алгоритма Крускала формировать дерево одновременно из нескольких поддеревьев является предпосылкой для разработки децентрализованного алгоритма построения покрывающего дерева. Однако в первоначальном виде данный алгоритм является централизованным, так как предполагает ряд операций, таких как просмотр списка ребер.

В свою очередь, алгоритм формирования минимального покрывающего дерева, ориентированный на динамические среды, в частности на *CFG*, должен обладать следующими свойствами:

- каждый узел сети должен строит асинхронно покрывающее дерево, не зная топологии всей сети;
- алгоритм должен быть устойчивым, то есть формировать покрывающее дерево связей даже в условиях часто изменяющейся топологии сети;
- алгоритм должен осуществлять реконфигурацию покрывающего дерева с минимальной временной сложностью.

С целью обеспечения максимальной устойчивости покрывающего дерева и минимальной временной сложности его реконфигурации в данной работе предлагается формировать покрывающее дерево относительно вершин с максимальной степенью.

2.Алгоритм построения и динамической реконфигурации покрывающего дерева *CFG*

По аналогии с алгоритмом Крускала на начальном этапе граф *CFG* рассматривается, как остовной лес в котором каждая вершина является корнем дерева. В процессе работы алгоритма вершины постепенно объединяются в деревья, в результате формируется одно покрывающее дерево.

Рассмотрим данный алгоритм более подробно. Каждая вершина v_i характеризуется своей степенью D_i, которая определяется числом смежных ребер данной вершины, и своим текущим приоритетом P_i, который соответствует приоритету корневой вершины поддерева ST_j (V_j, E_j, W_j) в которое входит вершина v_i. В качестве корневой вершины v^0_m поддерева ST_m (V_m, E_m, W_m) будем принимать вершину v_k, обладающую наибольшей степенью среди всех вершин v_i поддерева ST_m (V_m, E_m, W_m),

то есть: $v^0_j = \{ v_k \mid D_k \geq D_i \; \forall \; v_i \in V_m \}$.

В начале формирования покрывающего дерева текущий приоритет P_i каждой из вершин $v_i \in V$ равен ее степени D_i. На первом шага формирования минимального покрывающего дерева с помощью ребер $e_{i,j} \in E$ вершина v_i соединяется с смежной вершиной v_j с максимальной степенью D_j (рис.1). При этом текущий приоритет P_i вершины v_i становится равным степени D_j вершины v_j.

Таким образом, формируется некоторое множество поддеревьев $\{ST_m(V_m,E_m,W_m) | m=1,2..l;\ l<n\}$. На рис.1 представлен первый шаг формирования покрывающего дерева. Пунктирными линиями обозначены ребра исходного графа, а сплошными линиями – ребра сформированных поддеревьев.

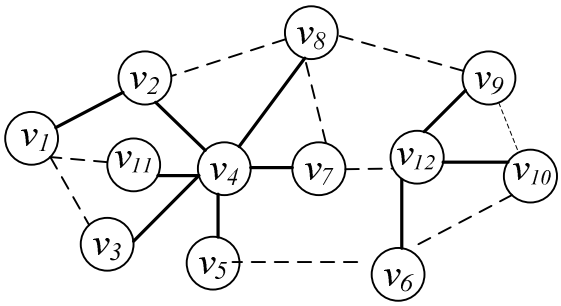


Рис. 1. Первый шаг формирования покрывающего дерева

На втором и последующих шагах формирования покрывающего дерева аналогичным образом происходит объединение между собой отдельных поддеревьев множества $\{ST_m(V_m,E_m,W_m) | m=1,2..l;\ l<n\}$. Каждый узел асинхронно опрашивает своих соседей, которые определены топологией сети, если оказывается, что узлы относятся к разным деревьям, эти деревья объединяются (рис.2).

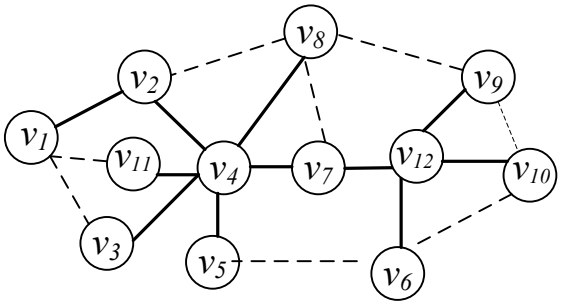


Рис. 2. Результирующее покрывающее дерево

Изменение топологии *CFG* приводит к изменению степени отдельных вершин подграфа $G_s(V, E_s, W_s)$, в результате чего формируется новое покрывающее дерево. В качестве примера на рис.3 представлено покрывающее дерево при удалении ребра e_{47}.

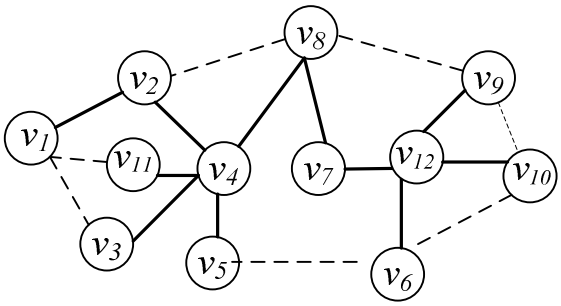


Рис. 3. Покрывающее дерево в результате удаления ребра $e_{i,j}$.

3.Анализ алгоритмов формирования покрывающих деревьев

С целью анализа временных характеристик предложенного алгоритма и сравнения его с известными алгоритмов были рассмотрены слабосвязанные графы с различным количеством узлов, которые наиболее характерны для структуры среды *CFG*. На рис 4. представлены полученные зависимости времени формирования дерева от количества узлов сети для рассмотренных алгоритмов .

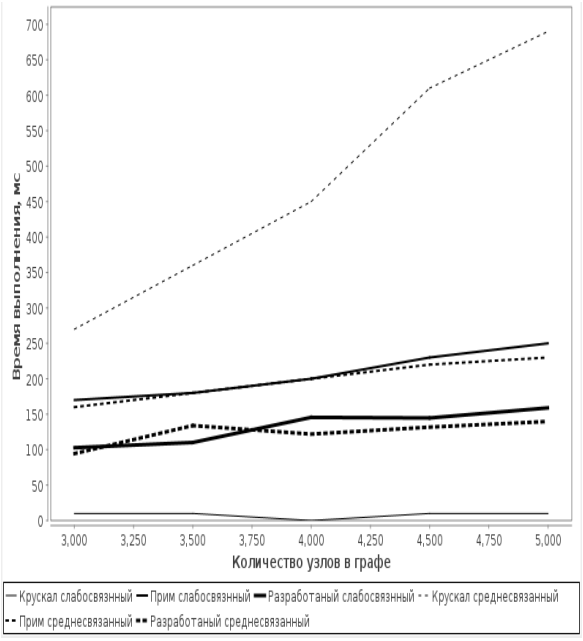


Рис. 4. Результаты моделирования алгоритмов построения дерева связей

Как видеть из графиков, переведенных на рис.4, предложенный алгоритм по сравнению с известными алгоритмами Прима и Крускала обладает меньшим временем формирования покрывающего дерева, что позволяет говорить о предпочтительном его использовании в динамически реконфигурируемой среде, к которой и относится *CFG*.

Кроме временных характеристик, согласно поставленной задаче, при сравнении алгоритмов учитывается их способность к параллельному и децентрализованному выполнению. Эта возможность реализована благодаря тому, что каждый узел асинхронно опрашивает только своих соседей, в то время как алгоритмы Прима и Крускала реализуют централизованное построение дерева связей, начиная от корня дерева.

Способность данного алгоритма к динамической реконфигурации является наиболее важной характеристикой в динамически изменяемой среде. Это свойство алгоритма связано с тем, что он функционирует полностью децентрализовано, адаптируясь к изменяющейся конфигурации *CFG*, не требует информации о топологии сети, обеспечивая тем самым высокую производительность и надежность.

Выводы

В работе предложен новый алгоритм децентрализованный алгоритм формирования покрывающего дерева, предназначенный для формирования виртуальной динамической структуры *Grid* – системы. На основе предложенного алгоритма осуществляется прозрачное распределение задач виртуальной динамической *Grid.* Предложенный алгоритм отвечает необходимым требованиям динамической *Grid* среды, обладает свойствами децентрализованного выполнения, что увеличивает надежность, а так же возможность функционирования в динамически изменяемой среде.

Список литературы

1. I. Foster, C. Kesselman. GRID: A Blueprint to the New Computing Infrastructure. Morgan Kaufman Publishers, 1999.

2. I. Foster, C. Kesselman and S.Tuecke The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International J .Supercomputer Applications 15(3)., 2001

3. C. Scott, P. Wolfe, and M. Erwin. Virtual Private Networks, 2nd Edition. O'Reilly, Sebastopol, CA, 1998.

4. Tsugawa, M. and J. A. B. Fortes, "A Virtual Network (ViNe) Architecture for Grid Computing," *20th Intl Parallel and Distributed Processing Symposium (IPDPS-2006)*, 04/06

5. S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu. From Virtualized Resources to Virtual Computing Grids: The In-VIGO System. Future Generation Computer Systems, 21(6):896-909, Jun. 2005.

6. Felix C. G¨artner and Henning Pagnia. Time-efficient self-stabilizing algorithms through hierarchical structures. In Proceedings of the 6th Symposium on Self- Stabilizing Systems, Lecture Notes in Computer Science, San Francisco, June 2003.

7. Liang, Z.: Self-Stabilizing Minimum Spanning Trees. Technical Report UofC/TR/TR-2001-688-11, University of Calgary, Department of Computer Science (2001)

8. Felix C. G¨artner and Henning Pagnia. Time-efficient self-stabilizing algorithms through hierarchical structures. In Proceedings of the 6th Symposium on Self- Stabilizing Systems, Lecture Notes in Computer Science, San Francisco, June 2003.

9. C. G´enolini and S. Tixeuil. A lower bound on dynamic k-stabilization in asynchronous systems. In SRDS 2002 21st Symposium on Reliable Distributed Systems, IEEE Computer Society Press, pages 211–221, 2002.

УДК 004.77

КОВАЛЬЧУК О.В.

МЕТОДИ ТА ЗАСОБИ ПЕРЕДБАЧЕННЯ ПЕРЕВАНТАЖЕННЯ КОМПЮТЕРНИХ МЕРЕЖ

Одним і з важливих аспектів підвищення продуктивності роботи комп'ютерної мережі є забезпечення її стабільної роботи. Пропонуються засоби для передбачення перевантаження комп'ютерних мереж, які дозволяють на основі аналізу попереднього завантаження мережі передбачити завантаження мережі у майбутньому та запропонувати засоби уникнення перевантажень.

One of important aspects of work productivity computer network increase is ensuring its stable functioning. means for prevision of overloading computer networks, which permit to forsight the loading of network in future and propose means for avoiding of overloadings, are proposed.

Вступ

В умовах розвитку сучасного світу постійно збільшується обсяг інформації якою обмінюється світова спільнота. У зв'язку із збільшення обсягу обміну інформації збільшується ризик додаткового навантаження комп'ютерних мереж , що призводить до перевантаження та падіння пропускної здатності мережі майже до нульової, тобто колапсу мережі, що в свою чергу призводить до втрати інформації в комп'ютерних мережах .

Для зниження імовірності виникнення перевантажень та колапсів мережі необхідно виконувати аналіз попередніх рівнів завантаження перевантажень сервера , тобто оцінювати ступінь завантаженості сервера в певний момент часу. Процедура на налізу ризику перевантаження сервера включає в себе: виділення певного проміжку часу, визначення навантаження на сервер в цей проміжок часу, порівняння навантажень на сервер в цей же час через певний період(день, тиждень, місяць, рік), на основі зміни ступеня навантаження вирахувати імовірність навантаження в той же самий час через період. В процесі аналізу імовірності повтору навантаження сервера формується графік прогнозу навантаження сервера, згідно якого формуються рекомендації по вибору та застосуванню засобів уникнення перевантажень комп'ютерної мережі. Кінцевою метою процедури прогнозування є передбачення перевантажень комп'ютерної мережі.

Загалом прогнозування завантаження та передбачення перевантажень направлене на раціональне використання ресурсів сервера та комп'ютерної мережі в залежності від умов роботи.

Мета статті

Основною задачею є представлення ефективного метода прогнозування перевантажень та розробка системи здатної обробляти інформацію по завантаженню сервера (мережі) і побудови прогнозу виникнення перевантажень сервера (мережі) на основі цього ж метода.

Сучасні методи прогнозування

На сьогоднішній день існує досить багато методів прогнозування трафіку та завантаження сервера.

Серед найбільш популярних та найбільш ефективних методів прогнозування виділяють методи математичної статистики, методи data minig , регресійний аналіз , фрактальний аналіз та методи нейронних мереж.

Методи математичної статистики базуються на побудові графіку завантаження на основі статистичних данних накоплених протягом певного періоду, вирахування приросту та періодичності виникнення перевантажень.

Методи data minig базується на врахуванні параметрів які є зовнішніми чинниками , такими як економічний стан країни чи певного регіону, політична ситуація та інші зовнішні фактори впливу.

Регресійний аналіз - статистичний метод дослідження залежності між залежною змінною та одною або кількома незалежними змінними. Основна мета визначення зв'язку між змінними та визначення характеру цього зв'язку.

Фрактальний аналіз – метод дослідження оснований на аналізі само подібності трафіка

мережі в різних масштабах . незалежно від того що самі фрак тали досить нестабільні вони веж несуть в собі деякі статистичні властивості які піддаються аналізу.

Нейроні мережі – основуються на методах збору статистики та знаходження закономірностей зміни параметрів. Нейронні мереж здатні проводити аналіз та накопичувати певний досвід для найбільш ефективного прогнозування навантаження сервера та мережі.

Порівняльна характеристика методів

В даний час постійно зростає величина можливих втрат інформації з огляду на збільшення кількості користувачів мережі. У зв'язку з цим в комп'ютерних мережах широко використовується прогнозування завантаження мережі та серверів. Оскільки структури мереж бувають досить різними за своїм складом та архітектурою використовуються досить різні методи прогнозування. Досить часто використовуються методи прогнозування які базуються на використанні даних знятих не тільки з мережі а й даних які залежать від впливу зовнішніх факторів, оскільки сучасні мережі залежать багато в чому саме від зовнішніх факторів.

Перевага методів математичної статистики базується на можливості вирахування закономірності виникнення перевантаження сервера на основі статистичних даних знятих протягом певного періоду. Це дозволяє побудувати графік завантаження на основі певних закономірностей, проте метод є не досить ефективним з огляду на те що для накоплення даних для обробки потрібен відчутно довгий проміжок часу і метод не враховує вплив зовнішніх факторів на зміну підраховуваних даних.

Перевага методів data minig базується в можливості прогнозування виникнення перевантажень сервера на основі вирахування трафіку з урахуванням впливу зовнішніх факторів таких як економічний стан країни(регіону), політична ситуація в країні(регіоні), демографічні фактори впливу. В прогнозуванні завантаження сучасних мереж є досить ефективним оскільки завантаження мереж в багатьох випадках залежить саме від вище перерахованих факторів. Даний метод дозволяє отримати більш точний прогноз завантаження ніж методи основані лише на математичній статистиці оскільки використовує окрім статистичної інформації більше корисної інформації із зовнішніх чинників. Недоліком даного методу є потреба в дослідженнях які не пов'язані з комп'ютерними мережами та збором інформації поза межами мережі, що значно ускладнює та збільшує трудоємність використання даного методу в прогнозуванні завантаження комп'ютерних мереж та серверів.

Фрактальний аналіз включає в себе методи статистики які базуються на само подібності трафіку мережі в залежності від масштабу. На даний момент є не надто ефективними оскільки самі мережі відрізняються не тільки масштабом але і своєю структурою. фрактальний аналіз є сенс використовувати для моделювання трафіку в мережах в залежності від їх збільшення, але лише для певної структури, у випадку моделювання трафіку для мереж із різними структурами данні прогнозу будуть хибними .

Нейронні мережі поєднують в собі методи статистики , знаходження закономірностей у зміні параметрів мережі та накопиченні досвіду уникнення небажаних ситуацій. В прогнозуванні завантаження є дуже ефективними оскільки вони здатні переналаштовуватись в залежності від того як змінюється структура самої мережі а також мають додаткову перевагу за рахунок постійного моніторингу стану мережі та завантаження. Завдяки такій сукупності є суттєво ефективнішими зі інші методи, оскільки здатні реагувати набагато швидше. Недоліком нейронних мереж є велика трудоємність обробки даних, що в свою чергу потребує додаткових ресурсів для своєчасного прогнозування завантаження мережі та сервера, а також негативним фактором являється затрата досить великого періоду часу для накопичення прикладів ситуацій та досвіду їх уникнення.

Система передбачення пеервантажень сервера

Пропонується система передбачення перевантажень сервера основана на методах математичної статистики та фрактального аналізу, яка дозволяє оцінювати завантаженість сервера, прогнозувати завантаженість, передбачити перевантаження сервера та пропонувати оптимальні методи їх уникнення.

Розглянемо детальніше п’ять підсистем в запропонованій системі передбачення перевантажень сервера.

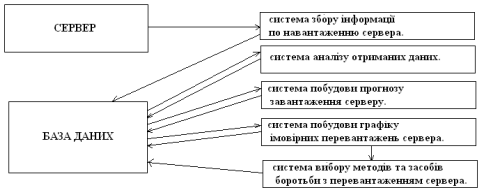


Рис.1. Узагальнена схема системи

1) система збору інформації по навантаженню сервера.

Даний збір інформації виконується методом зняття показів сервера а саме його завантаження протягом відрізків часу приміром в 10 хвилин, обраховується середнє значення завантаження і записується в базу даних відповідно з датою та проміжком часу до якого відноситься значення.

2) система аналізу отриманих даних.

Даний блок системи на основі отриманих даних здійснює порівняння отриманого значення із значенням отриманим певний відрізок часу тому одного і того самого часового проміжку і вираховує коефіцієнт приросту навантаження сервера у відповідний проміжок часу:

$$k_{приросту} = \frac{x_{n+1}}{x_n}$$

дійснюється корекція приросту відповідно того який це день тижня та святкові дні тому що одна і та сама дата припадає на різні дні тижня, корекція відбувається на основі даних по цій даті якщо вона вже припадала на вихідний день, або якщо не припадала то на основі даних знятих по вихідних днях які знаходяться поряд з цією ж датою за тією ж формулою:

3) система побудови прогнозу завантаження серверу.

Даний блок системи дозволяє отримати прогноз завантаження сервера на певний період (година, день, тиждень, рік). Для отримання прогнозу заданого проміжку часу в певний день система обирає отриманні та оброблені данні по цьому ж проміжку в минулому з урахуванням коефіцієнтів. Для підрахування прогноз коефіцієнта збільшення навантаження використовується наступна формула:

$$k_{прогноз} = \frac{\sum_{n=1}^{i} \frac{x_{n+1}}{x_n}}{n-1}$$

Також вираховується ступінь завантаження сервера, для чого обриється останній отриманий ступінь завантаження сервера для відповідного періоду та множиться на прогнозований коефіцієнт завантаження:

$$T_{прогноз} = T_{попередній} \times k_{прогноз}$$

Система на основі отриманих даних будує графік попереднього завантаження сервера та накладає на нього графік прогнозованого завантаження.

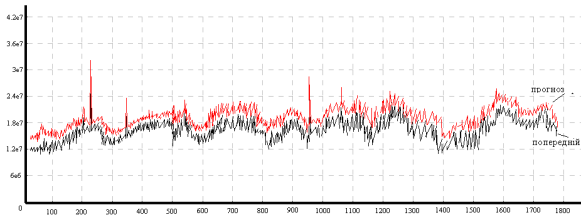


Рис.2. Графік прогнозованого завантаження сервера.
Попередній – графік попереднього завантаження сервера
Прогноз – графік прогнозованого завантаження сервера

Попередній та прогнозований графіки завантажень відрізнятимуться завжди, оскільки трафік сам по собі мінливий і може лише приблизно повторюватися відповідно до закономірностей. Загалом Прогнозований графік буде завжди знаходитись трохи вище по відношенню до попереднього завантаження по осі завантаження оскільки обсяг самого трафіку постійно зростає.

4) система побудови графіку імовірних перевантажень сервера.

Даний блок фактично і є системою передбачення перевантажень сервера.

Для передбачення перевантаження в систему вводяться допустимі параметри при яких сервер працює в нормальному режимі, тобто допустиме навантаження на сервер. Ці параметри порівнюються з графіком навантаження на сервер. Сам графік завантаження усереднюється з метою уникнення досить частого оповіщення про зміну завантаження. Усереднення здійснюється за формулою:

$$y_{усереднене} = \frac{\sum_{n}^{i} x_n}{i-n}$$

де x – це значення завантаження в певний момент часу, n та I - відповідно початок та кінець періоду усереднення.

Та побудова відповідного графіку з накладанням попереднього графіку:

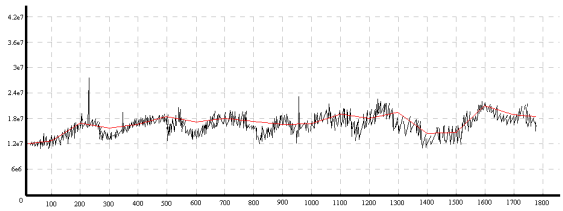


Рис.3. Усереднений графік навантаження сервера.

При виявленні в графіку завантаження в певний момент часу або протягом певного періоду значень які більші за допустимі параметри відмічається в базі даних час імовірного перевантаження та будується графік з відмітками перевантаження та його періоду:

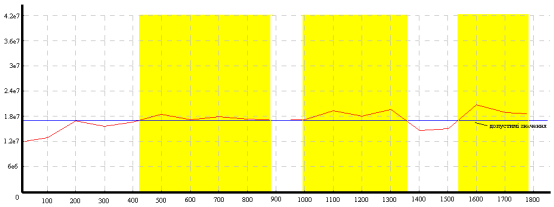


Рис.4. Графік перевантажень навантаження сервера.

Відповідно до кількості часу потрібного для реагування система завчасно сповіщає про загрозу перевантаження серверу.

5) система вибору методів та засобів боротьби з перевантаженням сервера.

Даний блок дозволяє обрати ефективні засоби та методи запобігання перевантаженню. Для вибору методів запобігання перевантажень сервера система опирається на рівень перевантаження який встановлюється у відповідності до можливостей сервера.

Згідно отриманого рівня перевантаження система пропонує най оптимальніший метод уникнення перевантаження сервера та мережі.

Головна задача засобів передбачення перевантажень сервера заключається в мінімізації імовірності непередбаченого перевантаження сервера (мережі).

Система класифікує п'ять рівнів перевантаження сервера:

- Миттеві перевантаження виникають під час одночасного звертання багатьох користувачів і так само швидко згасають. Данний клас перевантажень сервера виникає хаотично і не піддається прогнозуванню, так само данний клас первантажень призводить до мінімальних втрат інформації .

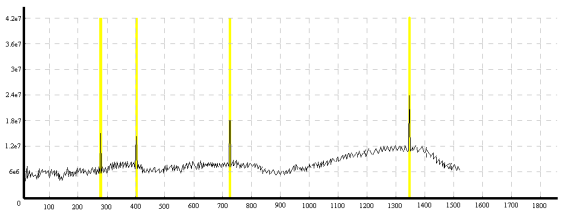


Рис.5. Миттєві перевантаження сервера.

- періодичні перевантаження сервера загалом виникають в певний час і тривають протягом відносно невеликого періоду часу. Такі перевантаження сервера призводять до відчутного зниження ефективності роботи сервера. Періодичні перевантаження піддаються прогнозуванню оскільки вони мають свою закономірність виникнення та тривалості.

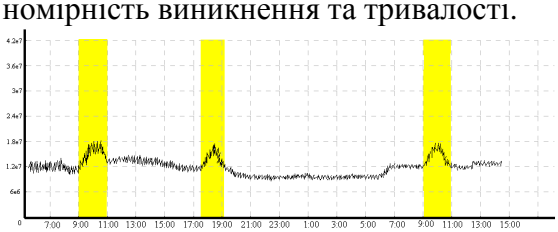


Рис.6. Періодичні перевантаження сервера.

- тривалі перевантаження сервера повязані з навантаженням на сервер протягом відносно довгого періоду часу. Такі перевантаження призводять до досить відчутного зниження ефективності сервера та повної зупинки його роботи. Тривалі перевантаження також як і періодичні перевантаження піддаються прогнозуванню оскільки також мають свою закономірність виникнення і тривалості.

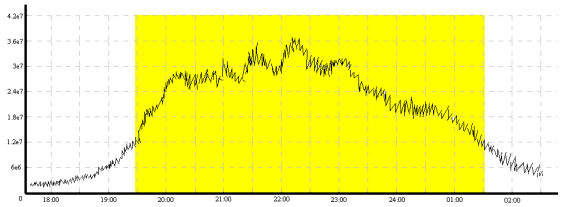


Рис.7. Тривалі перевантаження сервера.

- перевантаження повязані із виникненням екстремальних ситуацій в більшості залежать від впливу зовнішніх факторів які в свою чергу відносно важко спрогнозувати оскільки ці фактори мають швидше хаотичну природу аніж певну закономірність свого виникнення. Дані перевантаження не мають сталої закономірності, передбачення їх ґрунтується на вирахуванні імовірності виникнення зовнішніх факторів та міри їхнього впливу на ступінь завантаження сервера.

- перевантаження пов'язані із збільшенням користувачів мережі. У зв'язку з постійним зростанням кількості користувачів мережі та

постійним зростанням кількості інформації якою обмінюються в мережі навантаження на сервер має стале зростання. Вирахувати таке зростання навантаження можна з статистичних даних знятих з сервера, але для вирахування більш точного прогнозу завантаження система має потребу в вирахування коефіцієнту впливу зовнішніх факторів на навантаження сервера.

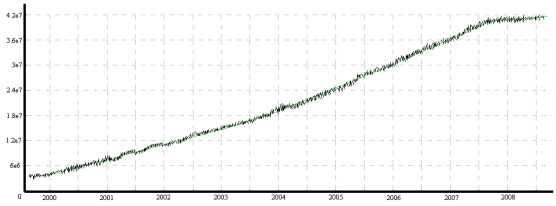


Рис.8. Перевантаження пов'язані із зростанням кількості користувачів

В залежності від рівня перевантаження система може за пропонувати використання різних засоби уникнення перевантаження. Це можуть бути як програмні так і апаратні засоби. До програмних засобів відносяться різні алгоритми передбачені в мережних протоколах. До Апаратних засобів відноситься застосування додаткового обладнання, а при постійному збільшення навантаження пропонується збільшення апаратної бази , або підвищення потужності існуючої бази.

Висновок

В даний час досить актуальними є дослідження області:

- Розробки методів Аналізу завантаження серверів з урахуванням різноманітних зовнішніх факторів які здійснюють вплив на ступінь завантаження.
- розробки методів прогнозування завантаження серверів та методів передбачення перевантажень серверів.
- розробки методів боротьби з перевантаженням та рівномірним розподілом завантаження між серверами в мережі.

Також актуальним являється вдосконалення системи прогнозування навантаження та передбачення перевантаження серверів для створення більш точних прогнозів на основі більш широкого набору вихідних даних не тільки роботи сервера та мережі, а й зовнішніх факторів, зокрема на основі статистичних даних знятих з серверів та даних які здійснюють корекцію на основі економічного , політичного або соціально стану країни або регіону , для отримання більш точних прогнозів та рекомендацій на практиці.

Список використаної літератури

1. Филаретов Г.Ф. Кабанов В.А. Применение искусственных нейронных сетей в системах управления // Приборы и системы управления. – 1999. № 4 – С. 3-6.
2. Вегенша Ш. Качество обслуживания в сетях IP: Пер. с англ. – М.: Издательский дом «Вильямс». – 2003. – 386 С.
3. Зенкин А.И. О математических методах прогнозирования – М.: – 1987. – 90 С.
4. Лабунская Н.Л. Система прогнозирования. М: – 1990. – 120 С.
5. Локтев С. Проблемы внедрения технологии "клиент-сервер". – 1996. №5 – С. 19-21 №6 – С. 20-22.
6. Дунаев С. Internet-технологии. – М.: "Диалог-МИФИ". – 1997. 117 С.

УДК 004.052.42

ТУРЧЕНКО Ю.А.

ИСПОЛЬЗОВАНИЕ АРИФМЕТИЧЕСКИХ ВЗВЕШЕННЫХ КОНТРОЛЬНЫХ СУММ ДЛЯ КОРРЕКЦИИ ОШИБОК В КАНАЛАХ СО СПЕКТРАЛЬНОЙ МОДУЛЯЦИЕЙ

В работе предлагается новый подход к повышению эффективности коррекции ошибок в каналах со спектральной модуляцией за счет использования арифметических взвешенных контрольных сумм. На основе исследования свойств ошибок, вызванных внешними помехами, разработано две процедуры их коррекции. Первая из них ориентирована на коррекцию одного символа, а вторая – на исправление двух смежных. Доказано, что предложенные технологии коррекции ошибок обеспечивают большую производительность и требуют меньшего числа контрольных разрядов по сравнению с корректирующими кодами.

In this paper a new approach is proposed to increase the effectiveness of error correction in spectrum modulation channel by using of arithmetic weighed check sums. Based on a study of the properties of data transmission errors caused by external noises a two new error correction prosedures is developed which use arithmetic weighed check sums. First of proposed procedures is oriented for one symbol correction and second – for two contiguous symbols. It is shown that the proposed error coorection techniques is more performance and demanded control bits less than error correcting codes.

Введение

Развитие технологий передачи цифровых данных является основой глобальных процессов информационной интеграции. Важное место в таких технологиях играет обеспечение надежности передачи данных. В современных условиях объективно действуют ряд факторов, снижающих надежность передачи данных. В частности, расширение использования беспроводных линий и систем мобильной связи имеет следствием заметный рост интенсивности внешних помех [1]. Ускорение скорости передачи вызывает рост числа ошибок, вызванных межсигнальной интерференцией. Применение спектральной модуляции, позволяющей повысить скорость передачи за счет кодирования группы бит одним канальным сигналом, приводит к резкому росту кратности возникающих ошибок.

Действие перечисленных факторов требует адекватного совершенствования средств обнаружения и коррекции ошибок, возникающих при передаче цифровых данных в компьютерных сетях и системах телекоммуникаций. Достигнутый в последние годы прогресс в области интегральных технологий открывает новые возможности создания эффективных средств контроля и коррекции ошибок. При этом важным направлением повышения эффективности средств контроля и исправления ошибок является учет специфики их возникновения в различных каналах передачи данных, в частности, в каналах со спектральной модуляцией (КСМ).

Таким образом, задача совершенствования средств контроля и коррекции ошибок передачи данных в КСМ является актуальной и значимой для практики.

Модели возникновения ошибок в КСМ и анализ средств их коррекции

Для повышения пропускной способности в интерфейсах компьютерных систем и сетей широко используются линии с фазовой, амплитудно-фазовой и амплитудно-импульсной модуляцией цифровых данных [1]. Основной особенностью такой модуляции является то, что передача группы битов контролируемого блока осуществляется одним канальным сигналом. В частности, спектральная модуляция цифровых данных применяется в телефонных, кабельных и беспроводных линиях компьютерных сетей. При спектральной модуляции каждые k смежных битов, которые можно рассматривать как канальный символ X, кодируются одним несущим (канальным) сигналом. Использование спектральной модуляции позволяет за счет более полного использования полосы пропускания линии в k раз повысить скорость передачи цифровой информации. Соответственно значение k называется эффективностью передачи сигналов.

Наиболее известным видом спектральной модуляции является квадратурно-амплитудная модуляция (quadrature ampli-

tude modulation-QAM). При использовании этого вида спектральной модуляции каждый канальный символ кодируется изменением фазы и амплитуды несущего синусоидального 2-х до 10. Так, стандартизированный протокол телефонных модемов V.32 предусматривает использование QAM с k=4, протокол V.90 предусматривает применение QAM со значением k равным 10 [1].

При использовании спектральной модуляции контролируемый блок B состоящий из m бит: $B=\{b_1,b_2,\ldots,b_m\}$, $b_l\in\{0,1\}$, $l=1,\ldots,m$ можно рассматривать состоящим из $q = m/k$ канальных символов: $B=\{X_1,\ldots,X_q\}$. Каждый j-тый из символов X_j, $j\in\{1,\ldots,q\}$ включает k смежных бит контролируемого блока $X_j=\{x_{j,1},x_{j,2},\ldots,x_{j,k}\}=\{b_{(j-1)\cdot k+1},b_{(j-1)\cdot k+2},\ldots,b_{j\cdot k}\}$ Каждому канальному символу X_j соответствует число $Z_j=\sum_{i=1}^{k}x_{ji}\cdot 2^{i-1}$.

В результате ошибок передачи канальных сигналов, соответствующие им кодовые группы (символы) искажаются, то есть при возникновении ошибки передачи j–то символа X_j их коды на приемнике и передатчике не одинаковы: $X_{sj}\neq X_{rj}$. Очевидно, что при этом могут изменить значение от одного до k бит, которые составляют j–тый символ X_j.

Выбор эффективного способа коррекции ошибок определяется их характером. Для беспроводных каналов основной причиной искажения канальных сигналов являются внешние помехи [2].

Существует несколько моделей возникновения ошибок передачи данных в каналах со спектральной модуляцией под воздействием внешних помех.

Если считать поток внешних помех пуассоновским (с интенсивностью λ), то длина m блока выбирается такой, чтобы за период τ его передачи вероятность появления искажающей данных помехи была достаточно мала. Это означает, что вероятность $P_0 = \exp(-\lambda\cdot\tau)$ отсутствия помех за время τ передачи блока существенно (на порядки) выше вероятности $P_1= \lambda\cdot\tau\cdot\exp(-\lambda\cdot\tau)$ появления одной, искажающей данные помехи. В свою очередь, вероятность $P_2= (\lambda\cdot\tau)^2\cdot\exp(-\lambda\cdot\tau)/2$ появления за период передачи блока двух, искажающих данные помех на 2-3 порядка ниже P_1.

При таких условиях способ коррекции должен обеспечивать эффективное гарантированное исправление битовых искажений передаваемого блока, вызванных одной помехой и надежное обнаружение битовых искажений, обусловленных двумя и более помехами.

Другая модель [1] возникновения ошибок в каналах со спектральной модуляцией под воздействием внешних помех учитывает длительность ν внешней помехи. Длительность ν воздействия внешней помехи является случайной величиной и для большинства реальных каналов передачи данных ее математическое ожидание меньше периода τ/m следования канальных сигналов: $M(\nu)<\tau/m$. При этом существует отличная от нуля вероятность того, что длительность ν воздействия внешней помехи может превысить τ/m. В этом случае, искажению могут подвергнуться несколько смежных символов информационного блока. Для большинства реальных каналов вероятность искажения одного символа на 1-2 порядка больше вероятности искажения пары смежных символов, а так, в свою очередь, на 1-2 порядка меньше вероятности искажения трех смежных символов. Поэтому на практике достаточно учитывать возможность искажения под воздействием помехи не более 2-х смежных символов.

Исходя из приведенной модели возникновения ошибок под воздействием внешних помех, необходимо обеспечить возможность коррекции битовых искажений в паре смежных символов блока.

Основными критериями эффективности средств коррекции ошибок являются:

- надежность адекватной классификации исправляемого класса битовых искажений;
- вероятностью обнаружения битовых искажений, вызванных более чем одной внешней помехой;
- числом контрольных разрядов:
- вычислительной сложностью формирования контрольного кода, проверки правильности передачи и исправления выявленных ошибок.

С ростом скорости передачи данных значимость последнего из перечисленных факторов эффективности неуклонно растет, поскольку важным является выполнение контроля в темпе передачи данных.

Обычно, для коррекции ошибок передачи данных используются корректирующие коды [1], наиболее известными из которых являются коды Рида-Соломона, БЧХ, турбо-коды. При использовании перечисленных корректирующих кодов число контрольных разрядов экспоненциально растет с увеличением кратности исправляемых ошибок. Такой же экспоненциальный характер носит зависимость вычислительной сложности процедуры формирования синдрома ошибок. Это означает, что при использовании корректирующих кодов для КСМ при имеющейся тенденции к увеличению числа k бит модулируемых одним канальным сигналом, резко возрастает число контрольных разрядов и время контроля. Другими словами, эффективность использования корректирующих кодов для КСМ с увеличением k снижается. Для увеличения эффективности применения корректирующих кодов в КСМ предложены специальные технологии [1]. Их суть состоит в том, что блок B разбивается на h субблоков $B=\{S_1,S_2,\ldots,S_h\}$, каждый u-тый, $u\in\{1,..,h\}$ из которых - S_u включает в себя m/h бит, отстоящих друг от друга в блоке на h бит: $S_u=\{b_u,b_{u+m/h},\ldots,b_{(h-1)\cdot m/h+u}\}$. Для каждого субблока вычисляется свой контрольный код. При возникновении ошибки в передаче символа, искаженные биты "рассеиваются" по разным субблокам, так, что в рамках каждого из них присутствует только один искаженный бит. Такая технология позволяет уменьшить суммарное число контрольных разрядов, но не позволяет производить контроль в темпе передачи блока, поскольку коды субблоков формируются после передачи всего блока. Это заметно увеличивает время контроля ошибок и требует больших аппаратных затрат для реализации параллельной работы h специализированных вычислителей корректирующих кодов.

С точки зрения контроля ошибок в темпе передачи блока, более эффективным является использование взвешенных контрольных сумм [2]. Предложенный в работе [2] метод коррекции ошибок в КСМ предполагает применение взвешенных контрольных сумм, при вычислении которых используются логические операции. Метод рассчитан на коррекцию битовых искажений, вызванных парой ошибок передачи канальных сигналов и не учитывает специфику ошибок, вызванных внешними помехами. Соответственно, основным его недостатком является большое число r контрольных разрядов, определяемое формулой [2]:

$$r = k \cdot (\log_2 \frac{m}{k} + 1) \qquad (1)$$

Вместе с тем, существует возможность уменьшить число контрольных разрядов за счет учета, в рамках метода формирования взвешенной контрольной суммы, особенностей возникновения ошибок в КСМ.

Целью исследования является повышение эффективности коррекции многократных битовых искажений, возникающих при передаче цифровых данных в КСМ под действием внешних помех.

Организация коррекции битовых искажений одного символа блока

Для решения задачи эффективной коррекции ошибок передачи данных возникающих при искажении под воздействием внешней помехи одного канального сигнала предлагается использовать арифметическую взвешенную контрольную сумму (АВКС). Сущность предлагаемого подхода состоит в следующем.

Символам X_1, X_2,…, X_q информационного блока ставятся в соответствие весовые коэффициенты $W_1,W_2,\ldots,W_q$ разрядностью по $\log_2 q$ бит. При этом значение j-того весового коэффициента W_j определяется его порядковым номером, то есть $W_j = j$. Например, при передаче блока данных, состоящего из 7-ми символов (q=7) 3-битовые весовые коэффициенты равны $W_1 = 1$, W_2=2,…,W_7 =7.

Контрольный код блока вычисляется в виде двух компонент. Первая компонента C_1 представляет собой арифметическую сумму кодов символов блока:

$$C_1 = \sum_{j=1}^{q} Z_j \qquad (2)$$

Вторая компонента C_2 представляет собой арифметическую сумму произведений кодов символов на их весовой коэффициент:

$$C_2 = \sum_{j=1}^{q} Z_j \cdot W_j \qquad (3)$$

Например, пусть контролируемый блок B объемом 28 бит состоит из 7-ми (q=7) 4-х битовых символов (k = 4): X_1={1111}, X_2={0011}, X_3={1101}, X_4={1010}, X_5={0010}, X_6={0101}, X_7={0100}. Z_1=15, Z_2=12, Z_3 =11, Z_4=5, Z_5=4, Z_6=10, Z_7 = 2. Значение C_1 = 59, а C_2 = 186.

Двухкомпонентный контрольный код $C_S=\{C_{S1},C_{S2}\}$ вычисляется передатчиком и отсылается приемнику. Последний по полученному блоку вычисляет код $C_R=\{C_{R1},C_{R2}\}$ и формирует двухкомпонентный код разности $\Delta=\{\Delta_1, \Delta_2\}$, компоненты которого вычисляются как арифметическая разность одноименных компонент контрольных кодов передатчика и приемника: $\Delta_1=C_{R1}-C_{S1}$, $\Delta_2=C_{R2}-C_{S2}$.

Если обе компоненты разности контрольных сумм равны нулю: $\Delta_1=0$ и $\Delta_2=0$, то блок считается переданным без ошибок.

При ошибочной передаче одного канального сигнала, например j–го, $j\in\{1,\ldots, q\}$, искажению подвергаются биты одного канального символа X_j. Соответственно, коды j-го символа на передатчике - Z_{Sj} и приемнике - Z_{Rj} будут отличными. Пусть α - разность этих кодов: $\alpha = Z_{Rj} - Z_{Sj}$, очевидно, что $-(2^k-1)\le\alpha\le(2^k-1)$.

Тогда компоненты разности арифметических взвешенных контрольных сумм равны: $\Delta_1=\alpha$, $\Delta_2=\alpha\cdot W_j$. Коррекция ошибок выполняется следующим порядком:

1. Осуществляется операция целочисленного деления Δ_2 на Δ_1. Если остаток от деления не равен нулю, то произошло искажение более, чем одного символа блока. В этом случае, коррекция не производится, а формируется сигнал на повторную передачу блока.

2. Если остаток деления Δ_2 на Δ_1 равен нулю, то частное равно значению весового коэффициента искаженного символа - W_j. Коррекция производится последовательным нахождением номера j искаженного символа по значению W_j и определением истинного значения кода Z_{Sj} j-го символа:

$$W_j=\frac{\Delta_2}{\Delta_1},\ j=W_j \qquad (4)$$
$$Z_{Sj}=Z_{Rj}-\Delta_1$$

Описанная процедура коррекции символа блока может быть проиллюстрирована следующим образом. Пусть в рамках приведенного выше примера ошибочно передан 3-й символ блока – $X_3=\{1101\}$ так, что на приемнике получен код $X_{R3}=\{1110\}$. Соответственно, $Z_{R3}=7$. Компоненты контрольного кода, вычисленные на приемнике равны: $C_{R1}=55$, $C_{R2}=174$. Компоненты разности контрольных кодов приемника и передатчиков определяются в виде: $\Delta_1=C_{R1}-C_{S1}=\alpha=55-59=-4$, $\Delta_2=C_{R2}-C_{S2}=174-186=-12$. Поскольку остаток от деления Δ_2/Δ_1 равен нулю, то ситуация классифицируется как искажение одного символа. Вычисляется его весовой коэффициент: $W_j=\Delta_2/\Delta_1=-12/(-4)=3$. Определяется номер символа: $j=W_j=3$ и исходный код: $Z_3=Z_{R3}-\alpha=7-(-4)=11$.

Важным аспектом является оценка вероятности Q_2 ложной классификации исправляемых ошибок. При искажении, в процессе передачи, двух канальных сигналов, возникают ошибки в двух символах блока. Пусть в процессе передачи искажению подверглись два символа: X_j и X_e, $j,e\in\{0,..,q-1\}$, $j<e$, причем $Z_{Rj}-Z_{Sj}=\alpha$, $Z_{Re}-Z_{Se}=\gamma$. Тогда $\Delta_1=\alpha+\gamma$, $\Delta_2=\alpha\cdot W_j+\gamma\cdot W_e$. Неверная классификация неисправимой ошибки в двух символах как исправимой в одном символе имеет место, если остаток от деления Δ_2/Δ_1 равен нулю, то есть существует целое N такое, что:

$$\alpha\cdot W_j+\gamma\cdot W_e=N\cdot(\alpha+\gamma) \qquad (4)$$

Проведенные экспериментальные исследования показали, что вероятность Q_2 в определяющей степени зависит от длины k символа. Эта зависимость носит экспоненциальный характер и выражается следующей эмпирической зависимостью при $k>4$: $Q_2\approx 0.1\cdot 2^{4-k}$. Это означает, что для современных систем передачи данных, которые характеризуются относительно большими значениями k, вероятность неадекватной классификации типа ошибки невелика. Например, при передаче блока длиной 1 Кбайт и длине символа в один байт, вероятность Q_2 составляет 0.0043. При этом следует иметь ввиду, что вероятность появления ошибки в паре символов на 2-3 порядка меньше вероятности искажения одного символа.

Число R контрольных разрядов определяется максимальной разрядностью первой C_1 и второй C_2 компонент арифметической взвешенной контрольной суммы. Поскольку максимальное значение C_1 равно $q\cdot(2^k-1)$, то разрядность этой компоненты не превышает $k+\log_2 q$. Максимальное значение компоненты C_2 равно $\sum_{i=1}^{q}(2^k-1)\cdot i=\frac{q^2\cdot(2^k-1)}{2}$. Соответственно, число бит для представления C_2 равно $k+2\cdot\log_2 q-1$. Таким образом:

$$R=2\cdot k+3\cdot\log_2 q+1 \qquad (5)$$

При использовании корректирующих кодов нижняя граница G числа контрольных разрядов определяется объемом информации, необходимым для локализации k искаженных бит в m-битовом блоке.

Соответственно численное значение G определяется выражением:

формулы:

$$G = \log_2 \prod_{l=0}^{k-1} \frac{m-l}{l+1} \qquad (6)$$

Например, при разрядности символа k=8 и числе символов в блоке q=1024, число контрольных разрядов, обеспечивающих коррекцию одного символа с использованием арифметической взвешенной контрольной суммы составляет R=45, а нижняя граница числа контрольных разрядов корректирующих кодов - G = 65. Таким образом, использование арифметической взвешенной контрольной суммы позволяет уменьшить число контрольных разрядов в 1.45 раза. С развитием технологии спектральной модуляции увеличивается разрядность символа и, соответственно, отношение числа контрольных разрядов при применении корректирующих кодов и АВКС. Так, при k=10, q=1024, соотношение G/R = 1.67.

Модификация АВКС для коррекции пары смежных символов

Важную проблему обеспечения надежности передачи цифровых данных в компьютерных сетях и системах телекоммуникаций представляет воздействие внешних помех большой длительности, которое может вызвать искажение нескольких смежных символов. Для каналов передачи данных со спектральной модуляцией наибольшее значение имеет случай воздействия помехи на пару смежных канальных сигналов.

При воздействии помехи на пару смежных символов, число потенциально подверженных искажению битов резко возрастает, что имеет следствием существенно снижение эффективности использования корректирующих кодов.

Гораздо эффективнее представляется использование для коррекции таких битовых искажений специальных модификаций АВКС.

Ниже предлагается одна из таких модификаций АВКС, которая предполагает формирование контрольного кода блока в виде четырех компонент. Каждая пара смежных символов информационного блока имеет одинаковый весовой коэффициент, так, что для j–го символа блока $j\in\{1,\ldots,q\}$, весовой коэффициент W_j для четных значений j равен $j/2$, а для нечетных значений j вычисляется в виде: $W_j = \lfloor j/2 \rfloor + 1$. В частности, первый и второй символы имеют весовые коэффициенты равные единице, для третьего и четвертого символов – W_3=W_4=2.

Первая компонента C_1 представляет собой арифметическую сумму кодов символов с четными номерами в информационном блоке:

$$C_1 = \frac{1}{2} \cdot \sum_{j=1}^{q} (1 + (-1)^j) \cdot Z_j \qquad (7)$$

Вторая компонента C_2 вычисляется как арифметическая сумма кодов символов с нечетными номерами в блоке:

$$C_2 = \frac{1}{2} \cdot \sum_{j=1}^{q} (1 + (-1)^{j+1}) \cdot Z_j \qquad (8)$$

Третья компонента C_3 представляет собой арифметическую сумму произведений кодов символов с четными номерами в блоке на их весовой коэффициент:

$$C_3 = \frac{1}{2} \cdot \sum_{j=1}^{q} (1 + (-1)^j) \cdot Z_j \cdot \frac{j}{2}) \qquad (9)$$

Четвертая компонента C_4 вычисляется как арифметическая сумма произведений кодов символов с нечетными индексами их весовой коэффициент:

$$C_4 = \frac{1}{2} \cdot \sum_{j=1}^{q} (1 + (-1)^{j+1}) \cdot Z_j \cdot (1 + \lfloor j/2 \rfloor) \qquad (10)$$

Например, пусть контролируемый блок B объемом 32 бит состоит из 8-ми (q=8) 4-х битовых символов (k = 4): X_1={0001}, X_2={1010}, X_3={1100}, X_4={0101}, X_5={1000}, X_6={0010}, X_7={1001}, X_7={0110}. Z_1=8, Z_2=5, Z_3=3, Z_4 =10, Z_5=1, Z_6=4, Z_7=9, Z_8= 6. В этом случае, значения компонент контрольной суммы на передатчике: Cs_1=25, Cs_2=21, Cs_3=61, Cs_4=53.

Четырехкомпонентный контрольный код C_S ={C_{S1}, C_{S2}, C_{S3}, C_{S4}} вычисляется передатчиком и отсылается приемнику. Последний по принятому блоку вычисляет код C_R ={C_{R1}, C_{R2}, C_{R3}, C_{R4}} и формирует четырехкомпонентный код разности контрольных кодов приемника и передатчика: Δ={Δ_1, Δ_2, Δ_3, Δ_4}, компоненты которого вычисляются как арифметическая разность одноименных компонент контрольных кодов передатчика и

приемника: $\Delta_1=C_{R1}-C_{S1}$, $\Delta_2 = C_{R2} - C_{S2}$, $\Delta_3=C_{R3}-C_{S3}$, $\Delta_4 = C_{R4} - C_{S4}$.

Процесс коррекции ошибок формально описывается следующим образом:

1. Если $\Delta_1=\Delta_2=\Delta_3=\Delta_4=0$, то блок считается принятым без ошибки. Переход на конец.

2. Если $\Delta_1\neq0$, $\Delta_3 \neq 0$, а $\Delta_2=\Delta_4 = 0$, то классификация типа ошибки производится анализом кодов Δ_1 и Δ_3 . Если Δ_3 не делится нацело на Δ_1, то это значит, что ошибка относится к классу неисправляемых и тогда формируется запрос на повторную передачу блока и переход на конец. Если результатом деления Δ_3/Δ_1 является целое число, то ошибка произошла в одном символе, причем этот символ находится на четной позиции в информационном блоке. Целое частное от деления Δ_3/Δ_1 определяет весовой коэффициент W_j искаженного символа. Номер j этого символа вычисляется как удвоенное значение весового коэффициента: $j = 2\cdot W_j$.

Коррекция идентифицированного описанным способом j–го символа осуществляется путем арифметического вычитания кода Δ_1 от кода j-го символа в блоке, полученном приемником. Формально процесс коррекции можно представить следующим образом:

$$j = 2\cdot\frac{\Delta_3}{\Delta_1}, \quad Z_{S,j} = Z_{R,j} - \Delta_1 \tag{11}$$

3. Если $\Delta_2\neq0$, $\Delta_4 \neq 0$, а $\Delta_1=\Delta_3 = 0$, для классификации типа ошибки необходимо анализировать коды Δ_2 и Δ_4. Если Δ_4 не делится нацело на Δ_2, то это значит, что ошибка относится к классу неисправляемых и тогда формируется запрос на повторную передачу блока и переход на конец.

Если результатом деления Δ_4/Δ_2 является целое число, то искажению подверглись биты одного символа с нечетным порядковым номером r в блоке. При этом частное от деления Δ_4/Δ_2 соответствует весовому коэффициенту W_r искаженного символа. Вычисление порядкового номера r искаженного символа в блоке по значению весового коэффициента W_r осуществляется в виде: $r = W_r\cdot2-1$. Процесс коррекции символа аналогичен описанному выше. Формально процесс коррекции одного символа с нечетным порядковым номером в блоке описывается в виде:

$$r = 2\cdot\frac{\Delta_4}{\Delta_2} - 1, \quad Z_{S,r} = Z_{R,r} - \Delta_2 \tag{12}$$

4. Если $\Delta_1\neq0$, $\Delta_2 \neq 0$, $\Delta_3\neq 0$ и $\Delta_4 \neq 0$, то классификация типа ошибки осуществляется путем анализа остатков от деления Δ_3/Δ_1 и Δ_4/Δ_2. Если хотя бы один из остатков не равен нулю: $\Delta_3 \bmod \Delta_1\neq0$ или $\Delta_4 \bmod \Delta_2\neq0$, то ситуация классифицируется как ошибка, которая не может быть исправлена. Формируется запрос на повторную передачу и переход на конец. В противном случае, то есть если $\Delta_3 \bmod \Delta_1=0$ и $\Delta_4 \bmod \Delta_2=0$, ситуация классифицируется как искажение, под воздействием внешней помехи большой длительности, пары смежных символов передаваемого блока. В этом случае определяются весовые коэффициенты W_j и W_r, $j,r\in\{1,\ldots,q\}$, пары искаженных символов с порядковыми номерами j и r: $W_j=\Delta_3/\Delta_1$ и $W_r=\Delta_4/\Delta_2$. Если $W_j = W_r$ или $W_j=W_r+1$, то номер r первого из искаженных символов $r = 2\cdot W_r$ -1, а второго – $j=2\cdot W_r$. В этом случае, коррекция j–го и r–го символов на приемнике выполняется путем вычитания из них кодов Δ_1 и Δ_2 соответственно. Формально процесс коррекции может быть представлен следующим образом:

$$j = 2\cdot\frac{\Delta_3}{\Delta_1}, r = 2\cdot\frac{\Delta_4}{\Delta_2} - 1, \quad Z_{S,j} = Z_{R,j} - \Delta_1, Z_{S,r} = Z_{R,r} - \Delta_2 \tag{13}$$

Если разница W_j и W_r превышает единицу, то есть $W_j \neq W_r$ и $W_j \neq W_r+1$, то ситуация классифицируется, как возникновение неисправимой ошибки.

Процедура коррекции пары смежных символов блока может быть проиллюстрирована следующим образом. Пусть в рамках приведенного выше примера ошибочно переданы 4-й и 5-й символы блока (j=4,r=5), так, что на приемнике вместо символов X_{S4}={0101} и X_{S5}={1000} получены искаженные их значения: X_{R4}={1000} и X_{R5}={1110}. Компоненты контрольного кода, вычисленные на приемнике равны: C_{R1} = 16, C_{R2} = 27, C_{R3} = 43, C_{R4} = 71. Компоненты разности контрольных кодов приемника и передатчика вычисляются в виде: $\Delta_1 = C_{R1}-C_{S1}$ = 16-25= -9, $\Delta_2 = C_{R2}-C_{S2}$ = 27-21 = 6, $\Delta_3 = C_{R3}-C_{S3}$ =43-61= -18, $\Delta_4 = C_{R4}-C_{S4}$ =71-53=18. Поскольку все значения Δ_1, Δ_2, Δ_3 и Δ_4

отличны от нуля и $\Delta_1 \neq \Delta_3$, то в соответствии с пп.2 и 3 анализируется остаток от деления $\Delta_3/\Delta_1 = (-18)/(-9)=2$ и $\Delta_4/\Delta_2 = 18/6 = 3$. Так как остатки равны нулю: $\Delta_3 \bmod \Delta_1 = (-18) \bmod (-9)=0$ и $\Delta_4 \bmod \Delta_2 = 18 \bmod 6 = 0$, то ситуация классифицируется как искажение пары смежных символов блока. Весовой коэффициент W_j искаженного символа с четным номером j определяется как $W_j = \Delta_3/\Delta_1=2$, а весовой коэффициент W_r искаженного символа с нечетным номером r в блоке равен $W_r = \Delta_4/\Delta_2=3$. Соответственно, четный номер j искаженного символа вычисляется как $j=2\cdot W_j = 4$, а нечетный номер r искаженного символа равен $r = 2\cdot W_r-1 = 2\cdot 3-1=5$. В соответствии с (13) осуществляется восстановление 4-го и 5-го символов: $Z_{S4}=Z_{R4}-\Delta_1 = 1-(-9)=10$, $Z_{S5}=Z_{R5}-\Delta_2 = 7-6 = 1$.

Оценка разрядности контрольного кода предлагаемой модификации АВКС может быть произведена следующим образом. Каждая из компонент C_1 и C_2 представляет собой арифметическую сумму $q/2$ положительных чисел, не превышающих 2^k-1. Соответственно, требуемое число разрядов для представления каждого кодов C_1 и C_2 равно $k+\log_2 q-1$. Каждая из компонент C_3 и C_4 является арифметической суммой $q/2$ положительных произведений кодов символов на весовые коэффициенты, образующие ряд: 1,2,…, $q/2$. Как было показано выше, число бит для представления такой суммы равно $k + 2\cdot\log_2\frac{q}{2} - 1 = k + 2\cdot\log_2 q - 3$. Таким образом, разрядность R' контрольного кода предложенной модификации АВКС определяется формулой:

$$R' = 4\cdot k + 6\cdot\log_2 q - 8 \qquad (14)$$

При использовании корректирующих кодов нижняя граница G' числа контрольных разрядов определяется объемом информации, необходимым для локализации $2\cdot k$ искаженных бит в m-битовом блоке. Соответственно численное значение определяется выражением:

$$G' = \log_2 \prod_{l=1}^{2\cdot k-1} \frac{m-l}{l+1} \qquad (15)$$

Например, при разрядности символа k=8 и числе символов в блоке q=1024, число контрольных разрядов, обеспечивающих коррекцию пары смежных символов с использованием предложенной модификации АВКС составляет R'= 84, корректирующих кодов - G' = 116, а логической взвешенной контрольной суммы – r =89. Таким образом, использование АВКС позволяет уменьшить число контрольных разрядов в 1.4 раза по сравнению с корректирующими кодами.

Изложенный подход к исправлению пары смежных искаженных символов информационного блока достаточно просто может быть обобщен на случай воздействия внешней помехи большой длительности на тройку или четверку смежных символов. Для коррекции тройки смежных символов контрольный код состоит из 6-ти компонент, а для восстановления четверки смежных символов – восьми компонент.

Выводы

В результате проведенных исследований предложены два варианта использования АВКС для коррекции ошибок в КСМ.

Доказано, что применение АВКС позволяет уменьшить число контрольных разрядов как по сравнению с корректирующими кодами, так и по сравнению с логической взвешенной контрольной суммой. Вычисление АВКС достаточно просты и осуществляются в темпе передачи данных в канале, а операции, связанные с коррекцией выполняются только при обнаружении ошибки.

Список литературы

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Издательский дом ”Вильямс”, 2004.- 1104 С.

2. Марковський О.П., Федоречко О.І., Турченко Ю.О. Корекція помилок передачі даних в каналах зі спектральною модуляцією з використанням зважених контрольних сум // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка.- К.:ВЕК+.- 2006.-№ 45.- С.104-110.

УДК 004.2

ЖАБИНА В.В.

ПАРАЛЛЕЛЬНОЕ ФОРМИРОВАНИЕ КОМАНД В ПОТОКОВЫХ СИСТЕМАХ

Предложен метод ускорения вычислений в системах, управляемых потоком данных, при реализации алгоритмов с ветвлениями, основанный на динамическом формировании параллельных потоков команд. Показана возможность идентификации слов акторов и данных на основе графа задачи в процессе компиляции для потоковых систем с несколькими средами формирования команд.

Method of computation acceleration in data-flow systems for branching algorithms realization is proposed. It is based on dynamic parallel instruction flows compilation. There is shown a possibility to identify actor and data words during compilation on the base of task graph for data-flow systems with several instruction formation units.

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. Ускорения вычислений в этом случае можно добиться распараллеливанием вычислений на уровне операций, а также аппаратной реализацией операций. Время решения параллельных задач во многом определяется эффективностью распараллеливания вычислений.

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.

Одним из перспективных подходов, позволяющих 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.

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.

Реализация данной модели вычислений для реализации мелкозернистых алгоритмов может быть обеспечена применением ПЛИС, которые содержат вычислительные ядра, модули памяти и средства коммуникации. Использование такой элементной базы и технологии SoC (System on chip) дают возможность эффективной реализации параллельных вычислений на уровне операций.

Поскольку динамическое распределение заданий осуществляется средствами самой системы, то важной задачей является уменьшение непроизводительных расходов времени на этот процесс.

Методы организации вычислений, управляемых потоком данных

К одним из первых работ в области организации вычислительных систем, управляемых потоком данных (потоковых систем), можно отнести работы [2-6]. Имея структурные особенности, такие системы используют общую модель вычислений. Потоковые системы содержат несколько вычислительных модулей (ВМ) (устройств обработки информации) и среду формирования команд (СФК), связанных коммуникационными средствами (КС).

В системах, управляемых потоком данных, команды выполняются не в заданной программой последовательности, а при наличии готовых данных (наличии 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 i-й вершине которого соответствует операция (функция), а каждой дуге – данные.

Операция для i-й вершины графа описывается информационным словом, которое называют актором (actor). Актор описывается кортежем

$$A_i = \langle I_i, F_i, N_i, T_i \rangle, \quad (1)$$

где I_i – идентификатор (уникальное имя) данного актора; F_i – функция преобразования данных (код операции); N_i – имя актора, для которого i-й актор подготавливает операнд, а T_i – совокупность признаков этого операнда.

Акторы связаны между собой только по данным. Каждой дуге графа соответствует слово данных

$$D_i = \langle I_i, Q_i, N_i, T_i \rangle, \quad (2)$$

где Q_i – значение операнда.

Из соответствующих элементов A_i и D_i в СФК формируется команда, которая выполняется в свободном ВМ или помещается в очередь.

Известны различные алгоритмы формирования команд [2-5]. Для организации СФК используется ассоциативная память или ее эмуляция с применением других технических средств.

Компиляторы позволяют автоматизировать подготовку акторов и данных, причем, без учета конкретного числа ВМ в системе.

Если время формирования команд в СФК меньше времени выполнения команд в ВМ, то в разных ВМ одновременно могут выполняться разные команды, за счет чего и достигается распараллеливание операций. Современные технологии реализации ВМ, в том числе, с использованием ПЛИС, позволяют применять аппаратные методы ускорения операций, в результате чего интенсивность формирования команд в СФК становится недостаточной для загрузки нескольких ВМ одновременно.

В связи с этим важной проблемой является увеличение интенсивности потока готовых команд с целью ускорения параллельных вычислений.

Простое дублирование СФК приводит к недостаткам, присущим статическим методам подготовки параллельных программ. Программист должен предварительно разрезать граф задачи на подграфы, предопределить идентификаторы акторов и данных в разных подграфах с учетом связи по данным между ними.

Более эффективным является подход, который позволяет автоматически формировать акторы и данные при наличии нескольких СФК [6, 7]. Система содержит модули, в состав которых входит СФК и ВМ. Модули организуются в кольцевую структуру. Данные циркулируют в такой структуре в поисках своего актора. Команды формируются в разных СФК и выполняются в соответствующих ВМ.

Недостатком такого подхода являются затраты времени на пересылку данных между модулями системы. Кроме того, выход из строя любого модуля приводит к неработоспособности всей системы, поскольку информация передается последовательно от одного модуля к другому.

В работе [8] предложен метод автоматического распределения акторов и данных между несколькими СФК, позволяющий автоматически формировать параллельные потоки команд. Однако указанный метод может быть использован в случае, когда граф задачи не имеет разветвлений потоков данных.

В данной работе предлагается метод параллельного формирования потоков команд, который не требует ручного вмешательства в процесс назначения идентификаторов соответствия акторов и данных различным СФК и может быть использован для алгоритмов, в которых допускается разветвление потоков данных.

Метод формирования параллельных потоков команд

Организация основных компонентов потоковой системы поясняется рис. 1. Система может иметь произвольное число ВМ. Количество СФК должно быть равно $k=2^j$, где $j=1,2,3\ldots$

Формирование акторов и данных осуществляется на основе графа.

Модифицируем форматы объектов (1) и (2). Формат акторов принимает следующий вид

$$A_i =< M_i, I_i, F_i, K_i, L_i, T_i >, \qquad (3)$$

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 запишем в виде

$$D_i =< Q_i, N_i, L_i, T_i >, \qquad (4)$$

где Q_i – значение операнда; N_i – количество акторов, для которых передается данный операнд.

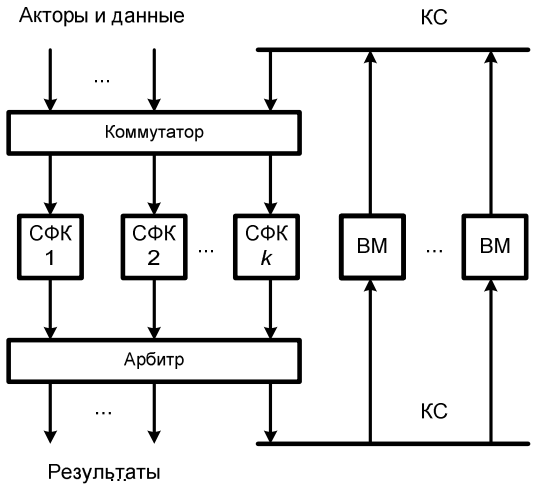


Рис. 1. Структура потоковой системы

Для формирования потоков команд, то есть нахождения значений идентификаторов СФК M_i, предлагается следующий алгоритм.

1. На основе исходного графа построить матрицу смежности, показывающую связь между акторами по данным. В качестве номеров строк и столбцов взять имена акторов I_i. или их индексы i. Первому потоку присвоить идентификатор M_0.

2. Для формирования параллельных потоков команд начать циклический просмотр полученной матрицы. В каждом цикле от первой до последней строки выполнять:

а) если текущая строка не содержит единичных элементов, перейти к следующей исходной строке;

б) если строка с индексом i содержит единичные элементы, то в качестве первого актора в потоке команд взять актор с именем I_i;

в) актор с именем, индекс которого равен номеру столбца первого единичного элемента в текущей строке, добавить в данный поток команд, а все единичные элементы данного столбца обнулить;

г) перейти к строке, номер которой соответствует номеру столбца, определенного в п. 2в;

д) если строка не содержит единичных элементов, закончить формирование очередного потока команд и присвоить ему соответствующий идентификатор СФК (если поток первый, то M_0, в остальных случаях M_{i+1}, где M_i – имя предыдущего потока); перейти к следующей строке для формирования следующего потока команд;

е) если строка содержит единичные элементы, то продолжить формирование текущего потока.

3. Если в результате прохода матрицы в ней остались единичные элементы, то проход повторить с п. 2; если единичных элементов нет, то выполнить п.4.

4. Составить список акторов I_i в соответствии с форматом (3) и учетом связей, заданных матрицей смежности, а также идентификаторов M_i.

5. Аналогичным образом составить список данных D_i.

При выполнении алгоритма реальное число ВМ и СФК в системе не учитывается. В результате выполнения алгоритма формируется максимально возможное число параллельных потоков команд. Если в системе имеется только одна СФК, то идентификаторы M_i системой игнорируется. При наличии нескольких СФК, число которых выбирается по формуле $k = 2^j$ $(j = 1,2,3...)$, реально будет формироваться столько потоков команд, сколько СФК имеется в системе. При распределении объектов (акторов и данных) между СФК учитываются только $\rceil \log_2 k [$ младших разрядов двоичных кодов идентификаторов M_j. Например, для множества идентификаторов $\{M_0, M_1, M_2, M_3, M_4, M_5\}$ соответственно с кодами $\{000, 001, 010, 011, 100, 101\}$ при наличии в системе двух СФК учитывается только один младший разряд кодов, а при четырех СФК –

два младших разряда. Соответственно в системе будут формироваться два или четыре потока команд.

Пример формирования параллельных потоков команд

Формирование параллельных потоков команд рассмотрим на примере графа, представленного на рис. 2.

На основе графа формируем матрицу смежности, как показано в табл.1.

В соответствии с приведенным алгоритмом формируем параллельные потоки команд. При первом проходе матрицы получаем последовательность команд

M_0: 1-5-10-14-16-19-21-22.

В табл. 1 рамками отмечены первые значимые элементы в строках при первом проходе матрицы. Затемненные единичные элементы обнуляются при первом проходе.

При дальнейших проходах аналогично получаем:

M_1: 2-6-11-15-17-20;

M_2: 3-7-12;

M_3: 4-8-13-18.

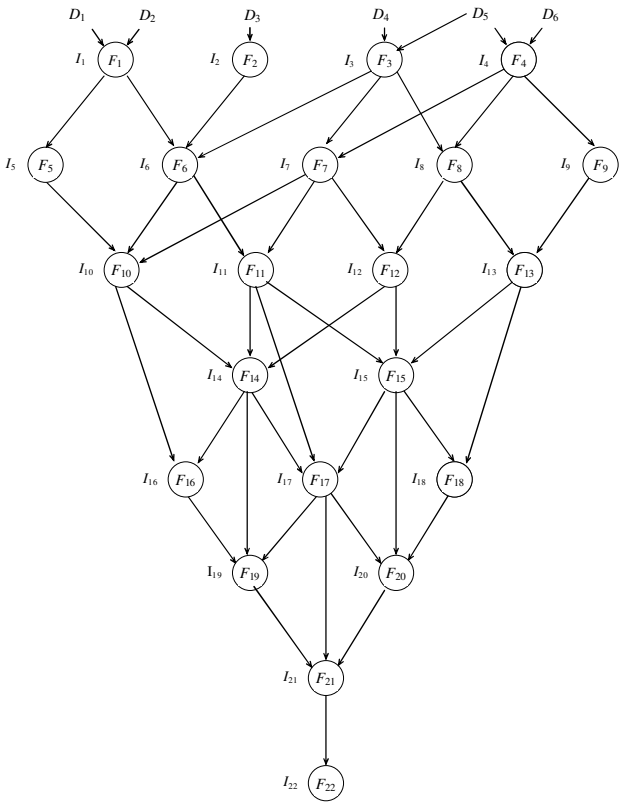


Рис. 2. Граф задачи

Поскольку матрица смежности не обнулена полностью, начинаем обход сначала и получаем последнюю строку потока команд

M_4: 9.

На основе полученных данных описываем каждый актор в соответствии с форматом (3):

$A_1 =< M_0, I_1, F_1, 2, I_5, M_0, I_6, M_1, T_1 >;$

$A_2 =< M_1, I_2, F_2, 1, I_6, M_1, T_2 >;$

$A_3 =< M_2, I_3, F_3, 3, I_6, M_1, I_7, M_2, I_8, M_3, T_3 >;$

$A_4 =< M_3, I_4, F_4, 3, I_7, M_2, I_8, M_3, I_9, M_4, T_4 >;$

$A_5 =< M_0, I_5, F_5, 1, I_{10}, M_0, T_5 >;$

$A_6 =< M_1, I_6, F_6, 2, I_{10}, M_0, I_{11}, M_1, T_6 >;$

$A_7 =< M_2, I_7, F_7, 3, I_{10}, M_0, I_{11}, M_1, I_{12}, M_2, T_7 >;$

$A_8 =< M_3, I_8, F_8, 2, I_{12}, M_2, I_{13}, M_3, T_8 >;$

$A_9 =< M_4, I_9, F_9, 1, I_{13}, M_3, T_9 >;$

$A_{10} =< M_0, I_{10}, F_{10}, 2, I_{14}, M_0, I_{16}, M_0, T_{10} >;$

$A_{11} =< M_1, I_{11}, F_{11}, 3, I_{14}, M_0, I_{15}, M_1, I_{17}, M_1, T_{11} >;$

$A_{12} =< M_2, I_{12}, F_{12}, 2, I_{14}, M_0, I_{15}, M_1, T_{12} >;$

$A_{13} =< M_3, I_{13}, F_{13}, 2, I_{15}, M_1, I_{18}, M_3, T_{13} >;$

$A_{14} =< M_0, I_{14}, F_{14}, 3, I_{16}, M_0, I_{19}, M_0, I_{17}, M_1, T_{14} >;$

$A_{15} =< M_1, I_{15}, F_{15}, 3, I_{17}, M_1, I_{20}, M_1, I_{18}, M_3, T_{15} >;$

$A_{16} =< M_0, I_{16}, F_{16}, 1, I_{19}, M_0, T_{16} >;$

$A_{17} =< M_1, I_{17}, F_{17}, 3, I_{19}, M_0, I_{21}, M_0, I_{20}, M_1, T_{17} >;$

$A_{18} =< M_3, I_{18}, F_{18}, 1, I_{20}, M_1, T_{18} >;$

$A_{19} =< M_0, I_{19}, F_{19}, 1, I_{21}, M_0, T_{19} >;$

$A_{20} =< M_1, I_{20}, F_{20}, 1, I_{21}, M_0, T_{20} >;$

$A_{21} =< M_0, I_{21}, F_{21}, 1, I_{22}, M_0, T_{21} >;$

$A_{22} =< M_0, I_{22}, F_{22}, 1, I_{22}, M_0, T_{22} >.$

Описываем данные в соответствии с форматом (4):

$D_1 =< Q_1, 1, I_1, M_0, T_1 >;$

$D_2 =< Q_2, 1, I_1, M_0, T_2 >;$

$D_3 =< Q_3, 1, I_2, M_1, T_3 >;$

$D_4 =< Q_4, 1, I_3, M_2, T_4 >;$

$D_5 =< Q_5, 2, I_3, M_2, I_4, M_3, T_5 >;$

$D_6 =< Q_6, 1, I_4, M_3, T_6 >.$

Выводы

Предложенный метод формирования параллельных потоков команд в системах, управляемых потоком данных, имеет ряд преимуществ по сравнению с программным управлением параллельными вычислениями. Управляющие слова и данные могут быть сформированы автоматически компилятором на основе графа. Нет необходимости предварительно учитывать число вычислителей в системе и обеспечивать синхронизацию обмена данными между параллельными ветвя-

ми алгоритма. Динамическое распределение операций позволяет выявить непосредственно в процессе вычислений скрытый параллелизм, связанный с различными длительностями обработки данных в различных ветвях алгоритмов. Вычислительные модули могут выполнять операции, относящиеся к различным задачам, в любой последовательности. В связи с этим в системе могут одновременно решаться независимые задачи, причем, нет необходимости синхронизации выполнения задач.

По сравнению с кольцевой организацией потоковых систем предложенный метод позволяет ускорить вычисления за счет сокращения непроизводительных затрат времени на последовательную пересылку данных между модулями системы. Кроме того, это способствует повышению надежности систем.

По сравнению с известным методом формирования параллельных потоков команд предложенный метод расширяет область применения потоковых систем за счет реализации алгоритмов с разветвлениями.

Все это повышается эффективность обработки данных в потоковых системах.

Табл. 1. Матрица смежности

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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.

УДК 004.056.5

МАРКОВСКИЙ А.П.,
ВИНОГРАДОВ Ю.Н.,
ПОВАЖНАЯ Н.С.

К ПРОБЛЕМЕ ОЦЕНКИ КАЧЕСТВА ДВОИЧНЫХ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДЛЯ СИСТЕМ ЗАЩИТЫ ИНФОРМАЦИИ

Статья посвящена проблеме повышения эффективности оценки качества двоичных псевдослучайных последовательностей для защиты информации. Предложен подход к повышению эффективности тестирования псевдослучайных последовательностей. Для определения сложности последовательности с *k*-кратной ошибкой разработана высокопроизводительная технология, использующая нелинейную вос-производящую модель. Показано, что применения разработанной технологии позволяет уменьшить вы-числительную сложность по сравнению с известными методами и, тем самым, сделать возможным тестиро-вание более длинных последовательностей, что обеспечивает повышение надежности оценки их качества.

Paper is dedicated to a problem of increasing of binary pseudorandom sequences for data protection testing efficiency. The approach for increasing the effectiveness of pseudorandom sequences testing for data security system has been proposed. The highly performance techniques for testing of *k*-errors complexity of sequences using nonlinear reproduction model has been developed. It has been shown that implementation of proposed techniques has low calculation complexity in compare to known ones, so it make possible testing of more long sequences and thus ensure highest reliability of testing.

Введение

Генераторы псевдослучайных объектов (чисел, двоичных последовательностей и функ-ций) широко используются в современных информационных технологиях в качестве ключевых элементов систем технической диагностики, статистического и имитационного моделирования, систем мобильной и спутни-ковой связи, при решении математических задач методами Монте-Карло, а также в компьютерных играх и развлечениях.

Важное место занимают псевдослучайные объекты в современных системах защиты информации. Особенно важную роль в этой динамично развивающейся области играют псевдослучайные двоичные последовательнос-ти (ПСДП), которые используются в качестве базового элемента одного из трех типов алгоритмов защиты информации - потоковых алгоритмов[1]. ПСДП широко используются для генерации ключей симметричных алгорит-мов защиты данных и псевдослучайных двоич-ных строк протоколов аутентификации удален-ных пользователей интегрированных систем[1].

В современных условиях роста производи-тельности вычислительных систем и возможностей объединения значительного числа компьютеров в сеть для решения задач наруше-ния защиты, актуальной становится проблема объективной оценки качества ПСДП, ис-пользуемых для защиты информации.

Защитные свойства ПСДП, в теоретическом плане, определяются принципиальной невоз-можностью аналитического решения систем не-линейных булевых уравнений [5]. Именно это свойство булевых преобразований лежит в основе использования двоичных последова-тельностей и булевых функций в системах защиты информации. Исходя из этого наиболее объективным представляется оценка качества указанных средств защиты информации через свойства булевых преобразований, лежащих в их основе.

Таким образом, на современном этапе разви-тия технологии защиты информации актуаль-ной является задача повышения надежности методов оценки качества ПСДП.

Анализ современных методов оценки качества ПСДП

Для эффективного использования ПСДП в современных системах защиты информации они должны удовлетворять определенным критериям. В обобщенном виде можно выделить следующие критерии качества ПСДП:

1. Статистические критерии качества ПСДП, которые включают в себя [6]: оценки распреде-ления, частотность и дисперсию основного и дифференциального распреде-

лений, оценки распределения серий, оценки стационарности статистических генерируемых чисел.

2. Критерий взаимной независимости элемен-тов ПСДП, который оценивается путем анализа двумерного распределения для выявления взаимного влияния элементов последователь-ности, числовых оценок взаимной корреляции, оценки энтропии, а также анализа спектральных характеристик, позволяющего выявить наличие зашумленных периодичностей последователь-ности [6].

3. Возможность экстраполяции последовательности ПСДП путем построения эффектив-ной воспроизводящей модели по заданному фрагменту [1]: формально эта возможность оценивается через параметры модели, вос-производящей заданный фрагмент. Фактически рассматриваемый критерий позволяет оценить с теоретико-информационных позиций возмож-ность предсказания заданного фрагмента по-следовательности.

В ряде работ [1,4] второй и третий критерии качества из приведенного выше перечня рас-сматриваются в качестве одного.

Проведенный анализ использований ПСДП для защиты информации показал, что основным критерием эффективности их применения в этой области является их непредсказуемость, невозможность построения воспроизводящей модели, способной интерполировать заданный фрагмент последовательности.

Формальной оценкой возможности экстра-поляции n-битовой двоичной последователь-ности $S=\{s_1,s_2,\ldots,s_n\}$, $\forall j\in\{1,\ldots,n\}$: $s_j\in\{0,1\}$ выступает сложность воспроизводящей ее модели [1]. В качестве воспроизводящих моделей используется сдвиговый регистр с линейной или нелинейной булевой функцией обратной связи. Соответственно, в первом случае воспроизводящая модель называется линейной, а во втором - нелинейной [1,2].

Установлены критерии [3] оценки возмож-ности экстраполяции последовательностей на основе параметров указанных моделей. Так, при тестировании линейной сложности $L(S)$ n-битовой последовательности S, в качестве критерия невозможности экстраполяции высту-пает равенство $L(S)=n/2$ [1]. Проблема состоит в том, что на практике возможна ситуация, при которой n-битовая последовательность S фор-мально проходит тест на линейную сложность, однако, если инвертировать в S k определенных битов (обозначим модифицированную двоич-ную последовательность через S''), то вполне может оказаться, что $L(S'')<<n/2$, то есть последовательность S'' может быть экстраполирована с использованием воспроизводящей модели $R(S')$. Это означает, что можно экстраполировать и исходную двоичную по-следовательность S с использованием модели $R(S')$ и вероятность битовой ошибки экстра-поляции будет близка к k/n [3].

Задача состоит в нахождении для фиксированного значения k такой локализации единичных компонент вектора $C=\{c_1,c_2,\ldots c_n\}$, $\forall j\in\{1,\ldots,n\}$: $c_j\in\{0,1\}$, $c_1+c_2+\ldots+c_n=k$, для кото-рой значение $L(S\oplus C)$ принимает минимальное значение. Следовательно, это минимальное зна-чение $L(S\oplus C)$ называют линейной сложностью двоичной последовательности S с k-кратной ошибкой (k-error linear complexity of sequence).

Основная проблема практического нахожде-ния линейной сложности последовательности с k-кратной ошибкой состоит в значительной вычислительной сложности решения этой зада-чи. Действительно, для заданной n-битовой двоичной последовательности S вычислитель-ная сложность построения линейной вос-производящей модели при использовании алгоритма Berlekamp-Massey [1] при объеме требуемой памяти $2\cdot n^2$ составляет $O(n^2)$. Следует отметить, что для упомянутого алго-ритма Berlekamp-Massey понятия вычислитель-ной и временной сложности тождественны, поскольку процедура построения линейной воспроизводящей модели в этом алгоритме имеет строго последовательный характер и не может быть распараллелена. Количество q_k возмож-ных вариантов локализации инвертируемых битов последовательности составляет $q_k=n/k$. Если выполняется условие $k<<n$, можно при-ближенно считать, что $q_k \approx n^k$. Следовательно, вычислительная сложность нахождения мини-мального значения $L(S\oplus C)$ составляет $O(n^{k+2})$. Учитывая, что длина последовательности, ко-торая трестируется, составляет 10^5-10^6 бит с перспективой увеличения уже в ближайшие го-ды до 10^8 [4], то очевидной становится пробле-ма практи-

ческой оценки тестирования средств получения случайных и псевдослучайных двоичных последовательностей для защиты информации в компьютерных системах и сетях.

Практика тестирования средств получения случайных и псевдослучайных двоичных последовательностей, ориентированных на ис-пользование в системах защиты информации, показала необходимость профиля изменения сложности последовательности при изменении в ней k бит (введении в нее k-кратной ошибки), где k изменяется от единиц до десятков [4]. Поэтому в последние годы интенсивно раз-вивается концепция построения линейных моделей, которые воспроизводят тестируемую последовательность с k-кратной ошибкой.

В последние годы предложены ряд подходов [4], которые позволяют частично снизить вычислительную сложность рассматриваемой задачи тестирования случайности двоичных последовательностей. Снижение вычислитель-ной сложности, достигается за счет частичного использования при формировании новой линей-ной воспроизводящей модели результатов построения предшествующей модели.

В современных условиях имеет место тенденция увеличения длины двоичной последо-вательности, которая потенциально может быть использована для нарушения системы защиты данных. Это суживает возможности тестиро-вания двоичных последовательностей с ис-пользованием линейной воспроизводящей мо-дели, вычислительная сложность построения которой составляет $O(n^2)$.

Поэтому к настоящему времени задача полу-чения оценки возможности построения при-ближенной воспроизводящей модели ПСДП не решена. Исходя из этого, целью исследований является разработка эффективного способа оценки возможности получения приближенной воспроизводящей модели ПСДП по заданному ее фрагменту.

Использование свойств нелинейной воспроизводящей модели для оценки возможности апароксимации ПСДП с заданной погрешностью

Для расширения возможностей тестирования двоичных последовательностей большой длины было предложено использовать нелинейную воспроизводящую модель[2]. Эта модель спо-собна воспроизвести заданную последователь-ность S и представляет собой m-разрядный сдвиговый регистр, текущие значения разрядов которого образуют вектор $X=\{x_1,x_2,\ldots,x_m\}$, с нелинейной, частично-определенной функцией $f(X)$ обратной связи. Нелинейной сложностью $M(S)$ двоичной n-битовой последовательности S является минимальная длина - m сдвигового регистра с нелинейной функцией обратной связи, который воспроизводит последователь-ность S. В работе [2] теоретически доказано, что последовательность S не может быть экстраполирована, если $M(S)=2\cdot\log_2 n$ и предло-жен алгоритм формирования нелинейной вос-производящей модели. Основное достоинство использования нелинейной воспроизводящей модели для тестирования двоичных последовательностей по сравнению с линейной состоит в том, что ее построение имеет меньшую вы-числительную сложность. Так, вычислительная сложность предложенного в [2] алгоритма по-строения нелинейной воспроизводящей модели составляет $O(n\cdot\log_2 n)$, что существенно меньше вычислительной сложности $O(n^2)$ построения линейной воспроизводящей модели.

В настоящей работе предлагается развитие рассмотренной концепции использования не-линейной воспроизводящей модели для тести-рования случайных и псевдослучайных двоич-ных последовательностей. В частности, предла-гается расширение возможностей нелинейной модели за счет ее применения для определения сложности последовательности S с k-кратной ошибкой.

Анализ возможностей применения нелиней-ной воспроизводящей модели для определения сложности n-битовой двоичной последователь-ности S с k-кратной ошибкой позволяет вы-делить два способа решения этой задачи.

Первый способ позволяет оценить упроще-ние воспроизводящей модели при инверти-ровании k битов последовательности на качест-венном уровне, второй - дает возможность про-извести количественную оценку уменьшения длины сдвигового регистра с нелинейной обратной связью.

Сущность первого способа состоит в оценке полученного значения нелинейной сложности $M(S)$. Предложенный в [2] алгоритм получения нелинейной воспроизводящей модели тестиру-емой двоичной последо-

вательности S заключа-ется в построении бинарного дерева G, описы-вающего частично-определенную нелинейную функцию обратной связи, причем число ярусов этого дерева соответствует значению нелиней-ной сложности $M(S)$. Как показано в [2], математическое ожидание $M(S)$ составляет $2\cdot\log_2 n$. Вполне очевидно, что превышение значения $M(S)$ уровня $2\cdot\log_2 n$ косвенно связано с несба-лансированностью бинарного дерева G. В свою очередь, несбалансированность дерева G свиде-тельствует о возможности уменьшения числа его ярусов за счет его балансирования путем изменения k ребер, что тождественно инверти-рованию k битов тестируемой двоичной после-довательности S. Доказано [2], что значение нелинейной сложности имеет нормальное распределение с математическим ожиданием $2\cdot\log_2 n$ и среднеквадратичным отклонением $0.5\cdot\log_2 n$. Соответственно, вероятность P_1 того, что для полностью случайной последователь-ности S значение нелинейной сложности $M(S)$ превысит $2\cdot\log_2 n$, определяется в виде [2]:

$$P_1 = \frac{\sqrt{2}}{\sqrt{\pi}\cdot\log_2 n}\cdot\int_{2\cdot\log_2 n}^{M(S)} e^{-\frac{(x-2\cdot\log_2 n)^2}{0.5\cdot\log_2^2 n}} \qquad (1)$$

Вычисленное с помощью функции Лапласа значение P_1 сравнивается с пороговым значени-ем $P_П$ и, при условии $P_1 > P_П$, принимается решение о несоответствии тестируемой по-следовательности критерию невозможности экстраполяции с небольшой ошибкой.

Предложенный способ прост в реализации и может быть, модифицирован для линейной воспроизводящей модели. В работе [5] указывает-ся, что "последовательность с большим(относи-тельно уровня $n/2$) значением линейной слож-ности может быть аппроксимирована последовательностью с низким уровнем сложности".

Второй способ не предусматривает перебор всех возможных локализаций модифицируемых битов тестируемой последовательности, а осно-ван на анализе функции обратной связи не-линейной воспроизводящей модели.

Сущность второго способа состоит в вычислении веса Хемминга дифференциала частично определенной булевой функции $f(X)$ обратной связи по каждой из переменных $x_1,\ldots,x_m$. Вычисленный вес Хемминга сравнива-ется с заданным значением k - допустимой кратности ошибки аппроксимации.

Вес Хемминга $HW(\partial f(X)/\partial x_1)$ дифференциала частично определенной булевой функции $f(X)$ по переменной x_1, определяется следующим образом [2]:

$$HW\left(\frac{\partial f(X)}{\partial x_1}\right) = \sum_{x_2,\ldots,x_m\in Z_1} \delta(f(x_1,x_2,\ldots,x_m), f(\overline{x_1},x_2,\ldots,x_m)) \qquad (2)$$

где функция $\delta(y_1,y_2) = y_1\oplus y_2$, если обе булевы переменные y_1,y_2 определены, то есть $\delta(y_1,y_2)=0$ и $y_1,y_2\in\{0,1\}$, если значение хотя бы одной из переменных y_1,y_2 не определено; Z_1 -множество всех 2^{m-1} возможных наборов значений буле-вых переменных $x_2,\ldots,x_m$. Поскольку нелиней-ная модель в виде сдвигового регистра с функцией обратной связи $f(X)$ воспроизводит последовательность S, то фактически значение $q=HW(\partial f(X)/\partial x_1)$ соответствует числу оши-бочных битов при воспроизведении последова-тельности S сдвиговым регистром с функцией обратной связи $f(x_1\oplus 1,x_2,\ldots,x_m)$. Так, значение $q=HW(\partial f(X)/\partial x_1)$ равно числу битов последо-вательности S, которые будут неверно воспро-изведены нелинейной моделью без учета стар-шего разряда сдвигового регистра, то есть с по-мощью $(m\text{-}1)$-разрядного сдвигового регистра.

Из сказанного следует, что если выполняется одно из условий $q=HW(\partial f(X)/\partial x_1)\le k$ или $q=HW(\partial f(X)/\partial x_1)>n-k$, тестируемая двоичная последовательность S может быть аппрокси-мирована последовательностью S' с ошибкой $d=min\{q,n-q\}$ не большей, чем в k битах, причем, нелинейная сложность последовательнос-ти S' на единицу меньше по сравнению с нелинейной сложностью заданной последовательности S: $M(S')=M(S)-1$.

Изложенное иллюстрируется следующим примером. Пусть задана двоичная последовательность $S=\{1,0,1,1,0,0,1,0\}$, в точности совпадающая с последовательностью, приведенной в качестве примера в работе [2]. Нелинейная сложность приведенной последовательности S равна 3-м: $M(S)=3$. Значения частично-определенной нелинейной функции $f(x_1,x_2,x_3)$ обратной связи сдвигового регистра приведены в таблице 1.

Табл. 1. Таблица истинности частично-определенной функции $f(x_1,x_2,x_3)$

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	-
0	0	1	0
0	1	0	-
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	0	-

Значение $q=HW(\partial f(x_1,x_2,x_3)/\partial x_1)=1$. Полу-чим, что заданную последовательность S можно аппроксимировать с одним ошибочным битом последовательностью $S'=\{1,0,1,1,0,1,1,0\}$, не-линейная сложность которой равна 2: $M(S'')=2$. Такое значение нелинейной сложности свиде-тельствует о том, что последовательность S' может быть воспроизведена нелинейной мо-делью в виде 2-разрядного сдвигового регистра и функцией $\varphi(x_2,x_3)$ обратной связи, значения которой представлены в таблице 2.

Важным является определения затрат вычислительных ресурсов, требуемых для реализации предложенного способа тестирования. Вычислительная сложность формирования веса Хемминга дифференциала частично определенной нелинейной булевой функции $f(X)$ от m переменных по переменной x_1 определяется перебором половины таблицы истинности $f(X)$ и составляет $O(2^{m-1})$. Для нелинейной модели $m\approx 2\cdot\log_2 n$ [2], поэтому $O(2^{m-1})=O(2\cdot n)$.

Табл. 2. Таблица истинности частично-определенной функции $\varphi(x_2,x_3)$

x_2	x_3	$\varphi(x_2,x_3)$
0	0	-
0	1	1
1	0	1
1	1	0

Если учитывать вычислительную сложность построения нелинейной воспроизводящей модели - $O(n\cdot\log_2 n)$ [2], результатом которого является формирование таблицы, получаем, что вычислительная сложность определения возможности уменьшения на единицу сложности нелинейной воспроизводящей модели за счет d ошибок аппроксимации заданной двоичной последовательности, составляет $O(n\cdot(\log_2 n+2))$. Это означает, что затраты вычислительных ресурсов на решение указанной задачи незначительно превышают вычислительную сложность построения самой нелинейной воспроизводящей модели.

Полученное значение вычислительной слож-ности существенно меньше по сравнению с использованием для решения аналогичной задачи линейной воспроизводящей модели [4], поскольку в последнем случае затраты вы-числительных ресурсов экспоненционально зависят от кратности k допустимой ошибки аппроксимации - $O(n^k)$. Важно, что значитель-ное уменьшение вычислительной сложности для определения возможности упрощения воспроизводящей модели за счет некоторого количества ошибок аппроксимации заданной двоичной последовательности достигается за счет следующих трех факторов:

- Использование нелинейной воспроизводя-щей модели, размерность $2\cdot\log_2 n$ которой существенно меньше размерности $n/2$ линейной модели.

- Применение предложенного подхода сопряжено с существенно большими объемами требуемой памяти по сравнению с методами [4,5], основанными на использовании линейной модели. Это объясняется тем, что при использо-вании нелинейной воспроизводящей модели формируемая нелинейная функция хранится в виде таблицы истинности, а при линейной вос-производящей модели функция обратной связи формируется в алгебраической форме, требую-щей намного меньшего объема памяти. Также, табличное представление функции обратной связи позволяет ускорить операции, связанные с вычислением ошибки аппроксимации. Вместе с тем, повышенный объем требуемой памяти не выходит за рамки возможностей практической реализации. Так, данный способ оценки воз-можности аппроксимации требует памяти, объем которой составляет $2\cdot 2^m = 8\cdot n$ бит. Для типовой длины тестируемой последователь-ности 10^6 бит, требуемый объем памяти - около одного мегабайта, что вполне реализуемо современными техническими средствами.

- В данном способе задача тестирования су-жена: в разработках [4,5] определяется мини-мальная сложность воспроизводящей модели при всех возможных k-битовых ошибках ап-проксимации. Предложенное решение позволя-ет выявить возможность

уменьшения сложнос-ти воспроизводящей модели при всех возмож-ных k-битовых ошибках аппроксимации. На практике тестирования двоичных случайных и псевдослучайных последовательностей определение такой возможности вполне может быть приемлемым результатом [3].

Для определения минимальной сложности нелинейной воспроизводящей модели при всех возможных k-битовых ошибках аппроксимации заданной последовательности, предложенная процедура может быть применена рекурсивно.

Пусть задана последовательность S длиной n бит и кратность допустимых ошибок аппрокси-мации S нелинейной воспроизводящей моделью с минимальной длиной сдвигового регистра составляет k битов. Надо определить минимать-ную длину сдвигового регистра с нелинейной функцией обратной связи, способного вос-произвести последовательность S с ошибками не более чем в k битах.

Рекурсивное применение предложенной про-цедуры осуществляется в следующем порядке:

1. Для заданной двоичной последовательности S строится нелинейная воспроизводящая модель и частично определенная функция $f(X)$ обратной связи, количество m аргументов кото-рой определяет нелинейную сложность $M(S)$ последовательности S.

2. Для функции по формуле (2) вычисляется Хеммингов вес ее дифференциала по перемен-ной, которая соответствует старшему разряду сдвигового регистра: $q=HW(\partial f(x_1,...,x_m)/\partial x_1)$. Вычисляется значение $d=min\{q,n-q\}$.

3. Если $d \le k$, то в последовательности S инвертируются биты, которые обуславливают зависимость функции f(X) от переменной x_1. Значение уменьшается на величину d: $k:=k-d$. Если k>0, то возврат на пп.1, в ином случае $M(S):=M(S)-1$.

4. Если $M(S) < 1.5\cdot\log_2 n$, то проведенное тестирование двоичной последовательности свидетельствует о нецелесообразности приме-нения генератора, с использованием которого получена последовательность, в системах защи-ты информации.

Вычислительная сложность реализации при-веденной рекурсивной процедуры определяет-ся количеством h шагов рекурсии и составляет $O(h\cdot(n+2)\cdot\log_2 n)$. Это значительно меньше по сравнению с вычислительной сложностью $O(n^{k+2})$ решения рассматриваемой задачи с ис-пользованием известных способов, основанных на линейной воспроизводящей модели.

Выводы

В результате проведенных исследований предложен способ тестирования генераторов случайных и псевдослучайных двоичных последовательностей, ориентированных для использования в системах защиты информации. Способ позволяет оценивать возможность экстраполяции двоичных последовательностей с использованием нелинейной воспроизводя-щей модели, а также, за счет меньшей по сравнению с известными методами вычисли-тельной сложности, обеспечивать возможность существенно увеличить длину тестируемой последовательности и, тем самым, позволяет повысить достоверность оценки пригодности генератора случайных или псевдослучайных последовательностей для систем защиты ин-формации.

Так, тестирование возможности аппроксимации последовательностей длиной 10^8 бит известными методами сопряжено со значительными техническими трудностями [3], в то время как экспериментально доказано, что тестирование таких последовательностей с использованием предложенного способа осуществляется достаточно просто. Обеспечи-ваемая предложенным способом возможность тестирования длинных последовательностей позволяет существенно повысить достовер-ность оценки качества генераторов случайных и псевдослучайных двоичных последователь-ностей применительно к их использованию в системах защиты информации. Достигаемое использованием разработанного способа увеличе-ние оперативности тестирования потенциально позволяет повысить качество проектирования средств генерации случайных и псевдослучай-ных последовательностей.

Список литературы

1. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. М.: КУДИЦ-ОБРАЗ.- 2003 – 260 с.

2. Марковский А.П., Мустафа Акрам Ареф Найеф, Бойко А.В. Об одном подходе к определению сложности случайных и псевдослучайных двоичных последовательностей // Вісник національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. – 2002.- №37.- С.120-129.
3. Самофалов К.Г., Марковський О.П., Абабне О.А. Оцінка якості генераторів двійкових послідовностей з використанням нелінійної відтворюючої моделі // Проблеми інформатизації та управління. Збірник наукових праць: Випуск 1(19).-К.,НАУ.- 2007.- С.142-147.
4. Чугунков И.В. Система оценки качества генераторов псевдослучайных кодов. // Научная секция МИФИ-2000.- Т.11,12.- С.45-46.
5. Gutierrez J., Shparlinski I.,A., Winterhof A. On the Linear and Nonlinear Complexity Profile on Nonlinear Pseudorandom Number Generators // IEEE Trans. Information Theory.-2003.- Vol. 49.- № 1. – P.60-64.
6. Kurosava K., Sato F., Sakata T., Kishimoto W. A relationship between linear complexity and k-error linear complexity. // IEEE Trans. Information Theory, 2000.- V.46,-№3.-P.694-698.
7. Meidl W., Niederreiter H. On the expected value of the linear complexity and the k-error linear complexity of periodic sequences // IEEE Transaction of Information Theory.- 2002.- Vol. 48.- № 11. - P.2817-2825.
8. NIST Special Publicaion 800-22: A Statistical Test Suite for Random and Pseudorandom Number. 2000. -348 p.

УДК 004.054

БОЛДАК А.А.
НЕВДАЩЕНКО М.В.
ДЕМЬЯНЕНКО М.В.

СИСТЕМА ФАКТОРОВ ДЛЯ КОМПАРАТИВНОГО АНАЛИЗА ПРОГРАММНЫХ ПРОЕКТОВ

В статье разрабатывается система факторов, вычисляемых на основе параметрической модели COCOMO, которая может применятся в рамках компаративного анализа программных проектов. Факторы определяются на основе результатов корреляционного анализа параметрических оценок COCOMO для выборки из 156 программных проектов. Редукция параметрического пространства с помощью метода главных компонент подтверждает корректность определения разработанной системы факторов.

In this paper, we develop a system of factors, calculated on the basis of a parametric model COCOMO, which can be applied in the comparative analysis of software projects. Factors are determined on the basis of the results of correlation analysis of COCOMO parametric estimates for a sample of 156 software projects. Reduction of the parameter space using the method of principal components confirms the correctness of determination of the developed system factors.

ВВЕДЕНИЕ

Программный проект (ПП) как процесс предполагает разработку программного продукта надлежащего качества при условии ограничений на сроки и ресурсы.

Сложностью планирования этого процесса и необходимостью объективной оценки его состояния объясняется интерес к использованию различных моделей параметрической оценки ПП[1-4]. Одной из таких моделей, которая широко используется при планировании ПП, является модель COCOMO (Constructive Cost Model), предложенная Барри Боэмом в 1981 году[2].

В соответствии с этой моделью для вычисления затрат, необходимых для выполнения ПП (в человеко-месяцах) используется формула:

$$effort = a \cdot KLOC^{b} \cdot \prod_{i=1}^{15} X^{i}, \qquad (1)$$

где *KLOC* – количество строк программного кода (тыс. стр. кода); *a*, *b* – константы, значения которых зависят от категории ПП, X^{i} – критерии затрат (метрики).

В этой модели используется разбиение программных проектов на 3 категории:

- organic – небольшие ПП, разработанные группой людей до 5 человек;
- embedded – ПП в рамках сложных взаимосвязанных операционных (встроенных) аппаратных средств и (или) программного обеспечения;
- semi-detached – ПП, которые занимают промежуточное положение между organic и embedded.

Такое разделение является важным, поскольку позволяет уточнить значения констант *a* и *b* (табл. 1).

Табл. 1. Значения констант a и b в зависимости от категории ПП

Категория ПП	a	b
organic	3.2	1.05
embedded	2.8	1.2
semi-detached	3.0	1.12

Метрики X^{i} для ПП сведены в табл. 2 и могут принимать одно из 6 значений: vl (very low) – неудовлетворительное, l (low) – удовлетворительно, n (normal) – нормальное, h (high) – хорошее, vh (very high) – очень хорошее, xh (extra high) – отличное. Для каждого из этих значений определены соответствующие числовые эквиваленты[2].

Ниже определяется система факторов, вычисляемых на основе метрик COCOMO, которая может эффективно использоваться при компаративном анализе ПП.

Табл. 2. Критерии затрат ПП

Категория критериев	Обозн.	Описание критерия
Критерии ПП	RELY	Требуемая надежность ПП
	DATA	Размер базы данных
	CPLX	Сложность ПП
Критерии техники	TIME	Ограничения по быстродействию
	STOR	Ограничения по оперативной памяти
	VIRT	Изменяемость виртуальной машины – степень, в которой изменяется аппаратура и ПО (ОС, СУБД и т.п.)
	TURN	Цикл обращения к аппаратуре
Критерии исполнителей	ACAP	Квалификация аналитика
	AEXP	Опыт работы в данной области
	PCAP	Квалификация программиста
	VEXP	Опит работы с виртуальной машиной (ОС)
	LEXP	Опыт работы с языком программирования
Критерии проекта	MODP	Использование современных методик программирования
	TOOL	Использование инструментальных средств
	SCED	Ограничение сроков разработки

ПОСТАНОВКА ЗАДАЧИ

В работе используются данные о 156 ПП, представленные в [5], которые содержат значения 15 метрик, а также количество строк кода (KLOC) и реальные затраты (REFF). Эти данные были дополнены уровнем планируемых затрат (PEFF), расчитанным по формуле (1). Эти данные можно представить в виде:

$$X_i = \left\langle x_i^1, x_i^2, ..., x_i^m \right\rangle, \qquad (2)$$

где X_i – оценка i-го ПП; $x_i^1, x_i^2, ..., x_i^m$ – значения параметров оценки ПП; m – количество параметров оценки ПП, которое в нашем случае равно 18.

Существует рад причин обусловливающих применение методов многомерного статистического анализа (МСА) для описанных выше данных. Во-первых, исходные параметры X^i в силу тех или иных причин являются взаимосвязанными. Незнание характера и степени этих связей может существенно искажать реальную оценку ПП. Указанные характеристики взаимосвязей определяются с помощью методов корреляционного анализа. Во-вторых, для удобства и наглядности компаративного анализа ПП необходимо свести исходные метрики к нескольким скрытым (латентным) факторам. В данном случае оценка ПП может быть упрощенно представлена не 18 параметрами, а несколькими, наиболее существенными факторами. Для этих целей используют метод главных компонент.

ИСПОЛЬЗОВАНИЕ МСА В РАМКАХ КОМПАРАТИВНОГО АНАЛИЗА ПП

Поскольку, все параметры $X^i, i = \overline{1,18}$ выражены в интервальных шкалах, то степень линейной связи между отдельными параметрами можно оценить с помощью коэффициента корреляции Пирсона [6]:

$$r_{k,l} = \frac{\sum_{i=1}^{N}(x_i^k - \overline{X^k}) \cdot (x_i^l - \overline{X^l})}{\sqrt{\sum_{i=1}^{N}(x_i^k - \overline{X^k})^2} \cdot \sqrt{\sum_{i=1}^{N}(x_i^l - \overline{X^l})^2}},$$

где $r_{k,l}$ – коэффициент корреляции параметров X^l и X^k; $\overline{X^l} = \frac{\sum_{i=1}^{N} x_i^l}{N}; \overline{X^k} = \frac{\sum_{i=1}^{N} x_i^k}{N}$ – средние значения параметров X^k и X^l; N – количество ПП.

Из [6, 7] следует, что коэффициенты корреляции, которые по модулю больше 0,70, говорят о сильной связи параметров (при этом коэффициенты детерминации > 50%, т.е. один признак определяет другой более, чем наполовину). Коэффициенты корреляции, которые по модулю меньше 0,7, но больше 0,5, говорят о связи средней силы (при этом коэффициенты детерминации меньше 50%, но больше 25%). Наконец, коэффициенты корреляции, которые по модулю меньше 0,5, говорят о слабой связи параметров (при этом коэффициенты детерминации меньше 25%).

Коэффициенты корреляции для параметров с сильной и средней степенью связи можно представить в виде графа, вершинами которого являются параметры, а ребра соот-

ветствуют корреляционной связи высокого и среднего уровней.

Такой граф для исследуемых данных представлен на рис. 1. Как видно, множество параметров разбивается на 7 групп:

1. RELY, CPLX, TIME и STOR.
2. KLOC, REFF и PEFF.
3. LEXP, VEXP и VIRT.
4. DATA и TURN.
5. ACAP и PCAP.
6. MODP и TOOL.
7. SCED.

Каждой группе в соответствие может быть поставлена одна интегральная оценка (фактор).

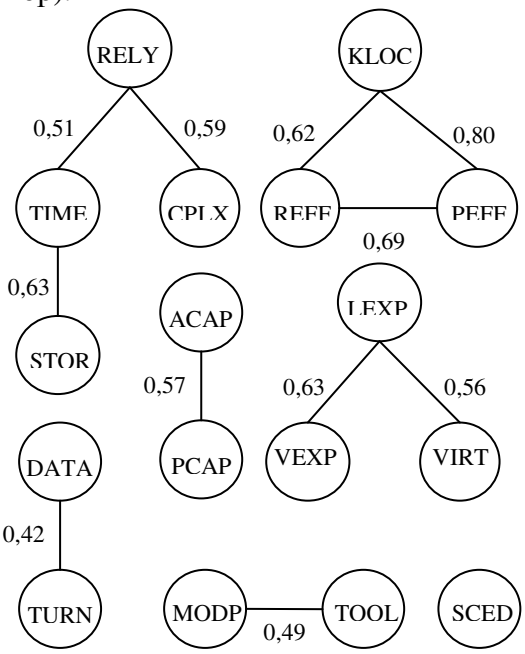


Рис. 1. Граф корреляционных связей

ПРИМЕНЕНИЕ МЕТОДА ГЛАВНЫХ КОМПОНЕНТ ДЛЯ АНАЛИЗА МЕТРИК ОЦЕНКИ ПП

Использование метода главных компонент позволяет выявить минимальное число скрытых (латентных) факторов, отображающих основные свойства исходных 18 параметров, и одновременно уменьшить степень зависимости этих факторов от своих остаточных случайных компонент.

Осуществим поиск нового ортонормированного базиса векторов-компонент [8,9], определив разложение матрицы X в виде:

$$X = T \cdot P^T + E, \qquad (2)$$

где T – матрица счетов размерностью $n \times m'; (m' \le m)$; P – матрица нагрузок размерностью $m' \times m$; (m' - размерность пространства параметров; m - количество главных компонент, выбранных для проецирования); E – матрица остатков.

Определение главных компонент (факторов) связано с вычислением собственных векторов ковариационной матрицы [9,10], определяемой как:

$$C = (c_{ij}, c_{ij} = \mathrm{cov}(X^i, X^j)), i = \overline{1,m},\ j = \overline{1,m}, \qquad (3)$$

где $\mathrm{cov}(X^i, X^j) = \dfrac{\sum_{k=1}^{n} (x_k^i - \overline{X^i}) \cdot (x_k^j - \overline{X^j})}{n-1}$ – ковариация признаков X^i и X^j.

Из работы [10] следует, что сумма дисперсий параметров равна сумме дисперсий всех факторов, и они упорядочены в соответствии с долями их дисперсий. Поэтому анализируя изменение относительной доли дисперсии, вносимой первыми m' факторами можно определить число факторов, которое целесообразно оставить для последующего рассмотрения. При этом для выбора достаточного числа $m' \le m$ факторов часто используют кумулятивную дисперсию [10]:

$$D_i = \frac{\sum_{j=1}^{i} \lambda_j}{m}, i = \overline{1,m},$$

где $\lambda_j, j = \overline{1,m}$ – собственные числа ковариационной матрицы C.

В соответствии с критериями Кайзера [9] (см. рис. 2) и критериями Кэттеля [11] (табл. 3) для последующего исследования достаточно оставить семь факторов. Это позволит представить около 77% данных о ПП (табл. 3).

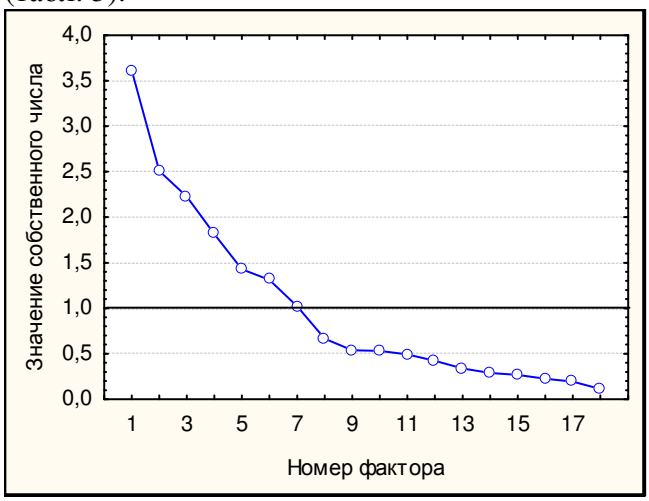


Рис. 2. График собственных чисел факторов

Матрица нагрузок P из формулы (2) задает отображение базиса исходного пространства X^i в пространство факторов T^j, в котором можно оценить значимость каждой из метрик.

Табл. 3. Таблица собственных значений и дисперсий факторов

Номер фактора	Собственное значение	Общая дисперсия, %	Суммарное собственное значение	Суммарная дисперсия, %
1	3,61	20,08	3,61	20,08
2	2,51	13,94	6,12	34,02
3	2,23	12,37	8,35	46,39
4	1,82	10,08	10,17	56,47
5	1,43	7,94	11,59	64,41
6	1,31	7,29	12,91	71,70
7	1,02	5,67	13,93	77,36

Проанализировав матрицу нагрузок в пространстве 18 метрик и семи факторов для исходных данных, можно сделать следующие выводы:

- метрики RELY, CPLX, TIME и STOR, с коэффициентами нагрузок 0.82, 0.77, 0.75 и 0.74 соответственно, составляют фактор 1 – требования к надежности, сложности, быстродействию и объемам оперативной памяти;
- метрики KLOC, REFF и PEFF, с коэффициентами нагрузок 0.91, 0.80 и 0.91, составляют фактор 2 – трудоемкость ПП и затраты на его производство;
- метрики LEXP и VEXP, с коэффициентами нагрузок равными 0.87, определяю фактор 3 – использование традиционных технологий;
- метрики DATA и TURN, с коэффициентами 0.74 и 0.84, определяют фактор 4 – требования к аппаратному обеспечению;
- метрики ACAP и PCAP, с коэффициентами 0.81 и 0.84, определяют фактор 5 – квалификация персонала;
- метрики MODP и TOOL, с коэффициентами 0.75 и 0.86, определяют фактор 6 – использование современных средств разработки программного обеспечения;
- метрика SCED, с коэффициентом нагрузки 0.92, определяет фактор 7 - ограничения на время разработки.

Согласованность результатов корреляционного и факторного анализов дает возможность предложить модель параметрической оценки ПП, суть которой заключается в использовании интегральных показателей. Для получения этих показателей необходимо провести корреляционный анализ и построить граф корреляционных связей. При этом число компонент связности соответствует числу интегральных показателей. Поскольку коэффициенты нагрузок параметров в каждом факторе примерно одинаковы, то каждый из показателей определяется как среднее арифметическое значений параметров, соответствующих компоненте связности.

ЗАКЛЮЧЕНИЕ

Установлено, что в рамках компаративного анализа ПП может быть использована система факторов, в которую входят требования к надежности, сложности, быстродействию и объемам оперативной памяти, трудоемкость ПП и затраты на его производство, использование традиционных технологий, требования к аппаратному обеспечению, квалификация персонала, использование современных средств разработки программного обеспечения, ограничения на время разработки.

Числовые значения этих факторов можно определять как средние арифметические параметров оценки COCOMO. Для этих целей необходимо провести корреляционный анализ и построить граф корреляционных связей для выборки ПП.

СПИСОК ЛИТЕРАТУРЫ

1. Andrew Stellman, Jennifer Greene. Applied Software Project Management // Sebastopol, MA: O'Reilly Media. – 2005. – 308 p.
2. Barry Boehm. Software engineering economics // Englewood Cliffs. – 1981. – 308 p.

3. Lawrence H. Putnam, Ware Myers. Five core metrics : the intelligence behind successful software management // Dorset House Publishing. – 2003. – 328 p.
4. Albrecht A. J. Measuring Application Development Productivity // Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium. – 1979. – P.83-92.
5. Данные для модели COCOMO // Режим доступа: http://unbox.org/wisp/trunk/cocomo/data, свободный. – Загл. с экрана. – Яз. англ.
6. Айвазян С. А. и др. Прикладная статистика: Основы моделирования и первичная обработка данных. Справочное изд. // С. А. Айвазян, И. С. Енюков, Л. Д. Мешалкин. — М.: Финансы и статистика, 1983. — 471с.
7. Айвазян С.А. и др. Прикладная статистика: Исследование зависимостей: Справ, изд. // С. А. Айвазян, И. С. Енюков, Л. Д. Мешалкин; Под ред. С. А. Айвазяна. – М.: Финансы и статистика, 1985. – 487 с, ил.
8. Lindsay I Smith. A tutorial on Principal Components Analysis. – 2002. – 208 p.
9. Kaiser H. F. The application of electronic computers to factor analysis // Educational and Psychological Measurement. – 1960. – P.141-151.
10. Прикладная статистика: Классификации и снижение размерности: Справ, изд. // С. А. Айвазян, В. М. Бухштабер, И. С. Енюков, Л. Д. Мешалкин; Под ред. С. А. Айвазяна. – М.: Финансы и статистика, 1989. – 607 с: ил.
11. Cattell R. B. The scree test for the number of factors // Multivariate Behavioral Research. – 1966. – P.245-276.

УДК 681.327.12.001.362

ДОРОГОЙ Я.Ю.,
ЯШИН В.Е.,
ЯЦУК С.В.

МНОГОПОТОЧНАЯ ЭМУЛЯЦИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ ПРОИЗВОЛЬНОЙ ТОПОЛОГИИ

В статье рассмотрена структура программного комплекса PANN, позволяющего использовать преимущества многопоточной обработки данных для эмуляции работы искусственных нейронных сетей произвольных топологий.

The structure of the software complex PANN was described in this article. PANN allows the use of the advantages of multithreading for emulation of artificial neural networks of arbitrary topologies.

Введение в проблему

На данный момент, изучение и использование нейронных сетей, снова набирает обороты [1]. Это обусловлено развитием компьютерной техники и постоянного роста вычислительных мощностей, доступных исследователю. Вместе с тем, инструменты для эмуляции работы и предварительного расчета нейронных сетей, развиваются слабо. Одним из преимуществ нейронных сетей является их скорость обработки данных. Хорошо спроектированная и обученная сеть работает на порядки быстрее аналогичных решений с использованием классических алгоритмов. Но, к сожалению, процесс проектирования и обучения нейронной сети занимает несравнимо больше времени, чем прямое ее использование. Зачастую для достижения необходимых результатов нужно перебрать десяток топологий. А сам процесс обучения занимает много времени ввиду необходимости обработки больших объемов данных. Таким образом, необходимо иметь инструмент для проектирования нейронных сетей, который позволил бы задействовать максимум ресурсов компьютера. Сегодня даже на домашних компьютерах стоят многопроцессорные системы, которые при оптимизированном программировании позволяют ускорить работу приложений в несколько раз. К сожалению, ни одна из известных на данный момент систем для проектирования нейронных сетей не использует возможности компьютера на полную мощность. Это объясняется тем, что большинство из таких пакетов написаны давно, когда многопроцессорные системы были мало распространены и не имело смысла оптимизировать код под такие системы. Сейчас же, имея материнскую плату с 2-мя процессорами по 4 ядра в каждом, можно ускорить обучение нейронной сети в 8 раз, что позволило бы существенно сократить время проектирования и тестирования.

Анализ существующих решений

Сейчас существуют такие пакеты для проектирования нейронных сетей:
SNNS – мощная библиотека, разработанная в Штуттгардском университете. Большая часть кода написана еще в начале 90-х на чистом C, без использования объектно-ориентированного подхода. Это сильно усложняет дальнейшее развитие.

Joone – более современный пакет, написан на Java, что несколько отражается на скорости работы. Преимуществом является полностью объектная модель. Но сама по себе библиотека малоразвита, основной упор поставлен на визуализацию, видимо, поэтому разработчики мало времени уделили развитию ядра.

Matlab Neural Network Toolbox - это пакет расширения MATLAB, содержащий средства для проектирования, моделирования, разработки и визуализации нейронных сетей. Основной недостаток - очень низкая скорость работы.

Общим недостатком для всех существующих пакетов, является однопоточность обработки, а также представление нейронной сети исключительно слоями. Слоистое представление упрощает разработку приложения, но отсекает возможность проектирования нейронных сетей с произвольной топологией.

Проблемы многопоточности

Современные операционные системы многозадачны. Это позволяет пользователю запускать одновременно несколько приложений. Операционная система позволяют

запускать большое количество приложений даже на одном физическом процессоре. Такая квазимногозадачность реализовывается путем постоянного переключения активной задачи. Каждое приложение в системе на какой-то квант времени захватывает процессор и выполняет необходимые действия, затем ОС переключает активность на другое приложение и так в бесконечном цикле.

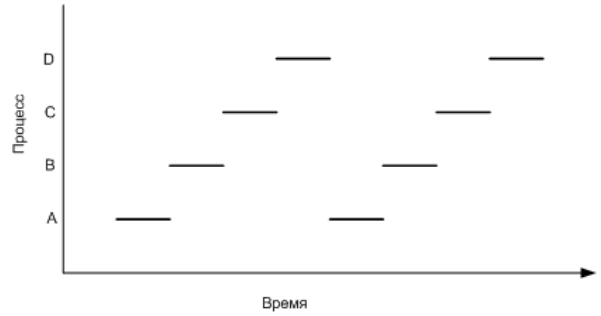


Рис. 1. Многозадачная работа процессора

Если в системе доступно множество процессоров (ядер), то ОС делит процессы на доступные ядра и тогда несколько процессов могут работать действительно одновременно. Более того, можно распараллелить вычисления одного приложения на несколько физических процессоров и тем самым увеличить скорость выполнения сложных вычислений [2].

Основной проблемой при работе многопоточных приложений являются ресурсы компьютера – устройства, потоки ввода/вывода, файлы, данные. Сложность состоит в том, что если несколько потоков используют один и тот же ресурс, то могут возникнуть коллизии. Представим ситуацию: один поток открывает файл для записи данных, и в это же время второй поток удаляет этот файл. В результате первый поток получает невалидный указатель на несуществующий файл. Другой пример, когда два потока пытаются одновременно получить доступ в какую-то область данных. Допустим, существует общая переменная, в которую два потока пишут данные, такое приложение будет нестабильным и работать будет неправильно. Для обеспечения уникальности и атомарности доступа к конкурентным ресурсам, операционная система дает несколько инструментов:

- Критическая область — часть исполняемого кода, который гарантировано выполнится без переключения текущего процесса.
- Семафор — объект, позволяющий войти в заданный участок кода не более чем n потокам.
- Мьютексы — это простейшие двоичные семафоры, которые могут находиться в одном из двух состояний — отмеченном или неотмеченном (открыт и закрыт соответственно). Когда какой-либо поток, принадлежащий любому процессу, становится владельцем объекта mutex, последний переводится в неотмеченное состояние. Если задача освобождает мьютекс, его состояние становится отмеченным.

В разных системах реализация таких системных вызовов отличается, поэтому мы использовали обертку boost::thread, которая унифицирует работу с потоками и дает удобные инструменты.

Цель работы

Таким образом, было принято решение о создании собственного программного пакета для проектирования нейронных сетей с использованием преимущества многопроцессорных систем. Основными требования к продукту стали:

- Высокая скорость обработки данных.
- Многопоточность
- Объектно-ориентированный дизайн
- Модульность программного пакета
- Мультиплатформенность
- Высокая стабильность.

Идея многопоточной обработки

Ввиду изложенных выше сложностей работы многопоточных приложений, было необходимо минимизировать взаимное использование одних и тех же данных разными потоками, так как неправильно спроектированное многопоточное приложение может работать медленнее однопоточного, ввиду возможности постоянной борьбы за уникальные ресурсы [3].

Ранее принятое решение об отказе от канонического послойного представления нейронной сети позволяет создавать сети произвольных топологий. Но так как обработка данных все равно идет последовательно, было введено понятие "расстояния между нейронами" - hop и "длины связи" - link latency [4]. Рассмотрим схему сложной части нейронной сети со сквозной связью. Кругами представлены нейроны N1, N2 и N3. Цифра возле круга - это hop. Цифра на связи - это

дальность связи. Такой подход позволяет проектировать топологии любой сложности (рис. 2).

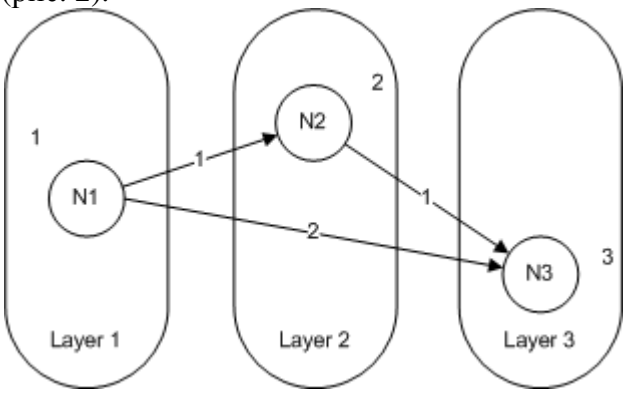


Рис. 2. Представление сложных топологий

Таким образом, мы делим сеть произвольной топологии на условные слои обработки.

Разберем многопоточную обработку нейронной сети на примере многослойного персептрона (рис.3).

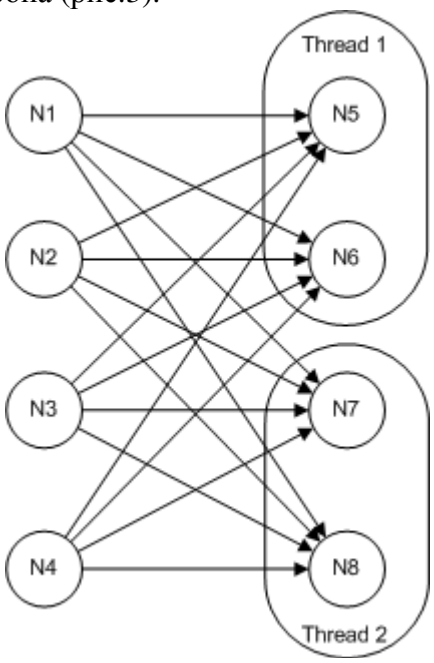


Рис. 3. Часть многослойного персептрона

Для расчета локального рецептивного поля нейрона N5 нужно прочитать значения выходов нейронов N1-N4, умножить на веса и просуммировать, затем те же операции нужно повторить со следующим нейронами N6-N8. Доступ к первому слою будет исключительно на чтение, а значит, обработку нейронов N5-N8 можно выполнять параллельно.

Точно так же можно применить данный алгоритм многопоточной обработки и для сетей более сложной топологии (например, к сверточным нейронным сетям).

Очевидно, что алгоритм обучения обратным распространением ошибки отличается от прямого прохода сети только направлением. Это означает, что данный подход позволяет легко выполнять обучение сети в несколько потоков.

В представленном пакете каждый условный слой делится на несколько частей (по количеству ядер в системе). Далее каждый поток обрабатывает свою часть данных. Очевидно, что для обработки следующего условного слоя, нам необходимо полностью посчитать результаты предыдущего слоя. Чтобы потоки двигались по сети равномерно, после каждого условного слоя ставиться специальный объект boost::barrier, который гарантирует синхронизацию потоков. Таким образом, обработка идет волнами от слоя к слою.

Выбор языка программирования и окружения

Существует множество языков программирования для реализации каких угодно задач. И важно правильно выбрать инструмент, ведь каждый язык имеет как преимущества, так и недостатки. В соответствии с предоставленными требованиями, были отвергнуты интерпретируемые языки, так как они существенно медленнее компилируемых. Для достижения стабильности пришлось отказаться от языков с динамической типизацией. В конце концов, список сузился до трех кандидатов: Java, C# и C++. Java была отвергнута ввиду наличия виртуальной машины. C# имеет проблемы с переносимостью. Поэтому мы остановили свой выбор на C++. Это достаточно старый, популярный, объектно-ориентированный язык программирования. C++ позволяет максимально использовать ресурсы компьютера, современные компиляторы генерируют настолько чистый код, что он очень близок к ассемблерной реализации. Таким образом, C++ дает нам возможность максимально использовать аппаратные ресурсы ПК. Также неоспоримым плюсом является наличие огромного количества библиотек для C++, что в какой-то мере упрощает разработку. Естественно не обошлось без препятствий – стандарт C++ не поддерживает многопоточность, но, к счастью, существуют библиотеки позволяющие писать потокобезопасные приложения на C++, в частности библиотека Boost. Эта

библиотека является очень мощным подспорьем при разработке приложений на C++, она имеет множество часто используемых компонентов, и новый стандарт разрабатывается на ее базе, что обеспечивает хорошую совместимость в будущем.

При написании вспомогательных утилит (графического интерфейса, юнит-тестов и т.д.) были задействованы следующие наборы библиотек:

✓ Для написания графического интерфейса для сопутствующих программ, использован Qt – кросс-платформенный инструментарий разработки ПО. Qt позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

✓ Для построения графиков и анализа тенденций использована библиотека построения графиков и анализа тенденций используется gnuplot – свободная программа для создания двух- и трёхмерных графиков. Gnuplot имеет собственную систему команд, может работать интерактивно (в режиме командной строки) и выполнять скрипты, читаемые из файлов.

Структура пакета

Чтобы обеспечить модульность и легкость изменения кода пакета, пришлось изначально заложить некоторые абстракции. Рассмотрим диаграмму классов пакета PANN (рис. 4):

Рис. 4. Диаграмма связей основных классов пакета

Для упрощения дальнейшей поддержки и развития проекта классы заведомо создавались со слабой связанностью. Все классы унаследованы от общего класса Object, что позволяет унифицировано работать с ними. Для динамического добавления различных временных свойств объектам введен класс AttributeName. Атрибуты позволяют быстро и удобно добавлять данные необходимые для текущих расчетов.

Каждый нейрон содержит набор ссылок на связи и веса, что позволяет работать со всеми нейронами одинаково (будь то входной или выходной слой). Веса отделены от связей для упрощения работы и экономии памяти на топологиях с общими весами (сверточные нейронные сети). Для уменьшения нагрузки процессора используются класс Cache и его наследник NetCache, который содержит информацию о текущем фронте обработки нейронной сети.

Особняком стоит набор классов Runner (рис. 5).

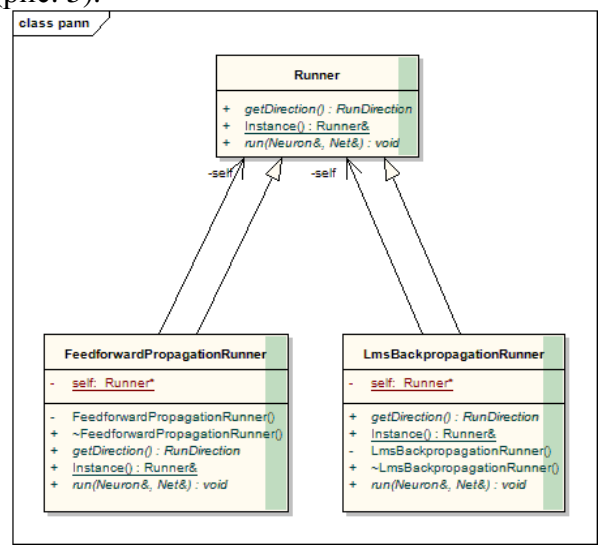


Рис. 5. Иерархия классов Runner

Класс Runner инкапсулирует в себе информацию о методе обучения сети. Как видно на первой диаграмме, класс Net содержит метод run. Именно тут формируется контекст запуска сети, создается необходимое количество потоков и их запуск.

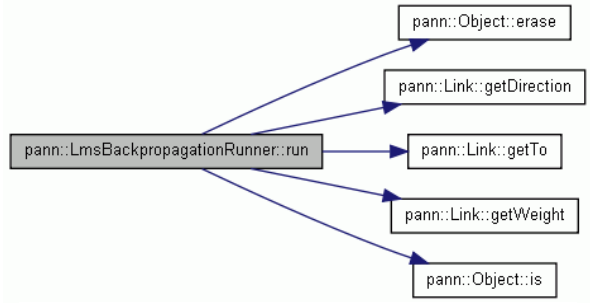


Рис. 6. Диаграмма вызова метода run у объекта LmsBackpropagationRunner

Как видно из рисунка 6, Runner сам вызовет необходимые методы обработки у каждого связи в соответствии с заданным алгоритмом.

Таким образом, мы имеем очень гибкую, слабо связанную систему базовых классов, которую можно с легкостью расширять и дополнять.

Пример работы со сверточной нейронной сетью

Рассмотрим структуру сверточной нейронной сети, топология которой в дальнейшем будет применятся для распознавания образов.

В 1981 году нейробиологи Торстен Визел и Девид Хабел исследовали зрительную кору головного мозга кошки и обнаружили, что существуют так называемые простые клетки, которые особо сильно реагируют на прямые линии под разными углами и сложные клетки, которые реагируют на движение линий в одном направлении.

Позже Ян ЛеКун предложил использовать так называемые сверточные нейронные сети, как аналог зрительной коры головного мозга для распознавания изображений.

Идея сверточных нейронных сетей заключается в чередовании сверточных слоев (C-layers), субдискретизирующих слоев (S-layers) и наличии полносвязных (F-layers) слоев на выходе (рис. 7).

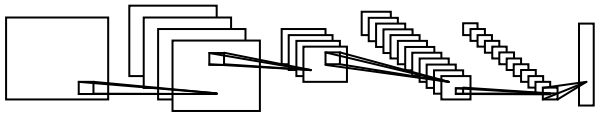


Рис. 7. Архитектура сверточных нейронных сетей

Такая архитектура заключает в себе 3 основных парадигмы:

1. Локальное восприятие
2. Разделяемые веса
3. Субдискретизация

Локальное восприятие подразумевает, что на вход одного нейрона подается не все 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 очень небольшой набор весов. Т.е. если у нас имеется на входе изображение размерами 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. Иными словами это будет карта признаков.

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, которое называется картой признаков. Следует сказать, что в идеале не разные фрагме-

нты проходят последовательно через ядро, а параллельно все 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, 2 по вертикали и 3 по горизонтали. Субдискретизация нужна 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.

Эксперименты и результаты

Для проведения ряда экспериментов с пакетом была выбрана топология сверточной нейронной сети [5]. Сеть состоит из семи слоев, в которых в общей сложности 5161 нейрон и 132418 связей. В качестве функции активации был выбран тангенс гиперболический.

Эксперимент проводился на персональном компьютере с 8-мю процессорами по 2 гигагерца каждый.

После проведения ряда запусков, были получены усредненные данные (табл. 1):

Табл. 1. Результаты эксперимента

Кол-во потоков	1	2	3	4	5	6	7	8	9	10	11	12
Время (с)	89,8	67,9	55,7	50,8	46,7	44,4	42,1	40,8	43,7	42,4	42,6	42,4

Для упрощения анализа построим график на рисунке 8:

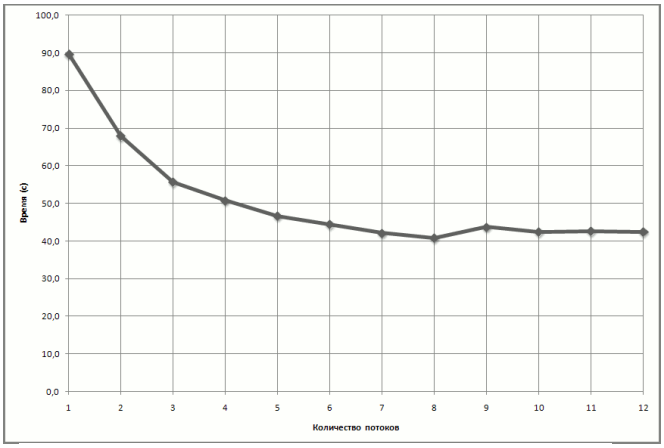


Рис. 8. График зависимости времени обучения сети от количества процессоров

Как видно, с ростом количества задействованных процессоров, время обработки падает. Так продолжается, пока количество потоков не начинает превышать количество доступных процессоров. В таком случае время обработки снова увеличивается, это обусловлено необходимостью переключения потоков между доступными процессорами. Также следует заметить, что скорость обработки от доступных процессоров не линейна - это результат синхронизации потоков с помощью boost::barrier. Больший выигрыш по времени можно получить при еще больших слоях, ведь, как было описано выше, параллельно обрабатывается каждый слой, а поле его обработки необходимо синхронизировать потоки обработки.

Теперь усложним обработку. Возьмем ту же топологию сверточной нейронной сети и проведем обучение алгоритмом обратного распространения ошибки. Получены такие усредненные результаты (табл. 2):

Табл. 2. Результаты эксперимента

Кол-во потоков	1	2	3	4	5	6	7	8	9	10	11	12
Время (с)	241,8	145,1	91,4	68,6	51,4	43,7	35,0	31,5	33,6	32,4	32,7	32,1

График затрат времени обработки при использовании разного количества ядер представлен на рисунке 9:

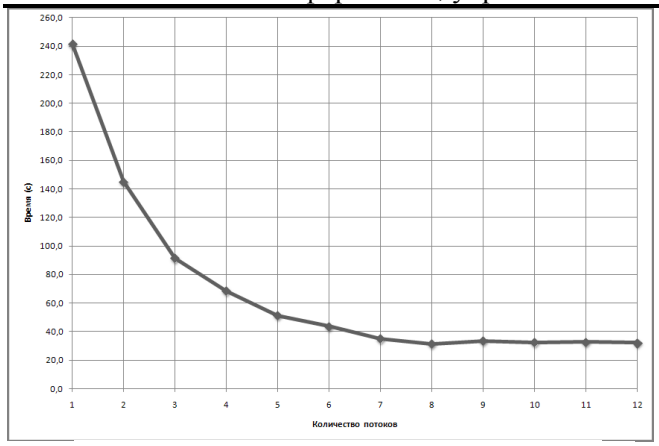


Рис. 9. График зависимости времени обучения сети от количества процессоров

Построенный график схож с графиком первого эксперимента. Но ввиду сложности вычислений для каждого нейрона получен значительно больший выигрыш от многопоточной обработки. При использова-нии 8-ми ядер скорость обработки возросла практически в 8 раз.

Выводы

В ходе разработки пакета PANN были созданы уникальные алгоритмы параллельной обработки нейронных сетей с произвольной топологией. В процессе создания данного пакета разработчики преодолели множество сложностей проектирования и разработки концепции. Пришлось уйти от канонического послойного представления топологии, что позволило проектировать самые сложные и немыслимые топологии. В результате имеем мощный, гибкий и быстрый пакет для проектирования, тестирования и использования нейронных сетей с произвольной топологией.

Список литературы

1. Neural Networks: A Comprehensive Foundation (2nd Edition), Simon Haykin, 842 pages,Prentice Hall; 2 edition (July 16, 1998),ISBN-10: 0132733501

2. Операционные системы: разработка и реализация, Таненбаум Э. С., Вудхалл А. С., 576 стр., 2005 г., изд. Питер, ISBN 5469001482.

3. Алгоритмы: построение и анализ, Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн, 296 стр., с ил.; ISBN 978-5-8459-0857-5, 0-07-013151-1; формат 70x100/16; твердый переплет2008, 4 кв.; Вильямс.

4. Дорогой Я.Ю., Яшин В.Е. Программный комплекс для симуляции многопоточных нейронных сетей.//Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №49. – 2008. – С. 123-127.

5. Gradient-Based Learning Applied to Document Recognition (Proc. IEEE 1998), Yann Lecun, Léon Bottou, Yoshua Bengio, Patrick Haffne.

Поступила в редакцию

УДК 52(15).003

KUSSUL O., NOVIKOV O.

UTILITY-BASED REPUTATION MODEL FOR VO IN GRIDS

В данной статье предложена модификация существующей модели репутаций для виртуальных организаций в Grid-системах, которая основана на оценке функции полезности. Модификация модели состоит в добавлении статистической модели поведения пользователя, которая ранее была разработана для компьютерных сетей и распределенных систем, а также компонентов, которые позволяют противостоять угрозам в области управления доверием и репутацией. К числу этих компонентов относятся: механизм присвоения начальной репутации для новых субъектов виртуальной организации; учет взаимосвязей между пользователями и ресурсами; функция учета времени; а также классификация предоставляемых сервисов в Grid-системе.

In this paper we extend the existing utility-based reputation model for VOs in Grids by incorporating a statistical model of user behaviour (SMUB) that was previously developed for computer networks and distributed systems, and different functions to address threats scenarios in the area of trust and reputation management. These modifications include: assigning initial reputation to a new entity in VO, capturing alliance between consumer and resource, time decay function, and score function.

Introduction

The key idea of Grid-systems is a coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [1]. Therefore; mutual trust between users and resource providers is of high importance.

Paper [2] demonstrates that trust is enabling technology and its implementation can provide the possibility to secure electronic transactions. Meanwhile trust is described as an important and sophisticated object dealing with honesty, truthfulness and reliability of trusted person or service. Nevertheless there still there is no common definition of trust [3]. Two main definitions can be given:

— ”When we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him. Correspondingly, when we say that someone is untrustworthy, we imply that that probability is low enough for us to refrain from doing so” [4]

— ”the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of a relative security, even though negative consequences are possible” [5]

A set of individuals and/or institutions defined by coordinated resource sharing rules form what we call a virtual organization (VO).

Virtual organization is a temporary or permanent union of geographically spread individuals and/or institutions defined by coordinated resource sharing rules, possibilities and information for reaching common goals [1].

VO’s are formed dynamically, exist for some time and then resolve. That is why the efficiency of their work depends on trust.

For monitoring and creation of such VO’s the reputation based trust management technologies are needed.

Currently, such trust models are built based on entity reputation assessment and mostly where created for other areas of distributed systems.

In common case under the reputation we understand the measure of reliability. With the help of reliability we can build a trust of one entity to another. According to [6], reputation is an assumption about the agent’s behavior based on existing information or observations about his behavior before.

In this paper we extended the existing utility-based reputation model [8] by incorporating a statistical model of user behaviour (SMUB) that was previously developed for computer networks and distributed systems [10, 11, 12, 13]. This model was originally used in intrusion detection systems to detect anomalous patterns of user actions. Other modifications include:

- *assigning initial reputation to a new entity in VO*: when organization provides a new resource to be integrated in VO there are no records from monitoring system to infer reputation value for this specific resource. One possible way of assigning initial reputation to a new

resource is to use the methodology of active experiment. There can be several benchmark tasks in the system to estimate the utility function and to provide initial reputation of the resource.

- *alliance between consumer and resource*: since reputation of resource is based on measure of satisfaction of a consumer in relation to this resource we should avoid cheating via collusions among a group of entities [14]. For this purpose, it is advisable to include into the model a factor that will reflect an alliance between the consumer and the resource.

- *time decay function*: reputation of resource is based on measuring average value of utility function over certain period of time [14]. But if VO exists for a considerable period of time (e.g. for years) reputation of resource may vary considerably. That is, it is unlikely to use, for example, two years data to estimate current resource reputation if more recent records are available. So, we propose to incorporate a time lag function into the model that will provide weights depending on the time of the transaction record between consumer and resource.

- *score function*: for different types of services offered by resource providers different reputation values will be used [15]. Namely, we will categorize services into categories, and a resource provider will get reputation value according to such a category. In Grid systems, tasks can be categorised by computational complexity. Successful execution of tasks with complex workflow and parallel programs (for example, environmental models like numerical weather prediction [16]) will provide to a resource provider higher reputation value.

We will also study the described reputation model against the existing attacks and how the model overcomes them. In particular, we will follow security threat scenarios presented in [15]. These scenarios include: individual malicious peers, malicious collectives, malicious collectives with camouflage, malicious spies, sybil attack, man in the middle attack, driving down the reputation of a reliable peer, partially malicious collectives, malicious pre-trusted peers.

Related Works

Up to date, there have been proposed several approaches of reputation models in grid systems.

PathTrust [14] is a reputation system suggested for choosing members of the VO whiles its formation. The organization has to register with the enterprise network by providing some certificates to enter the VO. Besides of user management enterprise network provides centralized reputation service. When the VO is resolved each member gives feedback values for reputation server and other members he had interacted with. The proposed model feels the lack of dynamics, because the feedback value is collected only when the VO is resolving.

Paper [8] proposes utility-based reputation model that can be used for users, resources and resource providers' assessment. This model is based on the calculating of the utility function that expresses the satisfaction of the entity with its interaction with other entities with respect to the key features specific for the assessable entity. For resource reputation assessment this trust model demands the existence of monitoring system for the quality of service data collection. But this paper does not highlight the question of assigning initial reputation values for new users or resources.

Paper [9] is solving the task of assessment of trust level between the agents of multi-agent system using the direct observation data and reputation information. Unlike other existing approaches that are based on the usage of heuristics the proposed approach HABIT (Hierarchical And Bayesian Inferred Trust Model) is using statistical Bayesian methods. The advantage of this model is the possibility of assessing trust values for new agents by searching for correlation with known groups of agents. It is extremely important in cases when the truster has no previous experience or reputation data of trustee. In the model truster-trustee trust is sighted from the providing desirable quality of service point of view. It is obvious that the rational agent is functioning in purpose to gain the maximum of expected utility.

Utility-based Reputation Model for VOs in Grids

The reputation model described in this paper is based on the model proposed in [8]. The main additions are associated with reputation model for VO users, and extensions of three functions to resource provider model. In our model we will also use slightly modified notations as comparing to [8]. All proposed modifications are described in details in the following subsections.

Basic notations used in our paper

The central notion in the reputation model is the organisation [8]. The organisation provides resources; and there exist users associated with this organisation (Fig. 1).

Definition 1. Organization is a set:

$$Org = \left\{ o_id, \bigcup_i r_i, \bigcup_j u_j \right\}, \quad (1)$$

where *o_id* is an organisation's identifier, $\bigcup_i r_i$ and $\bigcup_j u_j$ are resources and users associated with this organisation, respectively.

We will denote all existing organisations with $\bigcup_n Org_n$.

Virtual organization (VO) can be modelled as a set of organisations. Organisations integrate their resources on a temporary or permanent basis to achieve common goals [1]. It is to be noted that in general case organisation may provide to a VO only a subset of its resources, and the same resource can be used in different VOs. The same stands for users of the organisation.

Definition 2. VO is a set (Fig. 1):

$$VO = \left\{ vo_id, \bigcup_k r_k, \bigcup_l u_l, f_{vo_id}(), g_{vo_id}() \right\}, \quad (2)$$

where *vo_id* is a VO's identifier, $\bigcup_k r_k$ and $\bigcup_l u_l$ are resources and users from multiple organisations that participate in *VO*, respectively, $f_{vo_id}()$ and $g_{vo_id}()$ are membership functions defined in the following way

$$f_{vo_id} : \bigcup_k r_k \rightarrow \bigcup_n Org_n, \text{ i.e. } f_{vo_id}(r_k) = o_id \quad (3)$$

$$g_{vo_id} : \bigcup_l u_l \rightarrow \bigcup_n Org_n, \text{ i.e. } g_{vo_id}(u_l) = o_id \quad (4)$$

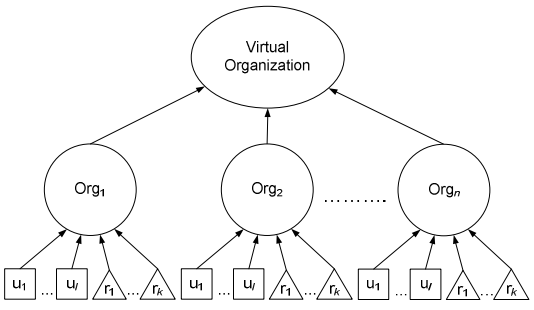


Fig. 1. Correspondence between virtual organization, organizations, resources and users

In general case, these functions can be maintained by the Virtual Organisation Membership Service (VOMS). Using these functions we can retrieve any required information on membership of organisations, resources and users in VOs. According to (3) a set of resources provided by the organisation with identifier *o_id* in a specific VO with identifier *vo_id* is given by

$$\{ r \in \bigcup_k r_k : f_{vo_id}(r) = o_id \} \quad (5)$$

We will denote this set as $f^{-1}{}_{vo_id}(o_id)$.

In the same, way according to (4), we can list all users from organisation with identifier *o_id* participating in VO with identifier *vo_id*:

$$\{ u \in \bigcup_l u_l : g_{vo_id}(u) = o_id \} \equiv g^{-1}{}_{vo_id}(o_id) \quad (6)$$

Suppose we want to list all organisations from $\bigcup_n Org_n$ that provide resources within specific VO with identifier *vo_id*, or whose users participate in this VO. According to (3), (4) such sets are given respectively by

$$\{ Org \in \bigcup_n Org_n : \text{if } \exists\ r \in \bigcup_k r_k \text{ that } f_{vo_id}(r) = o_id \} \quad (7)$$

$$\{ Org \in \bigcup_n Org_n : \text{if } \exists\ u \in \bigcup_l u_l \text{ that } g_{vo_id}(u) = o_id \}. \quad (8)$$

Let us denote with $\bigcup_m VO_m$ all existing VOs. Suppose we want to retrieve all VOs where a resource *r* from a specific organisation *Org* is used, or where user *u* from specific *Org* with identifier *o_id* participates in. According to (2), (3), (4) these sets are given respectively by

$$\{ VO \in \bigcup_m VO_m : f_{vo_id}(r) = o_id \} \equiv VO|_r, \quad (9)$$

$$\{ VO \in \bigcup_m VO_m : g_{vo_id}(u) = o_id \} \equiv VO|_u. \quad (10)$$

These basic notions are used in the following subsections to describe reputation models for resource providers and users.

Reputation model for resource providers

The reputation model is based on the utility function that measures the level of satisfaction of a user in relation to resource provider. In order to define utility function an auxiliary function that indicates the service level agreement (SLA) accorded between a VO user and a resource provider for a particular resource within a VO is implemented [8]

$$SLA : \bigcup_l u_l \times \bigcup_k r_k \times \bigcup_m VO_m \rightarrow \mathrm{R} \quad (11)$$

where R denotes the set of real numbers.

SLA value represents quality of resource provider as expected by user [8]. Quality of Service (QoS) metrics that can be used to measure the level of satisfaction is described in [17, 18]. Some papers also propose mechanisms for QoS negotiation and handling [19].

Definition 3. Event is a set:

$$Event = T \times \bigcup_l u_l \times \bigcup_k r_k \times \bigcup_m VO_m \times \{\text{QoS name}\} \times \mathrm{R} \quad (12)$$

where T is a time domain.

Therefore, the event is characterised by the following attributes

$\{t, u, r, vo_id, QoS, v\}$

where t indicates time, QoS is a name indicating QoS of interest, and v is a real QoS value measured by Grid monitoring system after user-resource interaction.

Definition 4. Trace is the sequence of events (12):

$$Trace = \bigcup_p Event_p = \bigcup_p \{t, u, r, vo_id, QoS, v\}_p . \quad (13)$$

Before defining utility function and reputation we will introduce three functions: the first one will characterise possible alliance between consumer and resource in order to avoid cheating, the second one will account for a time when utility was estimated, and the third one will provide different scores depending on the type of the provided service. These functions provide extensions to utility function and reputation originally proposed in [8].

Function $h(u, r)$ will take value between 0 and 1 and will show the level of alliance between user u and resource r. If there is no such alliance between targets, $h(u, r)$ will have a higher value. For example, one possible way of defining $h(u, r)$ is as follows

$$h(u,r) = \begin{cases} 1, \text{ if } f_{vo_id}(r) \neq g_{vo_id}(u) \\ \theta, \text{if } f_{vo_id}(r) = g_{vo_id}(u) \end{cases} \quad (14)$$

where θ is a parameter.

Function $z(t, t_c)$ will show what past records on user-resources interactions should be taken into consideration to estimate reputation of specific resource. Here t is the time, and t_c is a parameter. In a simplest form $z(t, t_c)$ could be a stepwise function

$$z(t,t_c) = \begin{cases} 1, t \geq t_c \\ 0, t < t_c \end{cases} \quad (15)$$

Function $s(type(r))$ will provide different values for different types of services provided by the resource r (function $type(r)$ maps into category of service).

Now, we can define a utility function according to (11), (12), (14):

$$utility : Event \rightarrow \mathrm{R},$$

$$utility(\{t, u, r, vo_id, QoS, v\}) = \begin{cases} h(u,r)s(r), & \text{if } v \geq SLA(u,r,vo_id) \\ \dfrac{v}{SLA(u,r,vo_id)} h(u,r)s(r), & \text{if } v < SLA(u,r,vo_id) \end{cases} \quad (16)$$

.

Let us denote a set of traces that are used to estimate the reputation of resource r in VO with identifier vo_id up to the current time t with

$$Trace|_{(vo_id,\ r,\ t)} = \left\{ \begin{array}{l} \{t', u', r', vo_id', QoS', v'\} \in Trace : \\ r = r', vo_id = vo_id', t' \leq t \end{array} \right\}. \quad (17)$$

Let us denote a set of *utility()* function values derived from traces $Trace|_{(vo_id,\ r,\ t)}$ (15), (17) with

$$O_{(vo_id,\ r,\ t)} = \{\ z(t, t_c) \cdot utility(\{t, u, r, vo_id, QoS, v\}) \mid \{t, u, r, vo_id, QoS, v\} \in Trace|_{(vo_id,\ r,\ t)}\} \quad (18)$$

Definition 5. Reputation is expectation of *utility*() function:

$$rep(vo_id, r, t) = \mathrm{E}[\ utility(O_{(vo_id,\ r,\ t)})\ ] = \int utility(O_{(vo_id,r,t)})\ p_{utility}(O_{(vo_id,r,t)})\ dO_{(vo_id,r,t)} \quad (19)$$

.

If we do not want to discriminate values from *utility*() function by time then we might use $z(t, t_c) = 1$.

In order to approximate expectation we can use a sample mean (18)

$$rep(vo_id, r, t) = \frac{1}{\left| O_{(vo_id,r,t)} \right|} \sum_{x \in O_{(vo_id,r,t)}} x \quad (20)$$

where $|\cdot|$ denotes the cardinality of a set.

The reputation of an organisation in VO is the aggregation of the reputation of all resources it provides to VO according to (5), (15), (19):

$$rep(vo_id, t) = \frac{1}{\left| f^{-1}_{vo_id}(o_id) \right|} \sum_{r \in f^{-1}_{vo_id}(o_id)} rep(vo_id, r, t) . \quad (21)$$

The reputation of a resource in all VOs according to (19) can be estimated as follows

$$rep(r, t) = \frac{1}{\left|\mathrm{VO}\right|_r} \sum_{vo_id \in \mathrm{VO}|_r} rep(vo_id, r, t). \quad (22)$$

Reputation model for users

In the reputation model proposed in [8] the corresponding model for user is built using a penalty function. If a user performs an action that does not conform to a VO or resource policy, the user is charged with a penalty. This penalty is used to estimate utility function for the user, and subsequently a reputation of the user. But audit of user actions and checking these actions against the VO or resource policy is a difficult task, especially from implementation perspective. There should be clearly defined criteria to assess user actions and corresponding program components to do that.

We propose to include into the reputation model a statistical model of user behaviour (SMUB) that was developed for computer networks and further modified for distributed systems, in particular Grid systems [10, 11, 12, 13]. This model is based on the analysis of statistical data that is gathered after a user executes actions in a Grid system. The model was trained and verified on real data that was collected from GILDA infrastructure (https://gilda.ct.infn.it) of the EGEE project. The model was able to discriminate behaviour of different types of users and to detect synthetically generated intrusions with a rate of more than 90%.

In particular, the model accounts for different statistical parameters given by the following attributes

{*S*, *ET*, *CPU*, *WT*, *CW*, *ES*, *CT*, *STD*, *RAM*, *VM*, *VO*, *RB*},

where *S* is a site where a user task was executed, *ET* is a execution target, *CPU* is a task CPU time, *WT* is a task wall time, *CW* is defined as CPUWall = CPU/W, ES is a task exit status (success or with errors), *CT* is a task creation time in Grid system, *STD* is a start time difference, i.e. difference between time when a task started to execute on computational resource and time when task was sent to Grid system by the user, *RAM* is a volume of RAM used by user task, *VM* is a volume of virtual memory used by user task, *VO* is a virtual organization a user belongs to, *RB* is a resource broker hostname that was used to schedule the user task.

This set of parameters is used to detect anomalous patterns in user actions to raising alarms in the system. Such patterns could include, for example, situations when a task is running for a significant amount of time, or when the CPU utilization is up to 100% [20]. In order to detect such patterns from data that was logged during user activities we use intelligent techniques, namely neural networks [21]—a multilayer perceptron (MLP) trained by means of extended-delta-bar-delta (EDBD) algorithm [22] which represents a fast modification of standard error backpropagation algorithm [23]. For each user a neural network is trained to form an opinion to discriminate between the normal and abnormal user behaviour. When neural network is trained a target output is set to 1 for input data that corresponds to the normal user behaviour, and to 0 for data that corresponds to the abnormal user behaviour. In order to represent both situations (normal and abnormal) in training data sets we use records from Grid monitoring system: records about past user actions represent the normal behaviour while records from other users and synthetically generated data represent the abnormal user behaviour. The synthetic data can be generated using generative models [24] and incorporated into the training data sets in order to represent abnormal patterns that were not present in data sets from monitoring system. Therefore, neural network acts as a classifier. If we put an independent sample (not present in the training data set) to neural network input the output value will be between 1 and 0. This value can be treated as a posterior probability of normal/abnormal user behaviour: higher values correspond to normal actions while lower values correspond to potential anomalous patterns.

In such a way, we propose to use the output of the SMUB model in order to estimate the reputation of user in VO. Such a user model will be specific to a VO depending on its goals and types of tasks being executed. For example, VO can be oriented on applications that require execution of large number of tasks with relatively small amount of data to be processed by a single task. In such a VO, tasks that consume almost full amount of RAM and virtual memory of resources would be considered as an anomalous pattern. In turn, other VOs can be oriented on applications where a single task consists of a number of elementary tasks each processing large amount of data (an example includes Earth science domain and satellite data processing [25, 26]).

Let us provide a formal description of the reputation model for user based on the SMUB model as it was done for the reputation model for resource provider in previous subsection.

Definition 6. *Event* for the user is a set:

$$Event = \{t, u, r, vo_id, \mathbf{x}\}. \quad (23)$$

where $\mathbf{x} = (S, ET, CPU, WT, CW, ES, CT, STD, RAM, VM, VO, RB)$.

Definition 7. Trace corresponds to the sequence of events:

$$Trace = \bigcup_p Event_p = \bigcup_p \{t, u, r, vo_id, \mathbf{x}\}_p . \quad (24)$$

An analogue of *utility*() function (as it was defined for resource providers) is defined in the following way

$$utility : Event \rightarrow \mathrm{R},$$
$$utility(\{t, u, r, vo_id, \mathbf{x}\}) = SMUB_{(u,\ vo_id)}(\mathbf{x}), \quad (25)$$

where $SMUB_{(u,\ vo_id)}(\mathbf{x})$ is an output of the SMUB model.

It should be noted that, in general case, under *utility*() function for users we can use other user behaviour models (e.g. [27, 28]) or a combination of different models to capture different aspects and patterns of user behaviour.

In our case, according to (24), the SMUB transformation is performed by a neural network, and specific to user and VO. Let us denote a set of traces that are used to estimate the reputation of user u in VO with identifier vo_id up to the current time t with

$$Trace|_{(vo_id,\ u,\ t)} = \left\{ \{t', u', r', vo_id', \mathbf{x}'\} \in Trace : \begin{array}{l} u = u', vo_id = vo_id', \\ t' \le t \end{array} \right\}. \quad (26)$$

Let us denote a set of *utility()* function values derived from traces $Trace|_{(vo_id,\ u,\ t)}$, according to (15), (25), (26) with

$$O_{(vo_id,\ u,\ t)} = \{\ z(t, t_c) \cdot utility(\{t, u, r, vo_id, \mathbf{x}\}) \mid \{t, u, r, vo_id, \mathbf{x}\} \in Trace|_{(vo_id,\ u,\ t)}\} \quad (27)$$

Definition 8. Reputation is the expectation of *utility*() function (25),(27)

$$rep(vo_id, u, t) = \mathrm{E}[\ utility(O_{(vo_id,\ u,\ t)})\ ] = \int utility(O_{(vo_id,u,t)})\ p_{utility}(O_{(vo_id,u,t)})\ dO_{(vo_id,u,t)} \quad (28)$$

.

In order to approximate expectation we can use a sample mean

$$rep(vo_id, u, t) = \frac{1}{\left|O_{(vo_id,u,t)}\right|} \sum_{x \in O_{(vo_id,u,t)}} x . \quad (29)$$

The reputation of an organisation in VO (from users' perspective) is the aggregation of the reputation of all users that participate in VO. According to (6), (29)

$$rep(vo_id, t) = \frac{1}{\left|g^{-1}_{vo_id}(o_id)\right|} \sum_{r \in g^{-1}_{vo_id}(o_id)} rep(vo_id, u, t) . \quad (30)$$

According to (2), (10), (29) the reputation of a user in all VOs can be estimated as follows

$$rep(r, t) = \frac{1}{\left|\mathrm{VO}\right|_u\right|} \sum_{vo_id \in \mathrm{VO}|_u} rep(vo_id, u, t) \quad (31)$$

Analysis of Attacks

In this section we will study different security threats scenarios in the area of trust and reputation management that were proposed in [15], and analyse how the proposed model responds to these attacks. It should be noted that some of these attacks can be handled by the existing mechanisms already implemented in Grid systems.

Individual malicious peers. Malicious peers always provide bad services. From Grid perspective, there can be either a resource that always provides unreliable services, or malicious user that always tries to harm a system. Such an unreliable resource will provide poor services to the users that will result in $v << SLA(u, r, vo_id)$, and thus the reputation of this resource will be low. Moreover, such resource will get no or few number of tasks when reputation is incorporated into resource broker to schedule tasks among resources of Grid system. As to the malicious users, the output of the user behaviour model $SMUB_{(u,\ vo_id)}(\mathbf{x})$ will be always low, close to 0. Thus, tasks submitted by the user could be cancelled.

Malicious collectives. This is a threat when malicious peers form a malicious collective. In Grids, there could be a user that tries illegally to improve the reputation of a particular resource. If the user and resource belong to the same organization that kind of behaviour will be captured by the alliance function $h(u, r)$. In order to improve the reputation value considerably the user will need to submit a lot of simple tasks. In such a case the reputation value of the resource will be bounded with the $s(type(r))$ function. If the user wants to keep illegally the reputation of

the resource at a high level over a long period of time, he or she will need to constantly submit tasks since the reputation is accumulated only over specific period of time. Such behavioural patterns will be detected by the SMUB model, and the user may be punished with a reputation for such actions.

Malicious collectives with camouflage. This is a threat which is not always easy to tackle, since its resilience will mostly depend on the behavioural pattern followed by malicious peers [15]. These correspond to the malicious collectives with the variable behaviour. In our model, such variability could be detected using SMUB model. Moreover, reputation value for such users will vary considerably over the time as well. Therefore, with such an approach it is possible to punish such wit the reputation.

But it is to be noted that patterns of user behaviour could be various, and it is very difficult to build a model that will capture all the possible situations. Thus, the success of detecting malicious collectives with camouflage greatly depends on the model of user behaviour.

Malicious spies. This a threat when malicious peers (spies) always provide good services when selected as service providers, but they also give the maximum rating values to those malicious peers who always provide bad services. In Grids this corresponds to the situation when a user with high reputation provides the maximum rating to unreliable resources. This type of threat was already described in *malicious collectives* subsection.

Sybil attack. An adversary initiates a large number of malicious peers in the system. Each time one of the peers is the system is scheduled as a resource provider it provides malicious service and after that it is disconnected and replaced by another peer [15, 20, 29]. In Grids, such an attack is hardly implemented since appropriate certificate should be obtained from certificate authority in order to integrate a resource into the Grid system. Another way to tackle this problem is to use the methodology of active experiment to monitor the availability of resources.

On the other hand, this attack may become a threat when Grids are integrated with the Sensor Web or sensor networks. Sybil attack implemented through the sensor web could cause harm to Grid entities. But security issues relating to the integration of Grids and Sensor Web are out of scope of this paper and should be investigated in the future works.

Man in the middle attack. A malicious peer can intercept the messages between other peers, rewrite the message and change reputation values. Our model relies on the existing Grid security mechanisms to tackle this attack [20].

Driving down the reputation of a reliable peer. Malicious peers give the worst rating to those benevolent peers, who indeed provide good services. Projecting on to the Grids, a malicious user will provide poor ratings to the resources, though user's tasks were completed successfully with appropriate QoS value. One possible way to tackle this problem is that tasks of users with low reputation value are never sent to resources with high reputation by the resource broker. Moreover, QoS metrics in Grids are measured by the monitoring system, and to change these values, and consequently reputation, the malicious user should crack it.

Partially malicious collectives. Malicious peers provide malicious actions for some kind of services, and, for others, they provide good services. In Grids, this threat corresponds to users with variable behaviour (covered in *malicious collectives with camouflage* subsection), and to resources that for some types of services provide poor. By just considering a different score for every service offered by a resource, this threat is mitigated most of the times [15]. That is why we included into our model a score function to provide different reputation values for different types of tasks executed by resources.

Malicious pre-trusted peers. Some models are based on the strategy that there is a set of peers that can be trusted before any transaction is carried out in the system (known as pre-trusted peers) [30]. Our model does not include the notion of pre-trusted peers.

Reputation Management System

Developed model could be used by VOs with reputation management system. Paper [8] presents the architecture of such reputation management system in order to facilitate the rating of both VO resources and VO users. The architecture is shown in Figure 2.

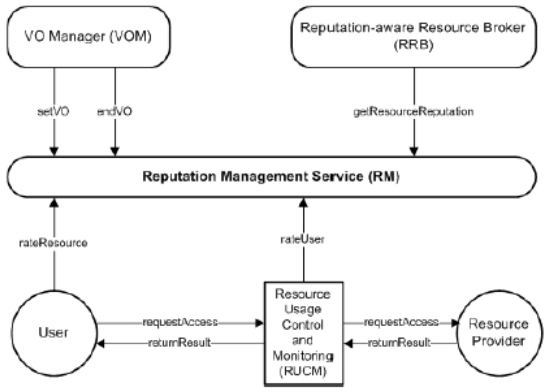


Fig. 2. Reputation management in VOs

Conclusions

In this paper we extended the existing utility-based reputation model [8] by incorporating a statistical model of user behaviour (SMUB) that was previously developed for computer networks and distributed systems, and different functions to address threats scenarios in the area of trust and reputation management that were proposed in [15]. These modifications include: assigning initial reputation to a new entity in VO, capturing alliance between consumer and resource, time decay function, and score function. We analysed the described reputation model against the existing attacks and how the model overcomes them following security threat scenarios presented in [15].

References

1. Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations // Int J of Supercomputing Applications. – 2000. – 15(3). – P. 200-222.
2. Grandison T., Sloman M. A Survey of Trust in Internet Applications // IEEE Communications Survey and Tutorials. –2000. – 3.
3. McKnight D.H., Chervany N.L. The Meaning of Trust. Technical Report MISRC Working Paper Series 96-04, University of Minnesota. Management Information Systems Research Center, 1996.
4. Gambetta D. Can We Trust Trust? In: D. Gambetta (ed). Trust: Making and Breaking Cooperative Relations. Department of Sociology, University of Oxford, 1988.
5. Josang A., Ismail R., Boyd C. A Survey of Trust and Reputation Systems for Online Service Provision // Decision Support Systems. – 2007. – 43(2). – P. 618-644.
6. Abdul-Rahman A., Hailes S. Supporting trust in virtual communities // In: Proceedings of the IEEE 33rd Hawaii Int Conf on System Sciences (HICSS '00). – 2000. – vol 6. – P. 6007.
7. Kerschbaum F., et al. A trust-based reputation service for virtual organization formation // In: Proceedings of the 4th Int Conf on Trust Management. Lecture Notes in Computer Science Springer. – 2006. – vol. 3986. – P. 193–205.
8. Arenas A., Aziz B., Silaghi G.C. Reputation management in grid-based virtual organisations // In: Fernandez Medina E., Malek M., Hernando J. (ed.) Proc. International Conference on Security and Cryptography (SECRYPT 2008), Porto, Portugal. – 2008. – P. 538-545.
9. Luke T.W.T., Jennings N.R., Rogers A., Luck M. A Hierarchical Bayesian Trust Model based on Reputation and Group Behaviour // In: 6th European Workshop on Multi-Agent Systems, Bath, UK. – 2008.
10. Shelestov A., Skakun S., Kussul O. Intelligent Model of User Behavior in Distributed Systems // Int J on Inf Theory and Applications. – 2008. – 15(1). – P. 70–76.
11. Shelestov A., Skakun S., Kussul O. Complex Neural Network Model of User Behavior in Distributed Systems // In: Proc. of XIII-th International Conference Knowledge-Dialogue-Solutions. Varna, Bulgaria. – 2007. – P. 42–49.
12. Skakun S., Kussul N., Lobunets A. Implementation of the Neural Network Model of Users of Computer Systems on the Basis of Agent Technology // J. of Automation and Inf. Sci. – 2005. – 37(4). –P. 11–18.
13. Kussul N., Skakun S. Neural Network Approach for User Activity Monitoring in Computer Networks // In: Proc. of the Int. Joint Conf. on Neural Networks, Budapest, Hungary. – 2004. – 2. – P. 1557–1562.
14. Azzedin F., Maheswaran M. Integrating Trust into Grid Resource Management Systems // In: First IEEE International Workshop on Security and Grid Computing. – 2002.

15. Gomez Marmol F, Martınez Perez G (2009) Security threats scenarios in trust and reputation models for distributed systems. Computers & Security 28: 545–556
16. Kussul N, Shelestov A, Skakun S (2009) Grid and sensor web technologies for environmental monitoring. Earth Science Informatics 2(1-2): 37-51
17. Menasce DA, Casalicchio E (2004) Quality of service aspects and metrics in Grid computing. In: Proc. 2004 Computer Measurement Group Conference, Las Vegas, USA.
18. Hong-Linh T, Samborski R, Fahringer T (2006) Towards a Framework for Monitoring and Analyzing QoS Metrics of Grid Services. In: Proc. Second IEEE Int Conf on e-Science and Grid Computing (e-Science'06).
19. Al-Ali R, von Laszewski G, Amin K, Hategan M, Rana O, Walker D, Zaluzec N (2004) QoS Support for High-Performance Scientific Grid Applications. In: Proc. IEEE International Symposium on Cluster Computing and the Grid 2004. (CCGrid 2004). pp 134–143.
20. Chakrabarti A (2007) Grid Computing Security. Springer-Verlag Berlin Heidelberg
21. Haykin S (1999) Neural Networks: A Comprehensive Foundation. Upper Saddle River, New Jersey, Prentice Hall
22. Minai AA, Williams RJ (1990) Back-propagation heuristics: A study of the extended delta-bar-delta algorithm. In: IEEE Int Joint Conf on Neural Networks vol I pp. 595-600
23. Werbos PJ (1994) The roots of backpropagation: from ordered derivatives to neural networks and political forecasting. John Wiley & Sons, Inc., New York
24. Bishop CM (2006) Pattern Recognition and Machine Learning. Springer
25. Shelestov A, Kussul N, Skakun S (2006) Grid Technologies in Monitoring Systems Based on Satellite Data. J of Automation and Inf Sci 38(3): 69–80.
26. Fusco L, Cossu R, Retscher C (2007) Open Grid Services for Envisat and Earth Observation Applications. In: Plaza AJ, Chang C-I (ed) High performance computing in remote sensing, 1st edn. Taylor & Francis Group, New York, pp 237-280.
27. Oh SH., Lee WS (2003) An anomaly intrusion detection method by clustering normal user behavior. Computers & Security 22(7): 596–612.
28. Wang W, Guan X, Zhang X, Yang L (2006) Profiling program behavior for anomaly intrusion detection based on the transition and frequency property of computer audit data. Computers & Security 25(7) 539–550.
29. Douceur JR, Donath JS (2002) The Sybil attack. In: Proceedings for the 1st international workshop on peer-to-peer systems (IPTPS ’02). pp 251–60.
30. Kamvar S, Schlosser M, Garcia-Molina H (2003) The EigenTrust algorithm for reputation management in P2P networks, Budapest, Hungary.

Поступила в редакцию 16.12.2009

УДК 004.04

МУХИН В.Е.,
ВОЛОКИТА А.Н.

АНАЛИЗ СОБЫТИЙ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ ДЛЯ ПРОВЕДЕНИЯ КОРРЕКТИРУЮЩИХ ДЕЙСТВИЙ ПО УПРАВЛЕНИЮ БЕЗОПАСНОСТЬЮ

Статья посвящена анализу событий информационной безопасности в соответствии с международным стандартом ISO/IEC 27001:2005 «Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Требования». Проведение данного анализа позволит выявить взаимосвязь между событиями информационной безопасности, применить соответствующие меры управления и провести корректирующие действия с целью предупреждения повторного возникновения инцидентов безопасности.

The article is devoted to the analysis of the information security events in accordance with international standard ISO/IEC 27001:2005 “Information technology. Security techniques. Information security management systems. Requirements”. This analysis allows to reveal the relationship between the information security events, to apply the appropriate management mechanisms and to perform the corrective actions for the security incidents recurrence prevention.

Введение

Информационная безопасность компьютерных систем и сетей (КСС) подразумевает сохранение информацией свойств конфиденциальности, целостности и доступности [1]. Построение системы управления информационной безопасностью в соответствии с международным стандартом ISO/IEC 27001:2005 «Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Требования» позволяет организациям получить следующие преимущества [2]:

- уменьшение и оптимизация стоимости поддержки системы безопасности;
- идентификация информационных активов, определение их владельцев и поддержка в актуальном состоянии перечня активов;
- регулярное выявление угроз и уязвимостей безопасности для существующих бизнес-процессов;
- выявление, оценка и контроль рисков;
- эффективное управление информационными активами предприятия в критичных ситуациях в соответствии с утвержденным планом по восстановлению бизнеса.

Проведение мониторинга и анализа системы менеджмента информационной безопасности является требованием стандарта ISO/IEC 27001.

Мониторинг безопасности представляет собой комплекс мер и мероприятий (технических, организационных и правовых), направленных на реализацию наблюдения, анализа и прогнозирования состояний безопасности сложных систем. Однако, существующие алгоритмы для системам менеджмента информационной безопасности (СМИБ) не позволяют провести комплексную оценку действий злоумышленников, выстроить цепочку реализованных уязвимостей, определить их конечную цель и оценить риски безопасности информационных ресурсов, что может приводить к финансовым потерям, а также повторным успешным атакам на компьютерные системы и сети.

Стандарт ISO/IEC 27001 содержит требования к разработке, внедрению, функционированию, мониторингу и анализу СМИБ, но не содержит готовых алгоритмов, которые могут быть использованы для управления информационной безопасностью.

Таким образом, актуальным является создание новых алгоритмов, обеспечивающих анализ событий информационной безопасности и проведение соответствующих корректирующих действий для своевременного предупреждение инцидентов безопасности.

Алгоритм анализа событий информационной безопасности

На основании анализа существующих подходов и методов мониторинга безопасности предложен новый алгоритм анализа событий

информационной безопасности, в котором формируется взаимосвязь между событиями и выполняется их группировка [3].

Для выявления взаимосвязи между событиями будем использовать коэффициент корреляции, характеризующий зависимость случайных величин. На основании статистических данных мониторинга строится матрица распределения событий информацион-ной безопасности (x_i, x_j), пример которой приведен ниже в виде таблицы 1.

Табл.1. Матрица распределения значений двух факторов.

x_b / x_a	x_{b1}	x_{b2}	..	x_{bm}
x_{a1}	$p_{xa1, xb1}$	$p_{xa1, xb2}$	..	$p_{xa1, xbm}$
..	..	..	$p_{xai, xbj}$	..
x_{an}	$p_{xin, xj1}$	$p_{xin, xj2}$	..	$p_{xin, xjm}$

Где x_a, x_b - события, произошедшие в КСС, между которыми устанавливается возможная зависимость; $x_{a1}..x_{an}$ диапазон возможных значений события x_a; $x_{b1}..x_{bm}$ - диапазон возможных значений события x_b; $p_{xa1,xb1}$ - вероятность нахождения фактора x_a в диапазоне x_{a1}, а фактора x_b в диапазоне x_{b1} [4].

Причем, $\sum_{i,j} p_{xai,xbj} = 1$.

Ряд распределения значений события x_a имеет вид:

x_{a1}	..	x_{an}
$\sum_j p_{xa1,xbj}$	..	$\sum_j p_{xan,xbj}$

Находим математическое ожидание события x_a:

$$MX_a = x_{a1} \cdot \sum_j p_{xa1,xbj} + .. + x_{an} \cdot \sum_j p_{xan,xbj} . \quad (1)$$

Затем находим дисперсию DX через второй начальный момент:

$$\alpha_2(X_a) = x_{a1}^2 \cdot \sum_j p_{xa1,xbj} + .. + x_{an}^2 \cdot \sum_j p_{xan,xbj} , \quad (2)$$

$$DX_a = \alpha_2(X_a) - MX_a^2 , \quad (3)$$

$$\sigma_{x_a} = \sqrt{DX_a} . \quad (4)$$

Аналогично, находится ряд распределения события x_b, суммируя вероятности $p_{xai,xbj}$ по столбцам, а также дисперсия DX_b.

Далее находим математическое ожидание событий x_a и x_b:

$$MX_i X_j = x_{i,1} \cdot x_{j,1} \cdot p_{xi1,xj1} + .. + x_{i,n} \cdot x_{j,m} \cdot p_{xin,xjm} \quad (5)$$

Затем находим значение ковариации x_a и x_b:

$$\begin{aligned} \operatorname{cov}(X_a, X_b) &= M\left[(X_a - MX_a)(X_b - MX_b)\right] = \\ &= MX_a X_b - MX_a \cdot MX_b \end{aligned} \quad (6)$$

На основании полученного значения находится значение коэффициента корреляции событий x_a и x_b:

$$cor_{xa,xb} = \frac{\operatorname{cov}(x_a, x_b)}{\sigma_{xa}\sigma_{xb}} . \quad (7)$$

Коэффициент корреляции характеризует степень зависимости факторов x_a и x_b. При этом положительная корреляция между событиями означает, что при наступлении (увеличении значения) одного события значение другого также имеет тенденцию к возрастанию. Значение положительных коэффициентов корреляции, характеризующих взаимосвязь событий, может быть использовано администратором безопасности при построении графов потенциальных действий и целей субъектов.

Отрицательная корреляция означает, что при возрастании значений одного из событий значение другого события имеет тенденцию к убыванию. Таким образом, данные сведения могут быть использованы для модификации системы безопасности КСС, при которой перераспределение защитных ресурсов будет происходить от объектов, не нуждающихся в защите к объектам, подвергающимся атакам.

Пример использования анализа событий информационной безопасности

На основании разработанного алгоритма анализа событий информационной безопасности предлагается реализация следующих этапов сбора и подготовки данных для принятия решений по управлению безопасностью [5]:

1) **Идентификация активов** – анализ конфигурации фрагмента корпоративной сети (рис.1), проводимый для выявления основных информационных активов (ресурсов), на которые могут быть нацелены атаки злоумышленников.

Для данного примера фрагмента КСС такими основными ресурсами являются база данных СУБД (*R1*), в которой находятся данные о клиентах компании, а также информация об их платежах и задолженностях.

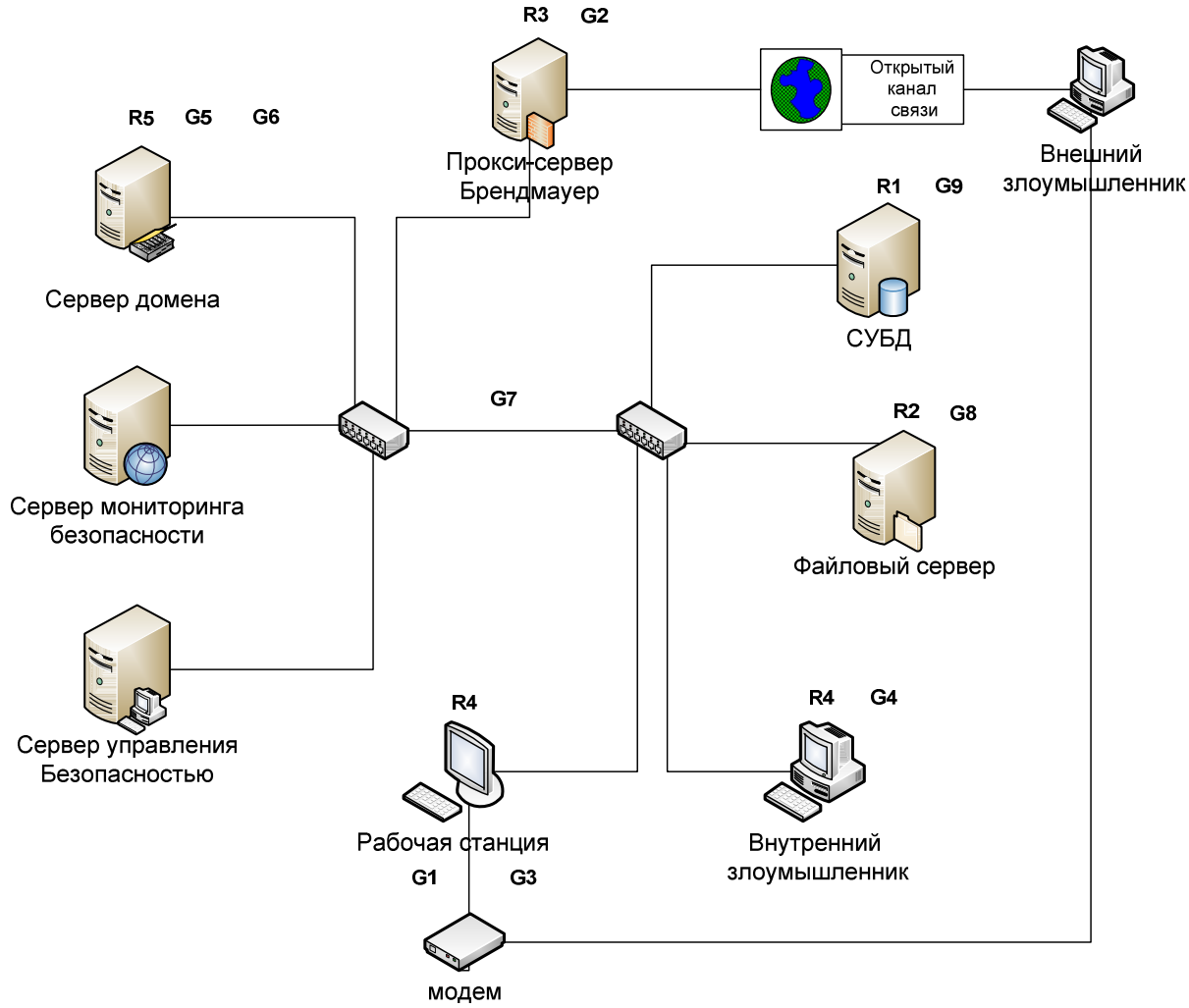


Рис.1. Фрагмент корпоративной сети

Кроме того, критичной является резервная копия данной базы, хранящаяся на файловом сервере (*R2*), а также сервер домена (*R5*). На основании полученной информации формируется вектор информационных ресурсов $r_{1\times f}$. Следует отметить, что выделение основных информационных ресурсов, требующих защиты, проводится на основании данных, полученных от владельцев информации. Собранная информация о конфигурации КСС позволяет приступить к следующему этапу – анализу угроз и уязвимостей данного фрагмента.

2) **Идентификация угроз и уязвимостей** – анализ уязвимостей КСС выполняется на основании топологии сети, статистических данных о событиях в компьютерной системе, полученных с помощью специализированного программного обеспечения, а также сведений об известных вторжениях. На этом этапе строится графовая модель уязвимостей данного фрагмента КСС (рис.2). Вершины графа представляют собой уязвимости, использование которых позволяет оказывать определенное воздействие на компьютерную систему, а дуги показывают возмож-

ность перехода от реализации одной уязвимости к другой.

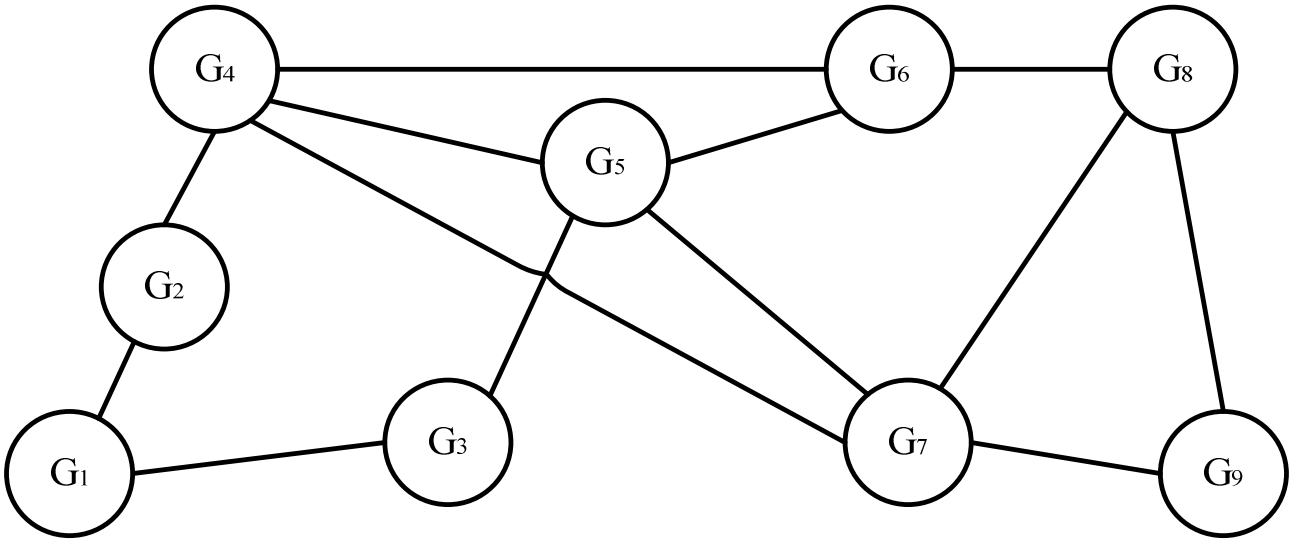


Рис.2. Возможная последовательность реализации уязвимостей

При построения данного графа, кроме приведенных выше формул для расчета взаимосвязи событий безопасности, могут быть использованы экспертные оценки, что вносит некоторый элемент субъективизма.

Например, для данного фрагмента КСС, вершины графа G_i представляют собой следующие уязвимости:

G_1 – открытый канал связи;

G_2 – уязвимости конфигурирования прокси-сервера;

G_3 – незащищенная точка доступа к сети (возможность запуска ПО и доступа к локальным ресурсам – уязвимость в «обе стороны»);

G_4 – уязвимости операционной системы (получение пароля локального администратора с помощью хеш-файла / подбора пароля);

G_5 – уязвимости DNS (получение таблицы DNS, удаленный доступ с помощью поврежденных RPC пакетов);

G_6 – уязвимости сервера домена (чтение активного каталога при инсталляции специализированного программного пакета);

G_7 – уязвимости стека протоколов TCP\IP (передача паролей в незащищенном виде);

G_8 – уязвимости конфигурирования файлового сервера (не отведение защищенной области данных);

G_9 – уязвимости сервера баз данных (возможность не доменной, а sql-аутентификации);

другие.

Определение потенциальных целей злоумышленников позволяет определить соответствующие им действия, с помощью которых данные уязвимости могут быть реализованы.

3) **Анализ этапов и трасс атак** проводится для отслеживания текущей активности злоумышленников и формирования очередности выполняемых ими действий. Для использования приведенных выше уязвимостей злоумышленник может выполнять определенную комбинацию действий a_i приводящую к использованию уязвимостей (Gj) (рис.3) :

a_1 – использование фрагментированных пакетов; (G_1)

a_2 – DOS - атаки; (G_1)

a_3 – несанкционированное подключение к модему; (G_3)

a_4 – локальный анализ log-файлов; (G_4)

a_5 – локальный подбор паролей; (G_4)

a_6 – запуск программ типа «троянский конь»; (G_4)

a_7 – получение IP-адресов и DNS-имен серверов; (G_5)

a_8 – изменение\захват IP-адреса; (G_5)

a_9 – получение полных сведений о доменных пользователях; (G_6)

a_{10} – получение паролей пользователей с помощью анализатора пакетов; (G_7)

a_{11} – копирование / модификация бекапов баз данных; (G_8)

a_{12} – доступ к рабочей базе данных. (G_9).

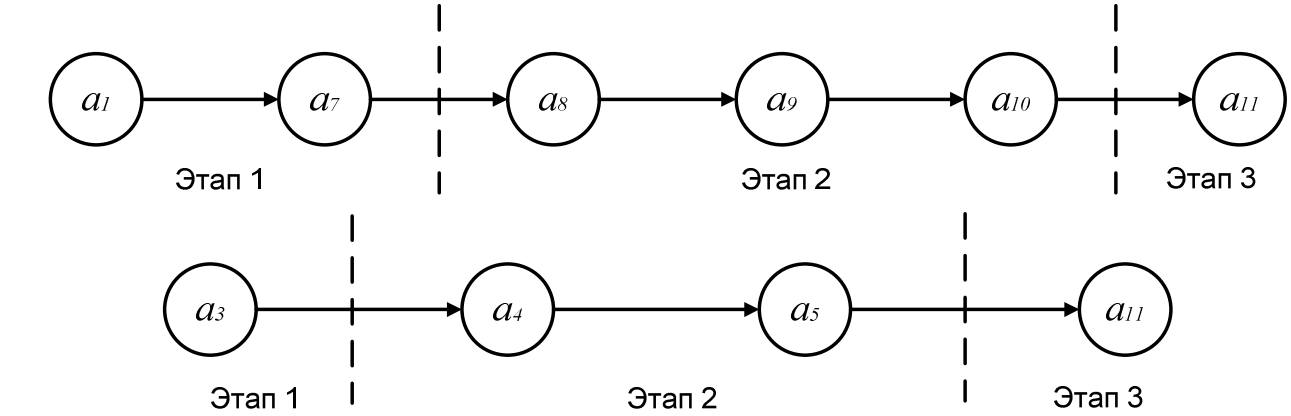


Рис.3. Этапы и трассы атак. Примеры развития 2-х атак

Для контроля событий безопасности и мониторинга действий пользователей необходима подстройка средств обеспечения безопасности, таких как системы обнаружения вторжений, средства защиты (антивирусы, межсетевые экраны, аппаратные средства защиты и контроля доступа) и анализаторы процессов КСС.

4)**Мониторинг событий безопасности** x_i, нуждающихся в дополнительном контроле:

- контроль трафика пакетов протоколов IP и ICMP; (a_1)
- контроль сканирования портов (ответы без запросов); (a_2)
- количество/интервалы подключений; (a_2)
- инициализация подключения к модему; (a_3)
- попытки доступа к log-файлам, журналам регистрации приложений; (a_4)
- контроль неудачных попыток аутентификации; (a_5)
- попытки подключения к службе удаленного реестра; (a_6)
- установка новых служб, помещение программ в автозапуск; (a_6)
- установка специализированных пакетов для доступа к папке “security” объектов домена. (a_9)
- другие.

Таким образом после 4-го этапа определены информационные ресурсы, уязвимости, возможные действия злоумышленников и события безопасности, позволяющие выявлять данные действия. Для этого с помощью средств защиты и проверки состоянии системы, выступающих в качестве агентов-приложений, выполняется сбор информации о происходящих в КСС событиях.

5) **Анализ и оценка рисков** безопасности – проведение анализа рисков реализации угроз безопасности информационных ресурсов позволяет ранжировать ресурсы по степени рисков для КСС.

Уровень рисков безопасности может быть оценен на основе эконометрической модели кредитно-финансовых рисков применительно к задачам защиты информации, обладающей определенной ценностью. Для анализа рисков необходимо сопоставить цели злоумышленника и конкретные ресурсы, подвергаемые атаке (табл.2).

Табл.2. Риски безопасности информационных ресурсов

Ресурсы / Цели	R_1	R_4	R_5
G_6	R_{16}	R_{46}	-
G_8	-	R_{48}	R_{58}

Таким образом, администратор безопасности получает данные о текущей активности по отношению к активам организации.

6) **Выбор мер управления для обработки рисков** безопасности – выполнение соответствующей настройки средств защиты информации. Так, например, целями злоумышленника являются рабочая станция и DNS-сервер. Учитывая ценность информации, а также степень критичности ресурсов администратор безопасности может предпринять следующие варианты действий:

- запретить модемные соединения на рабочих станциях, создать модемный пул;
- повысить защищенность DNS-сервера за счет специализированных средств защиты.

Граф уязвимостей (рис.2) позволяет выстроить цепочку возможных действий злоумышленника, исходя из его текущего положения в пространстве целей (уязвимостей) и текущей активности, а также спрогнозиро-

вать его следующие шаги и определить потенциальные и возможные конечные цели, и, соответственно, возможные действия. Это позволяет контролировать вторжения на их начальной стадии.

На основании построенного прогноза поведения злоумышленников с помощью обратного прохода по графу этапов атак определяются события безопасности, нуждающиеся в дополнительном контроле, а также может быть проведена перенастройка средств защиты.

Для обнаружения несанкционированных действий должны быть обеспечены ведение и хранение в течении определенного периода времени журналов аудита, регистрирующих действия пользователей, нештатные ситуации и события информационной безопасности, в целях помощи в будущих расследованиях и проведении анализа событий информационной безопасности. Часы всех систем, покрываемых СМИБ, должны быть синхронизированы с помощью единого источника точного времени. Средства регистрации и данные журналов регистрации должны быть защищены от вмешательства и несанкционированного доступа, действия администратора и системного администратора должны быть регистрируемыми.

Выводы

В соответствии с международным стандартом ISO/IEC 27001 организации должны проводить мероприятия по устранению причин несоответствий требованиям СМИБ с целью предупреждения их повторного возникновения. Применение предложенного анализа событий информационной безопасности позволяет выполнять корректирующие действия в рамках постоянного улучшения системы менеджмента информационной безопасности. Кроме того, полученные результаты могут быть использованы администраторами безопасности для определения потенциальных несоответствий требованиям СМИБ и проведения предупреждающих действий.

Список использованной литературы

1. ISO/IEC 27001:2005 “Information technology – Security techniques – Information security management systems – Requirements”.
2. Дмитриев А. А. ISO/IEC 27001 – Путь к информационной безопасности. Особенности внедрения на отечественных предприятиях.// Das Management, N1, 2009. – с. 36 – 39.
3. Мухин В.Е., Волокита А.Н. Алгоритм оценки вероятности вторжений для средств мониторинга безопасности компьютерных систем. // Системные исследования и информационные технологии, N2, 2007. – с. 47 – 58.
4. Кудрявцев В. Л., Демидович Б. П. Краткий курс высшей математики: Учебное пособие дли вузов. 7-е изд., испр. – М: Наука. Гл. ред. физ.-мат. лит., 1989. – 656 с.
5. Волокита А.Н. Система поддержки принятия решений для средств мониторинга безопасности компьютерных сетей. //Новые технологии, N1(216), 2007. – с. 253 – 258.

Поступила в редакцию 01.12.2009

УДК 681.51

СКАКУН С.В.

СТРУКТУРНО-ФУНКЦИОНАЛЬНЫЙ АНАЛИЗ КОМПОНЕНТОВ СИСТЕМ SENSOR WEB

В данной статье рассматривается задача структурно-функционального анализа компонентов систем Sensor Web. Приведена структурная модель системы Sensor Web, а также рассмотрены вопросы интеграции Sensor Web и Grid систем. Описана система прогнозирования и мониторинга наводнений, построенная на основе технологии Sensor Web.

The paper is devoted to the structural and functional analysis of Sensor Web systems. We described a structural model of the Sensor Web, and investigated different approaches to integration of Sensor Web and Grids. We presented a Sensor Web system for flood forecasting and monitoring.

Введение

Современный уровень информатизации общества характеризуется бурным развитием сложных распределенных систем экологического мониторинга, предполагающих активное взаимодействие с внешней средой путем получения и обработки разнородной измерительной информации. Технология построения таких систем получила название Sensor Web [31, 32, 33]. Sensor Web представляет собой координированную инфраструктуру наблюдений, включающую распределенные источники данных (сенсоры, модели, коммуникации) и предоставляющую доступ к исходным и обработанным данным, метаданным посредством использования стандартизированных сервис-ориентированных интерфейсов. Причем такая система является автономной, адаптивной, перенастраиваемой и управляемой [34].

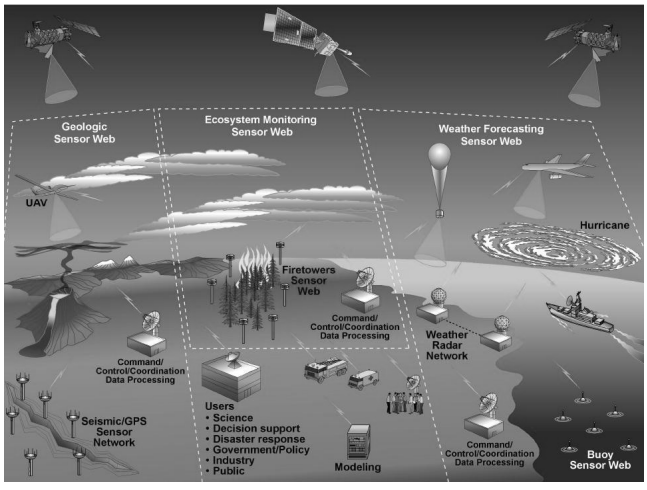


Рис. 1. Архитектура Sensor Web [34]

На современном этапе такие системы, как правило, подразумевают получение информации с наземных, дистанционных измерительных устройств, а также данных моделирования. Иллюстрация структуры такой системы, приведенная в отчете NASA [34], приводится на рис. 1.

Наблюдаются тенденции интеграции отдельных систем в еще более сложные структуры, включающие высокопроизводительную вычислительную технику и хранилища данных больших объемов на базе Grid-технологии, обеспечивающей для пользователя легкий («прозрачный») доступ к ресурсам виртуальной организации [35]. При этом возникает множество плохо формализуемых задач оптимизации затрат и производительности. Поэтому сегодня актуальной является не просто задача построения очередной Sensor Web системы, а создания эффективной инфраструктуры, объединяющей несколько работающих или вновь создаваемых региональных или специализированных сенсорных сетей и Grid-систем. Особый интерес исследователей вызывают системы, связанные с использованием наблюдений Земли из космоса, поскольку они представляют собой не просто высокопроизводительные вычислительные системы, а требуют использования измерительной информации разного типа (спутниковых наблюдений, наземных заверочных данных и данных моделирования, восполняющих пробелы в реальных измерениях). Проблемами создания таких систем занимаются международные рабочие группы и комитеты WGISS [36] и GEO [37], а также посвящены большие международные проекты, например “Sensor Web for Namibian Pilot project on integrated flood management and water related vector borne disease modelling”. Данный проект является совместной инициативой UN-SPIDER, NASA, NOAA, DLR и

ИКИ НАН-НКАУ и направлен на интегрированное использование спутниковых данных, наземных измерений и данных моделирования для мониторинга и прогнозирования наводнений для Африки. Кроме того, в рамках проекта решаются задачи, связанные с моделированием и прогнозированием распространения заболеваний, связанных с водой.

В данной работе предложенный в [38] метод структурно-функционального анализа (СФА) применяется к предметной области систем Sensor Web. Этот подход был ранее использован для анализа системы Earth observation Grid (EO Grid) [39, 40].

Постановка задачи структурно-функционального анализа сложных иерархических систем

Учитывая высокую стоимость создания распределенных систем, в которых используются гетерогенные данные и высокопроизводительная техника, к их разработке требуется подходить с позиций системного анализа, в частности метода структурно-функционального анализа (СФА). Согласно [38, с. 432], содержательная формулировка задачи структурно-функционального анализа сводится к следующему. Известны назначение, общие характеристики и свойства системы Sensor Web, а также требования к структурным, функциональным, эксплуатационным и экономическим показателям.

Конкретнее, система Sensor Web предназначена для решения задач экологического мониторинга, оценки рисков и прогнозирования последствий экологических и техногенных чрезвычайных ситуаций, поддержки принятия решений в интересах устойчивого развития на основе интеграции данных наземных и аэрокосмических наблюдений, а также данных моделирования. В зависимости от перечня задач, решаемых в конкретной системе, можно определить конкретные требования или уже реализованные свойства перечисленных выше категорий.

Требуется определить структуру Sensor Web как сложной иерархической системы, обосновать требования к каждому сегменту, узлу и функциональному элементу (ФЭ) всех иерархических уровней.

Заметим, что структурно-функциональный анализ может выполняться на разных стадиях жизненного цикла системы: как на этапе ее проектирования, так и в процессе ее эксплуатации. В первом случае формулируется перечень требований к системе, а во втором – набор свойств (существующих или желаемых в зависимости от постановки задачи). Поэтому без ограничения общности термины «требования» и «свойства» будем рассматривать как синонимы.

Формализация задачи структурно-функционального анализа.

Перейдем к формализации задачи структурно-функционального анализа системы EO Grid. Согласно выражению (7.1) из [38, с. 433] множество требований (свойств) рассматриваемой системы представляется в виде упорядоченной по значимости структуры классов (категорий):

$$B_0=\{B_i \mid i=\overline{1,m_1}\}, \qquad (1)$$

где B_0 – множество свойств системы, B_i – i-й класс (категория) свойств (m_1=5).

Для систем Sensor Web выделим следующее множество классов:

B_1 – класс структурных свойств (требований), характеризующих структуру системы и ее элементов;

B_2 – класс функциональных свойств, определяющих перечень функций системы и количественные показатели их выполнения;

B_3 – класс эксплуатационных свойств, определяющих удобство эксплуатации системы и требования к внешним условиям;

B_4 – класс свойств безопасности;

B_5 – класс экономических свойств, определяющих требования к стоимости построения и эксплуатации системы.

Каждая категория B_i определяется набором свойств b_{ij}:

$$B_i = \{b_{ij} \mid j=\overline{1,m_i}\}. \qquad (2)$$

Рассмотрим каждую категорию в отдельности. Если множество B_1 представляет класс структурных свойств

$$B_1 = \{b_{1j} \mid j=\overline{1,m_1}\},$$

то m_1=9 и к данному классу относятся следующие количественные свойства: b_{11} – число узлов в системе, b_{12} – число процессоров, b_{13} — пропусканная способность каналов связи, b_{14} – объем памяти для хранения данных, b_{15} – объем оперативной памяти.

К качественным (неколичественным) свойствам (требованиям к структуре системы) относятся: b_{16} – гетерогенность (гомогенность), b_{17} – межузловая топология, b_{18} – требования к интерфейсам, b_{19} – сервисная

ориентированность, или соответствие стандартам WSRF (Web Service Resource Framework).

Множество B_2 функциональных свойств

$$B_2 = \{b_{2j} \mid j = \overline{1, m_2}\}$$

характеризуется следующим набором параметров:

b_{21} – поддержка запросов пользователей на выполнения задач в заданном виде;

b_{22} – поддержка запросов пользователей на высокоуровневый доступ, передачу и репликацию данных;

b_{23} – поддержка единой регистрации;

b_{24} – балансировка нагрузки;

b_{25} – сбор статистики;

b_{26} – наблюдаемость (наличие информации о состоянии системы и ее элементов);

b_{27} – управляемость (возможность целенаправленно изменять состояние ФЭ (узлов) для достижения заданного критерия);

b_{28} – адаптируемость (возможность перенастройки системы в соответствии с изменением внешней среды, например, появлением новых задач или узлов системы).

Множество B_3 эксплуатационных свойств

$$B_3 = \{b_{3j} \mid j = \overline{1, m_3}\}$$

характеризуется следующим набором параметров

b_{31} – наличие определенного программного обеспечения;

b_{32} – дружественный интерфейс;

b_{33} – Quality of Service (QoS). К метрикам качества обслуживания Sensor Web относятся: доступность, надежность (отказоустойчивость и постепенное сокращение возможностей), оперативность (многомерная величина, характеризующая задержки и пропусканные характеристики), качество данных и т.д.;

b_{34}– должна поддерживать расширяемость без остановки работы.

Множество B_4 свойств безопасности

$$B_4 = \{b_{4j} \mid j = \overline{1, m_4}\}$$

содержит следующие элементы (согласно [41]):

b_{41} – конфиденциальность (доверительная конфиденциальность, административная конфиденциальность, повторное использование объектов, анализ скрытых каналов, конфиденциальность при обмене);

b_{42} – целостность (доверительная целостность, административная целостность, откат, целостность при обмене);

b_{43} – доступность (использование ресурсов, устойчивость к отказам, горячая замена, восстановление после сбоев);

b_{44} – наблюдаемость (регистрация, достоверный канал, идентификация и аутентификация, целостность системы безопасности, распределение полномочий, самотестирование, аутентификация при обмене, аутентификация отправителя, аутентификация получателя);

b_{45} – уровни гарантий (Г1-Г7).

Множество B_5 экономических свойств

$$B_5 = \{b_{5j} \mid j = \overline{1, m_5}\}$$

содержит следующие элементы: b_{51} – стоимость оборудования; b_{52} — стоимость эксплуатации (TCO – total cost ownership); b_{53} – стоимость услуг, предоставляемых системой.

Каждое свойство b_{ij} класса B_i характеризуется набором показателей

$$Y_{ij} = \{y_{ijk} \mid k = \overline{1, m_{ij}}\}, \qquad (3)$$

где m_{ij} – количество показателей для свойства b_{ij}, которые могут иметь как количественное, так и качественное представление.

Так, для параметра b_{21} из множества функциональных свойств B_2 будем иметь следующий набор показателей:

y_{211} – число запросов в минуту;

y_{212} – задержка на выполнения запроса (не более чем);

y_{213} –если число запросов больше заданного (y_{213}), часть запросов откланяется.

Требования к количественным показателям задаются с помощью интервальных оценок вида

$$\underline{y}_{ijk} < y_{ijk} < \overline{y}_{ijk}, \qquad (4)$$

а требования к качественным показателям удобно задать с помощью нечетких термов [42].

Структурная модель Sensor Web

Формализуем описание иерархической системы Sensor Web. Результат структурной декомпозиции системы показан на рис. 2.

Основными компонентами Sensor Web являются:

– *сенсорные узлы*, которые -

ответственны за получение данных. В общем случае, такие сенсоры могут быть *наземными*, выполняющие непосредственные контактные измерения; *дистанционными*, например, приборы дистанционного зондирования Земли, установленные на спутниках; или *моделями*, которые подразделяются на

модели сенсоров и окружающей среды. Модели сенсоров используются для описания работы сенсора для корректной интерпретации и обработки данных и обеспечения необходимого уровня качества данных. Модели окружающей среды в основном используются для прогнозирования явлений или значений параметров. Примерами могут служить, модели численного предсказания погоды [35];

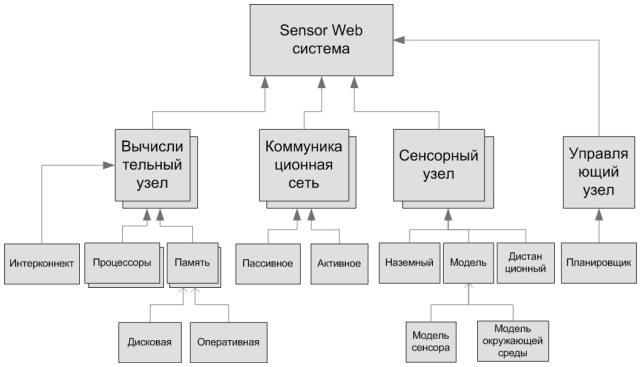


Рис. 2. Структурная модель системы Sensor Web

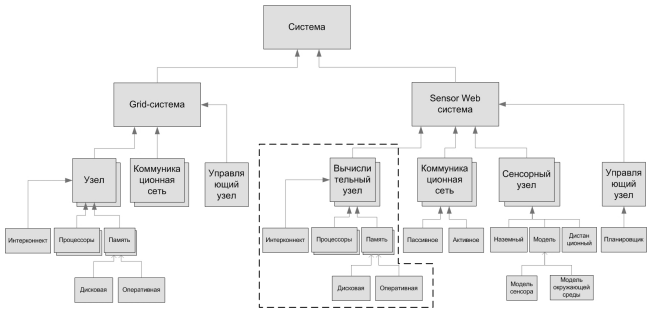
– *вычислительные узлы* используются для обработки данных. Стоит отметить, что данные, получаемые сенсорными узлами в системе Sensor Web, могут обрабатываться как этими узлами, так и использовать внешние системы, например Grid;

– *коммуникационная сеть* предоставляет средства для взаимодействия между различными сенсорами и узлами Sensor Web;

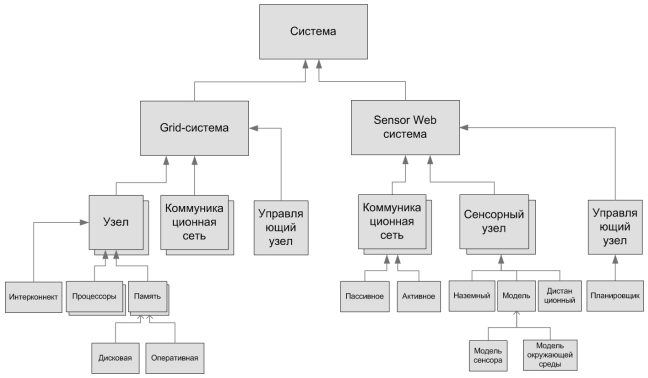
– *управляющий узел* ответственен за управление и контролем состоянием Sensor Web, а также за взаимодействие с внешними системами. В качестве основы взаимодействия может использоваться сервис-ориентированная архитектура SOA (Service-Oriented Architecture).

Стоит отметить, что Sensor Web редко используется как отдельная система, а интегрируется в более масштабные системы. Если основной задачей Sensor Web является сбор, получение и предоставление данных, то для их обработки необходимы соответствующие вычислительные ресурсы. Кроме того, во многих прикладных задачах требуется эффективное управление потоком выполнения заданий, что особенно актуально для Sensor Web, где данные интегрируются из разных источников. Для решения этих задач целесообразно использовать Grid-технологии [35, 39]. Таким образом, возникает задача интеграции систем Sensor Web и Grid.

Рассмотрим более детально следующие два способа интеграции таких систем:



(а)



(б)

Рис. 3. Способы интеграции Sensor Web и Grid

1. Система Sensor Web является независимым сегментом Grid (рис. 3а). Обработка данных, получаемых сенсорными узлами, и генерация информационных продуктов в основном осуществляется непосредственно вычислительными узлами Sensor Web. Однако в реальных системах требуется обработка больших объемов данных и использование для этих целей исключительно собственных вычислительных ресурсов Sensor Web маловероятно.

2. Гибридная Sensor Grid система (рис. 3б). В этом случае для обработки и хранения данных, получаемых узлами Sensor Web, используются ресурсы Grid. Именно в Grid реализуются сервисы, отвечающие за обработку исходных данных и создание информационных продуктов, передаваемых конечным пользователям.

Каждый q-й уровень иерархии состоит из P_q функциональных элементов. Тогда множество всех ФЭ системы можно описать следующим образом:

$$V = \{V_{qp} \mid q = \overline{1,N},\ p = \overline{1,P_q}\}, \qquad (5)$$

где N — общее число уровней иерархии; V_{qp} — p-й ФЭ q-ого иерархического уровня.

Каждый ФЭ V_{qp} системы характеризуется следующим вектором показателей:

$$\mathbf{x}_{qp} = (x_{qpj} \mid j = \overline{1, n_{qp}}), \qquad (6)$$

где n_{qp} — количество показателей ФЭ V_{qp}, выполняющего набор функций

$$\Phi_{qp} = (f_{qpk} \mid j = \overline{1, m_{qp}}), \qquad (7)$$

Каждая из функций f_{qpk} в (7) зависит от значений показателей вектора $\mathbf{x}_{qp}$

$$f_{qpk} = f_{qpk}(\mathbf{x}_{qp}). \qquad (8)$$

и влияет на реализацию требований, предъявляемых к ФЭ и системе в целом.

Согласно [38, с. 435] состав и вид функций (7-8) определяется в процессе системного анализа.

Пусть

$$X = \{\mathbf{x}_{qp}, p = \overline{1, P_q}, q = \overline{1, N}\} \qquad (9)$$

обобщенное множество показателей. Для простоты изложения откажемся от тройной индексации. Будем считать, что общее число показателей всех ФЭ всех иерархических уровней системы равно N_0. Тогда обобщенный вектор показателей можно представить в виде

$$\mathbf{x} = (x_1, x_2, ..., x_{N_0})^T, \mathbf{x} \in \mathbf{R}^{N_0}. \qquad (10)$$

Одной из главных задач структурно-функционального анализа является определении преобразования

$$F: X \to Y \qquad (11)$$

из множества X допустимых показателей системы в пространство Y требуемых свойств по набору количественных требований, заданных в виде (4), и качественных требований, заданных в виде нечетких термов.

При этом требуется решить задачу структурно-параметрической идентификации, позволяющей одновременно определить структуру иерархической системы в целом, структуру ФЭ всех иерархических уровней, а также вид преобразования (11).

Система Sensor Web для прогнозирования и мониторинга наводнений

На основе технологии Sensor Web Институтом космических исследований НАНУ-НКАУ (ИКИ НАНУ-НКАУ) разрабатывается система прогнозирования и мониторинга наводнений, архитектура которой представлена на рис. 4.

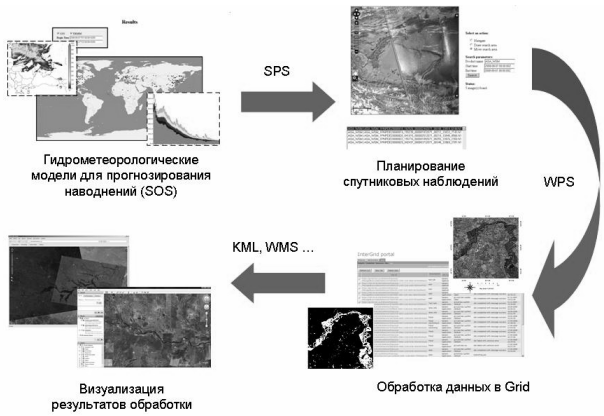


Рис. 4. Система прогнозирования и мониторинга наводнений

Для реализации системы используются стандарты консорциума Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE):

– OGC Observations & Measurements – общие термины и определения области Sensor Web;

– Sensor Model Language (SensorML) – язык на основе XML для описания различных видов сенсоров;

– Transducer Model Language – язык на основе XML для описания преобразований;

– Sensor Observations Service (SOS) – интерфейс для удаленного доступа к данным сенсора;

– Sensor Planning Service (SPS) – интерфейс для отправки заданий сенсорам.

Кроме того, для обработки используется стандарт Web Processing Service (WPS).

На сегодняшний день в рамках системы для прогнозирования и мониторинга осадков используются глобальная система прогнозирования погоды GFS и спутниковые измерения TRMM, для которых разработаны соответствующие интерфейсы SOS. Прогнозы осадков используются для идентификации потенциальных регионов, где возможны наводнения. На основе этих измерений осуществляется заказ и планирование спутниковых данных через интерфейс SPS. Такой интерфейс уже разработан специалистами NASA для спутника Earth Observing (EO-1). Для получения радиолокационных данных спутника Envisat используются оперативные архивы Европейского космического агентства (ЕКА). Полученные спутниковые данные передаются на ресурсы Grid-системы ИКИ НАНУ-НКАУ, где автоматически генерируются потоки выполнения заданий для их обработки. В настоящее время реализованы

сервисы картографирования наводнений на основе использования радиолокационных спутниковых данных [43]. Полученные карты предоставляются в форматах KML и WMS консорциума OGC.

Выводы

В данной статье реализован системный подход к построению и анализу функционирования систем Sensor Web. Построена иерархическая структурная модель Sensor Web, в рамках которой данные поступают с гетерогенных сенсорных узлов: наземных (контактных), дистанционных и модельных. Важным преимуществом развиваемого подхода по сравнению с другими работами, посвященными моделированию и оценке производительности распределенных систем, является системность, базирующаяся на методе СФА [38] и позволяющая рассматривать сложные иерархические системы на разных уровнях детализации одновременно.

В статье также рассмотрены вопросы интеграции Sensor Web и Grid. Преимущество интеграции таких систем состоит в том, что Sensor Web предоставляет доступ к разнородным источникам данных, обеспечивая такие свойства автономность, управляемость, настраиваемость и т.д., в то время как Grid представляет ресурсы для обработки данных и создания информационных сервисов для конечных пользователей.

Развиваемый подход апробирован при построении системы Sensor Web для прогнозирования и мониторинга наводнений с использованием данных моделирования (глобальная модель прогнозирования погоды GFS) и дистанционных наблюдений (TRMM для мониторинга за уровнем осадков и Envisat/ASAR для картографирования наводнений).

Работа выполнена при поддержке проектов НАНУ-УНТЦ №4928 «Разработка Grid-технологий интеграции данных разной природы» и гранта Президента Украины «Разработка каскада гидрометеорологических моделей для прогнозирования наводнений».

Литература

1. Moe K., Smith S., Prescott G., Sherwood R. Sensor Web Technologies for NASA Earth Science // Proc. of 2008 IEEE Aerospace Conference. – 2008. – P. 1-7.
2. Moe. K. NASA Technology for the Earth Observation Sensor Web // IEEE International Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. – 2008. – 5. – P. 128-131.
3. Mandl D., Frye S.W., Goldberg M.D., Habib S., Talabac S. Sensor Webs: Where They are Today and What are the Future Needs? // Proc. Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS 2006). – 2006. – P. 65-70.
4. Report from the Earth Science Technology Office (ESTO) Advanced Information Systems Technology (AIST) Sensor Web Technology Meeting. – San Diego, USA. – February 13-14, 2007. – http://esto.nasa.gov/sensorwebmeeting/files/AIST_Sensor_Web_Meeting_Report_2007.pdf.
5. Kussul N., Shelestov A., Skakun S. Grid and sensor web technologies for environmental monitoring // Earth Science Informatics. – 2009. – Volume 2, Numbers 1-2. – P. 37-51.
6. CEOS Working Group on Information Systems and Services (WGISS). — http://www.ceos.org/wgiss.
7. GEO - Group on Earth Observations. — www.earthobservations.org.
8. Згуровский М.З., Панкратова Н.Д. Системный анализ: проблемы, методология, приложения. – К.: Наукова думка, 2005. – 744 с.
9. Куссуль Н.Н., Шелестов А.Ю. Grid-системы для задач исследования Земли. Архитектура, модели и технологии. – К.: “Наукова думка”, 2008. – 452 с.
10. Шелестов А.Ю. Структурно-функциональный анализ компонентов Grid-систем // Проблемы управления и информатики. – 2007. – №5. – С. 119-132.
11. НД ТЗІ 2.5-004-99 «Критерії оцінки захищеності інформації в комп’ютерних системах від несанкціонованого доступу», ДСТЗІ, Київ, 1999.
12. Zadeh L. Fuzzy sets // Information and Control. – 1965. – V.8. – 338 p.
13. Kussul N., Shelestov A., Skakun S. Grid System for Flood Extent Extraction from Satellite Images // Earth Science Informatics. - 2008. - 1(3-4). - P. 105-117.

Поступила в редакцию 17.12.2009

УДК 004.02

КОНДРАШОВА Н.В.

МЕТОД ГРАМА-ШМИДТА-ГАУССА ДЛЯ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ В МГУА

Рассматривается рекуррентный метод решения систем линейных алгебраических уравнений (СЛАУ) для идентификации моделей с вложенными структурами. Метод Грама-Шмидта используется для проектирования исходной системы векторов переменных в ортонормированную, а обратный ход метода Гаусса – для определения коэффициентов СЛАУ. Показано, что такой способ решения является наименее вычислительно затратным по сравнению с известными рекуррентными и нерекуррентными прямыми («точными») методами решения СЛАУ

The recurrent decision method of linear algebraic equations systems (LAES) is considered for models identification with the imbedded structures. Grama-Shmidta's method is used for projection of initial variables vectors system into orthonormal one and backward motion of Gausse's method for LAES parameters determination. It is demonstrated that such decision method takes the least calculate time as compared to the known recurrent and non-recurrent direct (or exact) methods for LAES decisions.

Введение

Экономичный метод решения СЛАУ большой размерности за конечное число арифметических операций изложен в данной работе.

Методы численного решения делятся на две группы: итерационные методы и прямые («точные») методы [1].

Итерационные методы (методы последовательных приближений) состоят в том, что решение системы находится, как предел последовательных приближений параметров системы $\theta^{(\ell)}$ при $\ell \to \infty$, где ℓ номер итерации. К этим методам относятся метод Зейделя, Якоби и т.д.

Прямыми (или точными) методами называются методы, позволяющие получить решение системы за конечное число арифметических операций. Прямые методы подразделяются на нерекурентные и рекуррентные. Широко известными нерекурентными методами являются обычный метод Гаусса, LU–метод, метод квадратного корня, метод вращений линейных систем и др. Различают рекуррентные методы перерасчета неизвестного вектора параметров системы:

1) при добавлении условных уравнений (строк);

2) при добавлении аргументов (столбцов) $\widehat{\mathbf{u}}_i$, $\dim \widehat{\mathbf{u}}_i = 1 \times i,\ i = \overline{1,s}$.

Рекуррентные методы при добавлении аргументов применяют тогда, когда процесс перебора рассматриваемых моделей организуется по методу вложенных структур моделей [2], получившему развитие в последнее десятилетие и известному как модель вложенных множеств (англ. Joe Celko's nested sets). Рекуррентные алгоритмы вычисления параметров применяются в алгоритмах метода группового учета аргументов (МГУА) для структурно-параметрической идентификации, как более экономичные по сравнению с нерекуррентными. Известными являются рекуррентный метод наименьших квадратов с окаймлением [3], рекуррентный алгоритм Гаусса и модификация алгоритма Грама-Шмидта [4].

Следует заметить, что реализация прямых методов на компьютере приводит к решению с погрешностью, т.к. все арифметические операции над переменными с плавающей точкой выполняются с округлением. В зависимости от свойств матрицы исходной системы и точности выполнения арифметических операций эти погрешности могут достигать значительных величин. Представленный в данной работе пример решения хорошо обусловленной СЛАУ четвертого порядка при вычислении с точностью примерно до 16 значащих цифр (1E-16) программой EXEL демонстрирует, что данная погрешность вычисления не влияет на результат.

Постановка задачи моделирования

Выборка представлена таблицей данных $W = (X \vdots y)$, размерность таблицы $\dim W = n \times (m+1)$, где n – число строк; m+1 – число столбцов таблицы. Условимся, что данные, обозначенные как *X,* являются значениями m входных переменных в n точках

наблюдения, y – значениями выходной переменной в тех же точках. Эту же информацию можно представить в виде соответствующего размера матрицы $\mathbf{X}$ и вектора $\mathbf{y}$,

$$\dim \mathbf{X} = n \times m,\ \dim \mathbf{y} = n \times 1, \quad n \geq m.$$

Используя исходную информацию $(X : y)$, построим ряд моделей вида:

$$y = \theta_0 + \theta_1 x_1 + ... + \theta_i x_i + \xi_i, \qquad (1)$$

К случайным отклонениям ξ_i в классическом регрессионном анализе выдвигаются следующие требования:

$E\xi_i = 0$ где $i = \overline{1,n}$ в каждой точке измерения математическое ожидание шума равен нулю, или в векторной форме: $\mathbf{E}(\xi) = \mathbf{0}_n$, где $\mathbf{0}_n$ – вектор из n нулей;

$E\xi_i\xi_k = 0, \quad k \neq i$ – шум некоррелированный;

$E\xi_i^2 = \sigma^2 < \infty$ – дисперсия шума существует и имеет конечное значение. Необходимо найти параметры (1), как решение системы условных линейных алгебраических уравнений:

$$\mathbf{X}_i \mathbf{и}_i = \mathbf{y}, \qquad (2)$$

$$\dim \mathbf{X}_i = n \times i, \quad \dim \mathbf{и}_i = i \times 1, \quad \dim \mathbf{y} = n \times 1,$$

$$n \geq m,\ \mathbf{и}_i,\ i = \overline{0,s},\ s = \overline{s_\ell, s_j},\ s_\ell \in \{0,1,...,m\},$$

$$s_j \in \{m,(m-1)..,0\},\ s_\ell \leq s_j.$$

Положим, что столбцы и строки матрицы $\mathbf{X}_i$ не являются линейно зависимыми, т.е. решение системы (2) существует.

В комбинаторных алгоритмах МГУА перебор всевозможных сочетаний аргументов $x_i, \quad i = \overline{1,m}$ возможен по методу вложенных структур, например, с помощью следующих генераторов двоичных структурных векторов: счетчика Гарсайда [5] или оптимизированного последовательного счетчика [6], в которых перебираемые одна за другой модели отличаются только одним членом. Названные счетчики реализуют полный перебор структур моделей без повторений добавлением по одному аргументу к предыдущей модели, исходя из достигнутых ранее состояний счетчика.

Пусть фиксирован порядок усложнения моделей, как:

$\mathbf{y} = \theta_{00}$

$\mathbf{y} = \theta_{10} + \theta_{11}\mathbf{x}_1 = \mathbf{X}_1\boldsymbol{\theta}_1,\ \dim \mathbf{X}_1 = n \times 2,$

$\dim \boldsymbol{\theta}_1 = 2 \times 1$

$\mathbf{y} = \theta_{20} + \theta_{21}\mathbf{x}_1 + \theta_{22}\mathbf{x}_2 = \mathbf{X}_2\boldsymbol{\theta}_2,\ \dim \mathbf{X}_2 = n \times 3,$

$\dim \boldsymbol{\theta}_2 = 3 \times 1$

... и т.д.

$$\mathbf{y} = \theta_{s0} + \theta_{s1}\mathbf{x}_1 + \theta_{s2}\mathbf{x}_2 + ... + \theta_{s2}\mathbf{x}_s = \mathbf{X}_s\boldsymbol{\theta}_s, \qquad (3)$$

$$\dim \mathbf{X}_s = n \times s,\ \dim \boldsymbol{\theta}_s = s \times 1.$$

Изложение основного материала

Традиционно в алгоритмах МГУА от систем условных уравнений вида (2) переходят к нормальным системам уравнений $\mathbf{X}_i^T\mathbf{X}_i\mathbf{и}_i = \mathbf{X}_i^T\mathbf{y}$, решение которых существует, если определитель квадратной матрицы размерности $\dim \mathbf{X}_i^T\mathbf{X}_i = i \times i$ отличен от нуля. Затем применение известных способов решения (Гаусса, LU–метод и т. д.) сводится в конечном итоге к эквивалентному преобразованию матрицы $\mathbf{X}_i^T\mathbf{X}_i$ – к верхней треугольной матрице $\mathbf{R}_i$, $\dim \mathbf{R}_i = i \times i$, на которую справа умножается вектор параметров $\mathbf{и}_i$.

Переход к верхней треугольной матрице возможен также непосредственно от прямоугольной матрицы $\mathbf{X}_i$, как:

$$\mathbf{R}_i = \mathbf{V}_i^T\mathbf{X}_i \qquad (4)$$

После чего компоненты вектора параметров $\widehat{\mathbf{и}}_i$ находятся решением следующей преобразованной системы уравнений:

$$\mathbf{V}_i^T\mathbf{X}_i\mathbf{и}_i = \mathbf{V}_i^T\mathbf{y}, \qquad (5)$$

в обратном ходе обычного метода Гаусса. В работе [7] показано, что матрица преобразований $\mathbf{V}_i$ должна состоять из ортогональных векторов. $\mathbf{V}_i'$ – результат проектирования исходных векторов $\mathbf{x}_i, i = \overline{1,s}$ на ортогональную систему векторов $\mathbf{v}_i', i = \overline{1,s}$, а треугольная матрица $\mathbf{R}_i$ есть результат эквивалентного преобразования. Ортогонализация переменных может быть осуществлена рекуррентно с помощью процедуры Грама-Шмидта [8]:

$$\mathbf{v}_i' = \mathbf{x}_i - \sum_{j=0}^{i-1}(\mathbf{x}_i^T\mathbf{v}_j)\mathbf{v}_j, \qquad (6)$$

$$\dim \mathbf{v}_i' = \dim \mathbf{v}_i = n \times 1,\ i = \overline{0,s},$$

где $\mathbf{v}_i'$ – вектор ортогональный всем предыдущим i векторам. Если он не равен нулю, то

на него может быть спроектирован вектор $\mathbf{x}_i$, а если он равен нулю (в реальных вычислениях – заданной ε погрешности вычисления) – значит к уже найденной ортогональной системе векторов $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{i-1}$ данный вектор $\mathbf{x}_i$ является коллинеарным. И его нужно исключить из рассмотрения, т.к. он не может рассматриваться в качестве возможного аргумента модели. Вектора $\mathbf{v}_i'$ нормируются, как:

$$\mathbf{v}_i = \frac{\mathbf{v}_i'}{(\mathbf{v}_i'^T \mathbf{v}_i')^{1/2}} = \frac{\mathbf{v}_i'}{\sqrt{\sum_{j=1}^{n} v_{ij}'^2}},\ i = \overline{0,s}, \qquad (7)$$

где $|\mathbf{v}_j| = \sqrt{\sum_{j=1}^{n} v_{ij}'^2}$ есть длина вектора в n-мерном пространстве. Операция нормирования необходима для того, чтобы спроектировать вектор $\mathbf{x}_i$ на ортогональную систему полученных векторов $\mathbf{v}_j$, $j = \overline{1,(i-1)}$ и оценить искомый вектор параметров $\widehat{\mathbf{и}}_i$ как обратный ход в системе (5), поскольку $(\mathbf{x}_i^T \mathbf{v}_j) = |\mathbf{x}_i| \cos(\mathbf{x}_i, \mathbf{v}_j) = np_{\mathbf{v}_j} \mathbf{x}_i$, $j = \overline{1,(i-1)}$, $\mathbf{V}_i^T \mathbf{V}_i = \mathbf{I}_i$, $i = \overline{0,m}$, где $\mathbf{I}_i$ – единичная диагональная матрица, имеющая размерность $\dim \mathbf{I}_i = (i+1)\times(i+1)$.

Для приведения индексации векторов в формуле (6) в соответствие со схемой усложнения моделей (3) необходимо определить вектор $\mathbf{x}_0$, как единичный $\mathbf{x}_0^T \overset{\Delta}{=} (1,1,...,1)$.

Из формулы (6) видно, что для вычисления ортогонального вектора $\mathbf{v}_i'$ при добавлении в модель вектора $\mathbf{x}_i$ необходима только матрица $\mathbf{V}_{i-1}$ и после вычисления $\mathbf{v}_i$ не следует пересчитывать полученные ортонормированные вектора $\mathbf{v}_j$, $j = \overline{1,(i-1)}$.

Вектор $\mathbf{v}_0' \equiv \mathbf{x}_0$ и вектор $\mathbf{v}_0$ (в общем случае выбирается произвольно) совпадает с вектором $\mathbf{x}_0$ по направлению. В случае модели со свободным членом он имеет модуль $|\mathbf{v}_0| = 1$, $\mathbf{v}_0 = \left(\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, ..., \frac{1}{\sqrt{n}}\right)^T$, $\dim \mathbf{v}_i = n \times 1$.

Все последующие вектора строятся, как ортогональные к предыдущим, поэтому вся совокупность векторов не зависит от порядка включения последующих векторов при фиксированном множестве переменных $\mathbf{x}_i \in \mathbf{X}_s$. На порядок их следования в матрице $\mathbf{V}_s$ влияет только порядок включения переменных $\mathbf{x}_i$ в модель.

$$\mathbf{v}_1' = \mathbf{x}_1 - (\mathbf{x}_1^T \mathbf{v}_0)\mathbf{v}_0,\ \mathbf{v}_1 = \frac{\mathbf{v}_1'}{|\mathbf{v}_1'|},$$

$$\mathbf{v}_2' = \mathbf{x}_2 - (\mathbf{x}_2^T \mathbf{v}_0)\mathbf{v}_0 - (\mathbf{x}_2^T \mathbf{v}_1)\mathbf{v}_1,\ \mathbf{v}_2 = \frac{\mathbf{v}_2'}{|\mathbf{v}_2'|},$$

и т. д.

$$\mathbf{v}_s' = \mathbf{x}_s - \sum_{j=0}^{s-1} (\mathbf{x}_s^T \mathbf{v}_j)\mathbf{v}_j, \quad \mathbf{v}_s = \frac{\mathbf{v}_s'}{|\mathbf{v}_s'|}.$$

Матрица $\mathbf{V}_s = (\mathbf{V}_{s-1} \vdots \mathbf{v}_s)$ составлена из ортонормированных векторов $\mathbf{V}_s = (\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_{s-1}, \mathbf{v}_s)$. После этого получаем треугольную матрицу $\mathbf{R}_s$ по формуле (4).

Для системы уравнений $\mathbf{X}_s\boldsymbol{\theta}_s = \mathbf{y}$ из определенной системы $\mathbf{V}_s^T \mathbf{X}_s \mathbf{и}_s = \mathbf{V}_s^T \mathbf{y}$ находим решение. Выпишем соотношения для определения компонентов вектора параметров:

$$\theta_{ss} = \frac{\mathbf{y}^T \mathbf{v}_s}{\mathbf{x}_s^T \mathbf{v}_s},$$

$$\theta_{ss-1} = \frac{\mathbf{y}^T \mathbf{v}_{s-1} - (\mathbf{x}_s^T \mathbf{v}_{s-1})\theta_{ss}}{\mathbf{x}_{s-1}^T \mathbf{v}_{s-1}},$$

…

$$\theta_{s0} = \frac{\mathbf{y}^T \mathbf{v}_0 - (\mathbf{x}_1^T \mathbf{v}_0)\theta_{s1} - ... - (\mathbf{x}_s^T \mathbf{v}_0)\theta_{ss}}{\mathbf{x}_0^T \mathbf{v}_0}, \qquad (8)$$

Изменение индекса $s = \overline{s_\ell, s_j}$.организуется таким образом, чтобы был полный перебор вариантов моделей без повторов структур $s_j \in \{m,..,2\}, s_\ell \in \{1,2,...,s_k\}$, $s_k < m$, $s_\ell \le s_j$.

Оценим количество операций.

Учитывая, что операция сложения выполняется намного быстрее, чем операция умножения и деления, обычно ограничиваются подсчетом последних.

Известно [1] , что для решения СЛАУ методом Гаусса без выбора главного элемента требуется $\frac{s^3}{3} + s^2 - \frac{s}{3}$ умножений и делений, решение СЛАУ методом квадратного корня требует $\frac{s^3}{6} + \frac{3}{2}s^2 + \frac{s}{3}$ и s операций извлечения корней. Метод вращения предпола-

гает вчетверо больше операций умножения, чем в методе Гаусса. При больших значениях размерности *s*, можно сказать, что вычислительные затраты на операции умножения и деления в методе Гаусса составляют величину $O\left(\frac{s^3}{3}\right)$, в методе квадратных корней $O\left(\frac{s^3}{6}\right)$, в методе вращений $O\left(\frac{4}{3}s^3\right)$.

Для рекуррентных алгоритмов наименьших квадратов с окаймлением [6] и модификации алгоритмов Гаусса и Грама-Шмидта в работе [3] дана следующая оценка количества вычислительных операций: $3s^2 - 2s$.

Для того, чтобы можно было сравнивать метод Грама-Шмидта-Гаусса с другими методами не будем учитывать оценку свободного члена в уравнении (1).

Для получения ортонормированной системы векторов Грама-Шмидта $\mathbf{V}_s = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{s-1}, \mathbf{v}_s)$ требуется s^2 операций умножения и деления и s операций извлечения квадратного корня. Для получения треугольной матрицы $\mathbf{R}_s$ – $0{,}5s^2 + 2s$ операций и для умножения матрицы на вектор $\mathbf{V}_s^T\mathbf{y}$ – s операций. Для обратного хода при определении параметров s-ой модели по методу Гаусса необходимо сделать $0{,}5s^2 + s$ операций умножения и деления. Всего – $2s^2 + 5s$ операций умножения, деления и извлечения квадратного корня.

При больших значениях размерности *s*, можно сказать, что вычислительные затраты на операции умножения и деления для вложенных структур в рекуррентном методе Гаусса и методе наименьших квадратов с окаймлением составляют величину $O(3s^2)$, в методе Грама-Шмидта-Гаусса вычислительные затраты на операции умножения, деления и извлечения квадратного корня составляют величину $O(2s^2)$.

Пример

Продемонстрируем метод Грама-Шмидта-Гаусса на примере решения СЛАУ четвертого порядка.

Пусть имеем систему условных уравнений: $\mathbf{X}\boldsymbol{\theta} = \mathbf{y}$, $\dim\mathbf{X} = 5\times 4$:

$$\begin{aligned}
\theta_0 + 5\theta_1 - \theta_2 + 2\theta_3 &= 9{,}5\\
\theta_0 + 3\theta_1 + 4\theta_2 - 5\theta_3 &= -3{,}5\\
\theta_0 + \theta_1 + \theta_2 + \theta_3 &= 6{,}5\\
\theta_0 + \theta_1 + \theta_2 - \theta_3 &= 0{,}5\\
\theta_0 - 2\theta_1 + 3\theta_2 - 3\theta_3 &= -4{,}5
\end{aligned},$$

$$\mathbf{X} = \begin{bmatrix} 1 & 5 & -1 & 2\\ 1 & 3 & 4 & -5\\ 1 & 1 & 1 & 1\\ 1 & 1 & 1 & -1\\ 1 & -2 & 3 & -3 \end{bmatrix},\ \mathbf{y} = \begin{bmatrix} 9{,}5\\ -3{,}5\\ 6{,}5\\ 0{,}5\\ -4{,}5 \end{bmatrix}.$$

На первом шаге получаем первый нормированный вектор-столбец $\mathbf{v}_0$, совпадающий по направлению с первым вектором $\mathbf{x}_0$ матрицы $\mathbf{X}$.

$$\mathbf{v}_0 = \begin{bmatrix} 0{,}447213595\\ 0{,}447213595\\ 0{,}447213595\\ 0{,}447213595\\ 0{,}447213595 \end{bmatrix}$$

На втором шаге находим вектор-столбец $\mathbf{v}_1'$ – ортогональную проекцию $\mathbf{x}_1$ к первому вектору $\mathbf{x}_0$ по формуле (6) и нормируем ее по формуле (7). Получим:

$$\mathbf{v}_1 = \begin{bmatrix} 0{,}65192\\ 0{,}268438\\ -0{,}11504\\ -0{,}11504\\ -0{,}69027 \end{bmatrix}.$$

На третьем шаге находим ортогональный $\mathbf{v}_2$ к полученным двум ортогональным векторам $\mathbf{v}_0$ и $\mathbf{v}_1$, и находим проекцию третьего вектора-столбца $\mathbf{x}_2$ на $\mathbf{v}_2$, как $\mathbf{v}_2'$ после чего нормируем его. Получаем:

$$\mathbf{v}_2 = \begin{bmatrix} -0{,}40252\\ 0{,}850235\\ -0{,}23893\\ -0{,}23893\\ 0{,}030135 \end{bmatrix}$$

На четвертом и последнем шаге получаем четвертый вектор-столбец $\mathbf{v}_3$ матрицы $\mathbf{V}$, ортогональной к матрице плана $\mathbf{X}$:

$$\mathbf{v}_3 = \begin{bmatrix} -0{,}13311 \\ 0{,}020478 \\ 0{,}815284 \\ -0{,}53883 \\ -0{,}16382 \end{bmatrix},$$

$$\mathbf{V} = \begin{bmatrix} 0{,}4472136 & 0{,}65192 & -0{,}40252 & -0{,}13311 \\ 0{,}4472136 & 0{,}268438 & 0{,}850235 & 0{,}020478 \\ 0{,}4472136 & -0{,}11504 & -0{,}23893 & 0{,}815284 \\ 0{,}4472136 & -0{,}11504 & -0{,}23893 & -0{,}53883 \\ 0{,}4472136 & -0{,}69027 & 0{,}030135 & -0{,}16382 \end{bmatrix}$$

Умножаем транспонированную матрицу $\mathbf{V}^T$ на матрицу **X,** либо заполняем по формулам (8) соответствующие элементы верхней треугольной матрицы $\mathbf{R}_3$:

$$\mathbf{V}^T\mathbf{X} = \begin{bmatrix} 2{,}236067977 & 3{,}577708764 & 3{,}577709 & -2{,}683282 \\ 0 & 5{,}215361924 & -1{,}879064 & 2{,}032457 \\ -2{,}28983\text{E}-16 & -4{,}02456\text{E}-16 & 3{,}416009 & -5{,}146615 \\ -3{,}88578\text{E}-16 & 7{,}77156\text{E}-16 & -2{,}72\text{E}-15 & 1{,}476982 \end{bmatrix}$$

Умножаем транспонированную матрицу $\mathbf{V}^T$ на вектор **y** и получаем вектор-столбец:

$$\mathbf{V}^T\mathbf{y} = \begin{bmatrix} 3{,}801316 \\ 7{,}554605 \\ -8{,}60783 \\ 4{,}430945 \end{bmatrix}.$$

И, наконец, по формуле (8) вычисляем коэффициенты уравнения, как обратный ход метода Гаусса. Получаем: $\tilde{\mathbf{и}} = \mathbf{и} = \begin{bmatrix} 0{,}5 \\ 1 \\ 2 \\ 3 \end{bmatrix}$.

Выводы и перспективы

Метод Грама-Шмидта-Гаусса решения линейной системы уравнений является наиболее экономичным из известных рекуррентных и нерекуррентных методов, поскольку он пропорционален удвоенному квадрату числа операций. Данный метод непосредственно работает с исходной таблицей (прямоугольной матрицей) данных без предварительного преобразования ее к квадратному виду (к системе нормальных уравнений), что позволяет отсеивать коллинеарные векторы на этапе построения ортонормированных векторов в пространстве точек, проделав лишь s^2 операций умножения. Ввиду того, что по методу Грама-Шмидта-Гаусса при $s > 7$ в 1,5 раза делается меньше операций, чем известными экономичными рекуррентными методами (не говоря о нерекуррентных методах), его ошибка вычислений накапливается в меньшей степени, что дает преимущество данному методу при прочих равных условиях (одинаковый ранг матрицы, одинаковая степень ее обусловленности, одинаковая погрешность вычисления и т.д.) перед другими и делает его наиболее численно устойчивым.

Список литературы

1. А.Ф.Ляхов, Солдатов Е.В., Чернова Е.В. Прямые методы решения систем линейных алгебраических уравнений– Н.Новгород: ННГУ, 1999.–16 с.
2. Шелудько О.И. Самоорганизация математических моделей при решении некоторых задач надежности и контроля. // Дисс. на соиск. учен. степ. канд. техн. наук. К.: КПИ. – 1975. – 166 с.
3. Фаддеев Д.К., Фаддеева В.Н. Вычислительные методы линейной алгебры. Изд. 2-е.. – М.-Л.: Физматгиз, 1963 г.
4. Єфіменко С. М. Інструментальні засоби для дослідження та застосування методів моделювання за статистичними даними // Автореферат канд. дисертації на здоб. наук. ст. канд. техн. наук, К.: МННЦ ИТіС. – 2009. – 20 с.
5. Себер Дж. Линейный регрессионный анализ. – М.: Мир, 1980. – 456 с.

6. Ивахненко А.Г., Степашко В.С. Помехоустойчивость моделирования. – К.: Наук. думка, 1985. –214с.
7. Корн Г., Корн Т. Справочник по математике. – М.: Наука, 1973. – 831 с.
8. Хорн Р., Джонсон Ч. Матричный анализ. – М.: Мир, 1989. – 655 с.

Поступила в редакцию 12.12.2009

УДК 004.75

ТЕЛЕНИК С.Ф.
РОЛИК А.И.
БУКАСОВ М.М.
ЯСОЧКА М.В.

ОПРЕДЕЛЕНИЕ РАСПРОСТРАНЕНИЯ ВЛИЯНИЯ НЕИСПРАВНОСТЕЙ В СЕТИ ДОСТУПА НА КАЧЕСТВО ПРЕДОСТАВЛЯЕМЫХ СЕРВИСОВ

Предлагаются метод и алгоритм определения распространения влияния неисправностей на качество предоставления сервиса абонентам оператора телекоммуникаций, когда показателями качества являются параметры производительности сети доступа.

Problems of fault impact degree determination on the service quality in a communications network are considered, general approach is described. Method and algorithm of fault estimation in access networks are given. Fault management system based on this approach is developed.

Введение

В настоящее время успешность бизнеса предприятий и эффективность деятельности организаций существенно зависит от непрерывности и качества информационного взаимодействия, осуществляемого различными коммуникационными системами, как внутри предприятия, так и между головным офисом и филиалами, предприятием и сторонними организациями, между сотрудниками предприятия и клиентами и т. п. Коммуникационное взаимодействие обеспечивается операторами мобильной связи, телефонными компаниями, Интернет-сервис провайдерами (ISP) и другими поставщиками коммуникационных услуг. При этом провайдеры или операторы телекоммуникаций, как правило, предоставляют комплекс коммуникационных услуг.

Поскольку потери бизнеса от несвоевременности получения или низкого качества коммуникационных услуг могут быть очень значительными, предприятия и корпорации перекладывают их на поставщиков этих услуг в виде штрафных санкций, размер которых может быть весьма существенным. Большие объемы и стоимость телекоммуникационных услуг при одновременных жестких требованиях к непрерывности и качеству их предоставления, существенные штрафные санкции и опасность потери клиентов вынуждают телекоммуникационные компании непрерывно отслеживать состояние коммуникационной системы. При возникновении неисправностей необходимо не только осуществлять их локализацию и максимально быстрое устранение, но и оперативно определять степень влияния неисправностей в коммуникационной системе на качество предоставления сервиса всем клиентам, в первую очередь, наиболее важным.

Для этого системы управления ИТ-инфраструктурой (СУИ) [1] провайдеров телекоммуникационных услуг должны содержать дополнительные модули, позволяющие связать неисправности, возникающие в коммуникационной системе, с соответствующим ухудшением качества предоставления услуг абонентам. Разработка соответствующих модулей сопряжена с рядом трудностей, обусловленных большим количеством элементов коммуникационных систем, сложностью и запутанностью структуры системы, различным влиянием неисправностей и совокупности неисправностей на качество предоставляемого сервиса, наличием разнообразных схем резервирования, большим количеством клиентов, часто исчисляемых миллионами, и пр. Для разработки модулей СУИ, позволяющих в минимальные сроки выявлять клиентов, ощущающих снижение качества обслуживания вследствие неисправностей в коммуникационной системе, и оценить степень ухудшения качества получаемых телекоммуникационных услуг, необходимо разработать эффективные математические модели и алгоритмы определения степени влияния неисправностей в сетях провайдеров коммуникационных услуг на качество предоставления телекоммуникационных сервисов. Решению этих задач и посвящена предлагаемая статья.

Постановка проблемы в общем виде

В сфере телекоммуникационных услуг на доходы коммуникационных компаний непосредственное влияние оказывает качество обслуживания абонентов. Провайдеры коммуникационных услуг, кабельного и спутникового телевидения, провайдеры доступа в Интернет или услуг в Интернете, операторы мобильной связи несут существенные финансовые убытки из-за потери клиентов, недовольных обслуживанием, или штрафных санкций, накладываемых при нарушении непрерывности предоставления услуг или временном предоставлении некачественного сервиса.

Основным документом, регламентирующим отношения потребителя телекоммуникационных услуг и коммуникационной компании, является соглашение об уровне обслуживания (Service Level Agreement, SLA). SLA фиксирует качественное и количественное описание сервисов с указанием условий, параметров и других характеристик услуг, как с точки зрения клиента, так и с точки зрения провайдера телекоммуникационных услуг, а также взаимные ответственности. В SLA, в частности, определяются требования к: параметрам производительности и поддержки — среднему или максимальному времени реагирования; доступности — проценту времени обслуживания, в течение которого услуга фактически доступна пользователю; надежности — количеству инцидентов за определенный срок в течение согласованного времени обслуживания; устойчивости — времени восстановления услуги (полного или частичного) в случае повреждения инфраструктуры провайдера.

В случае невыполнения требований к качеству обслуживания, оговоренных в SLA, провайдер телекоммуникационных услуг несет убытки в результате потери клиентов или штрафных санкций. В то время как стабильно высокое качество обслуживания, разнообразие предоставляемых услуг, дифференциация обслуживания клиентов и другие факторы способствуют повышению среднемесячного дохода на абонента (Average Revenue per User — ARPU). Показатель ARPU определяется средним доходом от абонента за фиксированный интервал времени, который обычно равен одному месяцу, и используется ориентированными на потребителя компаниями, предоставляющими не только телекоммуникационные, но и ИТ-услуги.

Одним из ключевых компонентов телекоммуникационной инфраструктуры являются сети доступа, посредством которых обеспечивается доступ абонентов к телекоммуникационным сервисам. Крупные поставщики телекоммуникационных услуг могут насчитывать миллионы клиентов в сетях доступа, а затраты на создание, модернизацию и поддержание работоспособности сетей доступа составляют значительную часть бюджета операторов.

Большое количество абонентов и необходимость снижения затрат на обслуживание своей инфраструктуры вынуждают операторов производить дифференциацию обслуживания абонентов, уделяя больше внимания наиболее важным из них. В этом случае при обнаружении СУИ неисправностей в коммуникационной системе определяется степень влияния этих неисправностей на качество предоставления услуг абонентам и, в отношении наиболее важных из них, предпринимаются дополнительные мероприятия по обслуживанию, в то время как остальные клиенты довольствуются номинальным обслуживанием.

Для реализации эффективного дифференцированного подхода к обслуживанию абонентов в случае возникновения неисправностей в коммуникационной системе необходимо обеспечить в СУИ поддержание жизненного цикла снижения потерь из-за ухудшения качества предоставляемых услуг, состоящего из следующих этапов: выявление и локализация неисправностей; определение их влияния на качество предоставления услуг; выделение наиболее важных абонентов, пострадавших в результате получения некачественных услуг; проведение дополнительных мероприятий по отношению к этим абонентам.

Анализ публикаций по теме

Структура и принципы построения СУИ предложены в [1, 2]. Однако описание СУИ сделано без рассмотрения принципов построения подсистемы управления устранением неисправностей и, соответственно, без описания работы модуля, ответственного за определение распространения влияния неисправностей на элементы информационно-телекоммуникационной системы.

В [3] рассматриваются вопросы управления устранением неисправностей в ИТ-системах без учета влияния неисправностей в сети на качество предоставления абонентам телекоммуникационных услуг.

Анализу качества функционирования элементов информационно-телекоммуникационных систем посвящена работа [4], но в ней не предлагаются модели и методы анализа распространения изменения состояния элементов телекоммуникационной сети на параметры функционирования оборудования в точках доступа абонентов.

В [5] предложен метод и система определения влияния неисправностей в телекоммуникационных сетях на работоспособность сервисов. На основе анализа собранных данных, характеризующих работоспособность сервисов, предложенная система позволяет предсказывать их последующее состояние. Объединяя фактические и потенциальные неисправности в сети, система снабжает подразделение технической поддержки оператора мобильной связи информацией об отказах в оборудовании. Система не учитывает важность абонентов, а технология мобильной связи ограничивает возможности службы технической поддержки в выборе методов индивидуальной работы с наиболее важными абонентами.

В [6] предложена сервисно-ориентированная модель, рассматривающая различные фазы предоставления сервиса, позволяющая определить, каким образом неисправности, возникающие в отдельных сетевых элементах влияют на работоспособность сервисов в целом. Для этого вводится понятие индикаторов состояния компонентов сети применительно к отдельным сервисам, а умножение значений индикаторов на соответствующий вес позволяет получить приоритетный список неисправностей, оказывающих наибольшее влияние на качество предоставления сервисов. При этом не предлагаются методы оценки влияния неисправностей на качество предоставления сервисов отдельным абонентам.

Целью статьи является разработка метода и алгоритма определения влияния неисправностей в сетях доступа операторов телекоммуникационных услуг на качество предоставления сервисов абонентам.

Проблемы определения влияния неисправностей в сетях доступа на качество предоставления сервиса абонентам

Сети операторов связи можно разделить на два класса — магистральные сети передачи данных и сети доступа.

Магистральные сети, являясь ядром больших сетевых инфраструктур, связывают крупные телекоммуникационные узлы или региональные подсети. Магистральные сети используют каналы связи с большой пропускной способностью, объединяют информационные потоки большого количества подсетей и отличаются повышенной надежностью.

Одним из ключевых компонентов телекоммуникационной инфраструктуры являются сети доступа, посредством которых абонентам предоставляются различные телекоммуникационные услуги. В сетях доступа крупных операторов связи могут насчитываться миллионы абонентов, а затраты на создание, модернизацию и поддержание работоспособности таких сетей составляют значительную часть бюджета операторов.

Многообразие и большое количество оборудования, существенная протяженность линий связи, высокая стоимость резервирования, жесткие требования к экономичности обслуживания и другие факторы делают сети доступа существенно менее надежными, чем магистральные сети. Большие штрафные санкции и угроза потери клиентов вынуждают операторов немедленно реагировать на неисправности, оказывающие непосредственное влияние на качество предоставляемых услуг. Наряду с оперативным обнаружением и локализацией неисправностей со стороны СУИ и их устранением службой эксплуатации сетей доступа подразделение оператора, занимающееся обслуживанием клиентов, производит предупредительные мероприятия по отношению к важным клиентам, призванные нивелировать негативные последствия неисправностей, влияющих на качество услуг. Эти мероприятия включают в себя своевременное информирование клиентов о возникших неполадках, переключение на резерв или предоставление альтернативных источников сервиса и другие действия с целью максимально сгладить недовольство и негативный эффект от некачественного обслуживания у наиболее важных клиентов.

Поэтому очень важно быстро определять влияние неисправностей в сети доступа на качество абонентского обслуживания. В то же время, большое количество абонентов крупных операторов связи, иногда насчитывающее миллионы юридических и физических лиц, громоздкость и сложность коммуникационных систем, содержащих до десяти и более иерархических уровней в сетях доступа, большое количество типов и единиц коммутационного оборудования, непростые зависимости влияния неисправностей на качество сервиса, сложная интерференционная картина воздействия совокупности неисправностей и другие факторы превращают задачу быстрого определения распространения влияния неисправностей на качество сервиса для отдельного абонента в чрезвычайно трудоемкую задачу, решение которой требует больших вычислительных ресурсов. Трудность обусловлена большими размерами инвентаризационной базы данных оператора, поиск взаимосвязей в которой может потребовать существенного времени, в то время как подразделение обслуживания клиентов должно начать реагировать через считанные минуты после возникновения неисправностей, предпринимая предопределенные действия по отношению к наиболее важным клиентам.

Поэтому необходимо создать методы и простые модели, позволяющие СУИ максимально быстро получать картину распространения неисправностей по телекоммуникационной сети от проблемного узла до клиентского оборудования.

Учитывая большое количество разнообразных сервисов в сети, создать универсальную модель для оценки влияния неисправностей на все сервисы одновременно не представляется возможным, поэтому следует задействовать совокупность различных моделей, каждая из которых учитывает особенности отдельных или группы родственных сервисов.

Поэтому нужен общий подход, который позволит определить структурно-функциональную схему взаимодействия различных моделей оценки влияния неисправностей на сервисы и качество предоставления услуг абонентам. По меньшей мере, этот подход должен помочь разработчикам и пользователям СУИ определить последовательность действий, построить механизм подключения моделей, уточнить характеристики, влияющие на распространение неблагоприятных последствий неисправностей.

Ниже предлагается такой подход для определения распространения негативных последствий неисправностей в сетях доступа на качество предоставления услуг абонентам, построенный на определенных правилах распространения влияния неисправностей, учитывающих особенности структуры сети и сервисов, используемые технологии и ряд других важных факторов.

Общее описание предлагаемого подхода к оценке влияния неисправностей на качество предоставляемого сервиса

Из всех неисправностей, выявленных в сети доступа, СУИ выделяет только те, которые оказывают влияние на качество предоставления сервиса. Остальные неисправности, которые обнаруживаются СУИ, игнорируются при определении влияния на качество сервиса, но информация о них используется службой эксплуатации для их устранения в соответствии с определенными регламентами и с учетом их значимости для коммуникационной системы. Например, отказ резервного узла в кластерной системе или ненагруженного резервного канала связи снижает надежность коммуникационной системы оператора связи, но не отражается на качестве предоставления сервиса клиентам.

Качество предоставления каждого сервиса, определяется через значения группы параметров, оговоренных в SLA. Потому сначала создаются упрощенные схемы сети доступа, построенные с точки зрения возможного влияния неисправностей в узлах на значения отдельных параметров, зафиксированных в SLA, и учитывающие особенности распространения влияния неисправностей на эти параметры от места возникновения неисправности через промежуточные узлы до абонентского оборудования.

Далее для каждого типа узлов прописываются правила, нивелирующие особенности отдельных сетевых технологий, связывающие тип неисправности, произошедшей в узле, со степенью влияния на значения отдельных или группы однородных параметров.

Кроме того, для каждого типа узлов определяются правила распространения влияния неисправностей на значение рассматриваемого параметра, характеризующего качество

сервиса, в направлении от узлов, расположенных выше в иерархии сети, к узлам, расположенным ниже в иерархии сети или к абонентскому оборудованию, если между рассматриваемым узлом и оборудованием доступа абонентов к сети промежуточные узлы отсутствуют.

После этого, последовательно от узла верхнего уровня, в котором произошла неисправность, для каждого из зависимых узлов нижерасположенных уровней рассчитываются коэффициенты снижения значений показателей качества сервиса до тех пор, пока не будет достигнут уровень абонентского оборудования, для которого также рассчитываются значения коэффициентов, определяющих степень снижения качества предоставляемых услуг относительно нормативного значения.

При этом, если в сети доступа возникла только одна неисправность, для анализа распространения ее влияния не рассматриваются пути, исходящие с верхнего уровня на смежный нижний уровень, значением коэффициентов снижения качества предоставляемых услуг в которых можно пренебречь. В таких случаях говорят о затухании влияния неисправностей.

Если значение коэффициента, определяющего степень снижения качества предоставляемого оператором сервиса, на выходе абонентского оборудования превышает пороговую величину, задаваемую для каждого абонента или класса абонентов в зависимости от их важности, то СУИ вырабатывает информационные сообщения, поступающие в подразделение обслуживания клиентов для реагирования в соответствии с заранее прописанным регламентом.

Применение предлагаемого подхода для оценки влияния неисправностей на показатели производительности сети доступа

Рассмотрим случай, когда в качестве параметров, влияние на которые будет оцениваться при возникновении неисправностей, выбираются показатели производительности сети доступа. К таким параметрам относится, например, скорость доступа абонентов к сети Интернет. Этот параметр относится к категории важнейших, в связи с тем, что снижение скорости доступа негативно сказывается на качестве предоставления других услуг, таких, например, как качество передачи голосовых сообщений сервиса VoIP.

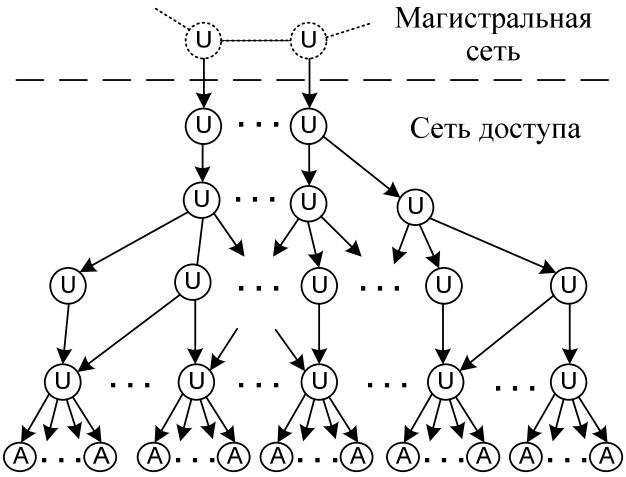


Рис. 1. Схема соединений узлов в сети доступа

Для анализа распространения воздействия неисправностей и оценки ухудшения качества сервиса, предоставляемого абонентам, когда в качестве параметров используются показатели производительности, достаточно использовать единственную схему, аналогичную схеме, приведенной на рис. 1. Эта схема представляет собой ориентированный граф, вершины *U* которого соответствуют узлам сети доступа, таким как маршрутизаторы, коммутаторы, мультиплексоры доступа и др., ветви — линиям связи, а листья *A* — абонентскому оборудованию.

Схема, приведенная на рис. 1, позволяет установить пути прохождения потоков данных от узлов к абонентскому оборудованию, определить поражаемые участки сети в результате распространения последствий неисправностей, получить интерференционную картину при возникновении нескольких неисправностей и т. п.

Эта схема, по сути, определяет лишь связность узлов сети доступа для передачи трафика. Такой схемы достаточно для решения поставленной задачи при рассмотрении сетей передачи данных, функционирующих на уровнях не выше второго уровня модели OSI. Однако ее не достаточно при сервисно-ориентированном подходе к определению влияния неисправностей. Так, в настоящее время превалируют мультисервисные сети, построенные с использованием IP-протокола, и для оценки влияния неисправностей в сети доступа на качество предоставления сервиса абонентам необходимо учесть состояние различных служб, обеспе-

чивающих работоспособность сервисов. В последнем случае дополнительно к схеме, приведенной на рис. 1, необходимо использовать схему, аналогичную той, что приведена на рис. 2. Эта схема, по сути, перечисляет сервисы, которыми пользуется абонент, и службы, обеспечивающие работу этих сервисов.

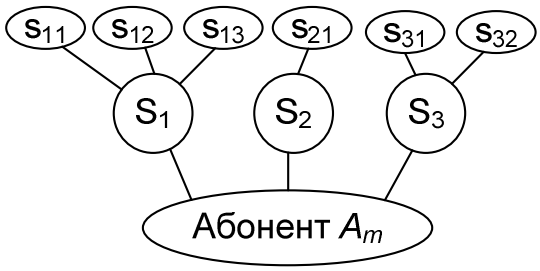


Рис.2. Сервисы, используемые абонентом A_m

В качестве примера на рис. 2 сервисы S_1, S_2, S_3, которыми пользуется m-й абонент A_m, поддерживают службы s_{11}, …, s_{32}. Такая схема составляется для каждого важного абонента A_m, $m = 1, \ldots, M$, где M — количество абонентов оператора связи, или для группы абонентов, и позволяет быстро определить влияние неисправностей в элементах сетевой инфраструктуры, обеспечивающих работу отдельных служб. Например, в случае отказа элемента, который обеспечивает работоспособность службы s_{21}, например, почтового сервера, прекращается предоставление сервиса S_2, например, электронной почты, что немедленно сказывается на абонентах $A_{i(S_2)}$, $i(S_2) = 1, \ldots, N(S_2)$ пользующихся этим сервисом (см. рис. 3).

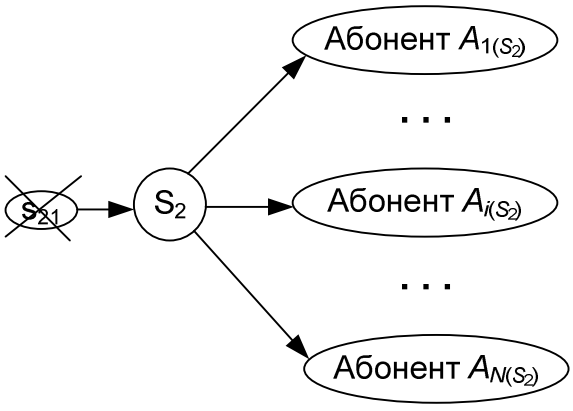


Рис. 3. Распространение неисправностей в элементах, поддерживающих работу служб, на качество сервисов для абонентов

Перечень абонентов услуг, например, клиентов электронной почты, целесообразно хранить в таблицах, которые уже отсортированы по важности абонентов. Тогда при снижении качества предоставления сервиса, вызывается соответствующая таблица, и предпринимаются необходимые мероприятия по работе с абонентами.

Таким образом, для оценки влияния неисправностей в сети доступа на качество предоставления услуг абонентам в мультисервисных сетях необходимо одновременно использовать обе схемы, приведенные на рис. 1 и рис. 2.

Остановимся на этом подробнее. Действительно, для оценки влияния неисправностей на значения отдельных параметров, характеризующих, например, продуктивность сети, таких как скорость доступа в Интернет или реальная полоса пропускания, схемы, приведенной на рис. 1 достаточно. Но представим, что в качестве метрики оценки влияния неисправностей в сети на качество предоставления услуг используется средняя величина или диапазон изменения джиттера при передаче голосовых сообщений. Тогда неисправность в узлах и линиях связи может привести к снижению качества передаваемого голоса или сделать обмен голосовыми сообщениями в принципе неосуществимым. Поэтому оценить качество голоса при использовании технологии VoIP с помощью только схемы, приведенной на рис. 1, не представляется возможным.

В силу изменчивости структуры и состава сетей доступа нецелесообразно строить совокупность моделей, которые будут непосредственно связывать все возможные неисправности в них с изменением значений параметров производительности каждого конкретного абонента. Более рациональным выглядит подход, который позволяет вычислить влияние неисправностей на абонентов на основе структуры, состава сети доступа и некоторых общих принципов распространения влияния неисправностей в сети на качество предоставления услуг. Иными словами, в основу подхода целесообразно положить модели распространения воздействия неисправного узла на зависимые узлы нижних иерархических уровней, до тех пор, пока не будет достигнут уровень абонентского оборудования. Только после этого можно будет определить степень влияния неисправности на

показатели качества для каждого абонента, если ее воздействие не затухло на верхних уровнях.

Итак, рассмотрим распространение влияния неисправностей, когда в качестве показателей качества используются параметры производительности. Для конкретизации изложения будем использовать в качестве такого параметра скорость доступа в Интернет.

Для формальной оценки влияния неисправностей на узлы и абонентов сети доступа введем коэффициент снижения скорости передачи $k_{i,j(n)}$, $i = 1, \ldots, I$, $j = 1, \ldots, J_i$, $n = 1, \ldots, N_{i,j}$, где j — номер узла в i-м уровне иерархии сети доступа, I — количество уровней иерархии, выделяемых в рассматриваемой сети доступа, J_i — количество узлов в i-м иерархическом уровне, n — номер исходящей линии связи из i,j-го узла сети, $N_{i,j}$ — количество линий связи, исходящих из i,j-го узла сети.

Коэффициент $k_{i,j(n)}$ принимает значения в диапазоне [0, 1] и определяет, во сколько раз уменьшается скорость передачи в n-й линии связи, исходящей из i,j-го узла сети, относительно нормативного показателя.

Если воздействие всех неисправностей в узлах и линиях связи представить таким образом, чтобы их влияние сводилось к изменению значения коэффициентов $k_{i,j(n)}$, то фактическая скорость доступа $Vp_{i,j(n)}$ в n-й линии, исходящей из i,j-го узла будет определяться следующим образом

$$Vp_{i,j(n)} = k_{i,j(n)} \times Vн_{i,j(n)}, \qquad (1)$$

где $Vн_{i,j(n)}$ — нормативная скорость доступа в n-й линии на выходе i,j-го узла.

С помощью соотношения (1) определяется фактическая скорость доступа Vp_m, для каждого m-го абонента A_m, $m = 1, \ldots, M$. Если полученная скорость Vp_m больше порогового значения, оговоренного в SLA, т.е. при выполнении условия

$$Vp_m \geq Vr_m, \qquad (2)$$

где Vr_m — пороговое значение допустимого снижения скорости для m-го абонента, то в отношении m-го абонента никакие меры не предпринимаются. В противном случае действия, предпринимаемые оператором связи, определяются степенью снижения скорости доступа и значимостью абонента.

Введем зависящий от состояния узла коэффициент $K_{i,j}$, определяющий снижение относительно нормативного значения параметра производительности j-го узла, $j = 1, \ldots, J_i$, i-го, $i = 1, \ldots, I$, уровня иерархии узлов сети доступа, J_i — количество узлов U на i-м уровне.

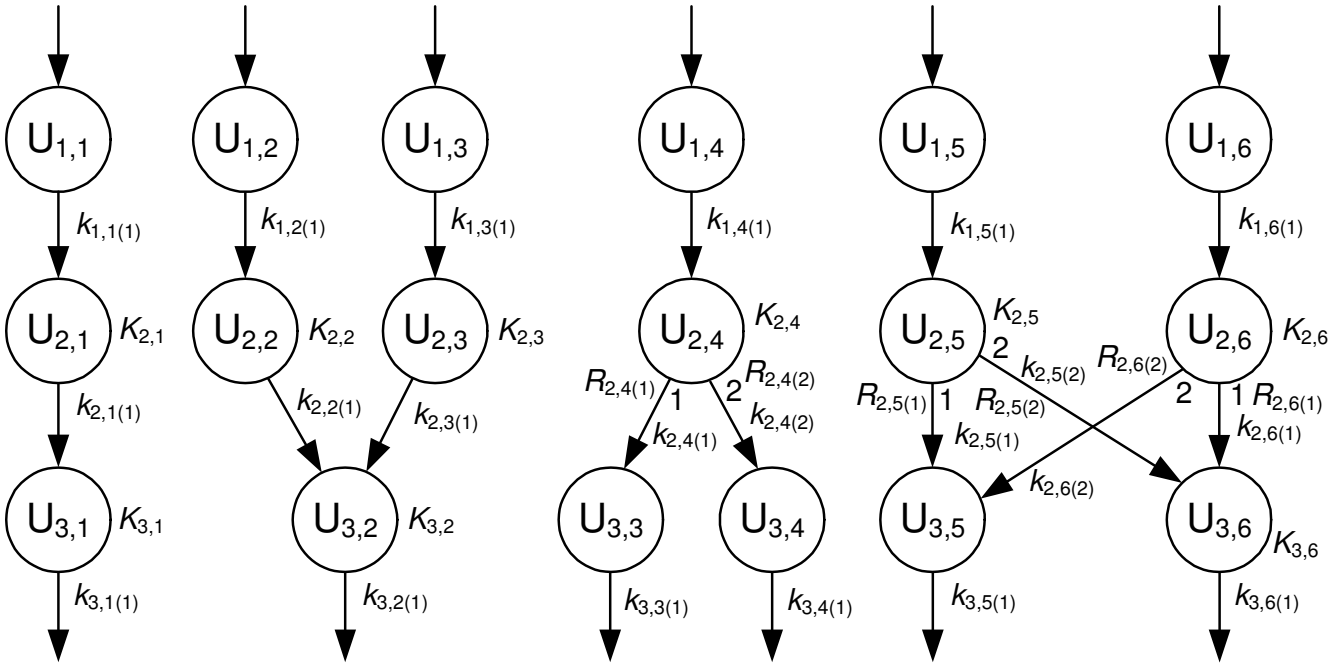
Для рассматриваемого случая коэффициент $K_{i,j}$ определяет степень снижения скорости передачи на выходах узла $U_{i,j}$ относительно нормативного, а значения коэффициента $K_{i,j}$ зависят от состояния узла, обусловленного неисправностями в сети доступа.

Коэффициент $K_{i,j}$ принимает значения в диапазоне [0, 1]. Значение $K_{i,j}$ устанавливается равным 0, если возможные неисправности приводят к тому, что трафик через узел $U_{i,j}$ не проходит. Значение $K_{i,j}$ устанавливается равным 1 при полностью исправном узле $U_{i,j}$, или, в случае, когда произошедшая неисправность в узле $U_{i,j}$ не влияет на скорость передачи данных. Для всех остальных неисправностей значения $K_{i,j}$ принимают промежуточные значения в диапазоне [0, 1].

Коэффициент $K_{i,j}$ можно определять на основании правил, прописываемых для каждого типа узлов в зависимости от структуры сети доступа. Правила должны адекватно учитывать особенности коммуникационных технологий, настройки оборудования, зависимость от величины коэффициентов снижения скорости передачи во входящих линиях, наличие резервирования и другие параметры. При обнаружении неисправностей в сети производится их локализация, определяется степень серьезности и производится сопоставление состояния узла $U_{i,j}$ конкретному значению коэффициента $K_{i,j}$ по соответствующему правилу.

Для повышения быстродействия модуля СУИ, ответственного за определение влияния неисправностей на качество абонентского обслуживания, правила должны быть максимально простыми.

Упомянутые правила, в первую очередь, должны учитывать структурно-технологические особенности сети доступа. Исходя из анализа существующих сетей доступа, принципов их функционирования и эксплуатации, используемых в них коммуникационных технологий и оборудования, целесообразно положить в основу определения правил следующие возможные способы соединения узлов сети доступа (см. рис. 4).



а) ***б)*** ***в)*** ***г)***

Рис. 4. Возможные способы соединения узлов сети доступа

Способ а). При исправном узле $U_{2,1}$ скорость доступа в исходящей линии определяется входной скоростью и ограничениями, накладываемыми на скорость передачи и приема, устанавливаемую для портов узла $U_{2,1}$ при его конфигурации.

Если узел $U_{2,1}$ неисправен, то возможны два случая. В первом случае, когда коэффициент $K_{2,1} = 0$ скорость в исходящей ветви равна 0 независимо от скорости на входе.

Во втором — $0 < K_{2,1} < 1$ и неисправность в узле $U_{2,1}$ приводит к снижению скорости в исходящей линии в $K_{2,1}$ раз.

Тогда коэффициент снижения скорости относительно номинальной в исходящей линии связи узла $U_{2,1}$ будет определяться следующим образом

$$k_{2,1(1)} = K_{2,1} \times k_{1,1(1)}. \qquad (3)$$

В свою очередь, коэффициент $k_{3,1(1)}$ снижения скорости на выходе узла $U_{3,1}$ с учетом (3) будет определяться с помощью следующего выражения

$$k_{3,1(1)} = K_{3,1} \times k_{2,1(1)} = K_{3,1} \times K_{2,1} \times k_{1,1(1)}. \qquad (4)$$

Способ б). Если в один узел, например, в узел $U_{3,2}$ ведут несколько линий связи, которые используются для распределения нагрузки, поступающей от абонентов, а не для целей резервирования, то при $K_{3,2} = 1$ для оценки влияния нужно использовать выражение

$$k_{3,2(1)} = \begin{cases} k_{2,2(1)} + k_{2,3(1)}, \text{ если } k_{2,2(1)} + k_{2,3(1)} \le 1, \\ 1, \text{ если } k_{2,2(1)} + k_{2,3(1)} > 1. \end{cases}$$

Если коэффициент $K_{3,2}$ не равен 1, то производится соответствующая коррекция коэффициента $k_{3,2(1)}$.

Способ в). Введем коэффициент $R_{i,j(n)}$, определяющий распределение полосы пропускания или скорости передачи между n-ми исходящими линиями связи j-го узла i-го иерархического уровня.

Коэффициент $R_{i,,j(n)}$ принимает значения в диапазоне [0, 1], причем, если скорость доступа во входящей линии полностью распределяется между портами исходящих линий без дополнительных ограничений, устанавливаемых при конфигурации узла $U_{2,4}$, то принимается

$$\sum_{n=1}^{N_{i,j}} R_{i,j(n)} = 1.$$

Коэффициент снижения скорости в 1-й и 2-й линиях связи на выходе узла $U_{2,4}$ при возникновении неисправностей будет определяться по формуле (3).

Способ г). Для повышения надежности в сети доступа используются резервные линии связи. На рис. 4*г* такими линиями являются

линии 2, исходящие из узлов $U_{2,5}$ и $U_{2,6}$, в то время как линии 1 являются основными. Переключение на резерв в IP-сетях происходит автоматически при выходе из строя основной линии связи или основного узла, а распределение скоростей в основной и резервной линиях прописывается в узловых элементах при их конфигурации. Тогда для оценки коэффициента $k_{3,6(1)}$ нужно использовать выражение

$$k_{3,6(1)} = \begin{cases} K_{3,6} \times k_{2,6(1)}, \text{ если } K_{2,6} \text{ равен 1,} \\ K_{3,6} \times k_{2,5(2)}, \text{ если } K_{2,6} \text{ равен 0,} \\ \qquad\qquad \text{а } K_{2,5} \text{ равен 1,} \\ 0, \text{ если } K_{2,6} = K_{2,5} = 0. \end{cases}$$

В свою очередь, для оценки коэффициента $k_{2,5(2)}$ нужно использовать выражение

$$k_{2,5(2)} = \begin{cases} R_{2,5(2)} \times K_{2,5} \times k_{1,5(1)}, \text{ при } K_{2,6} = 0, \\ 0, \text{ если } K_{2,6} \neq 0. \end{cases}$$

Естественно, что выделение части канальных ресурсов на поддержание резервной линии может повлечь за собой снижение скорости в основной линии. Тогда для оценки коэффициента $k_{2,5(1)}$ нужно использовать выражение

$$k_{2,5(1)} = \begin{cases} R_{2,5(1)} \times K_{2,5} \times k_{1,5(1)}, \text{ при } K_{2,6} = 0, \\ K_{2,5} \times k_{1,5(1)}, \text{ если } K_{2,6} \neq 0. \end{cases}$$

Таким образом, состояние одного из узлов, например, $U_{2,6}$ может оказывать влияние на коэффициенты снижения скорости в выходных линиях других узлов, в данном случае $U_{2,5}$, находящихся на том же или вышерасположенных уровнях относительно рассматриваемого узла и не связанных с ним физически.

Такую логическую взаимосвязь состояний можно установить на основе анализа резервных линий и узлов на схемах, подобных рис. 1 или 4. Учитывая низкую динамику создания и удаления резервных узлов и линий связи, информацию о взаимовлиянии можно отдельно фиксировать в качестве свойств или параметров узлов, отказы в которых оказывают косвенное влияние на другие узлы, или задавать в виде соответствующих правил. Для сохранения единства подхода взаимовлияние состояний узлов задается изменением состояния зависимого узла при изменении состояния основного.

На самом нижнем уровне иерархии сети доступа коэффициент $k_{i,j(n)}$ определяет, во сколько раз уменьшилась скорость передачи на входе или выходе абонентского оборудования относительно скорости, заданной в SLA. Точка контроля скорости на входе или выходе абонентского оборудования определяется организационно или ограничивается возможностями СУИ контролировать выход абонентского оборудования.

Фактическая скорость доступа Vp_m каждого m-го абонента A_m, $m = 1, \ldots, M$, определяется с помощью следующего выражения, полученного по аналогии с выражением (1)

$$Vp_m = k_m \times Vн_m,$$

где k_m коэффициент, соответствующий $k_{i,j(n)}$ на выходе абонентского оборудования, $Vн_m$ — нормативная скорость доступа m-го абонента.

Теперь принимаются решения, о которых говорилось выше. Если скорость Vp_m меньше порогового значения, оговоренного в SLA, т. е. при выполнении условия (2), то в отношении m-го абонента никакие меры не предпринимаются по причине отсутствия существенного влияния на уровень услуг. В противном случае, действия, предпринимаемые подразделением поддержки клиентов оператора связи, определяются степенью снижения скорости доступа и значимостью абонента.

При использовании нескольких пороговых значений Vr_m можно дифференцировать действия подразделения поддержки клиентов.

Ниже предлагается алгоритм работы модуля СУИ, ответственного за определение влияния неисправностей в сети доступа на показатели производительности качества обслуживания абонентов.

Алгоритм работы СУИ

Предлагается следующий алгоритм оценки неисправностей в сети доступа на качество предоставления услуг, которые можно оценить с помощью показателей производительности.

Шаг 1. Обнаружение и локализация неисправностей в сети доступа. Информация о них поступает в СУИ от CRM-системы и си-

стем управления отдельными коммуникационными технологиями, либо СУИ самостоятельно обнаруживает и анализирует неисправности. После фильтрации и корреляции сообщений о неисправностях выявляется проблемный узел или линия связи.

Шаг 2. Оценивается влияние неисправности на снижение производительности путем определения значений коэффициентов $K_{i,j}$ и $k_{i,j(n)}$ только для проблемного узла и исходящих линий связи. Если значения снижения производительности не превышают порога реагирования, который свидетельствует о существенности неисправности, и в сети имеется только одна неисправность, то дальнейшее исследование не производится.

Шаг 3. Если неисправность оказывается существенной или имеет место несколько независимых неисправностей, то выявляются узлы, расположенные на смежных нижерасположенных иерархических уровнях, на которые влияние неисправностей распространяться непосредственно, и узлы, на которые неисправность в рассматриваемом узле влияет косвенно. Для всех этих узлов производится расчет коэффициентов $K_{i,j}$ и $k_{i,j(n)}$. Если промежуточные коэффициенты не превышают порог реагирования и ошибок в сети не более одной, то считается, что влияние неисправности затухло и дальнейшее определение распространения влияния неполадок не производится.

Шаг 4. Осуществляется проверка, достигнут ли уровень абонентского оборудования. Если не достигнут, то повторяются шаги 2 и 3.

Шаг 5. Выявляются все абоненты, на которых неисправность сказалась снижением производительности. Абоненты ранжируются по важности и для каждого из них, начиная с наиболее важных, производится сравнение значений рассчитанных коэффициентов k_m с пороговыми значениями. Если величина снижения производительности не превышает порог реагирования, то никаких действий не производится. Если порог реагирования превышен, то СУИ генерирует соответствующее информационное сообщение.

Предложенный алгоритм позволяет аналитически рассчитать параметры производительности для каждого из абонентов оператора связи, на котором может сказаться неисправность в сети доступа.

Выводы

Предложен подход к определению распространения влияния неисправностей в сетях доступа на качество предоставляемых услуг абонентам операторов телекоммуникаций, когда показателем качества является скорость доступа. Предложенные модели позволяют аналитически рассчитать влияние неисправностей в узлах сети на снижение показателей продуктивности для каждого абонента. Подход реализован подсистемой локализации и устранения неисправностей СУИ SmartBase.ITSControl, которая внедрена в ряде крупных реальных объектов автоматизации.

Список литературы

1. Теленик С.Ф., Ролік О.І., Букасов М.М., Соколовський Р.Л. Система управління інформаційно-телекомунікаційною системою корпоративної АСУ// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. —2006. — № 45. — С. 112—126.
2. Теленик С.Ф., Ролик А.И., Букасов М.М., Волошин А.В., Галушко Д. А. Система управления ИТ-инфраструктурой — путь к повышению эффективности функционирования пред-приятия// Матеріали конференції «Інформаційні технології — інструмент підвищення конкурентоздатності підприємств»: 9—10 грудня 2009. — Київ. — УкрНЦ РІТ, 2009. — С. 30—33.
3. Ролик А.И., Тимофеева Ю.С., Турский Н.И. Управление устранением неисправностей в ИТ-системах// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. — К.: «ВЕК+», — 2008. — № 49. — С. 94—107.
4. Ролик А.И., Глушко Е.В. Анализ качества функционирования элементов информационно-телекоммуникационных систем// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. — К.: «ВЕК+», — 2008. — № 48. — с. 113—120.
5. Пат. 5872911 США, МКИ G06F 11/34. Method and system of service impact analysis in a communications network/ M. Berg; MCI Communications corp. — № 581748; Заявлено 29.12.95; Опубл. 16.02.99. — 16 с.

6. Пат. 7092707 США, МКИ H04Q 7/20, H04B 1/38, H04B 7/00. Service impact analysis and alert handling in telecommunications systems/ R. Lau, K. Kant, R. Khare; Telcordia Technologies, Inc., — № 10/779432; Заявлено 18.08.05; Опубл. 15.08.06. — 21 с.

Поступила в редакцию 12.12.2009

УДК 004.451

ХАРУК Р. В.

ОБРАБОТКА ДАННЫХ ИЗ ЖУРНАЛА АУДИТА В СИСТЕМАХ МОНИТОРИНГА СОБЫТИЙ БЕЗОПАСНОСТИ

В статье представлен метод и алгоритмы обработки данных о событиях безопасности в системе мониторинга событий безопасности, предлагается новый универсальный формат для представления событий аудита в системе. Новая структура позволит осуществить представление данных распространенных систем аудита в формализованном виде, удобном для дальнейшей обработки и автоматического анализа. Это способствует ускорению обработки данных событий. При обработке определяется принадлежность каждого события к классам событий. Данное решение позволяет выделять наиболее опасные (класс событий-тревог) из всех событий, что, в свою очередь, облегчает весь процесс анализа.

In article methods and algorithms of data processing about safety events in system of monitoring of events of safety are developed, the new universal format for representation of events of auditing system is offered. The new structure will allow implementing data presentation of widespread auditing systems in the formalized kind convenient for the further processing and the automatic analysis. It promotes acceleration of data processing of events. At processing the accessory of each event to classes of events is defined. The given decision allows allocating the most dangerous (a class of events-alarms) from all events that, in turn, facilitates all process of the analysis.

Введение

Наблюдение за действиями программ и компонентов операционной системы рабочего места является одним из важных компонентов системы безопасности информационной системы. В операционных системах существует несколько принципиально различных методов сбора событий, связанных с файловой системой: перехват запроса к драйверу файловой системы и перехват системных вызовов операционной системы.

Рассмотрим второй метод, который является наиболее распространенным.

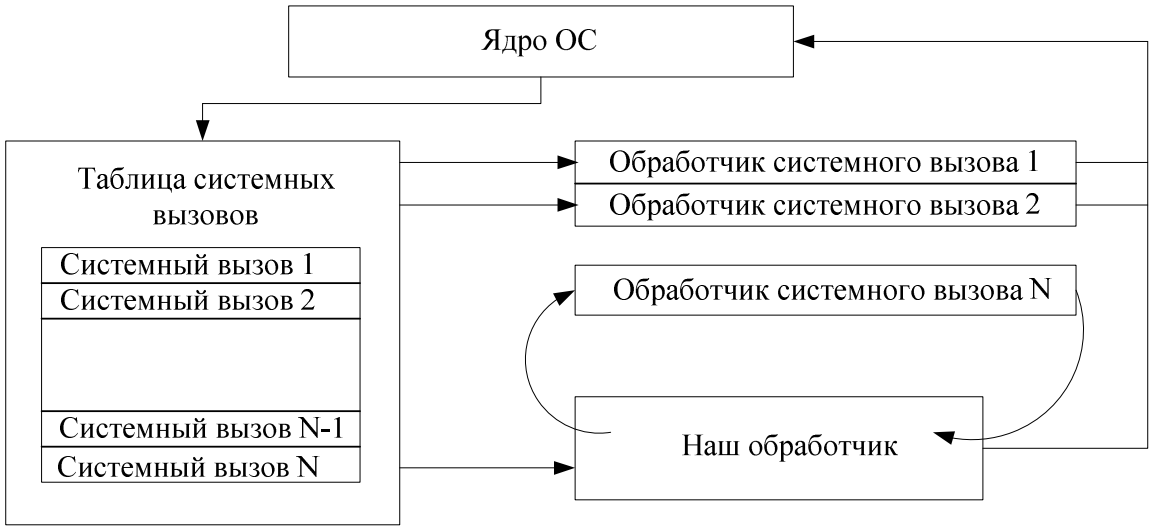


Рис. 1 - Схема перехвата системных вызовов операционной системы

Возможности регистрации событий безопасности реализованы в сетевых операционных системах (ОС) и прикладном программном обеспечении (ПО). Однако, ощутимый эффект от использования средств аудита достигается лишь тогда, когда зарегистрированные данные о событиях безопасности могут быть оперативно проанализированы. Только в этом случае становится возможным своевременное обнаружение вредоносных воздействий на элементы информационной системы – компьютеры, программное обеспечение, передаваемые и хранящиеся данные и т.д. Наличие подсистем регистрации событий безопасности является одним из основных требований, которое присутствует во всех совре-

менных стандартах и руководящих документах по информационной безопасности компьютерных систем[1].

Одним из наиболее важных этапов в системе мониторинга событий безопасности является этап первоначальной обработки данных из журналов аудита.

Постановка задачи

Целью статьи является упрощение процесса анализа событий безопасности посредством введения нового формата представления данных о событиях безопасности, а также уменьшение времени поиска нужного типа событий среди всего журнала аудита путем разделения событий на классы по уровню опасности.

Анализ данных

Анализ форматов и структур представления данных о событиях безопасности, принятых в существующих системах аудита позволяет определить и выделить основные поля данных, характеризующие события безопасности, и разработать универсальный формат для внутреннего представления событий безопасности в системе.

В ОС семейства Linux каждая строка текстового файла журнала аудита является отдельной записью о событии[2]. Значения отдельных полей разделяются пробелом. Каждая строка – запись о событии обычно содержит следующую информацию:

- дату и время регистрации события;
- символьный идентификатор модуля, зарегистрировавшего событие;
- подробное описание события.

События могут регистрироваться различными модулями. Например, идентификатор модуля «$AUDIT» обозначает, что событие зарегистрировано основным модулем аудита. Для записей о событиях, генерируемых этим модулем характерна структура подробного описания события, состоящая из следующих полей:

- идентификатор типа события;
- идентификатор пользователя (UID), в сеансе которого произошло событие;
- идентификатор группы пользователя (GID), в сеансе которого произошло событие;
- информация об операции (функции) и ее параметрах.

В ОС семейства Microsoft Windows NT для доступа к данным аудита необходимо использовать специальные функции, которые обеспечивают считывание данных аудита в буферы в оперативной памяти, формат которых хорошо документирован[2].

Запись о событии аудита в ОС семейства Microsoft Windows NT содержит следующую информацию[3]:

- порядковый номер события;
- время регистрации и время записи события (включая дату);
- идентификаторы типа события;
- категория аудита, к которой относится событие;
- идентификатор безопасности субъекта, в сеансе которого произошло событие;
- дополнительные параметры события, зависящие от типа события. Данные хранятся и представляются в двоичном виде.

С учетом рассмотренных структур данных, описывающих события безопасности **в** существующих распространенных системах аудита, можно разработать некоторый универсальный формат для представления событий аудита в системе, который бы способствовал ускорению обработки данных событий. При этом новый формат должен полно описывать событие.

Структура нашего представления событий безопасности в системе должна содержать следующие поля данных:

- N – порядковый номер события. Тип значения – целое число.
- RegistrationTime – время регистрации события. Тип значения – дата.
- StartTime – время начала обработки. Тип значения – дата.
- EventID – идентификатор разновидности события. Тип значения – целое число.
- EventType – идентификатор типа события. Тип значения – целое число.
- EventCategory – идентификатор категории (группы разновидностей событий), к которой относится событие. Тип значения – целое число.
- EventClasss – класс события – промежуточный результат обработки события в системе, класс события определяется системой на основе анализа данной структуры, описывающей событие. Тип значения - целое число.

- SourceName – имя источника события или модуля, зарегистрировавшего событие. Тип значения – строка символов.
- UserName – имя пользователя в сеансе которого зарегистрировано событие. Тип значения – строка символов.
- Domain – имя домена пользователя. Тип значения – строка символов.
- Host – идентификатор (имя или сетевой адрес) компьютера, на котором зарегистрировано событие. Тип значения – строка символов.
- EventParameters – дополнительные параметры события, зависящие от разновидности события. Тип значения - строка символов.

Такая структура позволяет осуществить представление данных распространенных систем аудита в формализованном виде, удобном для дальнейшей обработки и автоматического анализа.

Рассмотрим этап обработки (алгоритм), когда данные из распространенных систем аудита переводятся в новый вид, на основе которого будет проводиться анализ.

Предположим, что данные размещены в буфере.

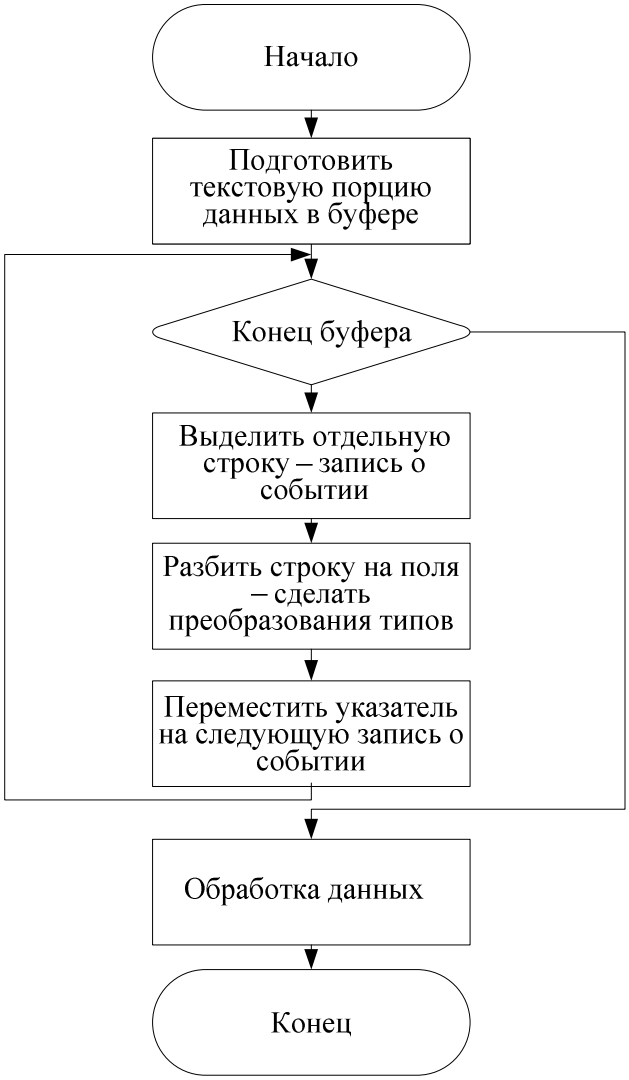


Рис. 2 – Алгоритм преобразования данных в новый вид

Проанализируем данные, которые мы извлекли из журнала аудита. Разделим все события на классы в зависимости от важности. То есть на основании полей записи будем относить события к какому-нибудь классу.

Каждое событие должно подвергнуться проверке на соответствие множествам правил, задающих классы событий. После проверки (при реализации – некоторая функция) поле EventClass получит значение, которое будет показывать принадлежность события к некоторому классу. Причем возможен случай, когда событие принадлежит сразу к нескольким классам.

Выделим следующие классы:

1. Архивные – это все события безопас-

ности, регистрируемые на компьютерах сети.

2. Отказы – это события, связанные с отказами в предоставлении доступа, а также со сбоями в работе средств обеспечения безопасности.

3. Информативные – это события, которые целесообразно отфильтровывать от всех архивных событий, поскольку они содержат важную информацию о важных происшествиях. Наличие этого класса объясняется тем, что обычно журналы аудита содержат большое количество малоинформативных событий, присутствие которых затрудняет дальнейший анализ, поэтому целесообразно отделять информативные события и, возможно, хранить их отдельно от архивных.

4. Предостережения – это класс важных событий, сигнализирующих о возникновении опасных ситуаций. Администраторы безопасности обязательно должны быть проинформированы о событиях этого класса, однако эти события не требуют немедленного оповещения.

5. Тревоги – это класс особо важных событий, сигнализирующих о возникновении особо опасных ситуаций, в случае обнаружения которых необходимо немедленно оповестить администратора безопасности.

Рассмотрим признаки событий, по которым можно добавлять к некоторому классу.

1. Архивные события

Причисление события к классу архивных событий не требует никаких условий. Любое событие, зарегистрированное на любом компьютере сети, может быть отнесено к классу архивных событий.

Таким образом, функция определения для класса архивных событий не зависит от полей данных записи о событии.

2. События-отказы

В среде ОС семейства Microsoft Windows NT события, содержащие информацию об отказах в доступе к ресурсам, могут быть однозначно идентифицированы по значению поля EventType записи о событии. Если поле EventType имеет значение EVENTLOG_AUDIT_FAILURE, то это означает, что событие сообщает о попытке доступа, закончившейся отказом в доступе.

Таким образом, функция определения для класса событий-отказов зависит от значения одного поля данных записи о событии.

3. Информативные события

В среде ОС семейства Microsoft Windows NT информативные события могут быть отделены от малоинформативных событий путем проверки идентификатора разновидности события, то есть поля EventID.

При настройке сетевой системы мониторинга событий безопасности администратор задает список идентификаторов информативных событий или список идентификаторов малоинформативных событий. Во втором случае, к информативным событиям относятся все события за исключением тех, идентификаторы которых перечислены в списке. Способ задания идентификаторов информативных событий не играет роли, поскольку метод их использования в процессе анализа не зависит от способа их задания при настройке системы.

В среде ОС семейства Microsoft Windows NT идентификаторы событий безопасности принимают значения от 512 до 1023. Другие диапазоны значений идентификаторов отведены для событий не связанных с безопасностью. Это позволяет применить быстрый метод определения принадлежности события к классу информативных событий.

То есть, для того чтобы определить относится ли некоторое событие к классу информативных событий достаточно проверить идентификатор разновидности события (поле EventID структуры TEventLogR.ec).

4. **События-предостережения и события-тревоги**

Для задания признаков событий, которые необходимо отнести к классам событий-предостережений или событий-тревог необходимо использовать сложные правила, способные описать ограничения, налагаемые на большинство полей структуры-события (структуры TEventLogR.ee).

Суть общего метода определения принадлежности события к классам событий-предостережений и событий тревог можно проиллюстрировать так:

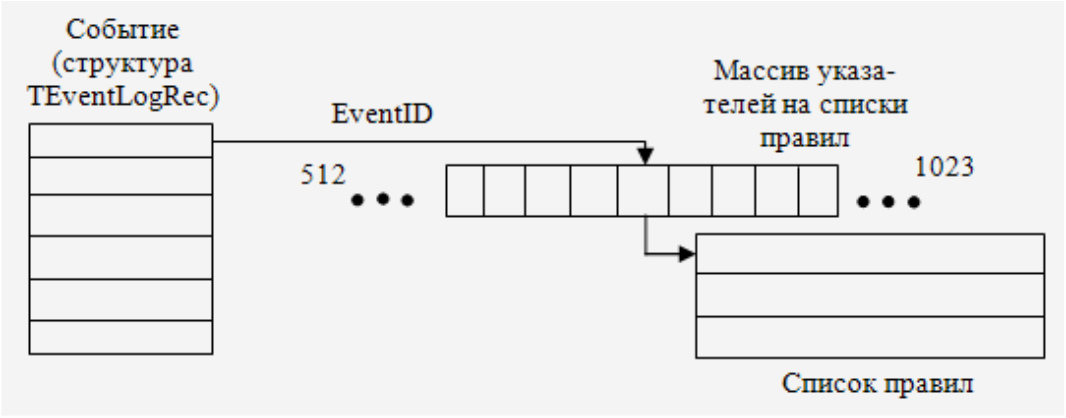


Рис.3 - Метод определения принадлежности события к классам событий-предостережений или событий-тревог

Все правила, определяющие признаки событий предостережений и событий-тревог, группируются в списки правил по идентификатору разновидности события. Таким образом, значение поля идентификатора разновидности события EvID в структуре записи о событии TEventLogRec является первичным признаком поиска, позволяющим существенно сократить количество правил, которые необходимо просмотреть для дальнейшего анализа события. По идентификатору EvID определяется элемент массива указателей – указатель на список правил, задающих крите-рии для других полей записи о событии.

Анализ событий на принадлежность к классам событий-предостережений или событий-тревог целесообразно проводить одновременно, поскольку эти классы отличаются друг от друга лишь требуемой реакцией системы на события указанных классов. Способ описания правил и метод их использования одинаков для обоих классов. Поэтому правила, задающие критерии принадлежности событий к указанным классам целесообразно хранить в одних и тех же списках, идентифицируя правила разных классов по некоторому признаку, включенному в структуру правила.

Для анализа события на принадлежность к классам событий-предостережений или событий-тревог последовательно перебираются правила, находящиеся в списке правил для заданного значения EvID. В этих правилах задаются условия, налагаемые на следующие поля записи о событии:

- GTime – дата и время регистрации события;
- EvType – тип события;
- SourceModule – модуль, зарегистрировавший событие;
- Host – имя компьютера, на котором произошло событие;
- UserName – имя пользователя;
- Domain – домен пользователя;

EvParams – дополнительные параметры события.

Выводы

Поскольку все события в системе имеют разные степени опасности, то нет необходимости в просмотре и анализе малоинформативных событий. Предложенный формат можно использовать в целях упрощения анализа базы событий в системе. В данной классификации класс событий-предостережений и событий-тревог сигнализируют о возникновении опасных и особо опасных ситуаций. Эти классы и вызывают особый интерес. Таким образом, внедрение предлагаемого формата представления данных сократит время, необходимое для обнаружения конфликтных ситуаций в системе и определения степени опасности.

Список литературы:

1. Люцарев В.С., Ермаков К.В., Рудный Е.Б., Ермаков И.В. Безопасность компьютерных сетей на основе Windows Ш\-М.:Изд.отдел «Русская редакция», 1998.-304с.
2. Брагг Р. Система безопасности Windows 2000. пер. с англ., - М: Издательский дом «Вильяме», 2001. - 592 с.
3. Немет Э., Снайдср Г., Сибс С, Хейн Т. UNIX: руководство системного администратора. Для профессионалов, пер. с англ., - С-Пб: Питер, К: Издательская группа BHV, 2002. - 928 с.
4. Фролов А. В., Фролов Г. В. Программирование для Windows NT. - М.: ДИАЛОГ-МИФИ, 1996.

Поступила в редакцию 23.12.2009

УДК 681.301

ЗАЙЦЕВ В.Г.,
ЛАН ЧУНЬЛИНЬ

ПРОЕКТИРОВАНИЕ И ВНЕДРЕНИЕ СИСТЕМЫ АВТОМАТИЧЕСКОЙ КЛАССИФИКАЦИИ ДОКУМЕНТОВ ДЛЯ КОНЕЧНОГО МНОЖЕСТВА ЯЗЫКОВ

В статье исследуется практические аспекты создания автоматической системы распознавания и классификация документов. Определяются основные методы автоматического распознавания, кодировки и классификации документов для конечного множества языков. Описывается предлагаемая архитектура автоматической системы классификации текстов для конечного множества языков.

The paper investigates the practical aspects of creating an automated system for automatic language recognition and classification of documents. The basic methods for automatic language identification and classification of documents on proposed. Describes the proposed architecture of the automatic text classification system for multilingual environment.

Введение

В современном мире, действует целый ряд систем классификации больших объемов текстовой информации, в основе которых лежат технологии компьютерной лингвистики и алгоритмов распознавания образов. В настоящее время существует большое количество текстовых документов на различных языках. Сегодня в Евросоюзе(ЕС) говорят на 23 официальных языках. Официальные языки ЕС – это английский, голландский, греческий, датский, испанский, итальянский, немецкий, португальский, финский, французский и шведский. Официальные тексты должны публиковаться на всех упомянутых языках. Все решения, принимаемые официальными органами ЕС, переводятся на все официальные языки, и граждане ЕС вправе обращаться в органы ЕС и получать ответ на свои запросы на любом из официальных языков. Официальными языками Организации Объединённых Наций(ООН) являются: английский, арабский, испанский, китайский, русский и французский. Обычные системы автоматической классификации текстов решают задачу автоматической классификации текстов на одном языке. Таким образом, необходимо решить задачу автоматической классификации текстов как для одного языка, так и для задач автоматической классификации текстов на разных языках.

Создание автоматической системы классификации текстов для конечного множества языков, можно разделить на две подсистемы. Первая подсистема служит для автоматического распознавания языка(Language Recognition) и идентификации(Language Identification). Вторая из подсистем – для автоматической классификации текстов, из первый подсистемы.

Первая подсистема: автоматическое распознавание и кодировка языка

Подсистема для автоматического распознавания языка состоит из двух частей. Блок автоматической идентификация зыка, извлечение текста из файла произвольного формата, определение кодировки текста. Блок автоматического распознавания языка, разделяет текст на абзацы, предложения, слова и определяет язык. Определение языка документа основано на принципе максимального распознавания слов каждого из языков. Алгоритм автоматической идентификации предложений языка использует тексты в кодировке UTF-8 или UTF-16 (двухбайтовые символы).

Подсистема автоматического распознавания распознает язык документа. Сейчас распознается больше 50 различных комбинаций кодировки и языка. Распознавание реализовано с использованием так называемой "N-Gram-Based Text Categorization" технологии. В статье[1] был предложен метод для определения языка и кодировки документа по его содержимому на основании статистик документов, для которых язык и кодировка известны заранее. Метод определяется частотой N-грамм(подстрок или сочетаний символов, длиной не более N) предполагает, что примерно 300 самых часто используемых N-грамм сильно зависят от языка. Алгоритм включается в нахождении частот N-грамм для всех тестовых документов, для которых

известен язык, а также для каждого документа, язык которого нуждается в определении. После этого среди всех тестовых документов находим тот, для которого расстояние от его N-граммной статистики до статистики тестируемого документа минимально. После того языком тестируемого документа считается язык найденного тестового документа. Расстояние между статистиками считается следующим образом: все N-граммы сортируются в порядке убывания частоты их появления, затем для каждой N-граммы вычисляется разница её позиций в отсортированном списке N-грамм тестового и тестируемого документов. Расстояние между статистиками определяется как сумма разностей позиций каждой N-граммы:

$$l = \sum_{i=1}^{300} \left| P_i - \tilde{P}_i \right| \qquad (1)$$

Где $P_i, \tilde{P}_i$ – позиции i-й N-граммы в тестовом и тестируемом документах соответственно. Значение N предлагается использовать равным 5(включительно). [2]

Распознавание работает хорошо на текстах в 500 байт и длиннее. Более короткие тексты могут распознаваться хуже.

Вторая подсистема: автоматическая классификация текстов

Классификация документов – одна из задач информатики, заключающаяся в отнесении документа к одной из нескольких категорий на основании содержания документа. Использует методы информационного поиска и машинного обучения. Следует отличать классификацию текстов от кластеризации, в последнем случае тексты также группируются по некоторым критериям, но заранее заданные категории отсутствуют.[3]

Постановка задачи классификации

Определим задачу формально. Пусть задано некоторое конечное множество категорий $C = \{c_1, ..., c_{|c|}\}$, конечное множество документов $D = \{d_1, ..., d_{|D|}\}$ и некоторая вначале неизвестная целевая функция *Ф*, которая для каждой пары <документ, категория> определяет, соответствуют ли они друг другу: $\Phi: D \times C \rightarrow \{0,1\}$. Задача состоит в том, чтобы найти максимально близкую к функции *Ф* функцию *Ф′*. Функцию *Ф′* называют классификатором. Машинное обучение основывается на начальном множестве документов $\Omega = \{d_1, ..., d_{|\Omega|}\} \subset D$. При этом, значение целевой функции *Ф* известно для каждой пары $< d_j, c_i > \in \Omega \times C$. Документы из *Ω* разделяют на два непересекающиеся множества:

"Учебное" множество $T_r = \{d_1, ..., d_{|T_r|}\}$ и множество документов, с помощью которой создается классификатор *Ф′*. *Ф′* обучается индуктивно, основываясь на замеченных характеристиках этих документов.

"Тестовое" множество $T_e = \{d_{|T_r+1|}, ..., d_{|\Omega|}\}$ и множество документов, на котором тестируется эффективность построенного классификатора. Каждый "тестовый" документ подается на вход классификатора *Ф′*, а затем сравнивается результат классификатора $\Phi'(d_j, c_i)$ с известным значением функции $\Phi(d_j, c_i)$. Классификатор считается тем эффективнее, чем чаще эти значения совпадают.

Документ $d \in \Omega$, называется положительным или отрицательным примером для категории *c*, если значение функции *Ф(d,c)* равно *1* или *0*, соответственно. [4]

Первичная обработка документов

Выбор веса признаков и уменьшение размерности.

В статье [5] приводится подробное исследование различных подходов к выбору весов признаков. Результаты экспериментов, описанных в этой статье, показывают, что одной из лучших формул вычисления весов является:

$$W_{ij} = TF_{ij} * IDF_i \qquad (2)$$

Где *TF* – частота термина в документе, *IDF* – обратная частота документа. Каждый документ – это просто набор слов (термов). Множество всех термов обозначим как *T*. Каждый терм $t_i \in T$ имеет вес w_{ij} по отношению к документу $d_j \in D$. Таким образом, каждый документ можно представить в виде вектора весов его термов $\overrightarrow{d_j} = < w_{ij}, ..., w_{|T|j} >$. Веса документов нормируют так, чтобы $0 \le w_{ij} \le 1$ для $\forall i, j : 0 \le i \le |T|, 0 \le j \le |D|$. Здесь TF_{ij} - это отношение числа термов t_i в документе d_i к общему числу термов в этом документе, а IDF_i – число, обратное количе-

ству документов, в котором встречается терм.

Нормализовать вес терма в документе можно следующим стандартным способом:

$$w_{ij} = \frac{TF_{ij} * IDF_i}{\sqrt{\sum_{s=1}^{|T|}(TF_{sj} * IDF_s)^2}} \quad (3)$$

После приведения всех слов документа к нормализованной форме, полученное пространство признаков имеет очень большую размерность. Эту размерность можно существенно уменьшить без ухудшения качества классификации, если исключить слова, слабо влияющие на результаты. [6]

Обычно из списка признаков удаляют так называемые "стоп-слова"(stopword) и из списка признаков можно удалить слишком редко встречающиеся слова.

Кроме того часто применяется методы выделения слов с использованием критерия информационного веса слова в категории (mutual information gain). Информационный вес слова в категории определяется по формуле:

$$MI(x_i, c) = \sum_{x_i \in \{0,1\}} \sum_{c \in \{0,1\}} P(x_i, c) \log \frac{P(x_i, c}{P(x_i)P(} \quad (4)$$

Здесь

$$P(x_i = 1) = 1 - P(x_i = 0) =$$

$$= \frac{\begin{array}{c}\textit{количество документов,}\\ \textit{содержащих слово } x_i\end{array}}{\textit{количество всех документов}},$$

$$P(c = 1) = 1 - P(c = 0) =$$

$$= \frac{\begin{array}{c}\textit{количество документов,}\\ \textit{принадлежащих категории } c\end{array}}{\textit{количество всех документов}}$$

$P(x_i, c)$ – вероятности совместного распределения слов и категории. Если предположить, что распределения слова x_i и категории c статистически независимы, то легко видеть $MI(x_i, c) = 0$. Если же между встречаемостью слова x_i и категории c имеется строгая логическая зависимость, то $MI(x_i, c)$ – максимально. Метод сокращения размерности на основе выделения наиболее информационно-значимых слов применяется, например, в работе [7].

Методы машинной классификации

Существует несколько наиболее известных методов классификации:

1. Метод Байеса.

Метод Байеса основан на анализе совместных распределений признаков документа и категорий [8]. Документу $D = < d_1, d_2, ..., d_n >$ сопоставляется наиболее вероятная апостериорная категория, определяемая по формуле:

$$c^* = \arg\max_{c \in C} P(c \mid x_1 = d_1, x_2 = d_2, ..., x_n = c \quad (5)$$

Апостериорная вероятность принадлежности документа некоторой категории вычисляется по формуле Байеса, связывающей априорную вероятность с апостериорной:

$$P(c \mid x_1 = d_1, x_2 = d_2, ..., x_n = d_n) =$$

$$= \frac{P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n \mid c) \cdot P(c)}{P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n)} \quad (6)$$

Подставляя (6) в (5), получаем:

$$c^* =$$

$$= \arg\max_{c \in C} \frac{P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n \mid c) \cdot P(c)}{P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n)} \quad (7)$$

Так как знаменатель не зависит от категории, его можно исключить:

$$c^* = \arg\max_{c \in C} P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n \mid \cdot P(c) \quad (8)$$

Условные вероятности $P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n \mid c)$ можно вычислить в предположении условной независимости переменных $x_1, x_2, ..., x_n$. В этом случае, формула для определения наиболее вероятной категории будет выглядеть следующим образом:

$$c^* = \arg\max_{c \in C} P(c) \cdot \prod_{i=1..n} P(x_i = d_i \mid c) \quad (9)$$

Для множества обучающих документов вероятности $P(x_i = d_i \mid c)$ вычисляются по формуле [9,10,11]:

$$P(x_i = d_i \mid c) = \\ = \frac{|\{D \in Ex \mid c \in Rub(D) \wedge D_i = d_i\}| + 1}{|\{D \in Ex \mid c \in Rub(D)\}|} \quad (10)$$

2. Метод опорных векторов SVM (Support Vector Machines).

Метод опорных векторов [12,13,14] разработан на основе принципа структурной минимизации риска – одновременного контроля количества ошибок классификации на множестве для обучения и «степени обобщения» обнаруженных зависимостей.

Нахождение оптимальной плоскости разделения множеств методом SVM сводится к решению оптимизационной задачи с линейными ограничениями типа равенств и неравенств [12]:

$$L_D(\alpha) = \\ = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \to \max, \\ 0 \le \alpha_i \le C, \sum_{i=1}^{l} \alpha_i y_i = 0 \quad (11)$$

Здесь $K(x_i, x_j)$ – функция ядра SVM, которая в простейшем случае равна евклидову скалярному произведению векторов x_i и x_j. Для решения задачи (11) предложены эффективные методы решения [15,16].

Оценка качества классификации

С целью определения наилучшего метода были предложены следующие критерии: полнота; точность. Для оценки качества классификации используются метрики из информационного поиска:

Полнота (recall): отношение количества найденных документов из категории к общему количеству документов категории.

Точность (precision): Определяется как отношение числа релевантных документов, найденных ИПС, к общему числу документов.

$$recall = \frac{|D_{rel} \cap D_{retr}|}{|D_{rel}|} \quad (12)$$

$$precision = \frac{|D_{rel} \cap D_{retr}|}{|D_{retr}|} \quad (13)$$

где D_{rel} – это множество релевантных документов в базе, а D_{retr} – множество документов, найденных системой.[17]

В статье [7,8,10,11] сравниваются различные методы машинного обучения. Результаты показали, что метод SVM имеет преимущество перед другими методами машинного обучения.

Результаты работы

Программа распознавания языка и классификации текстов написана на Java. Система распознает следующие языки: русский, китайский, украинский, английский, немецкий, французский. В данной работе для классификации использован SVM метод. Было протестировано три группы из 5000 документов. В качестве результатов средняя полнота: 86.3%. средняя точность: 89.2%. Основными недостатками описанного алгоритма являются: неустойчивость определения языка малых текстовых документов – отдельные предложения на похожих языках распознаются неуверенно и с ошибками. Кроме того, системы автоматической классификации текстов показывают лучшие временные результаты на 64-битной аппаратной платформе(ширина регистров современных процессоров), чем на 32-битной аппаратной платформе, которая работает медленней.

Выводы

Выбранный метод и результаты работы подтверждают возможность создания эффективной системы автоматической классификации документов для конечного множества языков по критерию принадлежности к определенной области знания, используя современные средства вычислительной техники.

Список литературы

1. Cavnar, W. B. and J. M. Trenkle, “N-Gram-Based Text Categorization” In Proceedings of Third Annual Symposiumon Document Analysis and Information Retrieval,Las Vegas, NV, UNLV Publications/Reprographics, pp. 161-175, 11-13 April 1994

2. Сотник С.Л. Идентификация языка UNICODE-текста по N-граммам длиной до 4-х включительно (квадрограммам). Журнал “Математичне моделювання” №1,2(15) 2006, стр. 111-112, Днепродзержинск, издательство ДГТУ.
3. http://ru.wikipedia.org/wiki/Классификация_документов
4. Юрий Лифшиц. Курс "Алгоритмы для Интернета" РАН 2006.
5. Salton G, Buckley C. Term-Weighting Approaches in Automatic Text Retrieval. // Information Processing and Management, —1988 — p. 513-523.
6. Yang Y., Pedersen J. A comparative study on feature selection in text categorization. // In: Proc. of ICML-97, 14th International Conf. On machine Learning — Nashville, USA, 1997. — p. 412-420.
7. Dumais S., Platt J,, Heckerman D., Sahami M. Inductive learning algorithms and representations for text categorization. // In Proc. Int. Conf. on Inform. and Knowledge Manage., 1998. — p. 2-6.
8. Yang Y., Liu X. A re-examination of text categorization methods. // Proc. of Int. ACM Conference on Research and Development in Information Retrieval (SIGIR-99), 1999 — p. 42-49.
9. Joachims T. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. // Proceedings of ICML-97, 14th International Conference on Machine Learning. — 1996. — p. 2-13.
10. Yang Y. An Evaluation of Statistical Approaches to Text Categorization. / Journal of Information Retrieval, 1999 — V.1 — p. 67--88.
11. Joachims T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. // Proceedings of ECML-98, 10th European Conference on Machine Learning — 1998. — p. 6-18.
12. Vapnik V. The Nature of Statistical Learning Theory. — Springer-Verlag — New York, 1995. — p. 123-167.
13. Burges C.J.C. A tutorial on support vector machines for pattern recognition. // Data Mining and Knowledge Discovery, 1998. — p. 955-974,
14. Вапник В.Н. Восстановление зависимостей по эмпирическим данным. — М.: Наука, 1979. —с. 223-255.
15. Joachims T. Making Large-Scale SVM Learning Practical. Advances in Kernel Methods // Support Vector Learning, Burges C., Smola A. (ed.), — MIT-Press, 1999. —p. 5-12.
16. Joachims T. Estimating the Generalization Performance of a SVM Efficiently. // Proceedings of the International Conference on Machine Learning, — Morgan Kaufman, 2000. —p. 5-22.
17. http://ru.wikipedia.org/wiki/Информационный_поиск

Поступила в редакцию 16.12.2009