ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

2(150) 2009

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с ноября 1995 г.

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Информация о журнале доступна по сети Internet по адресу http://www.**informika.ru/text/magaz/it/** или **http://novtex.ru/IT.**

Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора наук.

УДК 004.231.3; 004.382.6; 004.382.7; 004.384

А. К. Ким, канд. техн. наук, ген. директор,
В. М. Фельдман, канд. техн. наук,
ст. науч. сотр., нач. отделения,
ОАО ИНЭУМ
E-mail: kim@mcst.ru, feld@mcst.ru

Вычислительная система реального времени для реализации программ управления сложными объектами

Рассмотрены методы построения вычислительных систем, включенных в звено управления сложными объектами обороны. Описаны структура вычислительной системы, подходы в проектировании операционной системы реального времени. Приведен пример расчета надежности предлагаемой вычислительной системы.

Ключевые слова: *вычислительная система, микропроцессор, компилятор, операционная система реального времени, надежность.*

Введение

Задача создания вычислительной системы реального времени, включенной в звено управления сложными объектами обороны (как стационарными, так и подвижными, в том числе высокоскоростными) выглядит следующим образом. Система должна иметь высокую производительность, быстро и детерминировано реагировать на внешние воздействия и быть очень надежной, чтобы время ее простоя в случае прекращения вычислений было мало по сравнению со временем принятия решения по управлению объектом. Кроме того, если новая система приходит на замену ранее существовавшей, необходимо максимально сократить затраты на ее внедрение.

Из сделанного определения, которое подходит, прежде всего, для систем управления сложными объектами обороны, а также объектами атомной промышленности, транспорта, химической промышленности и т. п., следует ряд базовых принципов построения требуемой вычислительной системы.

1. Высокая производительность, обусловленная:
- сложностью алгоритмов для реализации возможностей системы;
- высоким темпом запуска алгоритмов для обеспечения высокого качества решения функциональных задач и соответственно высокой эффективности системы управления в целом;
- многообразием условий работы системы;
- необходимостью отмены ряда ограничений на работу алгоритмов, введенных при отработке функционального программного обеспечения (ФПО) для выполнения их реализуемости в реальном масштабе времени, в том числе при развитии и совершенствовании системы;
- необходимостью введения новых функциональных задач, расширяющих возможности системы при ее развитии и совершенствовании.

2. Высокая реактивность системы при обработке прерываний (управлении вычислительным процессом). Она обеспечивается характеристиками аппаратных средств и операционной системы.

3. Преемственность по связям с большим числом разных по характеристикам внешних абонентов. В частности, должен быть обеспечен:
- двусторонний обмен информацией по стандартным каналам;
- прием внешних сигналов прерывания;
- связь с системой единого времени.

4. Надежная, бесперебойная работа с возможностью обнаружения отказов аппаратуры и программного обеспечения, их обработки, локализации и восстановления вычислительного процесса.

Названные выше принципы разработки могут быть обеспечены только комплексом технических решений, реализованных в аппаратном обеспечении, операционной системе и компиляторе.

1. Эффективная реализация решения сложных задач управления

В качестве основы вычислительной системы, эффективно решающей задачи управления сложными объектами, предлагается высокопроизводительный отечественный комплекс "Эльбрус-3М1", построенный на базе микропроцессора "Эльбрус" с архитектурой широкого командного слова [1].

Архитектура широкого командного слова, благодаря параллельной работе аппаратуры, глубокому анализу программы и тщательному планированию ее выполнения транслятором на стадии компиляции, позволяет значительно повысить скорость счета как скалярных, так и векторных задач.

Для скалярных вычислений основным фактором, ограничивающим скорость счета таких за-

дач, является информационная зависимость между различными операциями.

Для векторных вычислений, как правило, ограничивающим фактором является не столько информационная зависимость, сколько общий объем вычислений и недостаток вычислительных ресурсов (в основном исполнительных устройств).

Спекулятивный и условный режимы выполнения команд, а также выполнение передач управления с использованием предварительной подкачки кода в направлении передачи управления способствуют ускорению выполнения скалярного кода.

Спекулятивный режим выполнения операций позволяет компилятору перемещать выше по коду отдельные операции и фрагменты программы и выполнять их параллельно на фоне текущих вычислений, что сокращает общее время выполнения программы. Спекулятивный режим выполнения арифметических операций позволяет проводить параллельно вычисления сразу по нескольким альтернативным ветвям программы на фоне вычисления условия ветвления.

Условный режим выполнения операций часто заменяет условную передачу управления. Условный режим выполнения операций позволяет управлять выполнением операции по заданному и ранее вычисленному условию. В частности, так может быть выбрана для продолжения вычислений одна из спекулятивно выполненных ветвей программы.

Предварительная подкачка кода в направлении ветвления (на фоне выполнения основной ветви) скрывает задержку по доступу к коду при передачах управления и, тем самым, позволяет выполнить передачу управления без остановки конвейера выполнения, когда уже известно условие ветвления.

Файл предикатов дает возможность параллельного вычисления нескольких условий. Это важно как для скалярных вычислений, обеспечивая условное выполнение операций, так и для работы в циклах, содержащих условные ветви.

Выполнение циклов методом программного конвейера является действенным методом ускорения векторных вычислений. Последовательные итерации цикла выполняются с некоторым наложением друг на друга. Шаг, с каким итерации накладываются, определяет темп выполнения итераций, который может быть существенно выше строго последовательного выполнения итераций.

В микропроцессоре "Эльбрус" введен буфер предварительной подкачки, в который данные из массивов вызываются отдельной ветвью программы, выполняемой параллельно на фоне вычислений. Имеются специальные средства управления, позволяющие подстраивать опережение вызова элементов массивов по отношению к их использованию в вычислениях.

Для микропроцессора с архитектурой широкого командного слова компилятор выполняет планирование параллельного выполнения программы и составляет последовательность широких команд, каждая из которых состоит из различных наборов простых команд типа сложения, умножения, деления, логических операций, обращений в память по чтению и записи и пр. Все простые команды, составляющие одну широкую команду, дешифрируются и выдаются на выполнение одновременно.

Параллельная структура микропроцессора "Эльбрус" позволяет в различных сочетаниях дешифрировать в каждом такте:

- до 10 арифметико-логических операций;
- до четырех операций обращений в память (до четырех — по чтению, до двух — по записи);
- до трех операций обработки булевских значений;
- до четырех операций формирования литералов;
- условную передачу управления;
- операцию приращения счетчика цикла;
- до двух групповых операций подкачки элементов массива с продвижением адресов;
- до четырех операций считывания подкаченных элементов массива;
- до шести операций вычислений условий, управляющих условным выполнением команд.

В результате, в одной широкой команде может содержаться в скалярном режиме до 14 и в векторном режиме до 23 простых команд.

На базе микропроцессора "Эльбрус" могут быть построены многопроцессорные и многомашинные вычислительные комплексы. Большая нагрузка по распараллеливанию вычислений как на уровне операций, так и на уровне потоков и задач ложится на оптимизирующий компилятор, являющийся неотъемлемой и очень важной частью архитектуры "Эльбрус" [2].

Объединение в единую вычислительную систему с общей памятью нескольких процессорных единиц открывает собой новые возможности по построению высокопроизводительных параллельных вычислительных комплексов. Характерной особенностью систем этого класса является многоуровневость совокупно развиваемого ими параллелизма, складывающегося из многопоточного параллелизма на микроуровне, и многозадачного параллелизма на макроуровне, в рамках вычислительной системы в целом.

Многопоточный параллелизм позволяет использовать все преимущества "близости" процессоров, "быструю" синхронизацию между потоками и возможность оперативной смены потоков на процессорах. Все это позволяет компилятору эффективно распараллеливать сильно связанные вычисления, разделяя их на "небольшие" потоки вычислений и распределяя потоки между процессорами.

Многозадачный параллелизм позволяет асинхронно исполнять крупные блоки программ на большом объеме оперативной памяти, которая является суммой оперативной памяти всех кристаллов распределенной системы. В итоге, распараллеливание на уровне задач дает возможность не только ускорить выполнение больших задач за счет распараллеливания, но и разместить в оперативной памяти данные, которые не могли быть размещены и обработаны на одном кристалле ввиду их большого объема.

Задача компилятора — проанализировать исходную программу и обнаружить фрагменты программы, которые могут быть исполнены параллельно.

Совершенно очевидно, что современные распараллеливающие компиляторы должны содержать мощные методы для обнаружения и использования параллелизма. Такие компиляторы должны использовать комбинации статических и динамических методов, включать символьную алгебру и анализ. При разработке новых методов необходимо проанализировать реальные программы для того, чтобы оценить эффективность этих новых методов.

2. Организация вычислений в реальном масштабе времени

Важным базовым принципом системы является ее реактивность, т. е. организация вычислений в реальном масштабе времени.

Сформулируем ряд требований к операционной системе реального времени (ОСРВ) [3] рассматриваемой системы.

1. Все средства ОСРВ должны работать с учетом многопроцессорности.

2. ОСРВ должна обеспечивать возможность работы ФПО в режиме жесткого реального времени.

3. ОСРВ должна обеспечивать и общепринятые функции управления:

- выполнение функций загрузчика программных модулей;
- мультипрограммный и многопользовательский режим;
- виртуализацию памяти;
- возможность создания параллельных процессов и синхронизацию для работы с общими данными.

4. ОСРВ должна иметь следующие характеристики:

- малые времена входа в процедуру прерывания и переключения процесса из процедуры прерывания на другой процесс (диапазон от единиц до нескольких десятков микросекунд);
- приоритетное планирование всех процессов, обработчиков внешних прерываний и семафорной синхронизации с возможностью управления приоритетами из ФПО;
- резидентация оперативной памяти;
- назначение одного или нескольких процессоров для исполнения конкретного процесса;
- возможность контроля целостности программного обеспечения ВК на внешних устройствах (дисках);
- регистрация информации на внешних носителях. Даже при отключении питания и последующем включении ВК архив регистрируемой информации должен быть доступен для анализа.

5. Системный интерфейс ОСРВ должен обеспечивать возможность работы в реальном времени со следующими объектами:

- процессами;
- потоками (*thread*) и мультипотоками (расширение POSIX) для исполнения заявок на некоторую деятельность на одном потоке с учетом приоритетов заявок;
- семафорами (*mutex*);
- массивами и областями фиксированного размера, выделяемыми из заранее выделенных пулов памяти;
- файлами и справочниками;
- устройствами (драйверами устройств).

6. Системный интерфейс ОСРВ при организации вычислительного процесса должен обеспечивать:

- динамическое распределение оперативной памяти из заранее выделенного пула памяти квантами фиксированного размера;
- вывод на экран монитора графической информации;
- управление процессами с использованием средств семафорной синхронизации для взаимодействия процессов;
- коррекцию часов ВК;
- управление системами файлов с возможностью их создания, чтения и записи информации;
- управление устройствами с возможностью чтения и записи информации и возможностью разработки и подключения дополнительных специализированных устройств;
- возможность назначения конкретного процессора для работы конкретного процесса;
- возможность установки процедуры реакции (ждущего прерывания процесса) СРВ на конкретное прерывание;
- средства сетевого взаимодействия с использованием классической схемы клиент—сервер;
- возможность построения отказоустойчивых ММК с использованием средств управления устройствами (драйверами) и сетевого взаимодействия;
- возможность "рестарта" ВК (перезапуска ВК и вычислительного процесса);
- возможность установки процедуры реакции (ждущего процесса) ФПО на сбой (выход из строя отдельного ВК в ММК) для организации

восстановления. Под сбоем понимается отсутствие периодичных сигналов от ВК из состава ММК;
- возможность получения информации о состоянии ВК и ММК;
- возможность вывода графической информации на мониторы.

Кроме мощных вычислительных средств для разрабатываемой системы необходим и максимально возможный сервис, поддерживаемый ОСРВ: для работы с различного рода файловыми системами, возможность работы графических библиотек, развитых средств отладки, поддержка возможности построения многомашинных комплексов и т. п.

В обычных ОС все механизмы настраиваются для достижения максимальной средней производительности. В ОСРВ производительность также стоит на первом месте, но еще важнее минимизировать временной интервал между максимальным и минимальным временем исполнения той или иной функции (минимальный выброс) [4]. Например, ключевой характеристикой для систем реального времени является время переключения процессора с одного процесса на другой. Вопрос не только в важности самого переключения, а в том, что переключение является составной частью основной деятельности ОСРВ.

При выборе любого варианта построения ОСРВ разработчикам придется решать задачи по следующим направлениям:
- сокращение возможных задержек при входе в прерывание;
- переключения процессов и синхронизация;
- обработка прерываний;
- планирование обменов;
- планирование процессов;
- управление прерываниями;
- реализация Posix-библиотеки;
- работа графических приложений в режиме реального времени;
- организация работы с единым временем системы;
- организация прямого доступа в память удаленного узла;
- средства отладки для ОСРВ.

3. Структура вычислительного комплекса

Поскольку предлагаемая вычислительная система приходит на смену ранее используемой, она должна быть совместима по внешним связям, иметь набор определенных структурных элементов, таких как вычислительное ядро, рабочие места операторов, объединенных с вычислительным ядром в единую сеть. На это накладывается требование по обеспечению высокой надежности, о чем будет сказано в следующем разделе.

Все перечисленное выше определило выбор структуры. Вычислительная система, представленная на рис. 1, состоит из основного и резервного ядер, восьми рабочих мест оператора АРМ0-АРМ7 и двух АРМ печати 0 и 1. Все это оборудование объединено двумя сегментами локальной вычислительной сети ЛВС1 и ЛВС2 типа Fast Ethernet. Кроме того, оба ядра соединены каналом межкомплексного обмена (КМО).

В состав основного и резервного ядер входит большое число специальных каналов для связи с внешними абонентами, в том числе и наследуемыми от ранее разработанной вычислительной системы. К таким каналам относятся:
- быстрые оптические каналы (МБКБО);
- каналы стандартных каналов (МКПМ);
- каналы Fast Ethernet;
- каналы внешних прерываний.

В качестве основного и резервного ядер используются вычислительные комплексы ВК "Эльбрус-3М1". В состав каждого ВК "Эльбрус-3М1" входят системный блок УВ 3М1, два устройства сопряжения с внешними абонентами УСВА-М0 и УСВА-М1, монитор и клавиатура/мышь. Каждое ядро подключено к обоим сегментам ЛВС через коммутаторы FE. Каждое рабочее место опе-

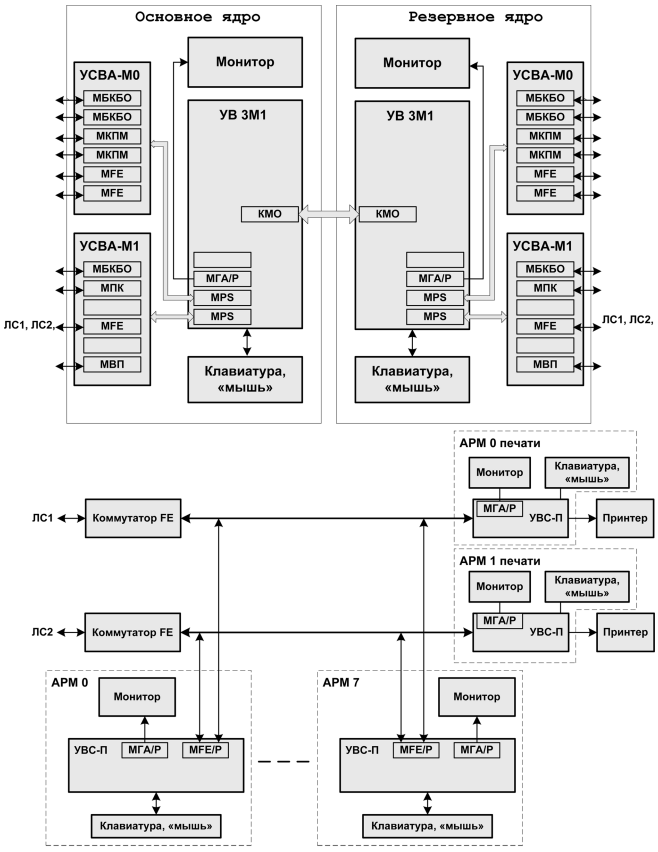


Рис. 1. Структурная схема вычислительной системы

ратора АРМ имеет два канала подключения к обоим сегментам ЛВС. Каждый АРМ печати подключен только к одному сегменту ЛВС. В качестве АРМ используется вычислительный комплекс "Эльбрус-90микро" [5].

Из рассмотрения структурной схемы видно, что основное и резервное ядра соединяются каналами для обмена информацией "контрольных точек" и прерываниями для обеспечения режима "горячего резерва". Варианты построения этих каналов во многом зависят от объема информации, которой будут обмениваться ядра, и от частоты этого обмена.

Технические характеристики ВК "Эльбрус-3М1"

Наименование	Значение
Производительность ВК	9,6 Гфлопс/с
Число микропроцессов/разрядность/архитектура	2 универсальных 64-разрядных микропроцессора с архитектурой широкого командного слова
Тактовая частота	300 МГц
Объем КЭШа команд/КЭШа данных первого уровня	64 Кбайт 64 Кбайт
Объем КЭШа второго уровня	256 Кбайт
Объем оперативной памяти	16 Гбайт
Пропускная способность оперативной памяти	9,6 Гбайт/с
Объем внешней памяти на НЖМД	80 Гбайт
Число слотов шины PCI 32/33 МГц	4
Пропускная способность каналов ввода-вывода: IDE (встроенный) SCSI (на ячейке) Ethernet (на ячейке) USB 1.0 (встроенный) RS-232 (встроенный) IEEE 1284 (встроенный) Мост PCI-Sbus (на ячейке) Быстрый оптический канал БК3 Шина SBus, подключение через УСВА, 6 слотов Ввод/вывод ЕС ЭВМ (канал) Ввод/вывод ЕС ЭВМ (абонент) Быстрый канал 1, дуплекс Быстрый канал 2, дуплекс	 33 Мбайт/с 20 Мбайт/с 10/100 Мбит/с 2 × 1,5 Мбит/с 2 × 115 Кбит/с 2 Мбайт/с 100 Мбайт/с 2 × 4,8 Мбайт/с 100 Мбайт/с 120 Кбайт/с 120 Кбайт/с 3,3 Мбайт/с 3,3 Мбайт/с
Число внешних прерываний	16
Напряжение питающей сети, В/Гц	220±22/50±1
Потребляемая мощность	Не более 740 Вт
Система охлаждения	Встроенная, воздушного типа
Условия эксплуатации: температура (рабочая) температура (хранения) влажность	 От 5 до 40 °С От −60 до 50 °С 98 % при температуре 25 °С

Основные технические характеристики вычислительного комплекса "Эльбрус-3М1" представлены в таблице.

4. Организация надежных вычислений

Для обеспечения требования работы в реальном времени должны быть реализованы программно-аппаратные средства, обеспечивающие режим "горячего" резерва ВК "Эльбрус-3М1" при введении в состав ФПО соответствующей программной компоненты. Переход на горячий резерв должен осуществляться программно аппаратным способом. Время перехода на горячий резерв должно обеспечивать восстановление вычислительного процесса управления средствами системы.

В ВК "Эльбрус-3М1" нет стопроцентного аппаратного контроля. В этих условиях определение, исправна система или нет, возможно, но только на основе исполнения каких-то заранее определенных действий и контроля за их исполнением. Например, основная и резервная машины постоянно обмениваются сообщениями, по каким-то каналам связи. Это может быть Ethernet или быстрые каналы передачи информации (на схеме обозначены КМО) и т. п. Признаком неисправности в этом случае может быть отсутствие сообщения. Время определения неисправности ("появление признака неисправного состояния") может составлять десятки миллисекунд и то, только в том случае, когда такого рода прослушиванием будут заниматься самые приоритетные процессы в основной и резервной машинах. Такой путь неприемлем для построения рассматриваемой системы. Необходимо специальное оборудование, которое ведет прослушивание независимо с минимальной поддержкой со стороны ОС. Основная и резервная машина должны быть связаны через специальные устройства прослушивания. Если в течение определенного времени не поступает сигнал из другой машины (ВК неисправен), то устройство прослушивания вызывает прерывание и активизируется специальный процесс ФПО, что и является сигналом о неисправности основной или резервной машины. Следующее прерывание от устройства прослушивания должно возникнуть при появлении первого сигнала-сообщения после ввода ВК в работу. Такое прослушивание можно организовать без участия ОС. ОС только инициализирует режим прослушивания на стадии инициализации системы.

Дополнительно к предложенной выше схеме прослушивания для определения исправности ВК "Эльбрус-3М1" необходимы и средства для обнаружения зависаний ОС.

Ниже приводится методология расчета показателей надежности вычислительной системы, построенной по схеме с "горячим" резервированием.

Контролируемые показатели:

- коэффициент готовности ВС — $K_{\text{Г\_ВС}}$ (определяется в результате расчета);
- коэффициент безотказной работы — $K_{\text{БР\_ВС}}$ за время безотказной работы $t_{\text{БР}} = 1$ ч (определяется в результате расчета);
- коэффициент планируемого использования $K_{\text{ПИ}} = 1$ (определяется принятой стратегией обслуживания ВС).

Математическая модель надежности ВС. Принятые предположения и допущения математической модели:

- все отказы — события независимые;
- каждое устройство (модуль) может находиться только в одном состоянии — работоспособность или отказ;
- законы распределения средней наработки на отказ и среднего времени восстановления являются экспоненциальными;
- контроль работоспособности — непрерывный, полный;
- резерв активный;
- время коммутации при включении/выключении резервных устройств пренебрежимо мало;
- восстановление и техническое обслуживание ограничены (одна ремонтная бригада);
- на техническое обслуживание одновременно выводится не более одного модуля из состава любого из резервированных узлов структурной схемы надежности и не более чем в одном резервированном узле;
- суммарный объем технического обслуживания по любому из устройств ВС составляет не более 19 ч в год;
- аппаратура эксплуатируется в непрерывном круглосуточном режиме работы.

Структурная схема надежности (ССН) ВС приведена на рис. 2. Она представляет собой последовательное соединение трех дублированных и одного резервированного (8 рабочих, из них один резервный, резерв скользящий) узлов.

Отказ единичного модуля в любом из узлов ССН не приводит к отказу ВС в целом.

Отказ ВС в целом происходит только в случае одновременного отказа не менее двух однотипных модулей ССН в составе какого-либо резервированного узла ССН.

Расчетные соотношения для показателей надежности резервированных узлов получены из предположения о том, что процессы деградации и восстановления ВС описываются марковскими процессами [6].

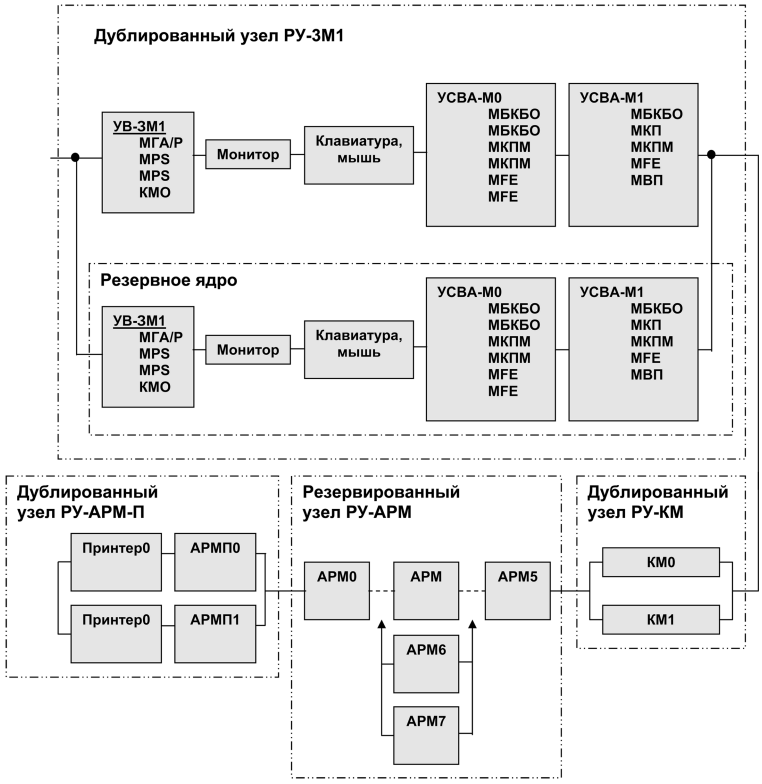


Рис. 2. Структурная схема надежности вычислительной системы

При полноформатной работе узла (все модули ССН узла в рабочем состоянии) в случае дублированных и резервированного узлов эти процессы представляются графами, изображенными на рис. 3 и 4.

Решение системы дифференциальных уравнений относительно вероятностей состояний узла позволяет получить все контролируемые показатели, приведенные выше [6]. В частности, для коэффициента готовности $K_{\text{г}}$ резервированного узла, состоящего из n рабочих и m резервных модулей:

$$K_{\text{г}} = 1 - \frac{\sum_{i=0}^{n-m-1} \frac{\gamma^i}{i!}}{\sum_{i=0}^{n} \frac{i^i}{i!}},$$

где $\gamma = \frac{T_{\text{о}}}{T_{\text{в}}}$, $T_{\text{о}}$ — средняя наработка на отказ модуля ССН в резервированном узле, $T_{\text{в}}$ — среднее время восстановления неисправности.

Рис. 3. Граф переходов для дублированного узла:
а — для узла в полном составе; λ — интенсивность отказов узла; μ — интенсивность восстановления узла; *б* — для узла в состоянии технического обслуживания

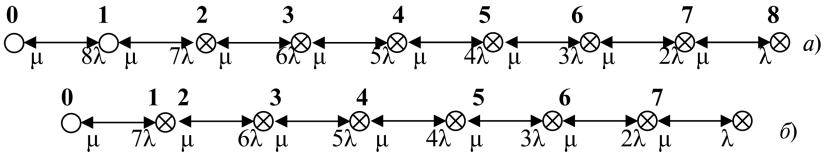


Рис. 4. Граф переходов для резервированного (8/1) узла:
а — для узла в полном составе; *б* — для узла в состоянии технического обслуживания

Для $n = 8$, $m = 1$ и средней наработке на отказ, много большей среднего времени восстановления, $\gamma = \dfrac{T_{\text{о}}}{T_{\text{в}}} \gg 1$, получим

$$K_{\text{г(8/1)}} = 1 - \left[\left(1 - \frac{T_{\text{пф}}}{8760}\right) \times \left(1 - \frac{8 * 7}{\gamma^2}\right) + \left(\frac{T_{\text{пф}}}{8760}\right) \times \frac{7}{\gamma}\right],$$

где $T_{\text{пф}}$ — время профилактического обслуживания.

$K_{\text{г}}$ для ВС определяется произведением $K_{\text{г}}$ резервированных узлов

$$K_{\text{г(ВС)}} = K_{\text{г(РУ – 3М1)}} \times K_{\text{г(РУ – КМ)}} \times \times K_{\text{г(РУ – АРМ)}} \times K_{\text{г(РУ – АРМ – П)}}.$$

Средняя наработка на отказ ($T_{\text{о}}$) для ВС определяется соотношением

$$T_{\text{о(ВС)}} = \frac{K_{\text{г(ВС)}} \times T_{\text{в}}}{1 - K_{\text{г(ВС)}}}.$$

Ниже приведены результаты оценки показателей надежности. Исходные данные по надежности устройств и модулей, входящих в состав системы, были взяты из ранее выполненных расчетов показателей надежности вычислительных комплексов "Эльбрус-3М1" и "Эльбрус-90микро", а также из справочной литературы [7].

Расчеты, сделанные по описанной выше методологии, подтвердили, что описанная выше система со значительным запасом соответствует обычно задаваемым требованиям по надежности ($K_{\text{г(ВС)\_ТТЗ}} = 0{,}999$, $K_{\text{БР\_ТТЗ}} = 0{,}998$ при времени безотказной работы $t_{\text{БР}} = 1$ ч). При этом модель надежности построена с учетом такой организации технического обслуживания системы, которая обеспечивает выполнение заданного требования по коэффициенту планируемого использования $K_{\text{ПИ}} = 1$.

Заключение

В работе предложены методы построения высокопроизводительной, надежной вычислительной системы управления сложными объектами в реальном масштабе времени.

Первое — это выбор архитектуры, обеспечивающей высокую производительность вычислительного комплекса, на базе которого строится система. Такая архитектура реализована в вычислительном комплексе "Эльбрус-3М1".

Второе — разработка операционной системы реального времени. В работе сформулированы основные принципы построения такой операционной системы.

Третье — создание новой структуры вычислительной системы, которая позволила бы обеспечить требуемые высокие технические характеристики при сохранении всех каналов и связей, наследуемых от старой системы.

Четвертое — обеспечение высокой надежности системы с использованием апробированных методов дублирования и "горячего" резервирования.

Совокупность рассмотренных методов позволит получить вычислительную систему, отвечающую всем требованиям, предъявляемым к вычислительным системам реального времени для реализации программ управления сложными объектами, в первую очередь, объектами обороны, а также объектами атомной промышленности, транспорта, химической промышленности и т. п.

Список литературы

1. **Diefendorf K.** The Russians are coming. Supercomputer maker Elbrus seeks to join x86/IA-64 melee // Microprocessor Report. 1999. Vol. 11. № 2. P. 1—7.

2. **Волконский В. Ю.** Оптимизирующие компиляторы для архитектуры с явным параллелизмом команд и аппаратной поддержкой двоичной совместимости // Информационные технологии и вычислительные системы. 2004. № 3. С. 3—26.

3. **http://ru.wikipedia.org/w/index/** ... Операционные системы реального времени.

4. **Семенихин С. В.** ЭТП Операционная система реального времени для моделей МВК "Эльбрус-3М" и многомашинных комплексов на их основе (ОС РВ "ЭЛЬБРУС"). Пояснительная записка к техническому проекту. М.: Изд. ЗАО МЦСТ, 1997.

5. **Фельдман В. М.** Вычислительные комплексы "Эльбрус-90микро" // Информационные технологии. 2005. № 1. С. 17—26.

6. **Половко А. М., Гуров С. В.** Основы теории надежности. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2006. 704 с.

7. **Надежность** электрорадиоизделий. Электронный справочник. Официальное издание Минобороны РФ, 2002. 565 с.

УДК 004.337

Г. В. Кристовский, канд. техн. наук, нач. отд.,
Ю. Л. Погребной, вед. инж.,
С. А. Соин, ст. инж.,
ЗАО "МЦСТ"
E-mail: soin_s@mcst.ru

Разработка и исследование маломощного быстродействующего трехпортового регистрового файла

На примере разработки трехпортового регистрового файла показано, как оптимизация структуры, схемотехники и топологии позволяют минимизировать мощность регистрового файла без потерь быстродействия. С применением КМОП-технологии с минимальным размером 90 нм создан трехпортовый регистровый файл 32 × 32 с рабочей частотой 1 ГГц и мощностью менее 1 мВт на порт.

Ключевые слова: *микропроцессор, КМОП-технология, топология, дешифратор.*

Введение

В последние годы в области разработки регистровых файлов наметились две основные тенденции. Для организации одновременного выполнения нескольких команд в микропроцессорах с архитектурой длинного командного слова и в суперскалярных микропроцессорах используются регистровые файлы (РФ) с большим числом портов считывания и записи [1]. Вместе с тем, для компактных мобильных приборов требуются микропроцессоры с минимальной рассеиваемой мощностью, для создания которых необходимы быстродействующие маломощные регистровые файлы с ограниченным числом портов.

В данной работе анализируется структура и схемотехника трехпортового РФ, содержащего 32 слова с разрядностью 32 бита и работающего с частотой 1 ГГц. Приоритетной задачей данной разработки являлась минимизация мощности, рассеиваемой РФ. Основными приемами, используемыми при создании маломощных КМОП-схем, являются:

- уменьшение суммарной нагрузочной емкости;
- уменьшение напряжения, на которое перезаряжаются емкости;
- уменьшение числа переключений;
- оптимизация размеров транзисторов.

Разработка регистрового файла велась таким образом, чтобы максимально использовать все указанные способы уменьшения мощности.

1. Запоминающая ячейка и структура массива

Для уменьшения суммарной емкости нагрузки необходимо минимизировать площадь РФ, следовательно, необходимо уменьшить размеры запоминающей ячейки. Исходя из этого была выбрана однофазная схема считывания с предзарядом и дифференциальная схема записи, исключающая необходимость инверсии входных данных в ячейке. Минимизация транзисторов ячейки выполнялась с учетом структуры запоминающего массива и предполагаемой топологии ячейки. Электрическая схема ячейки приведена на рис. 1. Транзисторы хранящего триггера и проходные транзисторы для записи выбраны минимально возможными для данной технологии (для технологии 90 нм фирмы TSMC ширина этих транзисторов выбрана равной 0,22 мкм).

Транзисторы портов считывания также были оптимизированы исходя из требуемого времени считывания информации из регистрового файла.

При обращении к РФ значительная часть мощности уходит на перезаряд битовых шин, число которых равно разрядности, умноженной на число портов. Для того чтобы уменьшить эту составляющую мощности, емкость битовых шин должна быть минимальной. Для достижения этой цели в рассматриваемом проекте выбрана иерархическая структура тракта считывания: выходы смежных восьми ячеек подключены к своей локальной битовой шине, четыре локальные битовые шины передают считанный сигнал на одну глобальную битовую шину. Дополнительное уменьшение длины битовых шин достигается за счет вытянутой формы запоминающей ячейки: размер вдоль битовой шины примерно в 2 раза меньше размера вдоль словарной шины. Топология запоминающей

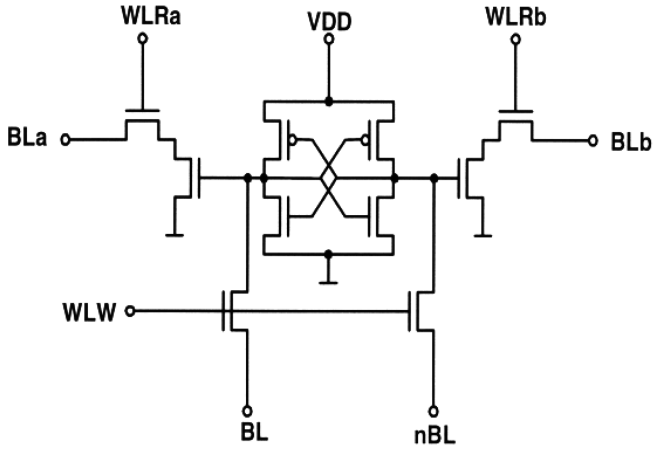


Рис. 1. Запоминающая ячейка

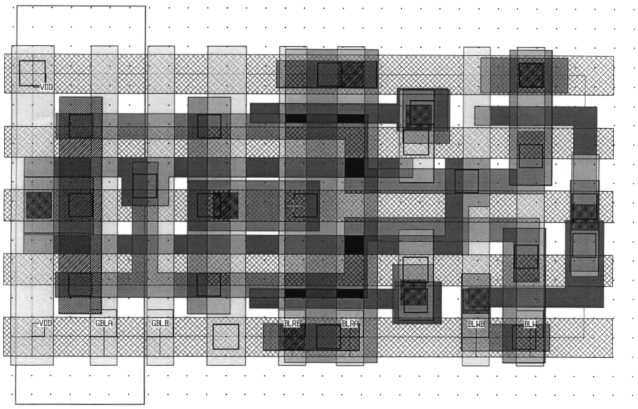


Рис. 2. Топология запоминающей ячейки

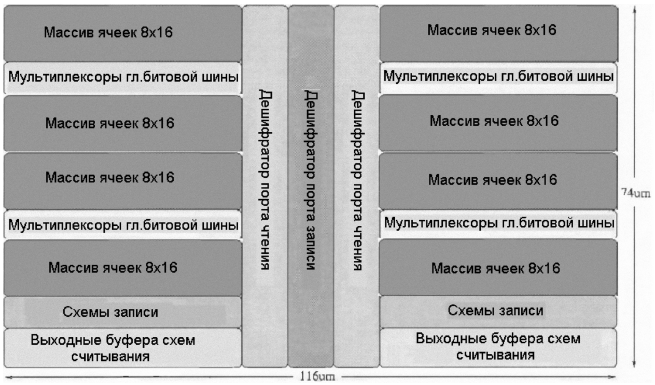


Рис. 3. Планировка регистрового файла

ячейки приведена на рис. 2. Из рисунка видно, что площадь ячейки определяется шинами: вертикальные битовые шины размещены во втором слое металлизации, а словарные шины размещены в третьем слое и идут горизонтально.

На рис. 3 приведен топологический план всего регистрового файла. Запоминающий массив разбит на четыре блока по восемь слов каждый. Для уменьшения мощности, рассеиваемой портом считывания, при каждом обращении активизируются только схемы выбранного блока. Это достигается посредством генерации индивидуальных синхросигналов для каждого блока. В качестве сигнала, разрешающего подачу синхросигнала на блок, используется дешифрированный сигнал двух старших разрядов адреса.

2. Схемотехника основных узлов регистрового файла

Дешифраторы. Для обеспечения необходимых значений времени выборки при считывании к дешифратору порта считывания предъявляются жесткие требования по задержке. Кроме того, этот дешифратор должен иметь минимальную мощность и малые времена опережения и удержания адресов.

Лучшие временные характеристики имеют дешифраторы, использующие динамическую схемотехнику. Однако динамические схемы имеют два недостатка: большая рассеиваемая мощность и большая площадь. Мощность обусловлена большой нагрузкой на синхросигнал, площадь — размерами транзисторов схем предзаряда. Особенностью схемы дешифратора является то, что в каждый момент времени работает только один выходной каскад, следовательно, можно организовать древообразную структуру, в которой синхросигнал подается только на один транзистор, вместо 32.

На рис. 4 показана структурная схема двухкаскадного дешифратора, оба каскада которого имеют древообразную структуру. На первом уровне используются схемы предварительной дешифрации адресов A1, A2.

Выходные сигналы со схемы предварительной дешифрации поступают на входы второго каскада дешифратора. Кроме того, на этот каскад поступает младший разряд адреса. Разряды адреса A3, A4 используются для генерации индивидуальных сигналов выборки для каждой секции дешифратора. Схема генерации этих сигналов такая же, как и схема предварительной дешифрации адресов A1, A2.

Ключевым элементом предложенной структуры дешифратора является "разветвитель на два направления", одновременно выполняющий функцию выходного элемента (рис. 5). Логически этот элемент является D-триггером, принимающим информацию по положительному фронту синхросигнала и обеспечивающим высокий уровень на одном из выходов в течение положительной фазы синхросигнала. Физически ток, поступающий на вход выборки, переключается в одну из выходных ветвей элемента (в зависимости от значений сигналов на адресных входах) и преобразуется в положительный импульс напряжения.

В отличие от типового динамического каскада в предложенном выходном элементе упрощена

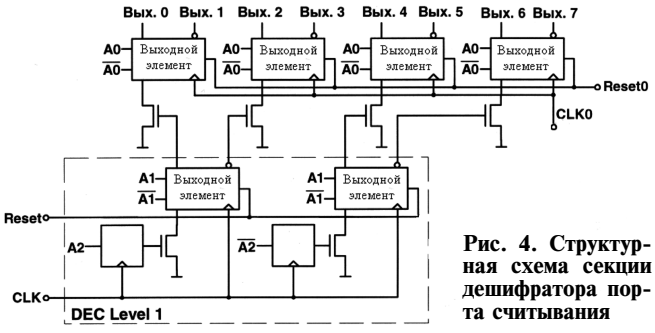


Рис. 4. Структурная схема секции дешифратора порта считывания

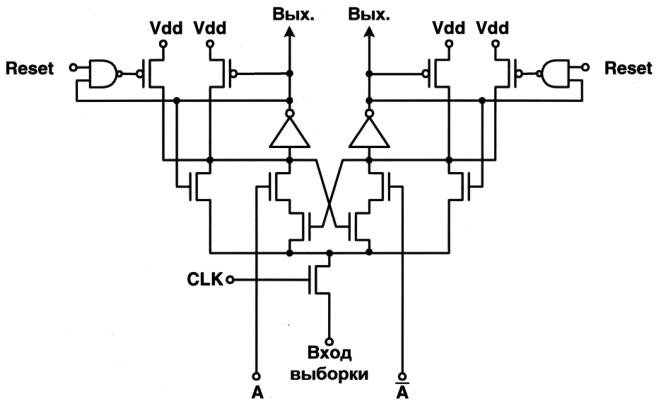


Рис. 5. Выходной элемент дешифратора порта считывания

схема восстановления исходного состояния. Это позволило уменьшить площадь и мощность, рассеиваемую выходным каскадом. Длительность положительного сигнала на словарной шине формируется с помощью техники копирования (*Replica technique* [2]). Идея заключается в том, что для учета влияния технологии, температуры и разброса напряжения на длительность сигнала выборки в каждом блоке из восьми слов реализованы контрольные словарная и битовая шины, по физическим характеристикам совпадающие с рабочими. Контрольные шины используются для формирования сигнала сброса, который определяет длительность сигнала выборки. Таким образом, при формировании длительности сигнала выборки учитываются реальные задержки распространения сигналов по словарным и битовым шинам. Кроме того, с помощью контрольной словарной шины формируются управляющие сигналы для схем подключения локальных битовых шин к глобальным и для схем предзаряда локальных битовых шин. Такой подход позволяет минимизировать размеры транзисторов предзаряда, оптимизировать временную диаграмму и обеспечить надежную работу регистрового файла в заданном диапазоне изменения внешних условий и технологических параметров.

Основное требование, предъявляемое к порту записи, это минимальная мощность. В соответствии с этим применена простейшая двухкаскадная схема дешифратора на статической КМОП-логике. Как в первом, так и во втором каскадах, используются элементы 3И-НЕ. Необходимая длительность сигналов формируется в первом каскаде дешифратора. Для этого на один из входов всех схем 3И-НЕ подается синхросигнал. Для уменьшения мощности в режиме молчания синхросигнал стробируется сигналом выборки порты. Таким образом, при отсутствии сигнала выборки порта переключаются только входные триггеры. Исследования показали, что такая реализация дешифратора позволяет получить минимальную мощность.

Тракт считывания. Структурная схема 1-го разряда тракта считывания приведена на рис. 6. Для уменьшения рассеиваемой мощности запоминающий массив регистрового файла разделен на четыре секции, каждая из которых содержит по восемь слов. В соответствии с этим в эквивалентную схему тракта считывания порта А включены: локальная битовая шина, объединяющая восемь запоминающих ячеек, схема подключения к глобальной битовой шине, глобальная битовая шина, выходной триггер с усилителем. Заметим, что выходной усилитель порта В содержит дополнительный инвертор, так как локальные битовые шины этого порта подключены к другому плечу запоминающей ячейки.

Оптимизация временной диаграммы работы порта считывания позволила реализовать полностью статические глобальные битовые шины, и тем самим уменьшить число переключений этих шин, так как при каждом считывании переключаются глобальные шины только тех разрядов, для которых изменилась считанная информация.

Наиболее сложной задачей оказалась передача сигнала от локальных динамических битовых шин к глобальной статической битовой шине, так как последняя имеет значительную емкость. Для ре-

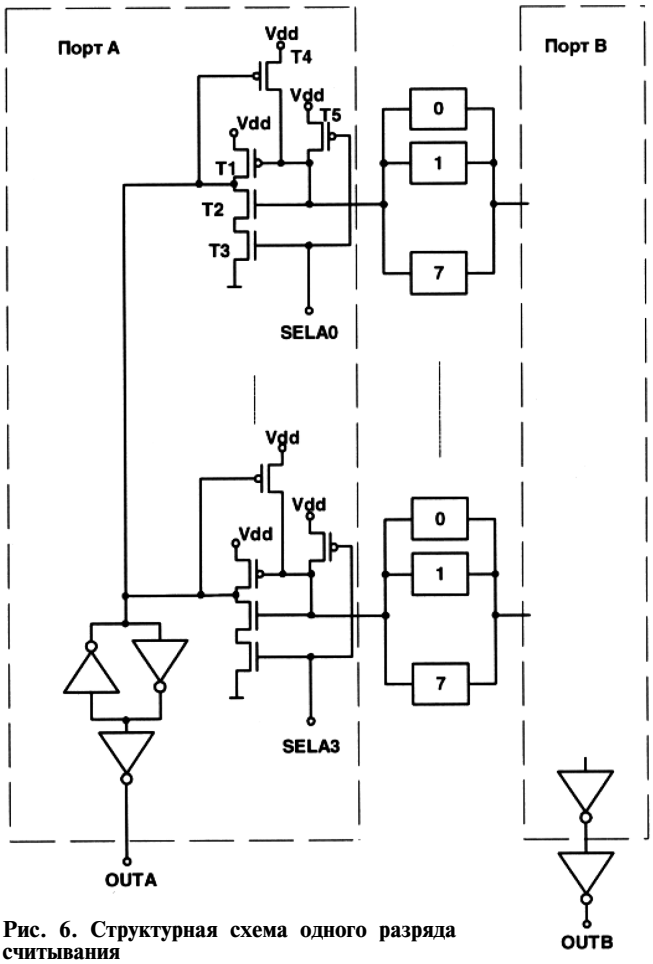


Рис. 6. Структурная схема одного разряда считывания

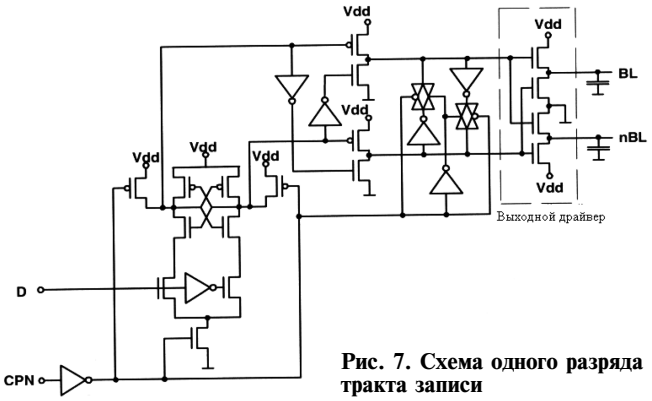


Рис. 7. Схема одного разряда тракта записи

шения этой задачи была разработана оригинальная схема, которая выполняет отключение невыбранных локальных битовых шин от глобальной, и обеспечивает необходимую скорость изменения потенциала глобальной битовой шины за счет сигнала с выбранной локальной битовой шины.

Схема работает следующим образом: когда на входе SEL низкий уровень, транзисторы T1 и T3 закрыты, и на выходе схемы — высокоомное состояние, транзистор T5 открыт и обеспечивает предварительный заряд локальной битовой шины до уровня, близкого к напряжению питания. При подаче на вход SEL высокого уровня транзистор T5 закрывается, транзистор T3 открывается, и схема работает как инвертор-усилитель, передавая изменения потенциала локальной битовой шины на глобальную. Назначение транзистора T4 заключается в повышении помехозащищенности схемы в режиме, когда на глобальной шине поддерживается низкий уровень. Помимо высокого быстродействия разработанная схема характеризуется малой мощностью, так как содержит всего пять транзисторов, причем транзистор T5 выполняет также функцию предзаряда локальной битовой шины.

Тракт записи. В цикле записи необходимо принять данные с 32-разрядной шины данных, обеспечить их хранение в течение времени, необходимого для записи, и в зависимости от значения разряда установить соответствующий дифференциальный сигнал на входах проходных транзисторов запоминающих ячеек данного разряда. В соответствии с этим каждый разряд тракта записи содержит входной триггер и выходной драйвер, нагруженный на парафазные битовые шины порта записи (рис. 7).

В качестве входного каскада триггера использован усилитель считывания, что позволяет уменьшить необходимое время опережения данных (D) по отношению к синхросигналу (CPN). Поскольку усилитель считывания хранит входные данные только во время отрицательной фазы синхросигнала, в триггер включен элемент хранения, реализованный на двух инверторах, работающих друг на друга через ключи, управляемые синхросигналом. Для уменьшения мощности триггера все транзисторы имеют минимальные размеры.

Выходной драйвер построен по схеме "тянитолкай", с применением только *n*-канальных транзисторов. Такой прием обеспечивает дополнительную экономию мощности, так как высокий уровень на соответствующей битовой шине (BL или nBL) не превышает напряжение Vdd-Vt, где Vt — пороговое напряжение *n*-канальных транзисторов.

3. Результаты моделирования

Изложенные выше принципы были использованы при проектировании топологии регистрового файла. При разработке топологии использовались правила проектирования для технологии TSMC 90 nm G. Одновременно с разработкой топологии проводилась оценка временных параметров и мощности регистрового файла с помощью программы электронного моделирования HSPICE. Отметим, что при моделировании учитывались не только характеристики транзисторов, но и паразитные параметры линии связи, экстрагированные из топологии с помощью программы STAR XT фирмы *Synopsis*. При наличии запаса по времени в соответствующих схемах уменьшались размеры транзисторов в целях получения минимальной площади и минимальной суммарной емкости.

На рис. 8 приведена временная диаграмма для основных сигналов регистрового файла в режиме считывания, полученная в результате моделирования с использованием программы SPICE. Отсчет времени начинается от переднего фронта синхросигнала CLK. Сигнал выборки секции BSEL формируется из выходных сигналов дешифратора, на вход которого подаются адреса A3, A4, и синхросигнала CLK. По переднему фронту

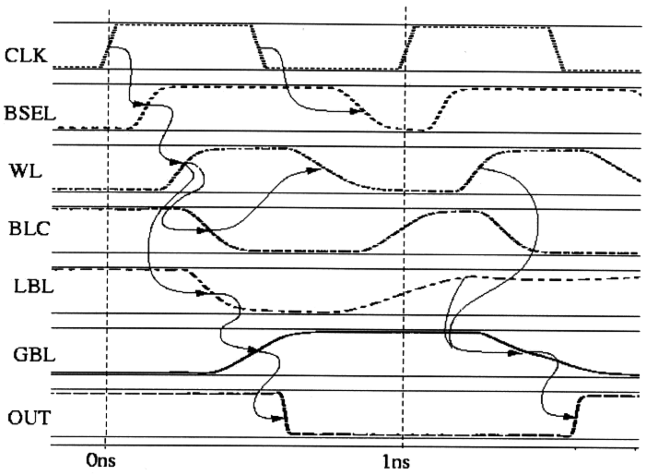


Рис. 8. Временная диаграмма работы регистрового файла

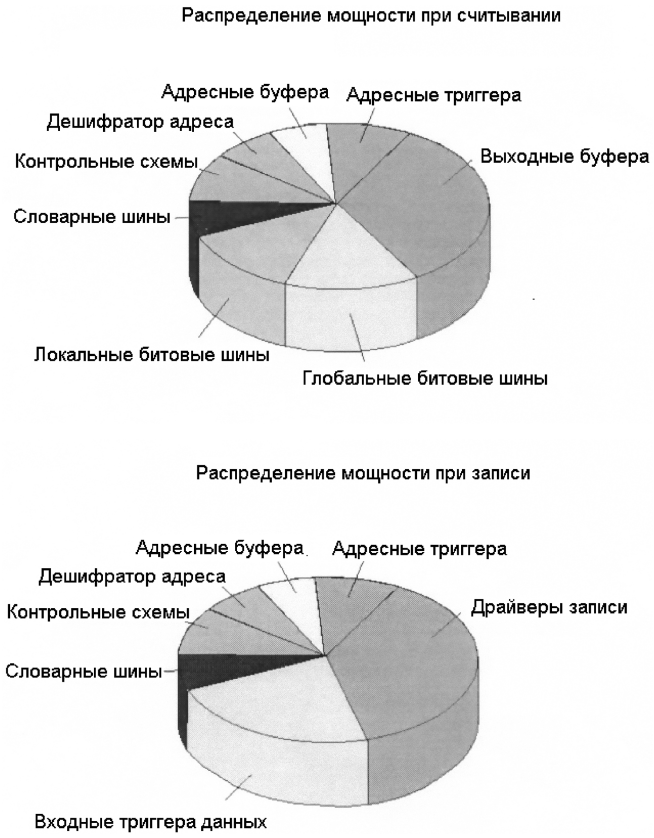


Рис. 9. Распределение мощности между блоками регистрового файла

этого сигнала формируется передний фронт сигнала на словарной шине — WL. Длительность сигнала на словарной шине определяется передним фронтом сигнала сброса BLC, который формируется с помощью техники копирования. С некоторой задержкой относительно переднего фронта сигнала WL появляется сигнал на локальной битовой шине LBL, который управляет схемой формирования сигнала глобальной битовой шины GBL, поступающего на выходной триггер.

Моделирование показало, что в типовых условиях максимальная частота работы регистрового файла равна 1,2 ГГц. Для повышенной температуры, напряжения питания 0,9 В и с учетом заданного технологического разброса параметров транзисторов максимальная частота равна 800 МГц.

На рис. 9 показано распределение мощности для режимов считывания и записи. При считывании основными потребителями мощности являются выходные усилители (измерения проводились при емкостной нагрузке, равной 5 фФ), локальные и глобальные битовые шины, дешифраторы и триггеры приема адресных и управляющих сигналов. При записи основная мощность выделяется на драйверах битовых шин, входных регистрах данных, дешифраторах и триггерах приема адресных и управляющих сигналов. Суммарная мощность регистрового файла в режиме считывании равна 0,58 мВт на порт, а в режиме записи — 0,55 мВт. Моделирование проводилось при температуре 125 °С, напряжении питания 0,9 В, на частоте 600 МГц.

Заключение

Для создания трехпортового регистрового файла, работающего с частотой порядка 1 ГГц и рассеивающего минимальную мощность, был использован комплексный подход, заключающийся в параллельной разработке конструкции, топологии и схемотехники. В результате был разработан регистровый файл, работающий с частотой 1 ГГц, с рассеиваемой мощностью менее 1 мВт на порт. Опыт разработки показал, что для достижения этих результатов определяющее значение имела оптимизация топологии. При разработке дешифраторов портов считывания, а также при разработке схемы объединения локальных битовых шин пришлось отказаться от известных схемотехнических решений для того, чтобы уменьшить площадь, занимаемую этими схемами, так как без уменьшения площади не удавалось достигнуть необходимого быстродействия. Доминирующая роль топологии объясняется тем обстоятельством, что для КМОП-технологии с минимальным размером 90 нм вклад линий связи в общую задержку превышает вклад транзисторов. Предложенные схемотехнические и топологические решения могут быть рекомендованы для использования в регистровых файлах следующих поколений микропроцессоров, изготовляемых по КМОП-технологии с минимальными нормами 0,65 нм и менее, поскольку вклад линий связи в задержку для этих технологий будет только возрастать.

Список литературы

1. **Подлесный А. В., Кириченко П. Г., Кристовский Г. В., Терентьев Ю. И.** Многопортовый регистровый файл: проблемы и способы их решения // Зарубежная радиоэлектроника. Успехи современной радиоэлектроники. 2002. № 7. С. 36—44.

2. **Amrutur B., Horowitz M.** A replica technique for wordline and sense control in low-power SRAMs // IEEE Journal of Solid-State Circuits. 1998. Vol. 33. № 8. P. 1208—1219.

УДК 004.382

С. Г. Бобков, канд. техн. наук, ст. науч. сотр.,
НИИ системных исследований РАН, г. Москва
E-mail: bobkov@cs.niisi.ras.ru

Методы повышения эффективности контроллера коммутатора Ethernet 10/100 Мбит/с

Рассматриваются методы повышения эффективности контроллера коммутатора Ethernet 10/100 Мбит/с, использованные при разработке микросхемы 1890КП1Я.

Ключевые слова: *контроллер коммутатора Ethernet, MAC-адрес, хэш-функция.*

Локальная сеть Ethernet стала самой популярной коммерческой сетью. Массовость выпуска микросхем контроллеров Ethernet позволила существенно снизить стоимость микросхем, создать развитое и надежное программное обеспечение. Это привело к использованию сети в самых различных областях промышленности, военных и научных системах. Специфика промышленных систем требует создания высоконадежных и эффективных систем. Повышение этих параметров является одной из самых актуальных задач развития сети Ethernet. В созданном контроллере коммутатора Ethernet использована разработанная методика поднятия эффективности управления коммутацией кадров на трех уровнях. На первом уровне определяются входные правила фильтрации кадров в соответствии с требованиями стандартов, которые должен поддерживать коммутатор: фильтрация сбойных кадров; фильтрация по виртуальным сетям; кадров с групповым адресом источника и т. д. На втором уровне происходит централизованная обработка кадра и определение его маршрута в соответствии с текущей конфигурацией сети. На третьем уровне определяются выходные правила фильтрации кадра в соответствии с настройками портов, на которые кадр коммутирован. Предложены специальные механизмы, позволяющие повысить эффективность использования коммутатора в кластерных системах: новая организация адресной таблицы и блока поддержки транковых соединений.

Метод построения адресной таблицы

Основным узлом "Блока обработки MAC-адресов" (рис. 1) является "Адресная таблица" (АТ), предназначенная для хранения MAC-адресов, масок коммутации и некоторой служебной информации, определяющей режим работы адресной таблицы и дополнительные правила коммутации.

Наиболее простым способом заполнения и поиска данных в адресной таблице является последовательная запись и последовательное чтение. Однако такой метод требует больших вычислительных затрат, что при больших загрузках может привести к сбоям в работе коммутатора, когда кадры будут поступать быстрее, чем обрабатываться.

Эффективным способом решения этой проблемы является хеширование — отображение точки пространства ключей (MAC-адресов) в точку пространства записей, т. е. зависимость адреса записи в память от данных [1].

При попытке отображения точек из некоторого широкого пространства в узкое неизбежны ситуации, когда разные точки в исходном пространстве отобразятся в одну и ту же точку в целевом пространстве. Ситуация, при которой разные ключи отображаются в один и тот же адрес записи, называется коллизией, или переполнением. Коллизии составляют основную проблему для хешированных таблиц. Если функция H, преобразующая ключ в адрес, может порождать коллизии, то однозначной обратной функции $k = H'(r)$, позволяющей восстановить ключ по известному адресу, существовать не может. Поэтому ключ должен обязательно входить в состав записи хешированной таблицы как одно из ее полей. К функции хеширования в общем случае предъявляются следующие требования:

- она должна обеспечивать равномерное распределение отображений фактических ключей по пространству записей;

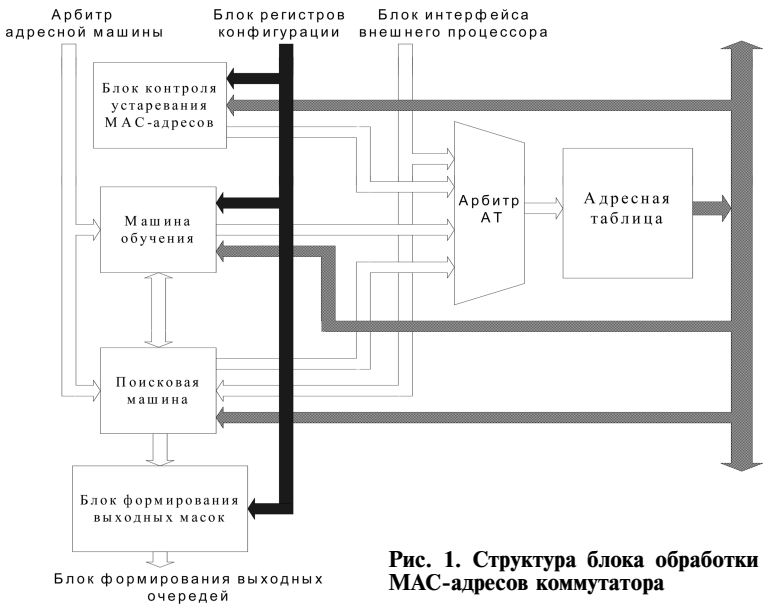


Рис. 1. Структура блока обработки MAC-адресов коммутатора

- она должна порождать как можно меньше коллизий для данного множества ключей;
- она не должна отображать какую-либо связь между значениями ключей в связь между значениями адресов;
- она должна быть простой и быстрой для вычисления.

Удачно подобранная хэш-функция может минимизировать число коллизий, но не может гарантировать их полного отсутствия. В работе [2] показано, что среднее число попыток поиска (записи) данных в таблице с использованием метода преобразования ключей в случае равномерного распределения можно оценить по формуле

$$E = -\frac{\ln(1-a)}{a},$$

где a — коэффициент заполнения, равный отношению занятой памяти ко всей имеющейся.

Из графика, приведенного на рис. 2, следует, что при заполнении области меньше 85 % запись с высокой вероятностью будет занесена в область хеширования.

Организация адресной таблицы и способ взаимодействия с ней являются частной разработкой компании-изготовителя и относятся к информации закрытого типа. В доступной документации на устройство указывается только размер адресной таблицы без описания алгоритмом поиска данных и заполнения АТ. Исключением является коммутатор AL126 компании Allayer Communications [3, 4].

Внутренняя адресная таблица AL126 рассчитана на 1031 запись. За счет подключения внешней статической памяти адресное пространство может быть увеличено до 8 или 16 килобайт (рис. 3). При использовании только внутренней адресной таблицы, поиск и обучение осуществляются с использованием функции хеширования CRC8 (порождающий многочлен кода не приведен в документации). Проблема коллизий решается с помощью метода пакетирования, сущность которого состоит в том, что записи таблицы объединяются в пакеты фиксированного и относительно небольшого размера — 8 записей, по значению кода CRC8 (*hash pointer*). Функция хеширования дает на выходе не адрес записи, а адрес пакета. После нахождения пакета в нем выполняется линейный поиск по ключу, или запись в первую свободную ячейку при автоматическом обучении. При полном заполнении любого из пакетов запись при обучении помещается в область переполнения (*overflow buffer*) размером 7, перезаписываемую в случае необходимости.

Недостатки организации адресной таблицы коммутатора в том, что при возникновении коллизии хэш-функции, гарантированное минимальное число адресов, которые могут быть запомнены в таблице, невелико и составляет лишь 1,45 % от максимально возможного числа, время поиска адреса при этом растет линейно от числа адресов, по которым произошла коллизия хэш-функции.

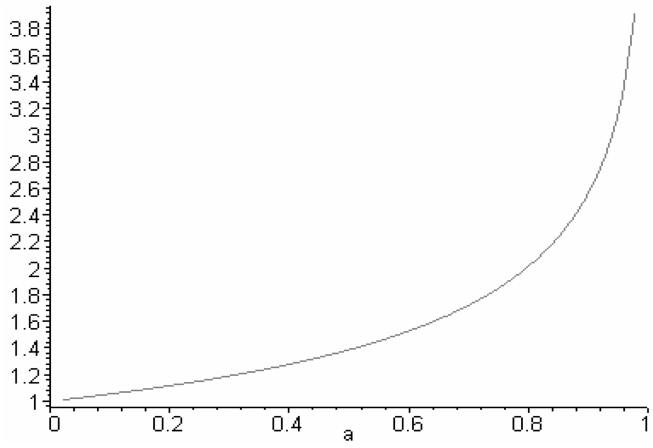


Рис. 2. Зависимость среднего числа попыток поиска (записи) данных в хешированной таблице от коэффициента заполнения

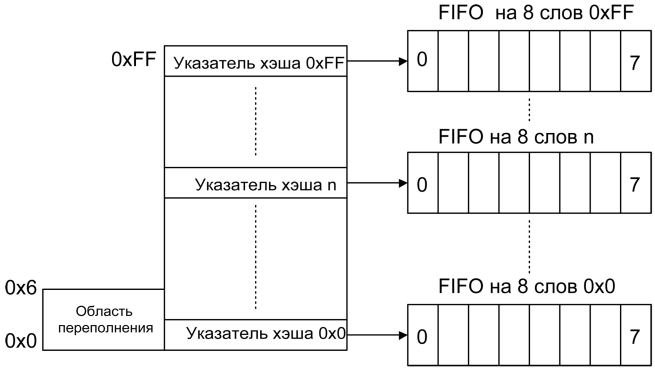


Рис. 3. Организация адресного пространства таблицы коммутатора AL126

Предложен метод организации адресной таблицы, который позволяет значительно повысить гарантированное минимальное число адресов, которые могут быть запомнены в таблице, и сократить при этом время поиска адреса в таблице. Адресная таблица имеет трехуровневую структуру. Первый уровень реализован с использованием хэш-функции на базе циклического кода CRC-10 [5]. Второй уровень введен для борьбы с коллизиями хэш-функции первого уровня и использует другую хэш-функцию циклического кода CRC-9. Третий уровень введен для борьбы с коллизиями хэш-функции второго уровня и использует алгоритм упорядоченного заполнения и бинарного поиска.

Адресная таблица состоит из двух страниц неравного размера, с возможностью одновременного обращения. Страница 0 АТ—область хеширования, страница 1 — область двоичного поиска. Ассоциативная память (страница 0) может содержать до 1536 адресов, логически разделенных на области хэш-функции первого уровня (1024 ячей-

Рис. 4. Трехуровневая организация адресной таблицы контроллера коммутатора

Значения параметров адресных таблиц

Коммутатор	Размер адресной таблицы (число записей)	Гарантированный минимум записей	Максимальное число итераций при поиске
Разработанная схема	2048	514 (25 %)	10
AL126, внутренняя таблица	1031	15 (1,4 %)	15
AL126, внешняя таблица	9223	19 (0,2 %)	19

ки) и второго уровня (512 ячеек); область переполнения (страница 1) может содержать до 512 адресов (рис. 4).

Если при попытке записи в таблицу оказывается, что требуемое место в таблице уже занято (происходит коллизия хэш-функции первого уровня), то значение записывается в ту же таблицу на какое-то другое место. Другое место определяется с помощью вторичной функции хеширования H2, аргументом которой также является исходное значение адреса. Хэш-функция второго уровня выбирается так, чтобы пространства записей для H1 и H2 не перекрывались.

Предложенный способ организации адресной таблицы [6] и механизмы заполнения таблицы и поиска адресов позволяют с хорошей долей вероятности поддерживать различные комбинации сетевых адресов в системе: при максимальном размере таблицы в 2048 записей минимальное гарантированное число адресов составляет 514, т. е. 25 % от максимального числа. Максимальное время, затрачиваемое на поиск адреса при коллизиях, растет при этом логарифмически от числа адресов, занесенных в общую область переполнения.

Образующие полиномы хэш-функций CRC-9 и CRC-10 подобраны таким образом, чтобы вероятности совпадения кодов для различных 48-разрядных сетевых адресов при использовании этих хэш-функций были минимальными. На основании результатов программ, написанных на языке программирования Си, дающих оценку использования различных полиномов в качестве порождающих многочленов кодов, были выбраны следующие полиномы:

CRC10: $x^{10} + x^9 + x^7 + x^3 + 1$, в восьмеричной системе счисления 32118.

CRC9: $x^9 + x^7 + x^5 + x^4 + x^2 + x + 1$, в восьмеричной системе счисления 12678.

Аппаратное вычисление кодов в потоке параллельных данных выполняется за один такт.

В таблице приведено сравнение характеристик адресной таблицы коммутатора AL126 и предложенной схемы.

На рис. 5 приведена структура устройства, реализующего метод трехуровневой организации адресной таблицы.

Принцип работы устройства обработки адресов коммутатора состоит в следующем. Адресная таблица коммутатора физически разделена на две страницы неравного размера. Первая физическая страница разделена логически на две части. Заполнение первой половины второй страницы и поиск адресов в этой части таблицы проводится с использованием хэш-функции типа 1. Хэш-функция типа 1 использует аппаратную реализацию алгоритма вычисления циклического избыточно-

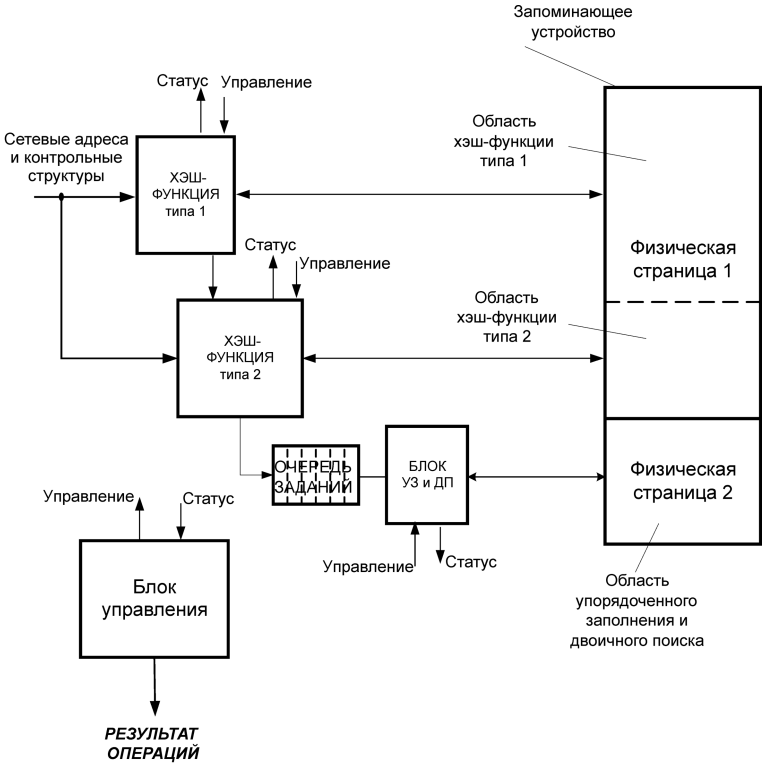


Рис. 5. Структура устройства, реализующего метод трехуровневой организации адресной таблицы контроллера коммутатора

го кода (CRC — *Cycle Redundancy Code*), который осуществляет отображение широкого поля MAC-адреса (48-разрядного) в узкое поле адреса ячейки в адресной таблице (10-разрядное). Заполнение второй части этой физической страницы, меньшего размера, проводится с использованием хэш-функции типа 2. Хэш-функция типа 2 использует аппаратную реализацию алгоритма вычисления циклического избыточного кода, который осуществляет отображение широкого поля адреса станции в сети (48-разрядного) в узкое поле адреса ячейки в адресной таблице (9-разрядное). Хэш-функция типа 2 отличается от хэш-функции типа 1 тем, что она использует другой порождающий многочлен, и разрядность результата на ее выходе на один разряд меньше разрядности результата на выходе первой хэш-функции. Заполнение второй физической страницы адресной таблицы проводится упорядоченно по возрастанию адресов станций. Адреса станций в этой части таблицы ищутся методом двоичного поиска. Соответствующие операции реализованы в блоке упорядоченного заполнения и двоичного поиска. Обращение к области упорядоченного заполнения и двоичного поиска выполняется медленнее, чем это происходит при использовании хэш-функций. В данном случае время обращения растет линейно при заполнении таблицы и логарифмически при поиске адресов в таблице. Поэтому, чтобы не блокировать работу устройств, с которых поступает адресная информация на устройство обработки адресов коммутатора, запросы к этой области памяти поступают через буферную очередь заданий, выполненную по принципу "первым пришел — первым вышел" (FIFO — *first input, first output*).

Обе хэш-функции используются одновременно, и если произошла коллизия по одной хэш-функции, то управление передается второй хэш-функции, если же произошла коллизия и по второй хэш-функции, то управление передается через очередь заданий в блок реализации операции упорядоченного заполнения и двоичного поиска. Управление работой устройства обработки адресов коммутатора локальной вычислительной сети осуществляется блоком управления.

Метод организации транковых соединений

При создании сетей, состоящих из нескольких коммутаторов, узким местом являются магистральные каналы, объединяющие коммутаторы друг с другом. Магистральный канал может быть выполнен либо как обычный порт коммутатора, либо как группа физических портов, объединенных в один логический порт. При группировке портов образуется канал с пропускной способностью, равной суммарной пропускной способности объединенных портов. Такие групповые соединения называются транковыми.

Основной проблемой при создании транкового соединения является проблема выбора коммутации кадров на тот или иной физический порт в составе транкового соединения так, чтобы оптимально сбалансировать трафик. Существуют два основных алгоритма распределения трафика в транковом соединении: фиксированное отображение портов вне транка на порты в транке и случайное распределение трафика.

При фиксированном отображении портов каждому порту, не входящему в транковое соединение, ставится в соответствие один из портов, входящих в транковое соединение. Все кадры с этого нетранкового порта, предназначенные для транка, будут направляться на этот фиксированный физический порт в составе транка.

При случайном распределении трафика все кадры, предназначенные для передачи через транковое соединение, распределяются между физическими портами в составе транка случайным образом по некоторому заданному псевдослучайному алгоритму. Распределение может быть равномерным, а может в качестве параметра учитывать характеристики кадра, такие как адрес назначения и адрес источника кадра.

Недостаток фиксированного отображения портов в том, что при назначении соответствия транкового порта с меньшей пропускной способностью нетранковому порту с большей пропускной способностью, транковое соединение используется неэффективно и часть пропускной способности нетранкового порта теряется.

Недостаток случайного распределения в том, что при такой балансировке трафика может непредсказуемо нарушаться последовательность следования кадров, и для ряда приложений, не восстанавливающих на конечных узлах последовательность прихода пакетов, это будет приводить к сбоям.

В разработанном коммутаторе используются оба рассмотренных механизма балансировки трафика, и кроме того, добавлен дополнительный механизм, устраняющий недостаток неэффективного распределения трафика, присущий фиксированным соединениям, и недостаток непредсказуемого распределения последовательности кадров, присущий случайным распределениям [7].

Предложенный механизм реализует циклический перебор целевых портов в составе транкового соединения при поступлении запросов на передачу кадров через это соединение. Каждый следующий кадр, предназначенный для передачи через транк, передается через очередной по порядку

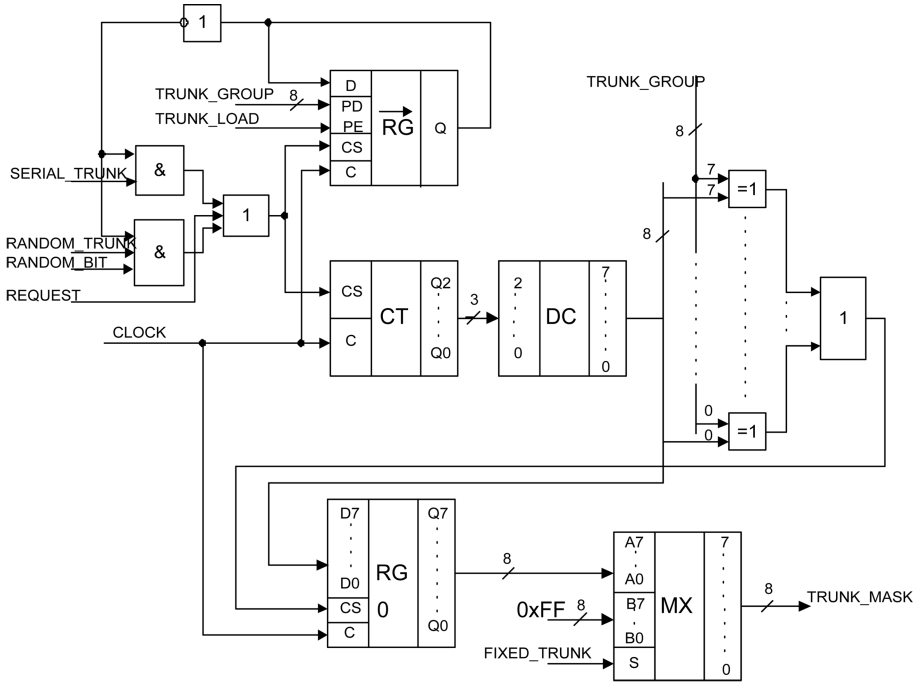


Рис. 6. Схема блока, вычисляющего маску транкового соединения

физический порт в составе транка. Так, например, если транковое соединение содержит 1,2,3 и 4-й порты, то 1-й кадр будет передан через 1-й порт, 2-й кадр через 2-й порт, 3-й кадр через 3-й порт, 4-й кадр через 4-й порт, 5-й кадр через 1-й порт, и т. д. В предложенном механизме трафик распределяется по портам внутри транка равномерно, как и для случайного распределения, но при этом сохраняется предсказуемость порядка следования кадров.

На рис. 6 приведена схема блока, реализующего все три механизма транкового соединения. Транковой маской является байт с одним единичным битом в любой из позиций единиц в регистре соответствующей транковой группы. В этот регистр на этапе инициализации записывается маска группы транка. По сигналу запроса, который является признаком завершения поиска в адресной таблице адреса назначения, содержимое регистра циклически сдвигается влево. Число позиций сдвига N подсчитывает счетчик на 8. Дешифратор преобразует двоичное число N в байт с одним единичным битом в N-м разряде. Если значения битов N-го разряда транковой группы и сформированного байта совпадают, то байт является транковой маской, в противном случае сдвиг продолжается — до появления в младшем разряде единицы. При случайном выборе порта из транка на формирование сигнала CS выборки регистров и счетчика накладывается дополнительное ограничение — случайный бит random_bit. В этом случае последовательный перебор допустимых транковых масок продолжается при нулевом значении random_bit и прекращается в противном случае. Для формирования случайного бита используется параллельный кодировщик циклического кода CRC8. В качестве порождающего многочлена кода выбран полином, используемый в технологиях ATM и Myrinet. Кодировщик вычисляет CRC8 для текущего значения 8-разрядного счетчика тактового сигнала clock. В качестве random_bit используется 5-й бит выхода кодировщика CRC8.

Заключение

Таким образом, разработана трехуровневая структура адресной таблицы сетевого коммутатора, позволяющая проводить поиск адреса в 1,5...2 раза быстрее, и иметь минимальное гарантированное число адресов в 30 раз большее, чем у рассмотренных зарубежных аналогов. Первый уровень реализован с использованием хэш-функции на базе циклического кода CRC-10. Второй уровень введен для борьбы с коллизиями хэш-функции уровня 1 и использует другую хэш-функцию циклического кода CRC-9. Третий уровень введен для борьбы с коллизиями хэш-функции уровня 2 и использует алгоритм упорядоченного заполнения и бинарного поиска.

Список литературы

1. **Огнев И. В., Борисов В. В.** Интеллектуальные системы ассоциативной памяти. — М.: Радио и связь, 1996. — 176 с.
2. **Вирт Н.** Алгоритмы и структуры данных: Пер. с англ. — М.: Мир, 1989. — 360 с.
3. **Understanding** address hashing. — USA.: Allayer Communications, 1999. — 4 pp. (http://www.allayer.com/pdf/An002.pdf)
4. **8 Port 10/100 Mbits/s** Dual Speed Fast Ethernet Switch AL126. — USA.: Allayer Communications, 1999. — 98 pp. (http://www.allayer.com/pdf/A1126rev.pdf)
5. **Акритас А.** Основы компьютерной алгебры с приложениями / Пер. с англ. Е. В. Панкратьева. М.: Мир, 1994.
6. **Бобков С. Г., Сидоров Е. А.** Устройство обработки адресов коммутатора локальной вычислительной сети, работающего по принципу прозрачного моста, Патент RU 2304802 C1, 2007.
7. **Сидоров Е. А., Бобков С. Г., Власов А. Ю., Пивоваров В. В.** Транковые соединения коммутатора ЛВС // Электроника, микро- и наноэлектроника. Сб. науч. тр. / Под ред. В. Я. Стенина. — М.: МИФИ, 2002. С. 107—111.

УДК 007.629.735

Аль-Аммори Али, канд. техн. наук,
Почетный вице-президент
Научно-методического центра
процессного анализа (НМЦПА)
Национальный транспортный университет,
г. Киев, Украина
E-mail: ammourilion@ukr.net

Оптимизация параллельного информационного резервирования методом вложенных модулей

Предложена методика выбора и обоснования оптимальной структуры параллельного информационного резервирования способом вложенных модулей с учетом технической надежности. Обоснованы экономические преимущества способа вложенных модулей по сравнению с альтернативным вариантом такого же резервирования, но без принципа вложенных модулей.

Ключевые слова: *информационное резервирование, вложенные модули резервирования, техническая надежность.*

Основным техническим требованием к информационным системам контроля и измерений является требование высокой достоверности информации. Такое требование количественно оценивается допустимо малыми вероятностями ошибок I и II рода, а именно, вероятностью ложной тревоги $p_{лт}$ и необнаружения $p_{но}$ контролируемого явления (пожар внутри авиадвигателя, распознавание информационно-управляющих сигналов) [1, 2, 3]. Это основное требование высокой достоверности, определяемое двумя условиями:

- высокие функциональные возможности контролируемого датчика информации (КДИ), которые определяют высокую достоверность информации, определяемую двумя основными характеристиками (требуемые малые $p_{лт}$ и $p_{но}$);
- высокая техническая надежность КДИ, которая также влияет на основные показатели качества $p_{лт}$ и $p_{но}$.

Как показывает практика конструирования всех видов КДИ, указанные технические условия являются диалектически противоречивыми. Как правило, устройство с высокими функциональными возможностями имеет сложную техническую конструкцию и, как следствие этого, низкую техническую надежность при высокой стоимости его изготовления и эксплуатации.

Решение этой сложной проблемы может быть достигнуто методом создания параллельных структур информационного резервирования, которые могут состоять из простых по конструкции, технически надежных КДИ, но имеющих низкие функциональные характеристики, т. е. немалые $p_{лт}$ и $p_{но}$. Как показывают расчеты, достигаются достаточно эффективные функциональные возможности таких систем параллельного информационного резервирования с мажоритарным принципом "m из n" принятия решения, в которых решение о наличии контролируемого явления принимается тогда, когда "m из n" КДИ заявляют о его наличии. При этом системные вероятности $p_{лт}$ и $p_{но}$ получаются достаточно низкими и удовлетворяют поставленным требованиям, хотя эти же характеристики для каждого отдельного КДИ могут быть неудовлетворительными.

Таким образом, появляется принципиальная возможность создания высокоэффективных технически надежных систем параллельного информационного резервирования с мажоритарным принципом принятия решения, состоящих из n КДИ с высокой технической надежностью, но с низкими показателями качества $p_{лп}$ и $p_{но}$.

Механическим аналогом эффективности таких информационных систем является **трос,** который, как известно, обладает значительно более высокой прочностью, чем цельнометаллический прут такого же диаметра, хотя сам **трос** состоит из отдельных тонких проводков с низкой прочностью.

Однако при создании сложных систем параллельного резервирования появляется проблема обеспечения высокой технической надежности соединений большого числа отдельных элементов в единую монолитную высокоэффективную информационную систему с высокой технической надежностью.

Такая проблема может быть решена с помощью объединения отдельных информационных элементов системы КДИ методом вложенных модулей.

Рассмотрим ряд конкретных примеров

1. Модуль $M_{2,\,3}$ — составная информационная система (рис. 1), объединяющая три контролирующих датчика информации x_1, x_2, x_3 и принимающая решения о наличии контролируемого явления по мажоритарному принципу "2 из 3" [1, 2].

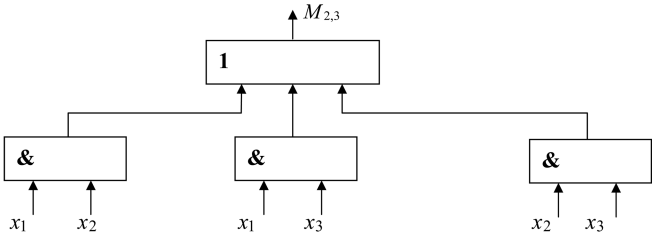


Рис. 1. Функциональная схема информационного модуля $M_{2,\,3}$

Объединение КДИ x_i в информационном модуле $M_{2,3}$ осуществляется с помощью логических элементов И и ИЛИ.

Системные вероятностные характеристики, а именно, вероятность p_{01} достоверного определения появления контролируемого события, вероятности p_{02}, p_{03} необнаружения и ложной тревоги, соответственно, можно определить на основании триномиального распределения [4, 5] по следующим формулам:

$$\left.\begin{aligned} p_{01} &= 1 - p_{02} - p_{03};\\ p_{02} &= \sum_{i=0}^{1} C_3^i (a_0 + b_0)^i d_0^{(3-i)};\\ p_{03} &= \sum_{i=0}^{1} C_3^i (a_0 + d_0)^i b_0^{(3-i)}, \end{aligned}\right\} \qquad (1)$$

где a_0, b_0, d_0 — соответственно вероятности достоверного обнаружения, ложной тревоги и необнаружения контролируемого явления отдельным КДИ.

Формулы (1) выведены исходя из условия, что КДИ системы имеют идентичные вероятностные характеристики a_0, b_0, d_0 и обладают одинаковой технической надежностью [1, 2, 3], определяемой технической надежностью k каждого отдельного элемента, равной 1. Если $k < 1$, то формулы (1) можно легко трансформировать к следующему виду:

$$\left.\begin{aligned} p_{01} &= k - p_{02} - p_{03};\\ p_{02} &= \sum_{i=0}^{1} C_3^i (a + b)^i d^{(3-i)};\\ p_{03} &= \sum_{i=0}^{1} C_3^i (a + d)^i b^{(3-i)}, \end{aligned}\right\} \qquad (2)$$

где $a = a_0 k$, $b = b_0 k$, $d = d_0 k$.

2. Модуль $M_{2,4}$ — составная информационная система (рис. 2), объединяющая четыре контролирующих датчика информации x_1, x_2, x_3, x_4 и принимающая решение о наличии контролируемого явления по мажоритарному принципу "2 из 4".

Аналогично, как и для формул (2), можно оценить вероятностные характеристики системы (рис. 2) на основании триномиального распределения вероятностей [4, 5] и с учетом технической надежности k с помощью следующих формул:

$$\left.\begin{aligned} p_{01} &= k - p_{02} - p_{03};\\ p_{02} &= \sum_{i=0}^{2} C_4^i (a + b)^i d^{(4-i)};\\ p_{03} &= \sum_{i=0}^{2} C_4^i (a + d)^i b^{(4-i)}. \end{aligned}\right\} \qquad (3)$$

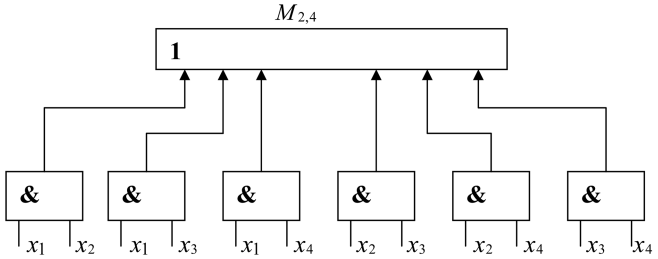


Рис. 2. Функциональная схема информационного модуля $M_{2,4}$

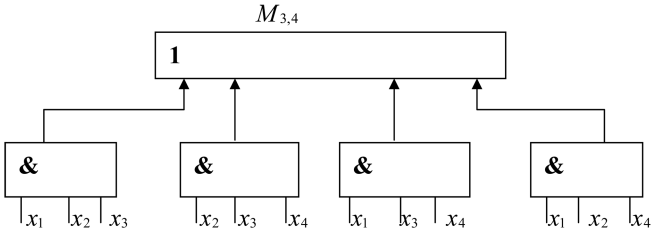


Рис. 3. Функциональная схема информационного модуля $M_{3,4}$

Системные вероятностные характеристики, а именно, вероятность p_{01} достоверного определения появления контролируемого события, вероятности p_{02}, p_{03} необнаружения и ложной тревоги для модуля $M_{3,4}$ (рис. 3), соответственно, можно определить на основании триномиального распределения [4, 5] по следующим формулам:

$$\left.\begin{aligned} p_{01} &= k - p_{02} - p_{03};\\ p_{02} &= \sum_{i=0}^{1} C_4^i (a + b)^i d^{(4-i)};\\ p_{03} &= \sum_{i=0}^{1} C_4^i (a + d)^i b^{(4-i)}. \end{aligned}\right\} \qquad (4)$$

Таким образом, формулы (3) отличаются от формулы (4) индексом мажоритарности "2 из 4" и "3 из 4".

3. Модуль $M_{3,5}$ — составная информационная система (рис. 4), объединяющая пять контролирующих датчиков информации x_1, x_2, x_3, x_4, x_5 и принимающая решение о наличии контролируемого явления по мажоритарному принципу "3 из 5".

Аналогично, как и для формул (3), можно оценить вероятностные характеристики системы

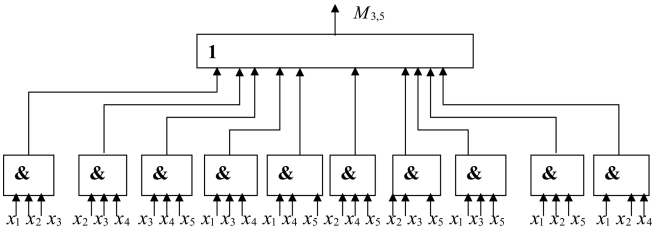


Рис. 4. Функциональная схема информационного модуля $M_{3,5}$

(рис. 4) на основании триноминального распределения вероятностей и с учетом технической надежности k с помощью следующих формул:

$$\left.\begin{aligned} p_{01} &= k - p_{02} - p_{03}; \\ p_{02} &= \sum_{i=0}^{2} C_5^i (a+b)^i d^{(5-i)}; \\ p_{03} &= \sum_{i=0}^{2} C_5^i (a+d)^i b^{(5-i)}. \end{aligned}\right\} \qquad (5)$$

Из приведенных примеров можно вывести общие формулы определения вероятностных характеристик p_{01}, p_{02}, p_{03} для модуля $M_{m,\,n}$, объединяющего n КДИ и принимающего решение по мажоритарному принципу "m из n", которые будут иметь вид

$$\left.\begin{aligned} p_{01} &= k - p_{02} - p_{03}; \\ p_{02} &= \sum_{i=0}^{n-m} C_n^i (a+b)^i d^{(n-i)}; \\ p_{03} &= \sum_{i=0}^{n-m} C_n^i (a+d)^i b^{(n-i)}. \end{aligned}\right\} \qquad (6)$$

Теперь применим принцип параллельного информационного резервирования способом вложенных модулей. Например, на основе модуля $M_{2,\,3}$ (см. рис. 1) можно легко организовать информационную систему $N_{6,\,9}$ (рис. 5), которая способом вложенных модулей $M_{2,\,3}$ легко и просто объединяет девять первичных КДИ x_i, реализуя мажоритарный принцип принятия решения "6 из 9".

Функциональная схема на рис. 5 отличается от схемы, приведенной на рис. 1, тем, что на ее информационных входах вместо первичных КДИ x_i подключены модули $M_{2,\,3}$, которые в совокупности объединяют девять первичных КДИ x_i, и при этом автоматически реализуется мажоритарный принцип принятия решения "6 из 9" вместо "2 из 3", как это свойственно информационному модулю $M_{2,\,3}$.

Показатели качества модульной информационной системы $N_{6,\,9}$, а именно, вероятности p_{11}, p_{12}, p_{13} оцениваются по тем же формулам (2), что и для модуля $M_{2,\,3}$, только аргументами в этих формулах будут показатели качества p_{01}, p_{02}, p_{03} информационного модуля $M_{2,\,3}$. Другими словами, если вероятности p_{01}, p_{02}, p_{03} есть функции от вероятностей a, b, d, т. е. $p_{01} = f(a,\ b,\ d)$, $p_{02} = f(a,\ b,\ d)$, $p_{03} = f(a,\ b,\ d)$, то вероятности p_{11}, p_{12}, p_{13} будут функциями от вероятностей p_{01}, p_{02}, p_{03}:

$$p_{11} = f(p_{01}, p_{02}, p_{03}),\ p_{12} = f(p_{01}, p_{02}, p_{03}),\ p_{13} = f(p_{01}, p_{02}, p_{03}). \qquad (7)$$

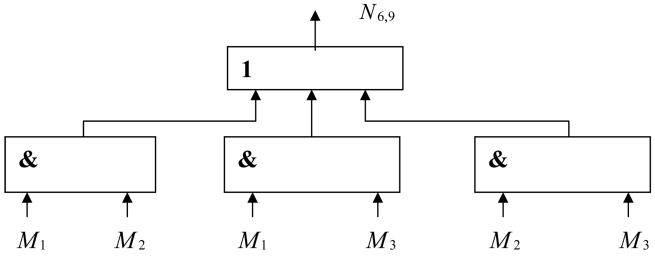


Рис. 5. Функциональная схема модульной информационной системы $N_{6,\,9}$

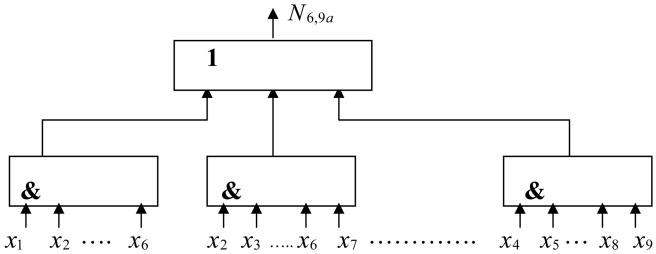


Рис. 6. Функциональная схема альтернативной немодульной информационной системы, реализующей мажоритарный принцип "6 из 9"

Это можно выразить следующими расчетными формулами:

$$\left.\begin{aligned} p_{11} &= k - p_{12} - p_{13}; \\ p_{12} &= \sum_{i=0}^{1} C_3^i (p_{01} + p_{03})^i p_{02}^{(3-i)}; \\ p_{13} &= \sum_{i=0}^{1} C_3^i (p_{01} + p_{02})^i p_{03}^{(3-i)}. \end{aligned}\right\} \qquad (8)$$

Для анализа и оценок преимуществ и недостатков информационного резервирования методом вложенных модулей приведем схему (рис. 6) альтернативного (немодульного) информационного резервирования системы $N_{6,\,9a}$, реализующей мажоритарный принцип "6 из 9".

На основании выражений (6) можно вывести формулы (9), определяющие показатели качества, а именно, вероятности p_{11a} достоверного обнаружения, вероятности p_{12a} необнаружения и вероятности p_{13a} ложной тревоги:

$$\left.\begin{aligned} p_{11a} &= k - p_{12a} - p_{13a}; \\ p_{12a} &= \sum_{i=0}^{3} C_9^i (a+b)^i d^{(9-i)}; \\ p_{13a} &= \sum_{i=0}^{3} C_9^i (a+d)^i b^{(9-i)}. \end{aligned}\right\} \qquad (9)$$

Определим число схем И, как основных соединительных элементов для схемы $N_{6,\,9}$ вложенных модулей (см. рис. 5).

Если взять для первичного модуля $M_{2,\,3}$ число $m_0 = 2$ и число $n_0 = 3$, то можно показать, что для

схемы $M_{6,\,9}$ параметры $m_1 = 6$ и $n_1 = 9$ соответственно определяются по следующим формулам:

$$\left.\begin{aligned} m_1 &= m_0 C_{n_0}^{m_0}; \\ n_1 &= n_0 C_{n_0}^{m_0}. \end{aligned}\right\} \qquad (10)$$

Анализируя схему вложенного модуля $N_{6,\,9}$ (см. рис. 5), можно определить общее число K_{u1} схем И для схемы $N_{6,\,9}$:

$$K_{u1} = m_1 C_{n_0}^{m_0} + C_{n_0}^{m_0}. \qquad (11)$$

Подставив в формулу (11) значение m_1 из формулы (10) и выполнив небольшие преобразования, получим общую формулу для определения K_{u1}:

$$K_{u1} = C_{n_0}^{m_0}(m_0 C_{n_0}^{m_0} + 1). \qquad (12)$$

Нетрудно подсчитать $K_{u1} = 21$ для схемы информационного модуля $N_{6,\,9}$ (см. рис. 5), где $m_0 = 2$, $n_0 = 3$.

Число K_{ua} схем И для альтернативного варианта схемы $N_{6,\,9}$ (рис. 6), очевидно, определяется числом сочетаний из n_1 по m_1, т. е.

$$K_{ua} = C_{n_1}^{m_1} = C_9^6 = 84. \qquad (12a)$$

Таким образом, число соединительных схем И для системы $N_{6,\,9}$ вложенных модулей (см. рис. 5) меньше в 4 раза, чем для альтернативной системы $N_{6,\,9a}$ (рис. 6), т. е. техническая надежность соединительных связей для системы вложенных модулей $N_{6,\,9}$ будет существенно выше при меньшей стоимости изготовления и использования, чем для альтернативной системы $N_{6,\,9a}$. Время контроля при этом не увеличивается, поскольку информация передается практически мгновенно без цифровой обработки.

Это достоинство можно оценить с помощью сравнительного коэффициента $V_{6,\,9}$, определяемого общей формулой

$$V_{m1n1} = \frac{K_{ua}}{K_{u1}} = \frac{C_{n_1}^{m_1}}{C_{n_0}^{m_0}(m_0 C_{n_0}^{m_0} + 1)} = \frac{84}{21} = 4. \qquad (13)$$

Рассмотрим теперь вторую иерархическую ступень информационного резервирования методом вложенных модулей. Если в схему $N_{6,\,9}$ (см. рис. 5) вместо модулей $M_{2,\,3}$ как источников информации поставить модули $N_{6,\,9}$, то получим иерархическую систему $L_{18,\,27}$ вложенных модулей (рис. 7).

Показатели качества модульной информационной системы $L_{18,\,27}$, а именно, вероятности p_{21}, p_{22}, p_{23} оцениваются по тем же формулам (2), относящимся к модулю $M_{2,\,3}$, только аргументами в этих формулах будут показатели качества p_{11}, p_{12}, p_{13}, т. е. вероятности p_{21}, p_{22}, p_{23} будут функциями

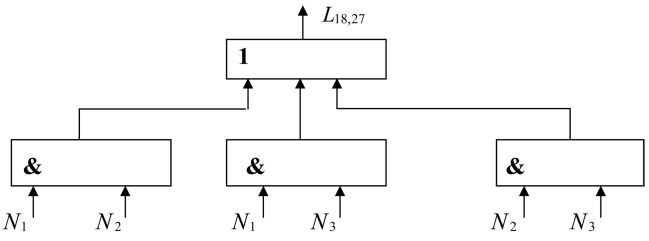


Рис. 7. Функциональная схема информационной резервированной системы, созданной методом вложенных модулей с двумя ступенями иерархии, реализующая мажоритарный принцип принятия решения "18 из 27"

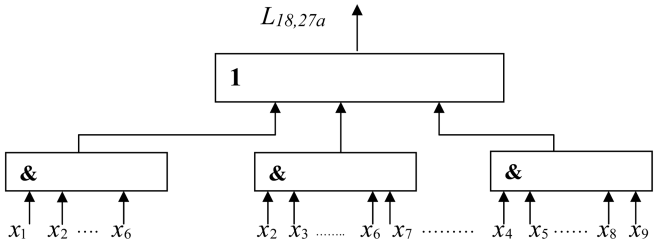


Рис. 8. Функциональная схема альтернативного варианта параллельного резервирования КДИ x_i с мажоритарным принципом "18 из 27*a*"

от вероятностей p_{11}, p_{12}, p_{13}, и их можно оценить с помощью следующих выражений:

$$\left.\begin{aligned} p_{21} &= k - p_{22} - p_{23}; \\ p_{22} &= \sum_{i=0}^{1} C_3^i (p_{11} + p_{13})^i p_{12}^{(3-i)}; \\ p_{23} &= \sum_{i=0}^{1} C_3^i (p_{11} + p_{12}) p_{13}^{(3-i)}. \end{aligned}\right\} \qquad (14)$$

Для того чтобы показать преимущества принципа вложенных модулей для системы $L_{18,\,27}$ (рис. 7), приведем функциональную схему альтернативной системы $L_{18,\,27a}$ (рис. 8).

На основании общих формул (6) легко вывести формулы, определяющие показатели качества, а именно, вероятность p_{21a} достоверного обнаружения, вероятность p_{22a} необнаружения и вероятность p_{23a} ложной тревоги:

$$\left.\begin{aligned} p_{21a} &= k - p_{22a} - p_{23a}; \\ p_{22a} &= \sum_{i=0}^{9} C_{27}^i (a + b)^i d^{(27-i)}; \\ p_{23a} &= \sum_{i=0}^{9} C_{27}^i (a + d)^i b^{(27-i)}. \end{aligned}\right\} \qquad (15)$$

Анализируя схему, приведенную на рис. 7, для вложенных модулей $L_{18,\,27}$, несложно вывести формулу, аналогичную выражению (12а), для определения числа K_{u2} схем И, применяемых в схеме:

$$K_{u2} = m_0 K_{u1} C_{n_0}^{m_0} + C_{n_0}^{m_0}, \qquad (16)$$

где $m_0 = 2$; $n_0 = 3$; $K_{u1} = 21$; при этом $K_{u2} = 129$.

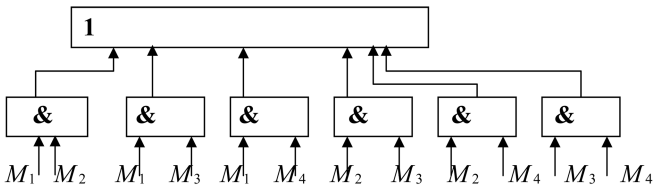


Рис. 9. Функциональная схема вложенного модуля $N_{12,\ 24}$

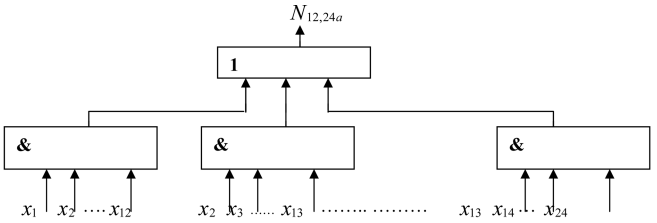


Рис. 10. Функциональная схема альтернативного варианта параллельного резервирования с мажоритарным принципом "12 из 24*a*"

Число K_{u2} схем И для альтернативного варианта $L_{18,\ 27a}$ (рис. 8) определяется числом сочетаний из 27 по 18 $K_{au2} = C_{27}^{18} = 246\ 675$.

Сравнительный коэффициент $V_{18,\ 27}$, определяемый формулой (13), показывает ошеломляющее преимущество принципа вложенных модулей перед альтернативным вариантом:

$$V_{18,\ 27} = \frac{C_{27}^{18}}{m_0 K_{u1} C_{n_0}^{m_0} + C_{n_0}^{m_0}} = \frac{246\ 675}{129} \approx 1912{,}3.$$

Таким образом, соединительные связи для схемы, приведенной на рис. 7, проще почти в 2000 раз по сравнению со схемой, представленной на рис. 8, которую практически почти невозможно реализовать.

Вернемся теперь к информационному модулю $M_{2,\ 4}$, и на его основе создадим вложенный модуль $N_{12,\ 24}$, который реализует мажоритарный принцип "12 из 24" и функциональная схема которого приведена на рис. 9.

Основные показатели качества информационного модуля $N_{12,\ 24}$, а именно, вероятность p_{11} достоверного обнаружения контролируемого события, вероятность p_{12} необнаружения и вероятность p_{13} ложной тревоги можно определить как функции от соответствующих вероятностей p_{01}, p_{02}, p_{03}, которые вычисляются с помощью формул (3).

Расчетные формулы для определения вероятностей p_{11}, p_{12} и p_{13} можно составить на основании выражений (3):

$$\left.\begin{aligned} p_{11} &= k - p_{12} - p_{13};\\ p_{12} &= \sum_{i=0}^{2} C_4^i (p_{01} + p_{03})^i p_{02}^{(4-i)};\\ p_{13} &= \sum_{i=0}^{2} C_4^i (p_{01} + p_{02})^i p_{03}^{(4-i)}. \end{aligned}\right\} \qquad (18)$$

Для того чтобы ярче выделить преимущества метода вложенных модулей, приведем альтернативный вариант модуля $N_{12,\ 24a}$, реализованный без принципа вложенных модулей и приведенный на рис. 10.

На основании общих формул (6) можно вывести формулы, определяющие показатели качества модуля $N_{12,\ 24a}$, а именно, вероятность p_{11a} достоверного обнаружения контролируемого явления, вероятность p_{12a}, необнаружения и вероятность p_{13a}, ложной тревоги:

$$\left.\begin{aligned} p_{11a} &= 1 - p_{12a} - p_{13a};\\ p_{12a} &= \sum_{i=0}^{12} C_{24}^i (a + b)^i d^{(24-i)};\\ p_{13a} &= \sum_{i=0}^{12} C_{24}^i (a + d)^i b^{(24-i)}. \end{aligned}\right\} \qquad (19)$$

Число K_{u1} применяемых схем И для соединения определяется на основании общей формулы (12):

$$K_{u1} = m_0 C_{n_0}^{m_0} C_{n_0}^{m_0} + C_{n_0}^{m_0}. \qquad (20)$$

Если подставить в формулу (20) значения $m_0 = 2$, $n_0 = 4$, то получим $K_{u1} = 78$, т. е. для реализации информационного вложенного модуля $N_{12,\ 24}$ (см. рис. 9) потребуется 78 схем И для обеспечения информационных связей с заданным мажоритарным принципом "12 из 24".

Для создания функциональной схемы альтернативного варианта $N_{12,\ 24a}$ (рис. 10) потребуется число K_{u1a} схем И, определяемое числом сочетаний из 24 по 12: $K_{u1a} = C_{24}^{12} = 2\ 704\ 156$.

Совершенно очевидно, что технически такую схему нет смысла создавать. Таким образом, вложенный информационный модуль $N_{12,\ 24}$ (см. рис. 9) обеспечивает фантастический экономический выигрыш V в требуемом числе соединительных схем И, а именно:

$$V = \frac{C_{24}^{12}}{2 C_4^2 C_4^2 + C_4^2} = \frac{2\ 704\ 156}{78} = 34668{,}6.$$

Следовательно, функциональная схема информационного модуля $N_{12,\ 24}$ (см. рис. 9) будет примерно в 35 000 раз дешевле и проще, чем схема с аналогичными функциональными характеристиками, приведенная на рис. 10.

На основании модуля $M_{3,\ 4}$ можно составить вложенный информационный модуль $N_{9,\ 12}$ (рис. 11), в котором вместо первичных КДИ x_i на входах подключены модули $M_{3,\ 4}$. Такой вложенный модуль $N_{9,\ 12}$ объединяет 12 первичных КДИ и реализует мажоритарный принцип принятия решения "9 из 12".

Расчетные формулы для определения вероятностей p_{11}, p_{12}, p_{13} для модуля $N_{9,\,12}$ можно составить на основании выражений (6):

$$\left.\begin{aligned} p_{11} &= k - p_{12} - p_{13}; \\ p_{12} &= \sum_{i=0}^{2} C_4^i (p_{01} + p_{03})^i p_{02}^{(4-i)}; \\ p_{13} &= \sum_{i=0}^{2} C_4^i (p_{01} + p_{02})^i p_{03}^{(4-i)}. \end{aligned}\right\} \qquad (21)$$

Можно показать, что число K_{u1} схем И для модуля $N_{9,\,12}$ равно $K_{u1} = 52$. Число K_{u2} схем И для альтернативного варианта равно числу сочетаний 12 из 9 и равно $K_{u2} = C_{12}^9 = 220$. Таким образом, экономический выигрыш W составит

$$W = \frac{K_{u2}}{K_{u1}} = \frac{220}{52} = 4{,}2.$$

В соответствии с приведенными расчетными формулами проведены расчеты зависимостей вероятностей $p_1(a)$ верного обнаружения контролируемого явления в зависимости от вероятности a верного сообщения для одного отдельного КДИ для приведенных функциональных схем параллельного информационного резервирования методом вложенных модулей. Результаты расчета показаны на графиках (рис. 12).

Анализ приведенных графиков $p_1 = f(a)$ (рис. 12) позволяет сделать следующие выводы:

- информационный модуль $M_{3,\,4}$ является наиболее эффективным по сравнению с модулями $M_{2,\,3}$, $M_{2,\,4}$, $M_{3,\,5}$ как по своим функциональным возможностям, определенным высокими вероятностями $p_1 = f(a)$, так и по простоте технической реализации схемы;
- функциональная эффективность вложенного модуля $M_{9,\,12}$ существенно выше по сравнению с вложенными модулями $M_{6,\,9}$, $M_{12,\,24}$;
- эффективность модульной системы особенно высока при низких функциональных качествах, определяемых вероятностью a верного обнаружения контролируемого явления для отдельного КДИ.

Для того чтобы более выразительно оценить эффективность отдельных модулей $M_{x,\,y}$, на графиках, представленных на рис. 13, приведены значения коэффициентов эффективности $Э_i$, определяющих отношения вероятности $p_1 = f(a)$ верного обнаружения контролируемого явления для различных модулей в зависимости от их функциональной характеристики — вероятности a верного обнаружения контролируемого явления одним отдельным КДИ.

В соответствии с приведенным графиком (рис. 13) можно сделать следующие выводы:

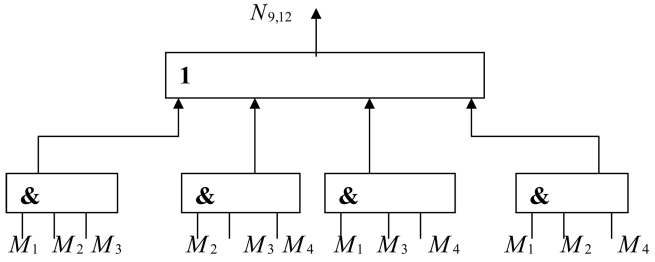


Рис. 11. Функциональная схема информационного модуля $N_{9,\,12}$

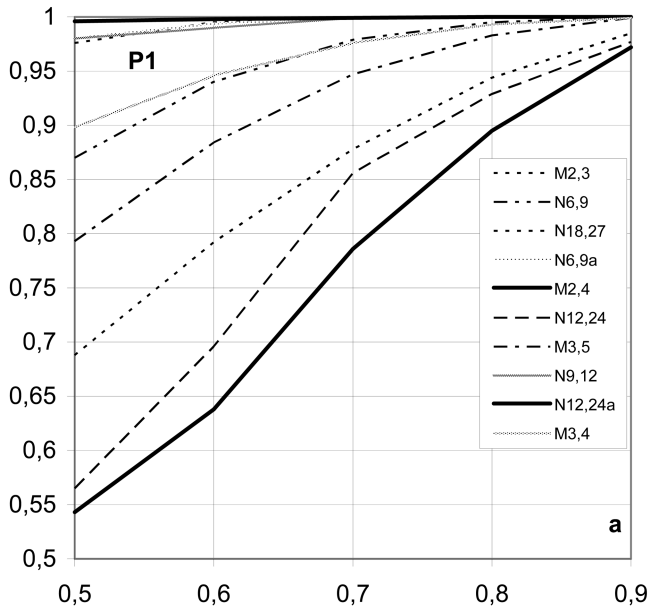


Рис. 12. Зависимости $p_1 = f(a)$ для модулей $M_{2,\,3}$; $N_{6,\,9}$; $N_{6,\,9a}$; $N_{18,\,27}$, $M_{2,\,4}$, $M_{3,\,4}$, $M_{3,\,5}$, $N_{9,\,12}$; $N_{12,\,24}$; $N_{12,\,24a}$

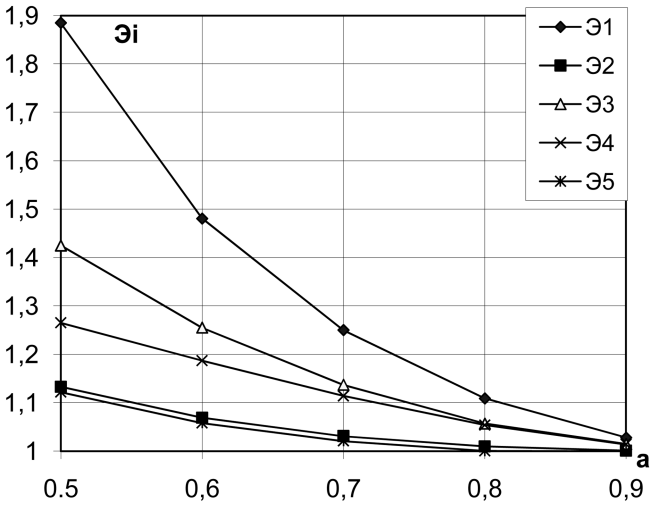


Рис. 13. Зависимости $Э_1 = \frac{p_{1(3,\,4)}}{p_{1(12,\,24)}}$, $Э_2 = \frac{p_{1(3,\,4)}}{p_{1(3,\,5)}}$, $Э_3 = \frac{p_{1(6,\,9)a}}{p_{1(2,\,3)}}$, $Э_4 = \frac{p_{1(6,\,9)}}{p_{1(2,\,3)}}$, $Э_5 = \frac{p_{1(18,\,27)}}{p_{1(6,\,9)}}$ вложенных модулей

- информационный модуль $M_{3,\,4}$ обладает наиболее высокой эффективностью по сравнению с модулями $M_{2,\,3}$, $M_{2,\,4}$, $M_{3,\,5}$;
- с повышением степени иерархического вложения при построении модулей $M_{6,\,9}$, $M_{18,\,27}$ их эффективность снижается;
- предложенная система вложенных модулей повышает качество контроля при низком функциональном качестве исходных КДИ.

Анализ полученных результатов исследования позволяет сделать следующие выводы:

- параллельное информационное резервирование, реализуемое способом вложенных модулей, экономически существенно более эффективно, чем альтернативный вариант такого же резервирования, но без принципа вложенных модулей;
- принцип вложенных модулей существенно повышает техническую надежность устройств параллельного подключения большого числа первичных источников информации;
- как показывают числовые результаты исследования, наиболее эффективными информационными модулями являются модуль $M_{3,\,4}$, вложенный модуль $N_{9,\,12}$; наименее эффективным являются модуль $M_{2,\,4}$, вложенный модуль $N_{12,\,24}$;
- эффективность модульного резервирования существенно повышается при низких вероятностных характеристиках первичных источников информации;
- эффективность применения принципа вложенных модулей уменьшается с увеличением числа иерархических структур вложенных модулей.

Список литературы

1. **Аль-Аммори Али.** Вероятностный способ обеспечения эффективности информационных систем // Управление проектами, системный анализ и логистика. К.: НТУ. 2006. Вып. 3. С. 178—180.
2. **Аль-Аммори Али.** Исследование влияния реальной технической надежности на эффективность информационного резервирования // Искусственный интеллект. 2006. № 4. С. 660—663.
3. **Аль-Аммори Али.** Экономические условия повышения достоверности данных в информационно-управляющих системах // Транспорт. Наука, Техника, Управление. М.: ВИНИТИ. 2006. № 12. С. 37—39.
4. **Абезгаус Т. Т., Тронь А. П. и др.** Справочник по вероятностным расчетам. М.: Воениздат, 1989. 656 с.
5. **Вентцель Е. С., Овчаров Л. А.** Теория вероятности и ее инженерные приложения. М.: Наука, 1988. 480 с.

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

УДК 004.8

Е. В. Щуревич, аспирант,
Алтайский государственный технический университет им. И. И. Ползунова
E-mail: scey.barn@gmail.com

Кластеризация знаний в системах искусственного интеллекта

Рассмотрены проблемы автоматической обработки текстов на естественном языке, предложено несколько вариантов проведения кластеризации знаний, а также даны сравнение и анализ их результатов. Описанные методики могут применяться на различных этапах жизненного цикла интеллектуальных систем, основанных на знаниях.

Ключевые слова: *база знаний, кластеризация, муравьиные алгоритмы, нейронные сети, визуализация знаний.*

Введение

В последнее время создается все больше интеллектуальных систем, основанных на знаниях и реализующих функции от поиска информации до принятия решений. Объем информации, которыми оперируют такие системы, очень велик, что усложняет и делает ресурсоемким контроль над формированием и развитием их баз знаний.

В данной работе представлены модели кластеризации базы знаний интеллектуальных систем, позволяющие упростить анализ качества сформированной базы.

Рассмотрим интеллектуальную систему, которая создавалась в предположении, что все множество фраз естественного языка представимо в виде суперпозиции понятий [1]. Элементарными структурными единицами знаний в этой системе являются *понятия*, объединяющие в себе слова, схожие по смыслу.

Перевод фраз естественного языка в вид суперпозиций понятий позволяет выделить *связи* между понятиями. Так, например, фраза "в вазе стоит букет" представима в виде суперпозиции

стоять (*букет*, *ваза*),

которая дает нам две связи:

— стоять — букет (функциональная связь) и

— стоять — ваза (пространственная связь).

Близость понятий, между которыми существует связь, определяется *повторность*ю этой связи — числом встречающихся в тексте суперпозиций, реализующих данную связь. Исходя из этого,

можно определить понятие *длины связи* — величины, обратно пропорциональной повторности:

$$L = \frac{R_{\max} - R + 1}{R_{\max}},$$

где L — длина связи; R — повторность связи; $R_{\max}$ — максимальная повторность в базе знаний.

Таким образом, длина любой связи будет находиться в промежутке $[1/R_{\max}, 1]$.

Однако знания не существуют обособленно, они объединены в некие блоки, представляющие собой те или иные знания о мире. Назовем такие блоки *кластерами*.

В данной статье мы опишем и проанализируем несколько вариантов кластеризации знаний на основе текстов на естественном языке.

Модель кластеризации

Для разложения текста естественного языка на суперпозиции в целях извлечения понятий и связей между ними мы использовали разработки проекта AOT.ru [2]. Понятия и связи из суперпозиций составляют базу знаний системы.

Далее к полученной базе знаний мы применяем "муравьиный" алгоритм. Муравьиные алгоритмы — метод оптимизации, базирующийся на моделировании поведения колонии муравьев.

Основу "социального" поведения муравьев составляет *самоорганизация* — множество динамических механизмов, обеспечивающих достижение системой глобальной цели в результате низкоуровневого взаимодействия ее элементов. Принципиальной особенностью такого взаимодействия является использование элементами системы *только локальной информации*. При этом исключается любое централизованное управление и обращение к глобальному образу, репрезентирующему систему во внешнем мире. Самоорганизация является результатом взаимодействия следующих четырех компонентов:

- случайность;
- многократность;
- положительная обратная связь;
- отрицательная обратная связь.

Для применения муравьиных алгоритмов к представленной задаче необходимо соответствующим образом уточнить реализацию четырех составляющих самоорганизации муравьев. В данной работе было предложено конкретизировать алгоритм следующим образом.

Многократность взаимодействия реализуется итерационным движением по графу взаимосвязей базы знаний одновременно несколькими муравьями. Каждый муравей начинает маршрут из своей точки — понятия базы знаний.

Положительная обратная связь реализуется как имитация поведения муравьев типа "оставление следов — перемещение по следам". Для данной задачи это поведение подчиняется следующему стохастическому правилу: вероятность выбора ребра в качестве следующего на пути пропорциональна количеству феромона на нем.

Применение такого вероятностного правила обеспечивает реализацию и другой составляющей самоорганизации — *случайности*. Количество откладываемого муравьем феромона на ребре графа связности обратно пропорционально длине маршрута.

Использование только положительной обратной связи приводит к случаю, когда все муравьи двигаются одним и тем же маршрутом. Во избежание этого используется *отрицательная обратная связь* — испарение феромона. Время испарения не должно быть слишком большим, так как при этом возникает опасность сходимости популяции маршрутов к одному. Вместе с тем, время испарения не должно быть и слишком малым, так как это приводит к быстрому "забыванию", потере памяти колонии и, следовательно, к некооперативному поведению муравьев.

Пусть для каждого муравья переход из понятия i в понятие j зависит от двух составляющих: видимости и виртуального следа феромона.

Видимость — величина, обратная расстоянию:

$$\eta_{ij} = 1/D_{ij},$$

где D_{ij} — длина связи (расстояние) между понятиями i и j.

Видимость — это локальная статистическая информация, выражающая эвристическое желание посетить понятие j из понятия i: чем ближе понятие, тем больше желание посетить его.

Виртуальный след феромона на ребре (i, j) представляет подтвержденное муравьиным опытом желание посетить понятие j из понятия i. В отличие от видимости след феромона является более глобальной и динамической информацией — она изменяется после каждой итерации алгоритма, отражая приобретенный муравьями опыт. Количество виртуального феромона на ребре (i, j) на итерации t обозначим через $\tau_{ij}(t)$.

Важную роль в муравьиных алгоритмах играет вероятностно-пропорциональное правило, определяющее вероятность перехода k-го муравья из понятия j в понятие i на t-й итерации. В данном случае было предложено использовать следующую формулу для этого правила:

$$P_{ij,\,k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}},$$

где α и β — два регулируемых параметра, задающих веса феромона и видимости при выборе маршрута.

После прохождения ребра каждый муравей k откладывает на ребре (i, j) количество феромона $\Delta\tau_{ij,\,k}(t)$, обратно пропорциональное параметру Q.

Для исследования всего пространства решений необходимо обеспечить испарение феромона. Обозначим коэффициент испарения феромона через $p \in [0, 1]$. Тогда правило обновления каждого вида феромона примет следующий вид:

$$\tau_{ij}(t + 1) = (1 - p)\tau_{ij}(t) + \Delta\tau_{ij}(t);$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij,\,k}(t),$$

где m — число муравьев в колонии.

В начале работы алгоритма количество феромона каждого вида принимается равным небольшому положительному числу τ_0. Общее число муравьев в колонии остается постоянным.

В результате работы муравьиного алгоритма мы имеем граф связности базы знаний с некоторым количеством феромона на его ребрах. Будем использовать эти данные для кластеризации знаний.

Рассмотрим два варианта кластеризации:

- на основании длины связей;
- на основании количества феромона на связях.

Для обоих вариантов кластеризация начинается с выбора ребра с максимальным (или близким к максимальному) количеством феромона на нем — оно будет являться начальной точкой распространения очередного кластера. Ограничим начальное для кластера ребро количеством феромона $K\tau_{\max}$, где K — некоторый коэффициент. Также начальное ребро не должно принадлежать ни одному из уже определенных кластеров.

Понятия, которые связывает выбранное ребро, становятся элементами кластера. Дальнейшее распространение кластера происходит по ребрам, исходящим из первых двух понятий, причем включение очередной связи (и понятия на другом ее конце) в кластер зависит от варианта кластеризации. Для кластеризации *по длине связей* ограничением является суммарная длина связей от одного из двух начальных понятий до любого из понятий, включаемого в этот же кластер. Для кластеризации *по количеству феромона* ограничена сумма величин, обратно пропорциональных количеству феромона на ребрах, входящих в граф связей (чем больше количество феромона на связи, тем меньше включение ее в состав кластера приближает нас к границе).

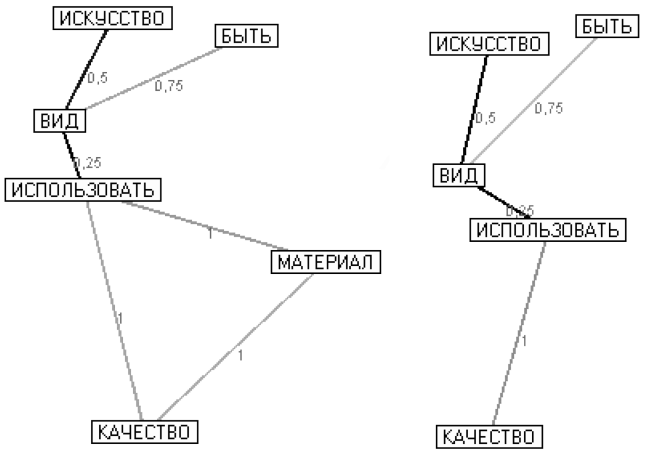
Сравнение алгоритмов кластеризации

Сравним результаты описанных выше вариантов кластеризации на одинаковых коротких текстах. На рисунках, иллюстрирующих примеры, длина каждой связи обозначена числом, а количество феромона — цветом (чем больше феромона, тем темнее связь). Понятия, объединенные в результате кластеризации в один блок, изображаются одним цветом. Будем проводить кластеризацию для каждого варианта по нескольку раз, чтобы отследить изменчивость результатов под влиянием случайной составляющей муравьиных алгоритмов.

Рассмотрим пример энциклопедического текста, характеризующегося высокой сложностью используемых языковых структур.

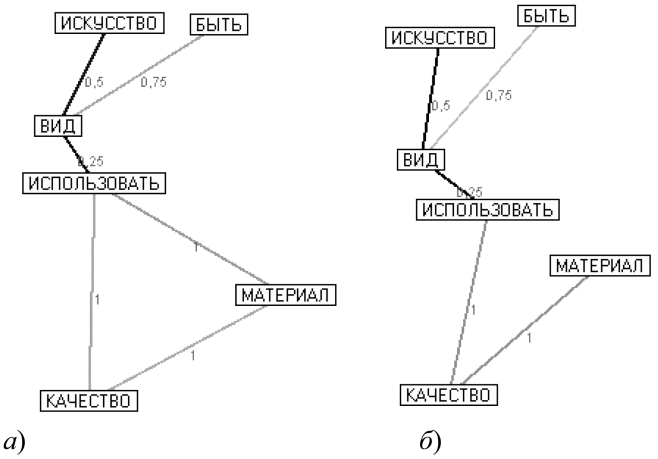
Пример 1

Художественная литература — вид искусства, использующий в качестве единственного материала слова и конструкции естественного языка. Специфика



а) *б*)

Рис. 1. Результат кластеризации примера 1 по длине связей:
а — вариант 1; *б* — вариант 2



а) *б*)

Рис. 2. Результат кластеризации примера 1 по количеству феромона:
а — вариант 1; *б* — вариант 2

художественной литературы выявляется в сопоставлении, с одной стороны, с видами искусства, использующими иной материал вместо словесно-языкового или наряду с ним, с другой стороны — с иными типами словесного текста: философским, публицистическим, научным и др. Кроме того, художественная литература, как и другие виды искусства, объединяет авторские произведения в отличие от принципиально не имеющих автора произведений фольклора.

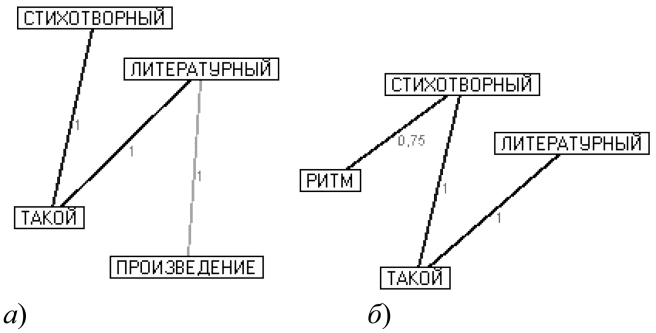
Очевидно, что на кластеризацию применительно к коротким текстам, когда база знаний имеет очень маленький размер, большое влияние оказывает случайность муравьиного алгоритма. Однако даже в этом случае результаты во многом схожи. В частности, основанием для выделения кластера послужила одна и та же связь, имеющая максимальную повторность (рис. 1, 2).

Попробуем рассмотреть текст такой же стилистики и на ту же тему, увеличив его объем.

Пример 2

Художественная литература — вид искусства, использующий в качестве единственного материала слова и конструкции естественного языка. Специфика художественной литературы выявляется в сопоставлении, с одной стороны, с видами искусства, использующими иной материал вместо словесно-языкового или наряду с ним, с другой стороны — с иными типами словесного текста: философским, публицистическим, научным и др. Кроме того, художественная литература, как и другие виды искусства, объединяет авторские произведения в отличие от принципиально не имеющих автора произведений фольклора. Прозаическим считается такой литературный текст, в котором отдельный, независимый от речевого ритм не вторгается в языковую ткань и не влияет на содержание. Известен, однако, целый ряд пограничных явлений: многие прозаики сознательно придают своим произведениям некоторые признаки стихотворности. О точных границах между прозой и поэзией не прекращается спор литературоведов разных стран на протяжении последнего столетия.

В общем случае, стихотворение — это литературное произведение, обладающее особой ритмической структурой, не вытекающей из естественного ритма языка. Характер этого ритма может быть различным в зависимости от свойств самого языка: так, для языков, в которых большое значение имеет различие гласных звуков по долготе, естественно возникновение стихотворного ритма, построенного на упорядочении слогов по признаку долготы и краткости, а для языков, в которых гласные различаются не долготой, а силой выдоха, естественно использование такого стихотворного ритма, который упорядочивает слоги по признаку ударности и безударности. Так возникают разные системы стихосложения.



а) *б*)

Рис. 3. Результат кластеризации примера 2:
а — по длине связей; *б* — по количеству феромона

Увеличение объема текста закономерно приводит к увеличению повторности связей, в результате чего объем кластеров уменьшается, и оба алгоритма кластеризации в большинстве случаев выдают практически идентичные кластеры (рис. 3).

Рассмотрим далее текст, взятый из детской энциклопедии.

Пример 3

Леса — самые важные экосистемы Земли. Лес — это природное сообщество, в котором живет много видов растений и животных. Различают леса таких видов: тайга, хвойные, смешанные, широколиственные, редколесье, субтропические, высокогорные и влажные тропические леса. В лесах обитает большая часть всех растений и животных суши. Леса не только обогащают воздух кислородом, они еще поглощают огромное количество газов и пыли. Кроме того, они дают кров многим животным. Лесами покрыто около 1/3 земной поверхности.

Такие тексты характеризуются меньшей сложностью языковых конструкций, а также более частым использованием синонимов для обозначения одних и тех же предметов. Результатом такой структуры текста является малая повторность связей между понятиями, в связи с чем, основная нагрузка на выявление кластеров ложится на муравьиные алгоритмы и, соответственно, увеличивается влияние стохастического фактора (рис. 4).

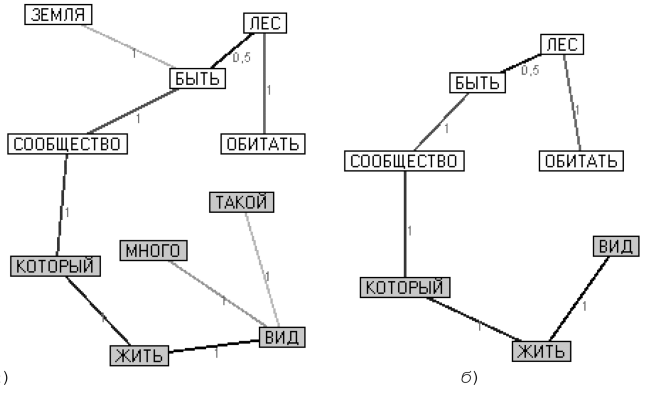


Рис. 4. Результат кластеризации примера 3:
а — по длине связей; *б* — по количеству феромона

Выводы

Проведенные эксперименты показали, что на коротких текстах различной стилистики стабильность результатов кластеризации составляет около 60 %. Мы считаем это хорошим показателем. Увеличение объема исследуемого текста приводит к повышению стабильности результатов, что позволяет сделать предварительный вывод о возможности проведения качественной кластеризации знаний на больших объемах данных. Полнота сформированной базы знаний позволит с высокой долей вероятности определять смысловое содержание коротких текстов на естественном языке путем проецирования их на пространство базы знаний.

Формирование кластеров, выражающих основной смысл текста, дает потенциальную возможность автоматического реферирования статей. Это порождает проблему преобразования понятий кластера в осмысленный текст, который должен строиться из понятий базы знаний на основании связей между ними. Мы предполагаем, что подобное реферирование станет возможным при введении различных типов связей и их направленности.

Список литературы

1. **Щуревич Е. В.** Моделирование и анализ знаний в системах искусственного интеллекта [Текст] / Е. В. Щуревич, Е. Н. Крючкова // Вестник Алтайского государственного технического университета им. И. И. Ползунова. 2006. № 2. С. 173—177.
2. **Автоматическая** обработка текста [Электронный ресурс]. — http://www.aot.ru.
3. **Лорьер Ж.-Л.** Системы искусственного интеллекта [Текст] / Ж.-Л. Лорьер. — М.: Мир, 1991. — 568 с.
4. **Сотник С. Л.** Основы проектирования систем искусственного интеллекта [Лекции]: Курс лекций / С. Л. Сотник. — 1997—1998.
5. **Штовба С. Д.** Муравьиные алгоритмы [Текст] / С. Д. Штобва // Exponenta Pro. — 2003. — № 4. — С. 70—75.
6. **Уинстон П.** Искусственный интеллект [Текст] / П. Уинстон. — М.: Мир, 1980. — 520 с.
7. **Щуревич Е. В.** Проблема визуализации и фильтрации знаний в системах искусственного интеллекта [Текст] / Е. В. Щуревич, Е. Н. Крючкова // Материалы конференции-конкурса "Технологии Microsoft в информатике и программировании". — Новосибирск, 2005. — С. 147—149.

УДК 529.58

Е. Г. Брындин, директор,
Исследовательский центр
"ЕСТЕСТВОИНФОРМАТИКА", Новосибирск
E-mail: bryndin@ngs.ru

Теоретические основы коммуникативно-ассоциативной имитации символически-языкового мышления

Мышление языковыми элементами знаний представляет собой коммуникативные процессы анализа, синтеза и расщепления предложений и суждений. Имитация мышления осуществляется на основе символически-языковой коммуникативно-ассоциативной логики над комбинаторно-расширяемым сетевым иерархическим коммуникативно-ассоциативным представлением предложениями и суждениями типовых информационных потребностей и их реализаций предметных областей знаний.

Ключевые слова: *символически-языковая имитация мышления, комбинаторно расширяемое сетевое иерархическое коммуникативно-ассоциативное представление типовой информационной потребности предложениями и суждениями, коммуникативно-ассоциативная логика реализации информационной потребности.*

Введение

Человечество приступило к созданию роботов. Наибольшие практические успехи демонстрирует Япония. Страна Восходящего Солнца собирается построить базу на Луне к 2025 г. и заселить эту базу передовыми версиями сегодняшних гуманоидных роботов. Первая деревня роботов должна появиться на Луне до 2014 г. В Японии действует программа *Humanoid Robotics Project* (HRP) стоимостью свыше 37 млн долл., предусматривающая создание серийно выпускаемых человекообразных роботов уже в ближайшие несколько лет.

Несмотря на практические достижения в области робототехники успехи в области имитации мышления пока еще малы. Специалисты по математической логике не могут решить эту проблему вследствие интерпретации семантики высказываний, которая нарушает коммуникативные и ассоциативные свойства языка. Лингвисты, которые занимались логическими проблемами естественного языка, остались далеки от понимания того, как в языке фиксируется логика рассуждений.

Основными препятствиями естественной имитации символического языкового мышления человека было, во-первых, отсутствие в языкознании системы понятий, характеризующих элементы знаний с точки зрения их соотнесенности с действительностью, во-вторых, не раскрытая

природа естественного представления знаний. Естественное представление знаний имеет важнейшее значение для имитации рассуждений, поскольку единицы знаний непосредственно участвуют в мыслительных процессах.

Мышление, осуществляемое на уровне смысловых языковых элементов знаний, представляет собой по существу коммуникативные процессы анализа, синтеза и расщепления, касающиеся слов, словосочетаний, предложений и суждений. На письменном уровне мышление осуществляется коммуникативными процессами с символическими языковыми элементами знаний. Имитация подражательного мышления на письменном уровне осуществляется на основе символически-языковой коммуникативно-ассоциативной логики и комбинаторно расширяемого сетевого иерархического коммуникативно-ассоциативного представления типовых информационных потребностей и их реализаций предметных областей знаний.

Имитация подражательного мышления реализуется на основе фундаментальных знаний:

- о языковом феномене, о природе и процессах символически-языкового мышления;
- о методологических основах имитации символически-языкового мышления и по символически-языковой коммуникативной логике;
- по структуре и функционированию роботов, имитирующих символически-языковое мышление и по реализации информационных потребностей человека роботом;
- по теоретическим основам построения программ с детерминированно-связанными модулями и по вычислительным процессам с непрерывной обработкой программ на виртуальной памяти.

Робот с символической языковой имитацией подражательного мышления, адаптивным поведением, имеющим дистанционную связь с человеком и суперкомпьютером, осуществляющим непрерывную обработку программ на виртуальной памяти с упреждающим замещением модулей программ типовых процедур, способен оперативно выполнять поручения человека во многих сферах жизнедеятельности. Символически мыслящие роботы имитируют мышление и формируют свое поведение в нормативной ситуации по информационной потребности человека для ее реализации. У такого робота поведение привязано к мышлению, что повышает автономность мобильных роботов. Робот работает круглые сутки и воспринимает знания несколько часов, которые человек усваивает десятилетиями. Дистанционное взаимодействие робота с человеком и суперЭВМ через спутниковый Интернет позволяет его использовать в космосе.

1. Символически-языковая коммуникативная логика

Информационной единицей языкового мышления и речевого общения человека является слово. Слово называет представителя либо видимого материального мира, либо невидимого рационального мира, либо невидимого духовного мира.

Языковая логика мышления — это процесс коммуникативного слияния на втором сигнальном уровне слов в словосочетания, словосочетаний в предложения, предложений в суждения. Коммуникативное слияние слов в словосочетания осуществляется по признаковой схеме первого сигнального уровня, словосочетаний в предложения — по ситуативно-признаковой схеме первого сигнального уровня, предложений в суждения — по ситуативной схеме первого сигнального уровня.

Коммуникативное слияние словосочетаний в предложения, предложений в суждения осуществляется по атрибутам информационной потребности человека первого и второго сигнального уровня. Полученные в результате коммуникативного слияния предложение и суждение являются реализацией информационной потребности.

1.1. Сущностно-ориентированный словарь

Пусть S — орфографический словарь, где $S = \{S_i\}$, S_i — морфологическое слово. Слово S_i называет признак Q_{ij} представителя M_{ij} из множества M_i, где $M_i = \{M_{ij}\}$. Обозначим лексическое значение слова S_i через $\{M_{ij}, Q_{ij}, S_i\}$. Связь лексических значений слов $\{S_i\}$ с элементами множества M_i зададим совокупностью признаковых отношений Q_i, где $Q_i = \{Q_{ij}, (M_i, M_{ij})\}$.

Множество лексических значений, связанных совокупностью признаковых отношений с представителями, является сущностно-ориентированным словарем. Слова в словаре снабжаются признаковыми индексами соответственно их признаковым отношениям с представителями. Сущностно-ориентированный словарь фиксирует признаковые сущности представителей. Словарь помогает употреблять слова со своим лексическим значением и различать представителей, которых они называют.

Слова употребляются ориентированно на основе признаковых индексов. Каждый признак имеет три индекса. Один индекс указывает на предметную область знаний, второй — на ситуацию, третий — на ситуативный момент. Слова с несколькими лексическими значениями имеют несколько наборов индексов. Каждый набор индексов определяет лексическое значение слова.

1.2. Признаки и грамматические виды

Представители реалии или абстракта имеют признаки изменения, пространства, объекта, субъекта, действия, качества, количества и т. д. Они выражаются в языке словами соответственно определенным грамматическим правилам.

Определение 1. Грамматическая характеристика множества слов категории признаков, морфологическое значение которых формируется по общим правилам грамматики, называется *грамматическим видом*.

Существительное, прилагательное, глагол, наречие, местоимение, причастие, числительное, деепричастие и междометие являются грамматическими видами русского языка.

1.3. Информационное семейство

Слова, которые образованы в языке от слова с главным значением, являются родственными. Для образования родственных слов служат приставка перед главным словом и суффикс после основы главного слова. От суффикса зависят окончания.

Определение 2. Множество слов одного грамматического вида, родственных слову с главным значением, называется *информационным родом*.

Определение 3. Два рода разного грамматического вида называются *родственными*, если их слова с главными значениями — родственные.

Определение 4. Множество родственных информационных родов называется *семейством*.

В семействе содержатся родственные слова разных грамматических видов. В семействах родственные слова упорядочены по видам и внутри вида.

1.4. Информационное сообщество

Определение 5. Множество слов, образующих словосочетания с заданным, называется его *информационным сообществом*.

Число представителей сообщества слова определяет его смысловую интенсивность.

1.5. Коммуникация словосочетаний в предложения

Теорема 1. Любое предложение можно сформировать по его синтаксической структуре из необходимого и достаточного набора словосочетаний путем морфологического, синтаксического и семантического их анализа и слияния.

1.6. Коммуникация предложений в суждения

Определение 6. Родственно сливаемые по содержанию предложения через словоформы семейств и ассоциативно сливаемые по содержанию предложения назовем *элементарным ассоциативным суждением*.

Определение 7. Последовательность предложений P_i, ..., P_n называется *ассоциативным суждением*, если для всех i, P_i, P_{i+1} есть элементарное ассоциативное суждение.

Теорема 2. Любое элементарное ассоциативное суждение C_0 можно сформировать по его синтаксической структуре CC из необходимых ситуативно-валентных предложений путем морфологического, синтаксического и семантического их анализа и слияния.

Теорема 3. Любое ассоциативное суждение C_0 можно сформировать по его синтаксической структуре CC из необходимых элементарных ассоциативных суждений путем морфологического, синтаксического и семантического их анализа и слияния.

1.7. Предметные области знаний

Предметные области знаний представляют собой сети ассоциативно связанных информационных потребностей и их реализаций из коммуникативно связанных символических языковых элементов (информационных следов коммуникативно-ассоциативного символически-языкового мышления). Реализация информационной потребности может быть сама информационной потребностью. Тогда ассоциативная ей сеть коммуникативно связанных элементов будет ее реализацией. Комбинированные информационные потребности состоят либо из вложенных информационных потребностей предметной области знаний, либо из последовательности информационных потребностей, либо из последовательности вложенных информационных потребностей. Комбинированные информационные потребности состоят либо из простых, либо из простых и комбинированных, либо из комбинированных. Информационные потребности предметной области знаний образуют комбинационно расширяемую сеть.

Совокупность элементов знаний с разметкой является базой знаний. Повелительное или вопросительное предложение базы знаний является информационной потребностью. Совокупность процедур реализации информационной потребности является базой умений.

Согласно теоремам 1—3, информационная потребность имеет реализацию в базе знаний и базе умений, если существует необходимый и достаточный набор элементов знаний в базе знаний и процедур реализации в базе умений.

Для информационной потребности, являющейся комбинацией информационных потребностей, для которых имеется реализация, существует необходимый и достаточный набор элементов знаний в базе знаний и процедур реализации в базе умений. Соответствующая комбинация необходимого и достаточного набора имеющихся реализаций будет реа-

лизацией комбинированной информационной потребности. Символически-языковая коммуникативная логика связывает реализуемые информационные потребности, т. е. истинные.

Пусть заданы:

- *CS* — сущностно-ориентированый словарь предметной области знаний;
- *BIP* — базовые информационные потребности предметной области знаний, составленные из слов словаря *CS*. Базовая информационная потребность является либо вопросным, либо повелительным предложением;
- *RBIP* — реализации базовых информационных потребностей;
- *PKIP* — правила комбинирования информационных потребностей из базовых;
- *KKASRIP* — конструктор коммуникативно-ассоциативной сети реализаций информационных потребностей пользователя;
- *KASRIP*(*b*) — коммуникативно-ассоциативная сеть реализаций базовых информационных потребностей предметной области знаний.

Пусть пользователем задана *IP*(*i*), составленная по правилам *PKIP*. Применим к ней конструктор *KKASRIP*. Получим *KASRIP*(*i*)-расширение сети *KASRIP*(*b*) с реализацией *IP*(*i*).

Реализация вопросной информационной потребности в коммуникативно-ассоциативной сети предметной области знаний осуществляется следующим образом. Пусть предложение *P1* с разметкой является вопросной информационной потребностью с ситуативно-признаковой схемой *SP1* из ситуативной схемы *S1*, вопросное слово в предложении *P1* к неописанному признаку *Q1* лексического значения *LZ1* информационной ниши *IH1*.

Нужно найти в коммуникативно-ассоциативной сети предметной области знаний предложение *P2*, содержащее словосочетания и лексические значения предложения *P1* и лексическое значение *LZ2*, являющееся описанием признака *Q1*.

Процедура реализации строит по предложению *P1* шаблон *P2Ш* с неописанным признаком *Q1* предложения *P2*. По шаблону *P2Ш*, его разметке, *SP1*, *S1* процедура реализации ищет предложение *P2* в сети коммуникативных словосочетаний предметной области знаний с помощью процедур морфологического, синтаксического и семантического анализа. Найденное предложение *P2* будет реализацией.

Повелительная информационная потребность заменяется эквивалентной вопросной информационной потребностью.

Вывод показывает, что символически-языковая коммуникативная логика алгоритмически разрешима.

2. Модель символически мыслящего робота

Методология естественной имитации символического языкового мышления и символически-языковая коммуникативная логика позволяют создать модель символически мыслящего робота [1—3]. Модель символически мыслящего робота имеет систему ввода информационных потребностей, систему реализации информационных потребностей (систему имитации мышления), нейросетевую систему оречевления (синтеза речи) реализации информационной потребности. Система реализации информационной потребности содержит систему усвоения новых знаний, систему обучения, базу знаний, базу умений, нейросетевую систему чтения, печатающую систему и систему графического отображения.

Информационной единицей общения между роботом и собеседником является информационная потребность либо речевая, либо письменная на естественном функциональном языке. Собеседник использует информационные потребности, которые содержатся в базе знаний робота. Он общается с роботом с помощью комбинаций имеющихся информационных потребностей, обогащая, тем самым, робота новыми информационными потребностями и их реализациями. Базовые информационные потребности и их реализации робот получает во время его обучения.

Система реализации информационной потребности использует сущностно-ориентированные словари предметных областей базы знаний, процедуры анализа, слияния (синтеза) и расщепления из базы умений, а также сеть поэлементной реализации информационных потребностей. Робот реализует информационную потребность, если база знаний содержит необходимый набор элементов знаний, а база умений — необходимый набор элементов реализации.

Имитация символического языкового мышления осуществляется на основе символически-языковой коммуникативной логики над символическими языковыми элементами знаний предметных областей. Предметные области знаний представляют сети коммуникативно и ассоциативно связанных символических языковых элементов с ситуативно-признаковой и языковой разметкой.

Определение 8. Сеть $G(Xt, \Gamma t, Vt)$ является концептуальным представлением знаний, если $Xt = x_t^1 \cup x_t^2 \cup x_t^3$, где x_t^1 — множество вершин суждений с ситуативной и языковой разметкой; x_t^2 — множество вершин предложений, имеющих языковую и ситуативно-признаковую разметку: {(лексическое значение, вид)}, (признак, ниша)}; x_t^3 — множество вершин словосочетаний с признаковой и языковой разметкой; Vt — множество коммуникативных связей элементов множества

Xt; Г*t* — множество ассоциативных связей элементов множества *Xt*; *t* — дискретные моменты появления новых элементов знаний; признак — признак представителя реалии, указывающий на смысловое употребление слова в речи и письменно; вид — грамматическая характеристика множества слов, морфологическое значение которых формируется по общим правилам грамматики; ниша — единое ситуативное использование множества слов различных видов в различных синтаксических структурах предложений (ниша может быть субъектно-объектной, процессуальной, обстоятельством, определительной, дополнительной).

Ассоциативные связи соединяют вершины, которые являются информационной потребностью и ее реализацией. Коммуникативные связи исходят из вершин коммуникативных элементов знаний в общую вершину.

Реализации информационной потребности формируются процедурами семантического синтаксического и морфологического анализа, слияния и расщепления элементов информационной потребности пользователя, слов сущностно-ориентированного словаря и элементов знаний предметной области и процедурами реализации информационных действий информационной потребности. Процедуры содержатся в базе умений.

Робот имитирует подражательное мышление. По новым информационным потребностям робот выдает версию реализации, если у него достаточно имеющихся коммуникативных и ассоциативных элементов знаний и процедур реализации.

2.1. Реализация информационной потребности

По информационной потребности пользователя ищется ее реализация в сети коммуникативных и ассоциативных символических языковых элементов предметной области.

Например, пусть задана повелительная информационная потребность "Сформулировать теорему Пифагора" (рис. 1). По лексическому значению слова "формулировка" находим через сущностно-ориентированный словарь представительное словосочетание "формулировка теоремы" в сети предметной области знаний, указанной в разметке информационной потребности. Затем по дереву коммуникативных словосочетаний с лексическими значениями информационной потребности путем их синтеза добираемся до ассоциативной вершины "формулировка теоремы Пифагора". Далее по ассоциативной связи от ассоциативной вершины информационной потребности переходим к ассоциативной вершине реализации "квадрат гипотенузы равен сумме квадратов катетов".

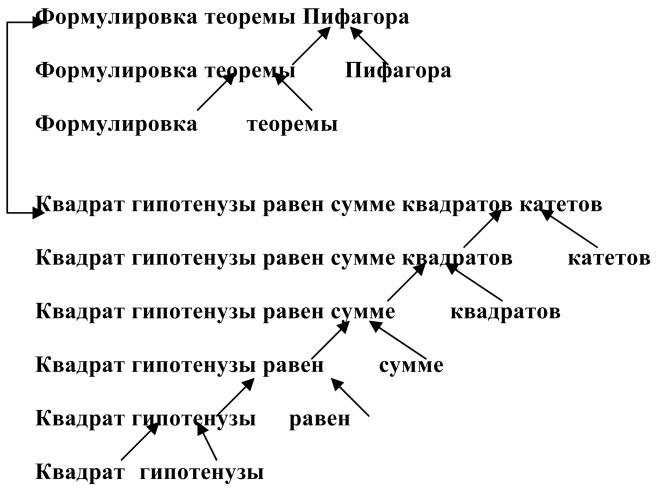


Рис. 1

Реализация информационной потребности сама может быть информационной потребностью. В этом случае по коммуникативному дереву лексических значений формулировки теоремы путем их синтеза добираемся до ассоциативной вершины. Затем по ассоциативной связи определяем реализацию "Формулировка теоремы Пифагора".

Если информационная потребность или реализация информационной потребности содержит информационное действие, тогда запускается типовая процедура реализации. В разметке информационного действия указывается ссылка на процедуру реализации.

Например, пусть задана повелительная информационная потребность "найти сумму *X* плюс *Y* при *X* равно 15, *Y* равно 10" (рис. 2). "*X* равно 15, *Y* равно 10" является информационным условием. Коммуникативные словосочетания "*X* равно" и "равно 15" в информационном условии информационного действия "плюс" инициирует замену *X* на 15. По информационному условию "*X* равно 15, *Y* равно 10" произойдет замена *X* на 15, а *Y* на 10 в типовой реализации информационной потребности. Лексическое значение информационного действия "плюс" в сущностно-ориентированном символическом языковом словаре содержит ссылку на аналитический знак "+" в аналитическом

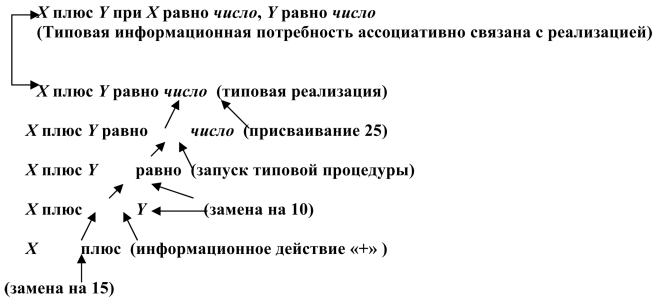


Рис. 2

сущностно-ориентированном словаре. Знак "+" в аналитическом сущностно-ориентированном словаре имеет ссылку на типовую процедуру сложения двух аргументов.

Реализация больших типовых процедур осуществляется на суперкомпьютере вычислительными процессами с непрерывной обработкой программ на виртуальной памяти с упреждающим замещением модулей программ типовых процедур [4—5].

Робот с символической языковой имитацией подражательного мышления, адаптивным поведением, имеющим дистанционную связь с человеком и суперкомпьютером, осуществляющим непрерывную обработку программ на виртуальной памяти с упреждающим замещением модулей программ типовых процедур, способен выполнять поручения человека во многих сферах жизнедеятельности. Это повышает автономность и мобильность роботов.

Список литературы

1. **Брындин Е. Г.** Природа информационной деятельности. Новосибирск: Наука, СО, 2001. 80 с.
2. **Брындин Е. Г.** Символически-мыслящие информационные роботы. Новосибирск: Наука, СО, 2002. 50 с.
3. **Брындин Е. Г.** Взаимодействие символически-мыслящего робота с человеком и внешней средой // Информационные технологии, 2004. № 6. С. 2—8.
4. **Брындин Е. Г.** Символически-языковая коммуникативно-ассоциативная технология подражательного мышления // Сб. трудов 4-й Междунар. науч.-практ. конф. "Высокие технологии, фундаментальные и прикладные исследования, образование". СПб.: Изд-во политехн. ун-та, 2007. Т. 11. С. 442—444.
5. **Брындин Е. Г.** Теоретические основы имитации мышления и непрерывной обработки на виртуальной памяти. Новосибирск: ИЦЕ, 2008. 257 с.
6. **Малинецский Г. Г.** Робототехника, прогноз, программирование. М.: ЛКИ, 2008.
7. **Сотник С. Л.** Конспект лекций по курсу "Основы проектирования систем искусственного интеллекта", 1997—1998. http://www.iskint.ru/?xid = books/sotnik
8. **Корендясев А. И.** Теоретические основы робототехники. В 2 кн. М.: Наука, 2006.
9. **Люгер, Джордж, Ф.** Искусственный интеллект: стратегии и методы решения сложных проблем. 4-е изд. М.: Издат. дом "Вильямс". 2003.
10. **Шамис А. Л.** Поведение, восприятие, мышление. М.: URSS, 2004.
11. **Шамис А. Л.** Пути моделирования мышления. М.: URSS, 2006.
12. **Joss Earl.** A Model of Consciousness. Электронный проект Programmer Stone, 2000, http://www.euro.ru/~programmerstone.
13. **Newpoisk** "Интегральная теория создания искусственного интеллекта", 2005, http://newpoisk.narod.ru/it.zip
14. **Эскизный** проект практической реализации интеллекта на базе данной теории, 2007, http://newpoisk.narod.ru/eskiz.zip

---

УДК 004:65

О. Н. Долинина, канд. техн. наук, доц., зав. каф.,
А. К. Кузьмин, аспирант,
Саратовский государственный
технический университет
E-mail: odolinina@rambler.ru

Применение методов технической диагностики для отладки баз знаний нейросетевых экспертных систем

Проведен анализ ошибок, возникающих в базах знаний экспертных систем, проанализированы существующие методы их отладки. Описан подход к решению задачи генерации исчерпывающих тестовых наборов, основанный на преобразовании структуры представления знаний экспертной системы к виду логической сети и применении к ней методов технической диагностики. Предложен метод тестирования нейросетевых экспертных систем, базирующихся на распространении данного подхода на модель многослойного персептрона.

Ключевые слова: *экспертные системы, нейронные сети, базы знаний, ошибки в базе знаний, отладка, тестовое множество.*

В последнее время все большую популярность приобретают экспертные системы (ЭС), в которых в качестве базы знаний (БЗ) и ядра принятия решений используются нейронные сети (НС). К большинству таких программных продуктов предъявляются повышенные требования к качеству принимаемых решений. В литературе приводится деление таких систем на следующие три класса [1]:

1. Системы, используемые для решения важных ответственных задач (**mission critical systems**). В военных приложениях функционирование таких систем чаще всего связано с выполнением таких функций, сбой которых может привести к провалу важных миссий. В промышленности подобные системы управляют процессами, безаварийное выполнение которых является важнейшим условием для организации функционирования всего производства.

2. Системы, чье функционирование тесно связано с обеспечением безопасности (**safety critical systems**). В качестве примера можно привести адаптивные системы, управляющие дорожным движением, или промышленные системы контроля опасных производственных процессов. Сбой в работе таких систем может привести к травмированию людей или их смерти.

3. Системы, связанные со здоровьем людей (**health critical systems**). В медицинских приложениях нейросетевых технологий сбой в работе по-

следних может послужить причиной неверной диагностики и, как следствие, ошибочно назначенного лечения. Это влечет за собой причинение вреда здоровью пациента или его смерть.

Рассмотрим структурные особенности нейросетевого механизма принятия решений. В теории нейроинформатики разработано множество концепций НС, однако для использования в качестве основы ЭС наибольшей популярностью пользуются многослойные персептроны [2, 3]. Это связано с тем, что большинство задач, решаемых ЭС, сводятся к задаче классификации входных векторов по выходным группам $[y_1, y_2, ...y_n]$. Совокупность входных векторов может быть представлена в виде матрицы $\begin{bmatrix} x_{11} & \dots & x_{m1} \\ \vdots & \ddots & \vdots \\ x_{m1} & \dots & x_{mn} \end{bmatrix}$, где m — число входных признаков; n — число объектов, подлежащих классификации. При этом каждая строка матрицы представляет собой входной вектор системы.

Многослойный персептрон представляет собой слоистую структуру, где в качестве элементов слоев выступают формальные нейроны [2, 3]. Каждый нейрон получает на вход взвешенную сумму значений выходов нейронов предыдущего слоя и применяет к ней некоторую нелинейную функцию активации. В качестве такой функции чаще всего выступает сигмоид вида

$$f(x) = \frac{1}{1 + \mathbf{e}^{-xa}}, \qquad (1)$$

где a — характеристика сигмоида, принимает положительные значения. Чем больше параметр a, тем больше нелинейность, вносимая функцией активации в систему; при $a = 0$ сигмоид вырождается в прямую. При функционировании сети сначала входной вектор подается на первый сенсорный слой. Затем послойно, начиная с первого ассоциативного слоя, рассчитываются состояния нейронов по формулам

$$Sum = \sum_i w_i x_i; \qquad (2)$$

$$Out = f(Sum). \qquad (3)$$

При этом выходы нейронов одного слоя поступают на входы нейронов следующего слоя. Состояния нейронов последнего слоя являются ответом сети на данный входной вектор. Алгоритм принятия решений данной системы задан неявно в виде совокупности K весовых коэффициентов связей между нейронами:

$$K = \begin{bmatrix} \{w(2)_1^1, ..., w(2)_{q(1)}^{q(2)}\}_1 & \dots & \{w(l)_1^1, ..., w(l)_{q(l-1)}^{q(1)}\}_1 \\ \vdots & \ddots & \vdots \\ \{w(2)_1^1, ..., w(2)_{q(1)}^{q(2)}\}_{q(2)} & \dots & \{w(l)_1^1, ..., w(l)_{q(l-1)}^{q(l)}\}_{q(l)} \end{bmatrix}. \qquad (4)$$

Здесь $q(x)$ — число нейронов в слое x; l — число слоев в сети; $w(k)_j^i$ — весовой коэффициент связи между нейроном i слоя k и нейроном j слоя $(k-1)$. Мощность P множества весовых коэффициентов находится по формуле

$$P = \sum_{i=2}^{l} q(i) \cdot q(i-1). \qquad (5)$$

Весовые коэффициенты связей нейронов рассчитываются автоматически в ходе обучения НС. Для этого необходимо наличие *обучающего множества*, под которым понимают известную совокупность $S(X, Y)$ пар входных и соответствующих выходных векторов системы. Здесь

X — множество входных векторов системы

$\{x_{11}, x_{21}, ..., x_{m1}, x_{12}, x_{22}, ..., x_{m2}, ..., x_{mn}\}$; (6)

Y — множество выходных векторов системы

$\{y_{11}, y_{21}, ..., y_{m1}, y_{12}, y_{22}, ..., y_{m2}, ..., y_{mn}\}$;

m — размер входных векторов (число входных признаков);

k — размер выходных векторов (число выходных признаков);

n — мощность обучающего множества (число обучающих примеров).

Введем понятие базы знаний (БЗ) нейросетевой ЭС. В классической теории ЭС под БЗ понимают некоторый набор известных фактов с правилами вывода новых фактов. Распространив это определение на нейросетевые ЭС, будем считать, что база знаний представляет собой набор входных признаков и конфигурацию нейросетевого механизма принятия решения. Набор входных признаков формируется экспертными методами и включает в себя факторы, которые, по мнению экспертов, оказывают влияние на выход системы. Под конфигурацией НС (для персептрона) будем понимать число слоев, число нейронов на каждом слое, вид и параметры активационной функции, а также множество весовых коэффициентов связей между нейронами.

В жизненном цикле интеллектуальных систем, выделяют следующие стадии [4]:

- идентификация;
- концептуализация;
- нормализация;
- выполнение;
- отладка.

Представляется логичным считать, что нейросетевые ЭС обладают всеми признаками ЭС в ее обычном понимании, и для них должны быть выделены те же основные этапы жизненного цикла.

На этапе *идентификации* определяется задача, подлежащая решению, выявляются цели разработки, ресурсы, эксперты и категории пользователей.

Концептуализация подразумевает содержательный анализ проблемной области, выявляются ис-

пользуемые понятия и их взаимосвязи, определяется метод решения задачи.

На этапе *формализации* определяются способы представления и интерпретации знаний, формализуются основные понятия [4].

При *выполнении* осуществляется программирование компонентов ЭС, происходит наполнение БЗ, создается прототип разрабатываемого программного продукта.

На этапе *отладки* проверяется компетентность ЭС на предмет соответствия поставленных целей и выдаваемых решений [5]. Для НС неотъемлемым этапом является процесс обучения, так как именно на этом этапе происходит извлечение неявных знаний из обучающего множества. Очевидно, что до процесса обучения НС неадекватна решаемой задаче и может выдавать на выходе только шум, а весовые коэффициенты на данном этапе проинициализированы малыми случайными значениями. Некоторые исследователи выделяют этап обучения нейросети в отдельный этап жизненного цикла, однако представляется целесообразным считать обучение нейросетевого механизма принятия решения частью общего процесса отладки системы. Целью отладки является доведение системы до состояния адекватности поставленной задаче и обеспечение нужного качества принятия решений, что делает задачу отладки ключевым этапом их жизненного цикла.

Введем понятие отладки в широком и узком смысле слова. Отладка ЭС в узком смысле слова — это процесс выявления, локализации ошибок в БЗ, а также коррекция БЗ, не связанная с выбором нового способа представления знаний [5]. Отладка в широком смысле слова подразумевает в качестве меры повышения качества принятия решений также и переход к другой системе представления знаний. В данной работе отладка рассматривается в узком смысле слова.

Для выполнения процесса отладки чаще для этого создаются особые группы по независимой отладке (*independent verification and validation groups* — IV & V) [6, 7], в распоряжение которых поступает уже прошедшая процесс обучения НС. В случае успешного завершения отладки ЭС может быть допущена к эксплуатации; в противном случае принимается решение либо о выборе иной топологии сети, ее параметров и конфигурации, либо о переобучении НС на вновь сгенерированном тестовом множестве.

Однако проблема состоит в том, что на данный момент методы и средства контроля качества НС недостаточно разработаны и носят в основном эвристический характер, а применение традиционных методов отладки программного обеспечения в таких случаях крайне неэффективно или даже невозможно [6]. Это значительно ограничивает возможности применения НС-систем [7]. На данный момент разработаны методики вычисления границ диапазонов возможных значений выходных нейронов НС. Например, в работе [8] для этой цели предлагается методика разделения области значений входных параметров на многомерные прямоугольные области, вычисления выходных значений и ошибки для угловых точек и нахождения максимальной производной с целью выявить границы возможных значений выходов. Однако упомянутые методы не осуществляют контроль корректности НС-систем.

Рис 1. Методы отладки программного обеспечения

Существующие методы отладки баз знаний можно условно подразделить на два класса: статический анализ и динамические методы (рис. 1).

Статические методы не требуют запуска системы и выявляют основные типы ошибок БЗ ЭС [5, 9], связанные с нарушением структурной целостности знаний, что является необходимым, но не достаточным условием отсутствия ошибок. Источником ошибок БЗ может служить как некорректно проведенный процесс обучения, так и недостатки, скрытые в обучающем множестве. В последнем случае ошибки можно сгруппировать по следующим категориям:

- избыточность;
- неполнота;
- противоречивость.

Избыточность означает наличие полностью или частично повторяющихся обучающих примеров, что ведет к неоправданному увеличению времени обучения.

Неполнота подразумевает отсутствие у обучающего множества свойства репрезентативности, т. е. наличия достаточного числа примеров для представления всей генеральной совокупности, а не только отдельных ее классов.

Противоречивость означает наличие обучающих примеров с идентичными входными, но различными выходными параметрами.

Приведенные категории ошибок могут быть выявлены методами статического анализа, которые хорошо представлены в литературе. Однако существуют ошибки, которые не связаны со структурной целостностью знаний и, следовательно, не выявляются статическими методами. Примером могут служить ошибки типа "забывание об исключении" [9, 12], когда некоторое правило или закономерность предметной области работает всегда, кроме одного единственного случая. Дан-

ный тип ошибок является наиболее общим и включает в себя также забывание о нескольких исключениях. Такие ошибки связаны не со спецификой представления знаний в той или иной модели, а с особенностями большинства предметных областей (наличием исключений из правил, используемых при описании объектов и принятии решений), а также со способностью к забыванию — объективного качества человека-эксперта, формирующего базу знаний. Медицина, например, являющаяся традиционной предметной областью для создания экспертных систем, содержит достаточно большое число различных исключений, трудно выявляемых на этапе формализации базы знаний.

В работах [5, 9] было показано, что единственным возможным способом выявления всех ошибок в БЗ является тестирование, т. е. запуск системы на выполнение на определенном множестве тестовых данных с целью обнаружить как можно большее число ошибок. Для экспертной системы, использующей нейросетевой механизм принятия решения, структура тестового множества T будет аналогична структуре обучающего множества $S(X, Y)$, приведенной выше (6).

Перенесение критериев тестирования традиционных программных средств на тестирование БЗ не дает возможности выявления всех ошибок, что определяется наличием скрытых противоречий в самой предметной области и, как следствие, — в базе знаний. В работе [5] предложен метод автоматизированной генерации тестовых данных для продукционных БЗ, выявляющих все возможные ошибки, в том числе ошибки типа "забывание об исключении". Под *продукционной* БЗ [5] будем понимать совокупность

$$P = (F, R, G, C, I), \qquad (7)$$

где F — конечное множество фактов о решаемой проблеме; K — множество продукций или правил, включающее правила вида

$$r_m: \text{ЕСЛИ } f_i \text{ И } f_j \text{ ... И } f_n \text{ ТО } f_k, \qquad (8)$$

где r_m — имя правила; $f_i, f_j, ..., f_n$ — условия выполнения правила; f_k — следствие правила; G — множество целей; I — интерпретатор правил, реализующий процесс вывода.

Данный метод предполагает преобразование модели представления знаний к виду логической сети и последующее применение к ней хорошо разработанных методов технической диагностики для обнаружения неисправностей. В теории логических сетей наряду с остальными типами ошибок выделяют константные и инверсные неисправности, которые могут служить базисом для построения полной модели неисправной системы [10]. Первый тип характеризуется наличием постоянного нулевого ("константный ноль") или единичного ("константная единица") сигнала на одном или нескольких выходах системы. Инверсная неисправность заключается в появлении на одном или нескольких выходах фиктивного инвертора.

Для использования методов технической диагностики для тестирования продукционных БЗ разработаны алгоритм преобразования БЗ в И/ИЛИ-граф, а затем в логическую сеть. И/ИЛИ-граф обеспечивает наглядность представления БЗ. Под И/ИЛИ-графом будем понимать граф $<F, R>$, вершины F которого соответствуют некоторым целям, фактам и правилам, а дуги R — отношения между ними [5]. Рассмотрим следующий пример продукционной системы (6):

ЕСЛИ A_1 ТО B
ЕСЛИ A_1 ТО C
ЕСЛИ A_2 ТО C (9)
ЕСЛИ A_2 ТО D
ЕСЛИ C И B ТО F
ЕСЛИ D ТО F

На рис. 2 представлен И/ИЛИ-граф системы (9), на рис. 3 — соответствующая логическая схема.

Ошибка типа "забывание об исключении" в базе знаний [9] является самой сложной для диагностики и эквивалентна неисправности "константный ноль" [10] соответствующей логической схемы. Для генерации минимального тестового набора, который обнаруживает данную ошибку, используется алгоритм PODEM [11]. Этот алгоритм широко известен в технической диагностике и позволяет гарантированно генерировать тест при условии его существования. Также доказано, что обнаружение в БЗ ошибок типа "забывание об исключении" гарантирует обнаружение и всех остальных типов ошибок.

Распространим описанный метод на нейросетевые БЗ. Будем считать, что вне процесса обуче-

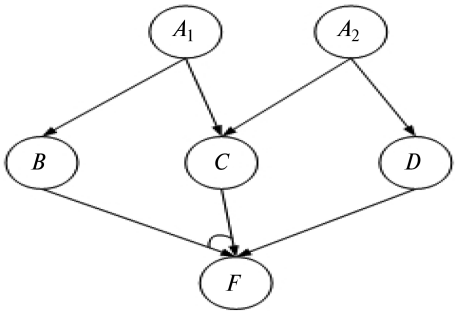


Рис. 2. И/ИЛИ-граф продукционной системы

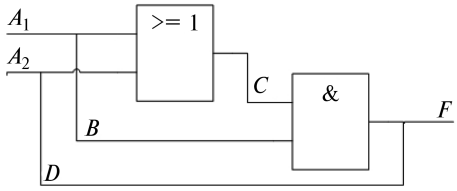


Рис. 3. Логическая схема продукционной системы

ния нейросеть статична и находится в некотором состоянии S_i, а по своей структуре похожа как на структуру, полученную в результате экспертной оценки взаимного влияния признаков [12], так и на продукционную модель представления знаний и может быть преобразована к продукционному виду. Для генерации тестового набора НС должна быть преобразована к виду И/ИЛИ-графа и затем к логической сети в целях дальнейшего использования методов технической диагностики. Сложность применения упоминавшихся методов для тестирования НС заключается в том, что связи между ее структурными элементами не носят дискретного характера, а выражаются весовыми коэффициентами, значения которых изменяются в широких пределах, которые, однако, можно фиксировать для некоторого состояния сети S_i. Поэтому задача построения метода создания тестовых наборов для НС сводится к задаче корректного преобразования последней к соответствующей логической схеме с учетом весовых коэффициентов связей между нейронами. Создание такой схемы сделает возможным применение имеющихся методов для генерации минимального тестового набора, который способен выявить все ошибки в БЗ.

Список литературы

1. **Rodvold D.** A Software Development Process Model for Artificial Neural Networks in Critical Applications // Proceedings of the 1999 International Joint Conference on Neural Networks (IJCNN'99), Washington D. C., 1999.
2. **Minsky M., Papert P.** Perseptrons. Cambridge, 1969. MA: MIT Press.
3. **Rosenblatt F.** Principles of neurodynamics: Perceptrons and the Theory of Brain Mechanisms. New York: Spartan Books, 1962.
4. **Попов Э.** Особенности разработки и использования экспертных систем // Искусственный интеллект: Системы общения и экспертные системы. Справ. изд. / Под ред. Э. Попова. М.: Радио и связь, 1990. С. 261—289.
5. **Долинина О. Н.** Разработка метода тестирования продукционных баз знаний экспертных систем с учетом ошибок типа "забывание об исключении". Саратов, 1997. 171 с. Деп. в ВИНИТИ, № 678-В97.
6. **Pullum L., Darrah M., Taylor B.** Independent Verification and Validation of Neural Networks — Developing Practitioner Assistance. Institute for Scientific Research, Inc. 2003.
7. **Schumann J., Nelson S.** Toward V&V of Neural Network Based Controllers. NASA, 2002.
8. **Hull J., Ward D., Zakrzewski R.** Verification and Validation of Neural Networks for Safety-Critical Applications. 2003.
9. **Поспелов И. Г., Поспелова Л. Я.** Динамическое описание систем продукций и проверка непротиворечивости продукционных экспертных систем // Изв. АН СССР, ТК, 1987. № 1. С. 184—192.
10. **Ярмолик В. Н.** Контроль и диагностика цифровых узлов ЭВМ. М.: Наука и техника, 1988. 240 с.
11. **Goel P., Rosales B. C.** "PODEM — X": an automatic test generation system for VISI logic structures // Proc. 18th IEEE Design Automation Conf. 1981. Paper 133. P. 260—268.
12. **Долинина О. Н.** Информационные технологии в управлении современной организацией. Саратов: СГТУ, 2006. 160 с.

БЕЗОПАСНОСТЬ ИНФОРМАЦИИ

УДК 004.414.23

И. В. Котенко, д-р техн. наук, проф., вед. научн. сотр.,
А. В. Уланов, канд. техн. наук, мл. научн. сотр.,
Санкт-Петербургский институт информатики и автоматизации РАН
E-mail: ivkote1@mail.ru

Многоагентное моделирование механизмов защиты от распределенных компьютерных атак

Предлагаются модели, методика и реализующая их система для моделирования механизмов защиты от распределенных компьютерных атак. Подход основывается на представлении сторон атаки и защиты в виде команд интеллектуальных агентов. На основе предложенной архитектуры разработана система моделирования. В качестве примера реализации моделей и методики исследуются механизмы защиты от атак, вызывающих "распределенный отказ в обслуживании".

Ключевые слова: *распределенные компьютерные атаки, механизмы защиты от компьютерных атак, многоагентное моделирование, распределенный отказ в обслуживании, Интернет.*

Введение

На практике достаточно сложно осуществить проверку и оценку применения того или иного механизма защиты. Необходимо либо располагать большим выделенным фрагментом сети, где можно проводить эксперименты, либо использовать для экспериментов сети реальных Интернет-провайдеров (*Internet Service Provider*, *ISP*). При этом важные условия научного эксперимента, такие как повторяемость и контролируемость, не могут быть соблюдены. Вследствие указанных причин для исследования механизмов защиты от распре-

деленных атак необходимо использовать имитационное моделирование (далее именуемое моделированием).

Для реализации данной цели в настоящей статье предлагается использовать многоагентное моделирование, так как оно упрощает сам процесс моделирования, позволяя представить изучаемое явление в виде совокупности автономных агентов и взаимодействий между ними.

Базовые компоненты моделирования

Предлагается использование подхода, основанного на многоагентных технологиях моделирования процессов поведения систем в сети Интернет [1]. На основе данного подхода выделены следующие базовые компоненты моделирования — модели противоборства команд агентов, реализующих распределенные атаки и механизмы защиты от них (рис. 1):

- модели команд агентов атаки;
- модели команд агентов защиты;
- модели взаимодействия команд;
- модель сети Интернет.

Модели команд агентов атаки и защиты предназначены для представления процессов реализации сетевых атак и механизмов защиты от них. Модели взаимодействия команд задают сценарии взаимодействия команд агентов. Модель сети Интернет определяет коммуникационную среду для агентов и их возможные взаимодействия.

Модели команд агентов и сети Интернет

Модели команд агентов включают:

- частные онтологии команд;
- базовые и специализированные функции агентов;
- перечень классов агентов;
- протоколы их взаимодействия;
- сценарии их поведения [2].

Онтология защиты от исследуемого класса атак строится на базе предложенных классификаций атак и механизмов защиты. Частная онтология команды защиты основана на классификации механизмов защиты от данного типа атак, а онтология команды атаки — на классификации соответствующих атак. Эти онтологии также используются для параметризации моделей защиты и атаки.

Предлагаемый перечень *базовых функций агентов* включает следующие функции:

- инициализации;
- окончания работы;
- доступа к частной онтологии агента;
- контроля списка активных агентов;
- базовой работы с модулями транспортного уровня (создание соединения, посылка сообщения, закрытие соединения).

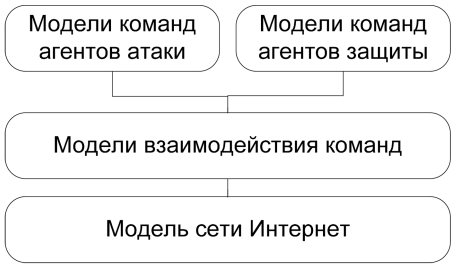


Рис. 1. Представление основных компонентов моделирования

Для моделей агентов атаки и защиты предлагаются *специализированные функции*, основанные на базовых. Для команды защиты их реализация зависит от применяемого метода защиты. В агентах атаки используются расширенные функции работы с модулями транспортного уровня стека протоколов Интернет для возможности реализации различных типов атак.

Модель команды атаки включает два *класса агентов*: "демоны", непосредственно реализующие атаку, и "мастер", выполняющий действия по координации остальных компонентов системы.

На предварительном этапе демоны и мастер устанавливаются на доступные (уже скомпрометированные) узлы сети Интернет. Здесь важными свойствами команды являются число и распределенность агентов. Затем происходит организация команды атаки: демоны посылают мастеру сообщения о том, что они существуют и готовы к работе, а мастер сохраняет информацию о членах команды и об их состоянии. Эта функциональность реализуется с помощью базовых функций агентов и протокола составления команды.

Функциональность агентов команды атаки задается с помощью сценариев поведения и протоколов, применяемых в этих сценариях.

Команда атаки использует следующие *протоколы взаимодействия*:

- протокол составления команды;
- протокол проверки работоспособности;
- протокол рассылки параметров атаки;
- протокол атаки.

В *модели команды защиты* выделены следующие *классы агентов*:

- первичной обработки информации ("сэмплеры"), обнаружения атаки ("детекторы");
- фильтрации ("фильтры");
- ограничения трафика ("ограничители");
- агенты расследования.

В начальный момент времени агенты защиты устанавливаются на соответствующие их ролям сетевые узлы:

- сэмплер — на пути следования трафика для защищаемого узла;
- детектор — на любой узел в подсети защищаемого узла;

- фильтры и ограничители — на входе в подсеть защищаемого узла;
- агент расследования — за пределами подсети защищаемого узла на любом доступном из Интернета узле.

Общая цель команды агентов защиты — противодействие сетевой атаке. За ее выполнением следит детектор. Функциональность агентов команды защиты задается с помощью сценариев поведения и протоколов, используемых в этих сценариях.

Команда защиты использует следующие *протоколы взаимодействия*:

- протокол составления команды;
- протокол проверки работоспособности;
- протокол сбора информации у сэмплеров;
- протокол рассылки адресов возможных источников атаки;
- протокол обезвреживания агентов атаки.

Протоколы взаимодействия агентов защиты представляются в виде последовательности директив (термин "директива" используется во избежание путаницы с командой в смысле сообщества агентов) с определенными параметрами. Вид директивы определяет, как использовать эти параметры. Директивы зависят от используемого метода защиты и схемы кооперации команды.

Протоколы взаимодействия агентов атаки также представляются в виде набора директив с определенными параметрами. В директивах указывается время и режим атаки, а также контролируется численность агентов атаки. Условия инициации протоколов обеспечивают селективность коммуникаций агентов команд.

Протоколы взаимодействия агентов используют механизмы транспортного уровня, предоставляемого коммуникационной средой. В данной работе для обмена сообщениями между агентами используется протокол TCP.

Например, в соответствии с протоколом составления команды агентов среди них выделяются "клиенты" и "серверы". Первые посылают сообщения о своем существовании "серверу". Он содержит список агентов в команде. Периодически этот агент опрашивает известных ему агентов для того, чтобы актуализировать свой список и знать, какие агенты активны в данный момент. Протокол составления команды агентов является составной частью механизмов обеспечения мониторинга и восстановления функциональности команды агентов.

В зависимости от используемого метода защиты и механизма кооперации может использоваться набор *сценариев поведения команды защиты*. Сценарии отражают различные этапы деятельности команды защиты: предупреждение атаки, определение факта атаки, противодействие атаке, отслеживание источника атаки. В сценариях реализована возможность адаптивного поведения в зависимости от действий команды атаки и реакции среды. Сценарии поведения команд агентов обеспечивают согласованность действий агентов.

Для атак "распределенный отказ в обслуживании" *сценарии поведения команды атаки* включают следующие этапы:

- захват уязвимых узлов и установка на них агентов атаки;
- формирование команды атаки путем составления списка активных агентов;
- посылка сообщения о начале атаки всем агентам (или группе агентов);
- контроль выполнения заданного режима атаки.

В сценариях реализована также возможность адаптивного поведения команды атаки в зависимости от действий команды защиты и реакции среды.

Модель сети Интернет включает топологию сети, модели каналов связи, протоколов, узлов и сетевого трафика. Модель позволяет задавать среды взаимодействия агентов, которые характеризуются различной степенью детальности представления сети Интернет, в зависимости от требований к точности и к масштабируемости моделирования. Используется модель сети на уровне пакетов. Топология может быть построена в виде графа на основе ранга узла и заданного распределения вероятностей.

Модели взаимодействия команд агентов

Модели взаимодействия команд включают модели антагонистического противоборства, модели кооперации команд и адаптации к поведению друг друга. *Антагонистическое противоборство* возникает в процессе взаимодействия агентов защиты и атаки. *Кооперация* между командами (например, между несколькими командами защиты) происходит для достижения их общих целей с большей эффективностью. *Адаптация* заключается в модификации реакции на действия агентов другой команды и изменение среды. Разработаны модели антагонистического противоборства между командами защиты и атаки, модели кооперации команд защиты, включающие различные методы кооперативной защиты, и модели адаптации команд защиты и атаки. Модели адаптации используются командой атаки для изменения режима атаки в зависимости от действий команды защиты и доступности агентов атаки. Модели адаптации используются командой защиты для реакции на изменения режима атаки и численности агентов атаки.

Антагонистическое противоборство. Команда атаки воздействует на сеть и цель атаки, и через

сеть это воздействие передается команде защиты. Команда защиты, обнаруживая атаку, на основе заданных методов защиты пытается применить определенные контрмеры. В их числе — применение правил фильтрации на различных узлах, где установлены агенты-фильтры, а также отслеживание и обезвреживание агентов атаки с помощью агентов расследования. Правила фильтрации, примененные в исходной (атакующей) подсети, могут нанести непосредственный вред команде атаки. Трафик от агентов атаки будет отбрасываться, и они не смогут получить директивы от агента-мастера.

Кооперативная защита. Команды агентов защиты могут *взаимодействовать по различным схемам кооперации* [3]:

- *без кооперации* — все команды агентов работают сами по себе;
- *кооперация на уровне фильтров* — команда, на сеть которой направлена атака, может применять правила фильтрации на фильтрах других команд;
- *кооперация на уровне сэмплеров* — команда, на сеть которой направлена атака, может получать информацию о трафике от сэмплеров других команд;
- *слабая кооперация* — команды могут получать информацию о трафике от сэмплеров некоторых других команд и применять правила фильтрации на фильтрах также некоторых других команд;
- *полная кооперация* — команда, на сеть которой направлена атака, может получать информацию о трафике от всех сэмплеров других команд и применять правила фильтрации на всех фильтрах других команд.

В зависимости от степени кооперации каждой команде задается то или иное число "известных" ей команд.

Адаптация. В соответствии с применяемой моделью адаптации при изменении показателя серьезности (мощности) атаки ($S(t)$) подсистема адаптации выбирает конфигурацию системы защиты ($K_D(t)$), которая минимизирует функцию эффективности F, представляющую собой сумму $C(S(t), K_D(t))$ компонентов стоимости защиты [4]:

$$F = \sum_{i=1}^{n} C_i(S(t),\ K_D(t)).$$

Например, для атак "распределенный отказ в обслуживании" может выбираться следующий критерий функционирования команды защиты [5]. При заданном объеме трафика атаки предлагается выбирать такую конфигурацию защиты, которая минимизирует долю ложных срабатываний C_{FP}, пропусков атак C_{FN} и продолжительность атаки C_T:

$$\min_{S(t)}\{C_{FP}(S(t),\ K_D(t)) + C_{FN}(S(t),\ K_D(t)) + C_T(S(t),\ K_D(t))\}.$$

Критерий функционирования команды атаки может быть следующим. Предлагается при заданном показателе действенности защиты ($E(t)$) и числе работоспособных демонов выбирать такую конфигурацию команды атаки $K_A(t)$, которая минимизирует число посылаемых пакетов C_P и возможность раскрытия агентов C_D:

$$\min_{E(t)}\{C_P(E(t),\ K_A(t)) + C_D(E(t),\ K_A(t))\}.$$

Архитектура системы моделирования

Для реализации представленного подхода к моделированию механизмов защиты от распределенных атак разработана многоуровневая программная система, отличающаяся от известных средств агентно-ориентированного моделирования (например, CORMAS, Repast, Swarm, MadKit, MASON, NetLogo и др.), в первую очередь, использованием в качестве базиса средств имитационного моделирования, позволяющих адекватно имитировать сетевые процессы. По этой причине для реализации декларируемого подхода используется архитектура системы моделирования (рис. 2), включающая:

- базовую систему имитационного моделирования (*Simulation Framework*);
- модуль (пакет) моделирования сети Интернет (*Internet Simulation Framework*);
- подсистему агентно-ориентированного моделирования (*Agent-based Framework*);
- модуль (библиотеку) имитации процессов предметной области (*Subject Domain Library*) [1].

Рис.2. Архитектура системы моделирования

Компонент Simulation Framework представляет собой систему моделирования на основе дискретных событий. Остальные компоненты являются надстройками или моделями для Simulation Framework.

Компонент Internet Simulation Framework — комплект модулей, позволяющих достаточно адекватно моделировать узлы и протоколы сети Интернет. Сеть состоит из узлов, узел содержит стек IP-протоколов. Дополнительно можно подключать модули, реализующие протоколы транспортного уровня.

Многоагентное моделирование реализуется посредством *компонента Agent-based Framework*, который использует модуль имитации процессов предметной области. Данный компонент представляет собой библиотеку модулей, задающих интеллектуальных агентов, реализованных в виде приложений. При проектировании и реализации модулей агентов подразумевается использование элементов абстрактной архитектуры FIPA. Передача сообщений между агентами происходит поверх TCP-протокола, реализованного в компоненте Internet Simulation Framework. Каталог агентов является обязательным только для агента, координирующего действия других. Агенты могут управлять другими модулями с помощью сообщений.

Компонент Subject Domain Library — это библиотека, служащая для имитации процессов предметной области, включающая также модули, дополняющие функциональность IP-узла (таблица фильтрации и анализатор пакетов).

С использованием OMNeT++ INET Framework и программных моделей, разработанных на C++, представленная выше архитектура была реализована для многоагентного моделирования атак DDoS и механизмов защиты от них. Модели агентов, реализованные в Agent-based Framework, представлены типовым агентом, агентами атаки и агентами защиты. Subject Domain Library содержит различные модели узлов, например, атакующего, брандмауэра и др., а также модели приложений (механизмы реализации атак и защиты, анализаторы пакетов, таблицы фильтрации).

Методика моделирования

Предлагаемая методика многоагентного моделирования механизмов защиты от распределенных атак определяет, каким образом задавать параметры модели сети, атаки и защиты (рис. 3). Методика описывает процесс получения выходных параметров и их анализа в целях выбора наиболее эффективного механизма защиты.

Основными этапами методики являются:

- подготовительный этап;
- этап задания параметров;

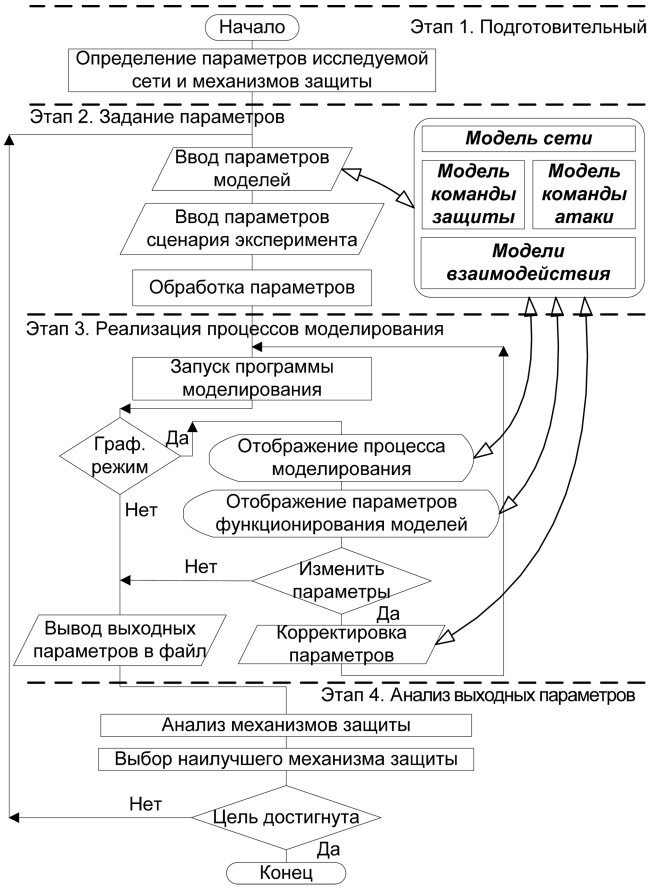


Рис. 3. Обобщенное представление методики моделирования

- этап реализации процессов моделирования;
- этап анализа выходных параметров.

Третий этап выполняется разработанной программной системой моделирования, остальные этапы — проектировщиком. Двусторонними стрелками на рис. 3 обозначено использование предложенных моделей на соответствующих этапах выполнения методики моделирования.

Реализация и эксперименты

Представленные выше модели, методика и среда моделирования были реализованы и использованы для предметной области защиты от атак, вызывающих "распределенный отказ в обслуживании" (DDoS) [1—3, 5].

Для исследования данного класса атак и механизмов защиты от них была разработана библиотека предметной области, содержащая модели атак и механизмов защиты (*DDoS attack and defense library*). Модели реализованы в виде параметризованных модулей, которые устанавливаются как приложения на узлы моделируемой сети, а также включаются в состав агентов атаки и защиты. Для этого реализованы протоколы взаимодействия между ядром агента и модулем атаки или за-

щиты. Библиотека подключена к разработанной системе моделирования.

Созданная система моделирования позволяет проводить различные эксперименты в целях исследования стратегий реализации атак DDoS и методов защиты от них. В процессе использования системы моделирования можно варьировать топологию и конфигурацию сети, структуру и конфигурацию команд атаки и защиты, механизмы реализации атак и защиты, параметры кооперации команд и др. Задавая различные адаптивные стратегии действий команд, можно наблюдать за процессом выполнения командами своих функций и эволюции их поведения и изменения состояния глобальной сети. На основе экспериментов проводятся измерения различных показателей эффективности механизмов защиты, и выполняется анализ условий и возможности их применения.

Проведено представительное число экспериментов по исследованию как известных, так и предложенных авторами механизмов защиты от DDoS-атак. В том числе анализировались различные кооперативные схемы защиты и схемы адаптации. Проведены эксперименты по изучению таких известных кооперативных механизмов защиты, как DefCOM [6] и COSSACK [7]. Исследованы также предложенные кооперативные механизмы защиты (с кооперацией на уровне фильтров, с кооперацией на уровне сэмплеров, со слабой и полной кооперацией).

В соответствии с архитектурой системы DefCOM введены следующие классы агентов [6]:

- "Alert generator" (строится на базе детектора и сэмплера);
- "Rate limiter" (представляет собой агента "ограничитель");
- "Classifier" (задается агентом "фильтр", получающим данные по фильтрации от детектора).

Для моделирования системы COSSACK представлены следующие классы агентов [7]: "snort" (на базе агента "сэмплер"); "watchdog" (на базе агента "детектор"). Для имитации фильтра на маршрутизаторе применяется агент "фильтр".

Использованы методы защиты HCF (*Hop Count Filtering*, фильтрация по числу "скачков" при передаче пакетов), Source IP Address Monitoring (мониторинг IP-адресов отправителей) и BPS (мониторинг объема трафика, *Bit Per Second*, бит в секунду).

Топология сети генерировалась на основе степенной функции плотности распределения числа связей узлов и числа узлов. Задавались защищаемый сервер, клиенты и параметры осуществления запросов к нему. В команду атаки входило 10 и более демонов, реализующих атаку UDP flood. Команды защиты конфигурировались в соответствии с указанными кооперативными схемами.

Проведенные эксперименты показали эффективность различных схем кооперативной защиты от атак DDoS. Лучшей из них оказалась схема с полной кооперацией. Решающую роль в защите от атаки сыграла кооперация сэмплеров, благодаря чему осуществлялся постоянный обмен данными по трафику в различных командах защиты.

Исследована следующая адаптивная схема взаимодействия команд. Команда атаки начинала атаку в заданный момент времени с определенными интенсивностью и методом подмены адреса отправителя. Во время атаки, если мастер обнаруживал, что какой-то из демонов неработоспособен, он изменял параметры атаки, чтобы сохранить интенсивность и предотвратить обнаружение. Команда защиты изначально работала, используя наименее ресурсоемкий способ защиты. Как только обнаруживалась атака, делалась попытка заблокировать пакеты от атакующих, проследить их и обезвредить. Если это не удавалось, метод защиты заменялся на другой, более сложный. Для эксперимента использовались кооперативные схемы команд "без кооперации", "на уровне фильтров", "на уровне сэмплеров" и "полная кооперация".

Заключение

Представлены модели и методика для проведения многоагентного моделирования механизмов защиты от распределенных атак. Полученные в работе результаты позволяют описывать стороны атаки и системы защиты в виде команд агентов, используя основные параметры атаки и защиты, проводить моделирование в соответствии с предложенной методикой и оценивать эффективность анализируемых механизмов защиты. Полученные результаты могут быть использованы для анализа систем защиты существующих сетей, а также для выработки рекомендаций по созданию перспективных систем защиты. В качестве примера реализации моделирования исследованы различные механизмы защиты от атак "распределенный отказ в обслуживании". Особое внимание уделено кооперативным и адаптивным механизмам защиты.

Направления дальнейших работ связаны с исследованием механизмов защиты от различных типов атак, а также с совершенствованием системы моделирования.

Работа выполнена при финансовой поддержке РФФИ (проект № 07-01-00547), программы фундаментальных исследований ОИТВС РАН (контракт № 3.2/03), Фонда содействия отечественной науке, проектов Евросоюза POSITIF (контракт IST-2002-002314) и RE-TRUST (контракт № 021186-2) и других проектов.

Список литературы

1. **Котенко И. В., Уланов А. В.** Многоагентное моделирование защиты информационных ресурсов в сети Интернет // Изв. РАН. Теория и системы управления. 2007. № 5. С. 74—88.

2. **Котенко И. В., Уланов А. В.** Команды агентов в киберпространстве: моделирование процессов защиты информации в глобальном Интернете // Тр. Института системного анализа РАН. Т. 27. Проблемы управления кибербезопасностью информационного общества. М.: КомКнига, 2006. С. 108—129.

3. **Kotenko I., Ulanov A.** Investigation of Cooperative Defense against DDoS // International Conference on Security and Cryptography. SECRYPT 2007. Barcelona, Spain. July 28—31. 2007. P. 180—183.

4. **Zou C., Duffield N., Towsley D., Gong W.** Adaptive Defense Against Various Network Attacks // Proc. of SRUTI: Steps to Reducing Unwanted Traffic on the Internet. Boston, July 7—8, 2005. Berkley: USENIX Association. 2005. P. 10—20.

5. **Kotenko I. V., Ulanov A. V.** Multi-agent Framework for Simulation of Adaptive Cooperative Defense against Internet Attacks // Lecture Notes in Artificial Intelligence. 2007. Vol. 4476. P. 212—228.

6. **Mirkovic J., Robinson M., Reiher P., Oikonomou G.** Distributed Defense Against DDOS Attacks // Report CIS-TR-2005-02. University of Delaware CIS Department. 2005. 12 p.

7. **Papadopoulos C, Lindell R., Mehringer I., Hussain A., Govindan R.** Cossack: Coordinated suppression of simultaneous attacks // Proc. of DARPA Information Survivability Conference and Exposition. 2003. Vol. 2. P. 94—96.

УДК 004.332.3

О. Д. Жуков, канд. техн. наук, вед. науч. сотр.,
Московский государственный университет
им. М. В. Ломоносова
E-mail: zhukovo@dec1.sinp.msu.ru

Модулярные числовые системы в криптографии

Модулярные экспоненциальные вычисления составляют основу многих криптографических методов. Эта статья обсуждает возможность создания эффективных систем, ориентированных на использование модулярных числовых систем и известного метода умножения Монтгомери.

Ключевые слова: *модулярные числовые системы, вычет, экспоненциальные вычисления, информационная безопасность.*

Введение

Существует множество приложений, основанных на экспоненциальных вычислениях. К ним относятся приложения, связанные с процедурами полиномиальной алгебры, полиномиальной аппроксимации, интерполяции и экстраполяции функций, криптографических методов и др. Интенсификация информационных процессов в современном обществе порождает ряд попутных проблем, без решения которых нельзя говорить об эффективности информации. Одной из них является проблема защиты информации. Важную роль в этих системах, например в криптографии, играют модулярные экспоненциальные вычисления выражений вида $C = Q^E \bmod P$ или $C \equiv |Q^E|_P$, где C, Q, E, P суть чрезвычайно большие целые числа. Основу таких вычислений, как правило, составляет умножение по методу Монтгомери, или умножение Монтгомери [1], от эффективности реализации которого зависит, в конечном счете, эффективность используемого криптографического метода.

В следующих разделах излагаются и сравниваются различные подходы к решению этой задачи на основе традиционных двоичных технологий, систем вычетов и дуальных смешанных систем.

Для полноты материала рассмотрим некоторые аспекты метода Монтгомери. Метод использует в качестве модуля простые числа P и выполняется в конечном поле $GF(P)$. Операнды A, $B \in GF(P)$, участвующие в умножении Монтгомери как элементы поля $GF(P)$, преобразуются в элементы домена Монтгомери в виде $\widetilde{A} \equiv |AR|_P$. Произведение Монтгомери $\widetilde{C}$ определяется как

$$\widetilde{C} \equiv |CR|_P \equiv |(AR)(BR)R^{-1}|_P \equiv |AB|R^2|_P R^{-1}|_P, \quad (1)$$

где $\widetilde{C}$ является также элементом домена Монтгомери; $C \equiv |AB|_P$; $|RR^{-1}|_P \equiv 1$; A, $B < P < R$; НОД$(P, R) = 1$.

Операции преобразования между двумя доменами могут быть также выполнены в соответствии с выражениями

$$\widetilde{A} = (A, R^2) \equiv |AR^2R^{-1}|_P \equiv |AR|_P; \quad (2)$$

$$\widetilde{B} = (B, R^2) \equiv |BR^2R^{-1}|_P \equiv |BR|_P; \quad (3)$$

$$C = (\widetilde{C}, 1) \equiv |CRR^{-1}|_P, \quad (4)$$

где $|R^2|_P$ вычисляется в качестве константы для заданных P и R.

В этом случае вычисление модулярной экспоненты переопределяется как повторное умножение Монтгомери. В рамках процедуры определения модулярной экспоненты $|C = Q^E|_P$ мы вычисляем $Q' \equiv |QR|_P$ (стандартная модулярная редук-

ция), $C' \equiv |(Q')^E R^{1-E}|_P$ (экспонента Монтгомери), $C \equiv |C'R^{-1}|_P$ (редукция Монтгомери).

Метод Монтгомери основан на двух конгруэнциях:

$$A - |A|_R |P^{-1}|_R \equiv |A|_P; \quad (5)$$

$$A - |A|_R |P^{-1}|_R \equiv 0, \quad (6)$$

где $\|P^{-1}|_R P|_R \equiv 1$; значение $|A|_R |P^{-1}|_R$ есть целое число; $|P^{-1}|_R$ является заранее вычисляемой константой.

Следующее свойство приводит непосредственно к алгоритму умножения Монтгомери. Если P и R суть взаимно простые числа и $P' \equiv |-P^{-1}|_R$, тогда $(T + QRP)/R \equiv |TR^{-1}|_P$ есть целое, удовлетворяющее конгруэнции

$$(T + QP)/R \equiv |TR^{-1}|_P, \quad (7)$$

где $Q \equiv |TP'|_R$. То, что $(T + QP)/R$ есть целое, может быть проверено подстановкой Q.

На основе приведенных выше выражений выполняется обобщенный алгоритм вычисления произведения Монтгомери $|AB|_P$.

Обобщенный алгоритм

$A, B, C \in GF(P)$, $(0 \leqslant A, B < P < R)$.

1. $P' := |-P^{-1}|_R$.
2. $T := AB$.
3. $Q := |TP'|_R$.
4. $T := T + QP$.
5. $U := T/R$.
6. Если $U > P$, то $U - P$, иначе п. 5,

где T и U суть промежуточные переменные (неозначенные целые длиной $[\log_2 R]$ бит. Шаг 6 требуется потому, что не гарантируется выполнение условия $T + QP < PR$.

В следующих разделах представляются и сравниваются методы и алгоритмы умножения и возведения в степень на основе двоичных и модулярных числовых систем.

Умножение на основе двоичных систем

При вычислениях в позиционной системе в качестве значения R берется степень основания r^e такая, что $P < R = r^e$. Выполнение умножения Монтгомери с учетом приведенных выше выражений в рамках привычных двоичных технологий реализуется в соответствии с исходной конгруэнцией

$$C \equiv |AB2^{-D}|_P. \quad (8)$$

где $A, B < P < 2^D$ и НОД$(P, 2) = 1$.

Если младшие D разрядов (битов) произведения AB равны нулю, то (8) может быть вычислено с помощью обычного сдвига на D разрядов без модулярного деления. В общем случае обнуление D младших разрядов произведения AB осуществляется прибавлением к нему подходящего числа, кратного P.

Напомним, что числа, используемые в криптосистемах с открытыми ключами, имеют длину от нескольких сотен до нескольких тысяч битов. Обычная разрядность современных вычислительных средств варьируется в пределах 8—128 бит. Поэтому для реализации действий над числами, по длине многократно превышающими эту разрядность, они разбиваются на w блоков (машинных слов) длиной d бит ($D = dw$).

Тогда операнд A может быть представлен с помощью w d-разрядных слов a_i $(0 \leqslant i \leqslant D - 1)$ с весами 2^{id} в виде

$$A = a_0 + a_1 2^d + a_2 2^{2d} + \ldots + a_{w-1} 2^{(w-1)d} = (a_0, a_1, \ldots, a_{w-1}). \quad (9)$$

Одна из типичных процедур вычисления (8) сводится на каждом шаге к накоплению суммы C частных произведений $a_i B$ и делению ее на 2^d:

$$C := |C + a_i B)/2^d|_P, \quad (10)$$

где $i = 0, 1, \ldots, w - 1$.

Операция деления в (10) заменяется сдвигом на d бит после прибавления определенного числа $k_i P$, кратного P, в целях обнуления младших d разрядов текущей величины C:

$$C := (C + a_i B + k_i P)/2^d. \quad (11)$$

Значение d-разрядного числа k_i определяется конгруэнцией

$$k_i \equiv |(c_0 + a_i b_0) P'|_{2^d}, \quad (12)$$

где c_0, b_0, p_0 суть наименьшие значащие слова соответственно в C, B, P, а d-разрядное число P' определяется как

$$g \equiv |-P^{-1}|_{2^d} \equiv |-p_0^{-1}|_{2^d}. \quad (13)$$

С учетом формул (11)—(13) получим наименьшее значащее слово в (11), равное нулю:

$$c_0 + a_i b_0 + k_i P|_{2^d} \equiv \equiv |c_0 + a_i b_0 + (c_0 + a_i b_0)(-P^{-1})|_{2^d} \equiv 0. \quad (14)$$

Процедура, описанная выше, сводится к типичному для метода Монтгомери двоичному алгоритму 1.

Алгоритм 1

Ввод: $A, B, C, P \in GF(P)$, $A, B < P < 2^D$.
$A = (a_0, a_1, \ldots, a_{w-1})$,
$B = (b_0, b_1, \ldots, b_{w-1})$, $P' \equiv |-P^{-1}|_{2^d}$.

Вывод: $C = |AB2^{-D}|_P$.

1. $C := 0$.
2. for $i = 0$ to $w - 1$.
3. $k_i := |(c_0 + a_i b_0)P'|_{2^d}$.
4. $C := (c_i + a_i b_0 + k_i P)/2^d$.
5. if $(C \geqslant P)$ then $C := C - P$ else p. 4.

Напомним, что деление на 2^d в предпоследнем операторе заменяется на простой сдвиг. Алгоритм 1 лежит в основе большинства известных алгоритмов, различающихся лишь характером и последовательностью операций. Например, алгоритм, описанный в оригинальной статье [2], основан на суммировании полноразрядных чисел длиной $D = dw$ бит. Ниже приведен алгоритм, который реализует один из известных возможных подходов, использующих вместо длинноразрядных сложений d-разрядные умножения.

Алгоритм 2

1. $C := 0$.
2. for $i = 0$ to $w - 1$.
3. $u := 0$.
4. $k_i := |(c_0 + a_i b_0)P'|_{2^d}$.
5. for $j = 0$ to $w - 1$.
6. $S := c_j + a_i b_j + k_i p_j + u$.
7. if $(j \neq 0)$ then $c_{j-1} := |S|_{2^d}$.
8. $u := S/2^d$.
9. $c_{w-1} := u$.
10. if $(C > P)$ then $C := C - P$ else p. 6.

Длина промежуточных результатов C равна $2d + 1$ бит и ее старшие d бит и младшие $d + 1$ бит запоминаются в u и c_j соответственно. Размерность c_{w-1} равна $d + 1$ бит и соответствует размерности поступающего в c_{w-1} содержимого и в качестве переноса для следующего суммирования. При этом $(d + 1)$-й бит запоминается в отдельном однобитовом регистре, а не в d-битовом слове памяти. Отметим, что D не обязательно должно быть кратным d. В этом случае старшие биты старшего слова просто заполняются нулями.

Суммарное число операций во вложенных циклах, определяющих не в полном объеме сложность умножения Монтгомери в двоичной системе, оценивается примерно как $2w^2$ умножений и $3w$ сложений машинных слов длиной d бит.

Далее представляются методы и алгоритмы для вычисления произведений и экспонент по Монтгомери на основе модулярных числовых систем: систем вычетов и смешанных систем. Показываются преимущества использования этих систем, обеспечивающих существенное увеличение скорости выполнения названных процедур. Достаточно подробное рассмотрение двоичных алгоритмов было выполнено в целях проведения в дальнейшем сравнительной оценки сложности (эффективности) алгоритмов на основе исследуемых модулярных числовых систем относительно этих алгоритмов.

Умножение Монтгомери на основе системы вычетов

В [3] показан метод умножения Монтгомери на основе Китайской теоремы о вычетах. Метод использует две системы вычетов (СВ): исходную (базовую) и дополнительную, позволяющую получить расширенную СВ для размещения в ней произведений чисел. Вычислительная сложность данного метода определяется в основном двумя процедурами расширения кодов чисел из дополнительной системы в исходную, и наоборот.

Рассмотрим другой возможный метод умножения Монтгомери с применением системы со смешанными основаниями.

В сравнении с арифметикой позиционных систем простое умножение в системе вычетов является самой эффективной операцией. Но умножение Монтгомери характеризуется существенной спецификой. В рассмотренных выше алгоритмах метод Монтгомери на каждом шаге использует самую младшую цифру (слово) позиционного представления. СВ является непозиционной системой. Однако имеется возможность преобразования чисел из СВ в систему со смешанными основаниями (ССО), использующую модули той же СВ. И здесь основная идея состоит в том, что самая младшая цифра (слово) числа в ССО может быть представлена любым вычетом числа в СВ, если он соответствует первому модулю в рамках процесса преобразования в ССО. Поскольку каждый шаг требует точного деления на один из модулей, образуются неопределенные вычеты. Для определения этих вычетов используется расширенная СВ с дополнительным набором модулей, достаточным для покрытия исходного числового диапазона.

Представим приведенные в предыдущем разделе выражения для умножения Монтгомери в следующем виде:

$$Q \equiv |TP'|_M; \qquad X = (T + QP)/M, \tag{15}$$

где $P' \equiv |-P^{-1}|_M$, A, $B < P < M$, НОД$(P, M) = 1$, $T = AB$, $M = \prod_{i=1}^{n} m_i$ — диапазон СВ и ССО с модулями m_i.

Здесь P' есть предвычисленная константа. Поскольку обозначение M означает динамический

диапазон СВ, оно используется в (15) вместо обозначения диапазона позиционной системы R.

Деление на M в СВ приводит к неопределенности вычетов частного по модулям m_i ($i = 1, ..., n$). Для определения этих вычетов вводится дополнительная СВ с модулями расширения m_j ($j = n + 1, ..., n + v$) и динамическим диапазоном $\tilde{M} = \prod_{i=n+1}^{n+v} m_i$. Обозначим базовую (исходную) СВ как СВ1, а дополнительную СВ — как СВ2. Обе системы образуют интегрированную СВ с общим диапазоном $M^+ = M\tilde{M} \prod_{i=1}^{n+v} m_i$.

Из (15) следует, что $X < 2P$. Поэтому для СВ2 должно выполняться условие $\tilde{M} > 2P$. Ниже показаны основные шаги алгоритма умножения Монтгомери в СВ в соответствии с (15).

Алгоритм 3

1. СВ1 : $|T|_M := |AB|_M$, $|T|_{\tilde{M}} \equiv |AB|_{\tilde{M}}$.
2. СВ1 : $|Q|_M := |TP'|_M$.
3. СВ1 и СВ2: $|Q|_{\tilde{M}} \equiv \|Q|_M|_{\tilde{M}}$ (расширение представления Q из СВ1 в СВ2, т. е. в диапазоне $\tilde{M}$).
4. СВ2: $|QP|_{\tilde{M}}$.
5. СВ2 : $|T + QP|_{\tilde{M}}$.
6. СВ2 : $|X|_{\tilde{M}} \equiv |(T + QP)M^{-1}|_{\tilde{M}}$.
7. СВ! и СВ2: $\|X|_{\tilde{M}}|_M$ (расширение представления X из СВ2 в СВ1).

Очевидно, немодулярные процедуры преобразования в ССО и расширения СВ составляют основную вычислительную нагрузку в алгоритме, поскольку само умножение является короткой модулярной операцией. Чтобы избежать первую процедуру расширения СВ в алгоритме 3, мы представляем число A в ССО.

Здесь возможны, по крайней мере, два способа реализации конкретного алгоритма: последовательный и параллельный.

Последовательный способ

Чтобы избежать первой процедуры расширения СВ1 в алгоритме 3, мы представляем число A в ССО. Кроме того, этот подход предполагает примерное равенство модулей СВ1 и СВ2 и машинного слова двоичного компьютера. В этом случае сравнение сложности умножения Монтгомери в СВ и позиционной двоичной системе будет проводиться в равных условиях.

Для нашего алгоритма умножения Монтгомери все числа, кроме A, представляются в СВ:

$$B = (b_1, b_2, ..., b_n); \quad (16)$$

$$P = (p_1, p_2, ..., p_n); \quad (17)$$

$$P' = (p'_1, p'_2, ..., p'_n), \quad (18)$$

где $p'_i \equiv |(m_i - p_i)^{-1}|_{m_i}$;

$$A = A' = \sum_{i=1}^{n} a'_i \prod_{v=1}^{i-1} m_v = (a'_1, a'_2, ..., a'_n). \quad (19)$$

Результат обозначим как

$$X = (x_1, x_2, ..., x_n). \quad (20)$$

С учетом формул (15)—(21) представим вариант алгоритма умножения чисел A и B по Монтгомери.

Алгоритм 4

Ввод: B, P — в СВ1, A — в ССО, $M = \prod_{i=1}^{n} m_i$ для СВ1, A, $B < P < M/4$, НОД(P, M) = 1, $\tilde{M} = \prod_{i=n+1}^{n+u} m_i$ для СВ2, $\tilde{M} > 2P$.

Вывод: $X \equiv |ABM^{-1}|_P$

$X := 0$
for $i = 1$ to n

$$t_i := |a_i' b_i|_{m_i}$$
$$s_i := |x_i + t_i|_{m_i}$$
$$q_i' := |s_i(m_i - p_i)^{-1}|_{m_i}$$
$$X' := X + a_i' B$$
$$X := X' + q_i' P$$
$$X := X/m_i = Xm_i^{-1}$$

Расширение представления X из СВ2 в СВ1
end

В сравнении с двоичным алгоритмом 2 данный алгоритм показывает более высокую эффективность, поскольку его сложность эквивалентна примерно $6n$ (вместо $2n^2$) умножений и $4n$ сложений (включая процедуру расширения).

Отметим, что алгоритм по характеру и последовательности операций подобен двоичному алгоритму. Именно поэтому он не демонстрирует в полной мере достоинства СВ по сравнению с позиционными системами.

Параллельный способ

Другой подход в большей степени отвечает возможностям СВ, которая обеспечивает параллельную обработку числовых цифр (вычетов) без межразрядных переносов. Это свойство приобретает все большее значение по мере возрастания числовых данных. Здесь мы имеем как раз этот случай. Поэтому самый эффективный способ реализации (15) состоит в выполнении модулярных операций над полноразрядными числами, а не отдельными позиционными разрядами, т. е. в прямом выполнении алгоритма 3, более близкого по характеру к чисто математической интерпретации (15) и языкам высокого уровня. Данный подход предполагает наличие разрядной сетки для кодирования полноразрядных чисел в СВ.

Сложность этого алгоритма может быть оценена как $n + 3$ умножений и $n + 1$ сложений (включая второе результирующее расширение СВ2) или $2n + 3$ умножений и $2n + 1$ сложений, включая оба расширения СВ1 и СВ2. Первая оценка относится к варианту алгоритма, предполагающему обработку чисел удвоенной длины, т. е. в интегрированном диапазоне СВ1 и СВ2 $M^+ = M\tilde{M}$. В обеих оценках слагаемые, включающие n, определяют сложность немодулярных процедур расширения и основную вычислительную нагрузку алгоритма. Этот пример наглядно демонстрирует зависимость степени снижения потенциально высокой эффективности СВ от удельного веса немодулярных процедур в вычислительном процессе.

Метод Монтгомери на основе смешанных систем

Поскольку смешанные системы (СС), как и СВ, обеспечивают возможность параллельной обработки числовых цифр без межразрядных переносов, оптимальным для них является алгоритм, подобный алгоритму 3, т. е. вариант обработки полноразрядных чисел с максимальной эффективностью для метода Монтгомери. Отличие состоит лишь в том, что смешанные системы объединяют две составляющие: позиционную СС и изоморфную к ней (ИСС). Все модулярные операции выполняются в ИСС. Однако структура позиционной СС предполагает, как и в случае классической позиционной системы (например, двоичной), выбор в качестве диапазонов M и $\tilde{M}$ степеней модуля (основания) $M = m^{D_1}$, $\tilde{M} = m^{D_2}$, которые в сумме составляют общий диапазон $M^+ = M\tilde{M}$, покрывающий максимально возможное значение числителя в (15):

$$M^+ > P^2 + PM > 3P^2,$$

где $\tilde{M} \geqslant M > 2P$.

Произведения $T = AB$ и TP', участвующие в вычислении Q по модулю M, могут выйти за пределы диапазона M. В этом случае необходимо отбросить (обнулить) разряды этих произведений в позиционной СС, начиная с номера D_1 в общем диапазоне M^+. Поскольку все действия выполняются в ИСС, обнуление осуществляется с помощью двух операций в соответствии с [3]:

- простой сдвиг на D_2 разрядов в сторону старших разрядов (искомые D_1 разрядов Q займут старшие разряды диапазона M^+). Сдвиг на один разряд выполняется с помощью одного сложения (вычитания) и одного умножения на модуль m. Таким образом, сдвиг на D_1 разрядов требует D_1 умножений и столько же сложений, т. е. столько же, сколько требуется для немодулярной операции расширения для CB;
- циклический сдвиг в ту же сторону на D_1 разрядов (D_1 разрядов Q займут требуемую позицию в диапазоне M); данный сдвиг выполняется с помощью простой модулярной операции умножения на M.

После вычисления числителя в (15), кратного M, его деление на M выполняется простой операцией умножения на мультипликативную инверсию $|M^{-1}|_{M^+}$.

Резюмируя сказанное выше, можно сказать, что сложность алгоритма на основе смешанной системы примерно такая же, как и для СВ. Преимущество СС по сравнению с СВ состоит в ее однородности (истинной модульности), поскольку для этой системы используется единый простой или составной модуль (основание) m. Ниже приводится алгоритм умножения Монтгомери на основе СС.

Алгоритм 5

1. $T := AB$.
2. $T' := |T|_M$ (сдвиги для определения вычета T по модулю M).
3. $Q' := T'P'$.
4. $Q := |Q'|_M$ (сдвиги для определения вычета Q' по модулю M).
5. $U := QP$.
6. $C' := T + U$.
7. $C := C'|M^{-1}|_{M^+}$.
8. if $C > P$ then $C := C - p$.

Сдвиги в данном алгоритме могут быть заменены вычитанием из T' и Q' чисел соответственно $[AB/M]M$ и $[T'P'/M]M$, определяемых разрядами за пределами диапазона M.

Для покрытия сверхбольших чисел, с которыми оперируют криптографические методы, необходимы модули m соответствующей величины, поскольку максимально возможное число разря-

дов в смешанных системах ограничивается конгруэнцией вида

$$|x^D \pm 1|_m \equiv 0. \qquad (21)$$

Чтобы обойти это ограничение, в качестве модулей СС могут быть использованы теоретически любые целочисленные степени простых чисел.

При кодировании показателя E степени A^E в двоичной системе возможный вариант алгоритма возведения в степень может быть представлен цепочкой умножений Монтгомери в следующем виде.

Алгоритм 6

Ввод: $A < P < M \leqslant \tilde{M}$; $E = \sum_{i=0}^{n-1} e_i$; $M^+ = M\tilde{M}$.

Вывод: $X = |A^E|_P$.

$i := n - 1$; $X := 1$
While $i \geqslant 0$ and $e_i = 0$ do
$i := i - 1$
If $i \geqslant 0$ *d*0
$X := A$
$i := i - 1$
While $i \geqslant 0$ do
$X := |X * X|_P$
If $e_i = 1$ then
$X := |X * A|_P$
$i := i - 1$
End

Заключение

Экспоненциальные вычисления находят широкое применение в различных компьютерных приложениях, связанных, например, с криптографией. Многие криптографические методы включают модулярные экспоненциальные вычисления, характеризуемые параметрами больших и сверхбольших величин с длиной от нескольких сотен до нескольких тысяч битов. Естественно, с возрастанием длины операндов двоичные методы для возведения в степень по методу Монтгомери становятся все менее эффективными.

Использование модулярных числовых систем, таких как системы вычетов и смешанные системы, позволяет оперировать не полноразрядными числами названной выше размерности, а вычетами значительно меньшей величины при отсутствии межразрядных переносов. Криптографические алгоритмы на основе этих систем дают возможность проводить необходимые вычисления одновременно над всеми совокупностями вычетов, т. е. над целыми числами, а не отдельными машинными словами, составляющими числа в случае применения двоичных технологий. Как было показано, эффективность модулярных числовых систем по сравнению с двоичными системами в криптографических приложениях существенно выше, несмотря даже на то, что процедуры умножения и возведения в степень по Монтгомери включают "неудобные" немодулярные операции расширения или сдвиги.

Отметим, что здесь предпочтение может быть отдано смешанным системам, которым в отличие от систем вычетов с модулями разной величины присущи свойства привычных позиционных систем, включая однородность (модульность) структуры представления чисел. Эти свойства, с одной стороны, обеспечивают более высокую технологичность и, с другой стороны, более широкую применимость систем СС вследствие более естественного формирования в них структур чисел в целочисленном формате и с плавающей точкой. Конечно, если для некоторого вычислительного средства система вычетов выбрана по другим соображениям, она может быть эффективно использована и для криптографических приложений.

Другое важное достоинство модулярных числовых систем состоит в возможности создания на их основе высокоэффективных средств коррекции ошибок не только при передаче, но и при обработке информации. Для позиционных, в том числе и двоичных, систем не существует технологии с подобными характеристиками.

Таким образом, модулярные числовые системы обеспечивают сочетание высокой степени безопасности и надежности обработки и передачи информации.

Список литературы

1. **Montgomery P.** Modular multiplication without trial division // Mathematics of computation. Apr. 1985. Vol. 44. N 170. P. 519—521.

2. **Wu H.** Montgomery multiplier and squarer in GF(2^m) // Proc. Cryptographic hardware and embedded systems (CHES 2000). Aug. 2000. P. 264—276.

3. **Жуков О. Д.** Численные процедуры в компьютерной модулярной алгебре // Информационные технологии. 2003. № 10. С. 22—26.

УДК 004.056

П. Д. Зегжда, д-р техн. наук, проф., зав. каф.,
Д. П. Зегжда, д-р техн. наук, проф.,
М. О. Калинин, канд. техн. наук, доц.,
E-mail: max@ssl.stu.neva.ru
ГОУ "Санкт-Петербургский государственный политехнический университет"

Реализация логического подхода к оценке безопасности состояния ОС семейства MS Windows в системе анализа настроек безопасности "Декарт"

Рассмотрен подход к оценке защищенности информационных систем путем представления и проверки безопасности их состояний с использованием логики предикатов. Представлен пример применения логического анализа настроек безопасности. Предложенный подход является основой для разработки средств автоматического доказательства безопасности информационных систем. Он реализован с помощью механизмов вычислительного ядра системы "Декарт", которая в автоматизированном режиме выполняет анализ безопасности настроек операционной системы семейства Windows.

Ключевые слова: *автоматизированный анализ, оценка безопасности, критерий, логика предикатов, настройки безопасности, информационная безопасность, уязвимость.*

Современные операционные системы (ОС) для защиты от деструктивных информационных воздействий включают разнообразные механизмы безопасности: контроль и управление доступом, шифрование трафика и хранилищ информации, биоидентификация, виртуализация и другие. Программные ошибки, допущенные при проектировании и реализации средств защиты, исправляют путем обновления программного обеспечения. Несоблюдение мер безопасности или некорректная реализация администраторами политик информационной безопасности при настройке ОС вследствие чрезвычайной сложности и перегруженности механизмами безопасности приводят к тому, что многокомпонентная эшелонированная защита становится неэффективной. Просчеты администрирования приводят к нарушениям безопасности ОС, проявляющимся в виде так называемых уязвимостей настроек безопасности. Типичными примерами таких нарушений являются разрешение модификации системных каталогов и использование настроек, задаваемых по умолчанию. По этой причине ОС и построенные на их основе системы обработки информации нуждаются в регулярно проводимом доказательстве безопасности, что позволит гарантировать выполнение необходимых ограничений на доступ к информации.

Авторами предложен подход к автоматизированному доказательству защищенности информационных систем, основанный на логическом описании состояния настроек безопасности ОС, алгоритмическом моделировании контроля и управления доступом к информации и проверке состояния ОС на соответствие критериям безопасности. Логическое описание состояния настроек безопасности ОС — это представленная в форме предикатов абстракция состояния системы в контексте модели контроля и управления доступом, реализованной в ОС. Правила модели определяют ограничения, накладываемые ОС на принятие решения о доступе. Условия безопасности ОС формализуются в виде логических критериев. Такие условия позволяют разделить множество состояний на безопасные и небезопасные.

Процесс оценки соответствия системных состояний критериям безопасности в контексте настоящей работы будем называть разрешением проблемы безопасности. Формально проблема безопасности представляется как $\Lambda = \{M, \Sigma, D\}$.

Модель контроля и управления доступом M — кортеж $\{S, R, C\}$, где S — множество состояний модели (состояний безопасности); R — множество правил контроля и управления доступом, описывающих условия перехода между состояниями безопасности; C — множество критериев безопасности, описывающих условия безопасности состояний. Множество R состоит из предикатов r: $S \times S \to$ {ИСТИНА, ЛОЖЬ}, которые истинны, если разрешен переход из состояния $s \in S$ в состояние $s' \in S$. Множество C состоит из предикатов c: $S \to$ {ИСТИНА, ЛОЖЬ}, которые истинны, если состояние безопасно.

Система Σ — кортеж $\{S^\Sigma, T, s^\Sigma_{\text{init}}, Q\}$, где S^Σ — множество состояний системы; D — функция соответствия состояний, D: $S^\Sigma \to S$; Q — множество запросов, обрабатываемых системой; T — функция перехода из состояний в состояние, T: $Q \times S^\Sigma \to S^\Sigma$, $s^\Sigma_{i+1} = T(q, s^\Sigma)$; s^Σ_{init} — начальное состояние системы. Состояние s^Σ достижимо в системе $\Sigma = \{S^\Sigma, T, s^\Sigma_{\text{init}}, Q\}$ тогда и только тогда, когда существует последовательность $<(q_0, s^\Sigma_0), ..., (q_n, s^\Sigma_n)>$, в которой $s^\Sigma_0 = s^\Sigma_{\text{init}}$, $s^\Sigma_n = s^\Sigma$, а $s^\Sigma_{i+1} = T(q_i, s^\Sigma_i)$, $0 \leqslant i \leqslant n$. Система Σ представляет собой конечный автомат, описывающий пути изменения состояний в зависимости от текущего состояния и входных запросов, при условии конечности общего возможного числа состояний.

Таким образом, система Σ является безопасной согласно правилам R модели M и проверяемым критериям C тогда и только тогда, когда выполняются условия:

1) $\forall s_i^\Sigma,\ s_{i+1}^\Sigma \in S^\Sigma$: $s_{i+1}^\Sigma = T(q,\ s_i^\Sigma)\ \exists s_i$,

s_{i+1}: $s_i = D(s_i^\Sigma)$, $s_{i+1} = D(s_{i+1}^\Sigma)$,

$\forall r \in R$: $r(s_i, s_{i+1})$ = ИСТИНА;

2) $\forall c \in C : c(D(s_{\text{init}}^\Sigma))$ = ИСТИНА;

3) $\forall s_i^\Sigma \in S^\Sigma$, $s_i^\Sigma = T(q_{i-1}, T(..., T(q_0, s_{\text{init}}^\Sigma)...))$,

$0 \leqslant i < n$: $\exists s_i$, $s_i = D(s_i^\Sigma)$,

$\forall c \in C : c(s_i)$ = ИСТИНА.

Тем самым система безопасна, если переходы между состояниями выполняются по правилам модели контроля и управления доступом и в каждом состоянии выполняются условия критериев безопасности.

Доказательство проблемы безопасности для общего случая выполнено в рамках мандатных моделей контроля и управления доступом [1] (известно как "Общая теорема безопасности"). Для дискреционных моделей показано, что проблема безопасности неразрешима в общем случае [2]. Однако проведение доказательства возможно в частном случае путем проверки состояний на соответствие критериям безопасности.

Поскольку абсолютное большинство защищенных информационных систем, в том числе ОС Windows и UNIX, реализуют именно дискреционные модели, то доказательство их безопасности является актуальной задачей. В практическом плане доказательство безопасности для реальных систем эквивалентно установлению корректности администрирования безопасности, так как согласно приведенным ранее условиям требуется проверка истинности критериев, эквивалентная проверке отсутствия уязвимостей настроек безопасности. Таким образом, для доказательства безопасности системы необходимо иметь формальное задание состояния настроек безопасности, модели контроля и управления доступом, реализованной в системе, и критериев безопасности. Однако решение задачи путем перебора состояний рабочих станций и серверов в сети, их проверки на наличие нарушений критериев безопасности и, как следствие, обеспечение корректного администрирования, невыполнимо вручную. Например, для сети на базе ОС Windows с учетом числа объектов защиты, пользователей и групп, их атрибутов безопасности (владения объектом, прав доступа, других) и доменных настроек, число возможных комбинаций настроек безопасности, а значит и состояний, исчисляется миллионами. Авторами разработан автоматизированный инструментарий, который помогает справиться с объемом обрабатываемых состояний и множеством проверок для компьютеров локальной сети, — система "Декарт". Она предназначена для анализа настроек безопасности, составления и проверки критериев безопасности, обнаружения нарушений безопасности, связанных с некорректным администрированием систем, функционирующих под управлением ОС семейства Windows.

Метод логического моделирования, использованный в системе "Декарт", позволяет добиться простоты практической реализации, проведения автоматической проверки критериев и обнаружения причин их невыполнения (уязвимостей настроек безопасности). Удобство метода заключается в том, что сама процедура доказательства безопасности анализируемой системы задана в терминах логики предикатов. Способ анализа посредством логических методов основан на описании системы и правил ее поведения в виде логических структур, задания на их основе правил логического вывода. Формальная база такого метода, а именно математическая логика, теория автоматов и автоматическое доказательство теорем, позволяет получать спецификацию системы, применять аппарат предикативного представления и резолюции (логического вывода) для моделирования и анализа состояния механизмов обеспечения безопасности информационных систем.

Рассмотрим использование принципов логического моделирования и вывода, реализованных в системе "Декарт", при проверке безопасности ОС Windows на примере проверки прав доступа пользователей к файлу-шаблону редактора MS Word *normal.dot*. В системе "Декарт" состояние ОС представляется как конфигурация настроек безопасности (субъектов, объектов и их атрибутов), зафиксированных в некоторый момент времени (множество S в условиях безопасности). Например, средствами системы "Декарт" фиксируется состояние файла:

```
subject ('s-1-5-21-...-500',
[type (user), name ('admin'),
privileges ([security, ..., remotelogon]),
groups (['s-1-5-32-544'])]).
subject ('s-1-5-32-545',
[type(group), name(['users']),
privileges ([shutdown, ..., networklogon])]).

object ('c : \\...\\templates\\normal.dot',
[type (file), owner (['s-1-5-21-...-500']),
inheritance(yes)],
[['s-1-5-21-...-500', [0, 1, 2, 3, 4, 5, 6, 7, 8, 16,
17, 18, 19, 20]],
['s-1-5-18', [0, 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18,
19, 20]],
['s-1-5-32-544', [0, 1, 2, 3, 4, 5, 6, 7, 8, 16, 17,
18, 19, 20]],
['s-1-5-32-545', [0, 1]]]).
```

В примере заданы предикаты описания пользователя, группы и файла. Каждый предикат инкапсулирует информацию об идентификаторе субъекта, а также его атрибутах безопасности. Например, пользователь *admin* имеет идентификатор безопасности S-1-5-21-...-500, обладает привилегиями и входит в группу *Administrators*. Второй предикат описывает группу *Users* и ее привилегии. Третье выражение описывает файл *normal.dot* и его атрибуты: SID владельца, флаг наследования прав и права доступа, заданные в виде списка в формате "SID-биты прав". Логическое представление скрыто от пользователя и используется на этапе проверки критериев безопасности. Оператору предоставляется графическая информация о текущем состоянии ОС.

Описание функционирования подсистемы контроля и управления доступом записывается в виде логической программы, в которой правила определяют логические условия, заданные на множестве настроек безопасности:

```
allow_file_read(U, F) :-
% Get effective permissions list for user U.
effective_permissions(U, F, EPL),
% Get privileges granted to user U.
privileges_list(U, PL),
...
% Privilege "Backup files and directories".
(member(backup, PL);
% Permission Read data is granted.
member (0, EPL),
% Permission Read attributes is granted.
(member(7, EPL); ...).
```

В приведенном фрагменте доступ на чтение файла предоставляется, если пользователь имеет, например, бит-разрешение на чтение, установленное в дискреционном списке контроля доступа, или у пользователя есть привилегия резервного копирования [3]. Таким образом, выполняется замена атрибутов безопасности, определяющих возможность доступа, на множество эквивалентных им эффективных разрешений, указывающих, какие виды доступа разрешено выполнять в данном состоянии системы (множество R в условиях безопасности) согласно реализованной модели безопасности.

Для ОС семейства Windows множество атрибутов безопасности (например, владелец, группа, привилегия) образуют иерархию из 15 уровней, на каждом из которых выполняется проекция соответствующего атрибута на множество эффективных разрешений. В общем случае для i-го уровня иерархии множество эффективных разрешений $EP_i = f(subject, EP_{i-1}, A_i)$, где $2 \leqslant i \leqslant m$, m — число уровней иерархии. Функция f вычисляет множество эффективных разрешений EP_i, доступных пользователю или группе *subject*, на i-м уровне с учетом влияния атрибута безопасности A_i. Например, для уровня 5, на котором учитываются запреты, множество EP_5 вычисляется следующим образом:

$$EP_5(subject) = EP_4^+(subject)\ XOR\ EP_4^-(subject),$$

где EP_4^+ — EP_4, вычисленный для разрешений; EP_4^- — EP_4 вычисленный для запретов; *subject* — пользователь или группа;

$$EP_4(subject) = EP_3(subject)\ AND\ EP_3(\{groups_j\} | \forall i\ groups_j \ni subject);$$

$$\ldots;\ EP_1(subject) = \{bits_k\},$$

где $\{bits_k\}$ — множество битов доступа в маске прав дискреционного списка прав доступа данного субъекта, приведенного в атрибутах объекта. Наличие такой процедуры вычисления эффективных разрешений позволяет свести задачу проверки критериев безопасности в заданном состоянии к сравнению множеств — вычисленного множества эффективных разрешений и множества прав, заданных в критерии. Жесткость иерархии и взаимооднозначность соответствия "атрибут—разрешения" позволяют при обнаружении нарушения, т. е. при несовпадении множеств прав, заданных в критерии и рассчитанных для состояния системы, определить атрибут, привнесший "нарушающие" права в качестве причины нарушения.

Проверяемый критерий (элемент множества C из условий безопасности), по сути, представляет собой логическую цель, которая должна быть согласована на множестве настроек безопасности, например:

```
criterion ('Crit#1: Users can not edit Normal.dot', mask,
[object ('c: \\...\\templates\\normal.dot'),
check_zone('this_object'),
's-1-5-32-544' (0, 1, 2, 3, 4, 5, 7, 8, 6, 16, 17,
18, 19, 20),
's-1-5-18' (0, 1, 2, 3, 4, 5, 7, 8, 6, 16, 17, 18,
19, 20)]).
```

Данный логический предикат описывает критерий, заданный относительно файла *normal.dot*. Тип критерия *mask* указывает на то, что это критерий проверки дискреционного списка прав доступа, а именно проверки установленных битов для конкретных субъектов. В критерии задано условие безопасности: только субъект *SYSTEM* (S-1-5-18) и группа *Administrators* (S-1-5-32-544) могут иметь полный доступ к файлу. Отклонение от заданного условия считается нарушением безопасности. Критерии в системе "Декарт" задаются графически путем заполнения формы критерия элементами состояния и задания параметров проверки. Перед проверкой критерии автоматически формализуются.

Проверка критерия системой "Декарт" заключается в проверке эквивалентности множества эффективных прав, которое доступно субъекту в системе, множеству прав, указанных в проверяемом критерии.

Рассмотрим механизм проверки критериев. Пусть R_{all} — множество всех прав доступа, которые может иметь пользователь для объекта некоторого типа. Права R_{PA} — необходимые, т. е. множество прав доступа, которые должны быть предоставлены пользователю, чтобы система была безопасна. Данное множество прав указывают в критерии безопасности. Права R_S — множество эффективных прав, которые разрешено иметь пользователю в ОС с учетом влияния настроек безопасности, $R_S \subseteq R_{\text{all}}$. Данное множество прав рассчитывается по иерархии атрибутов безопасности на основе заданного состояния системы. Следующие множества прав являются правами "нарушителями безопасности". Права R_{excess} — "лишние". Так именуется множество прав, разрешенных настройками безопасности, но не являющихся необходимыми правами. Права R_{miss} — "недостающие". К числу таких прав относится множество прав, которых недостает пользователю для получения множества прав, разрешенных настройками безопасности. Тогда для критериев можно сформулировать в терминах множеств следующее условие безопасности системы: система безопасна (согласно критерию), если множество прав доступа R_S, разрешенных пользователю настройками безопасности, эквивалентно множеству необходимых прав R_{PA}, $R_S = R_{PA}$ (рис. 1).

Для выявления уязвимости необходимо провести следующие проверки.

1. $R_{\text{excess}} = R_S - R_{PA}$. Если $R_{\text{excess}} \neq \varnothing$, то система уязвима, так как пользователю действующими настройками разрешены права, которые ему запрещены (рис. 2).

2. $R_{\text{miss}} = R_{PA} - R_S$. Если $R_{\text{miss}} \neq \varnothing$, то система уязвима, так как пользователю действующими настройками запрещены права, которые ему необходимы (рис. 2).

Для проверки выполнения условия необходимо выполнить оба теста, чтобы выявить все несоответствия между множеством прав доступа, указанных в критерии, и множеством прав доступа, разрешенных настройками. Если при выполнении теста 1 обнаружено, что множество R_{excess} не является пустым, то система уязвима, так как настройки безопасности разрешают пользователю права доступа из множества R_{excess}. Если при выполнении теста 2 обнаружено, что множество R_{miss} не является пустым, то система уязвима, так как настройки безопасности не разрешают пользователю права доступа из множества R_{miss}.

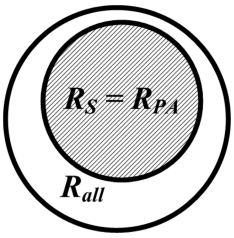


Рис. 1. Проверка условия

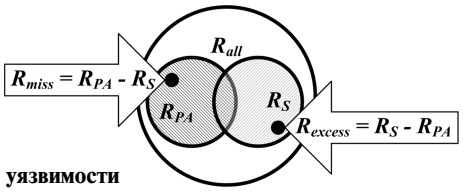


Рис. 2. Выявление уязвимости

Результатом проверки критерия является отчет, содержащий сведения о выявленном нарушении, например, в текстовом виде:

```
*** SAFETY RESOLUTION [CRITERION #1] ***
Users are not allowed to edit
the file 'c:\\...\\templates\\normal.dot'
CHECKING RESULTS
> > VIOLATION DETECTED:
subject
      group < Users > has unauthorized
      permissions bits [0, 1]
                      [Read Data, Write Data]
for object(s):
file c :\\...\\templates\\normal.dot
```

Проверяемый критерий не выполнился. Этот факт означает наличие уязвимости в настройках безопасности системы. Анализ причин уязвимости, выполняемый системой "Декарт" по иерархии атрибутов безопасности, определяющих эффективные права, показывает, что члены группы *Users* могут выполнять доступ на чтение и запись.

Таким образом, система "Декарт" позволяет проводить оценку защищенности системы путем проверки критериев безопасности и выявления уязвимостей настроек безопасности в заданном состоянии системы с использованием метода логического программирования. Набор встроенных критериев реализует множество проверок ОС семейства Windows на соответствие различным требованиям.

Список литературы

1. **LaPadula L. J., Bell D. E.** Secure computer systems: A mathematical model, ESD-TR-278, Vol. 2, The Mitre Corp., 1973.

2. **Harrison M. H., Ruzzo W. L., Ullman J. D.** Protection in operating systems // Communications of the ACM. 1976. 19 (8). P. 461—471.

3. **Руссинович М., Соломон Д.** Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер-класс. СПб.: Питер, М.: Изд. дом "Русская редакция", 2005. 992 с.

УДК 519.8

А. А. Колоколов, д-р физ.-мат. наук, проф.,
зав. лабораторией,
А. В. Адельшин, канд. физ.-мат. наук,
ст. науч. сотр.,
Омский филиал Института математики
им. С. Л. Соболева СО РАН,
Д. И. Ягофарова, канд. физ.-мат. наук,
ст. преподаватель,
Омский государственный университет

Решение задачи выполнимости с использованием метода перебора *L*-классов

Рассматривается задача выполнимости логической формулы в конъюнктивной нормальной форме. Предлагается комбинаторный алгоритм решения указанной задачи, основанный на моделях и методах целочисленного линейного программирования и переборе L-классов. Приводятся результаты вычислительного эксперимента, проведенного для различных серий задач из специализированной электронной библиотеки SATLIB.

Ключевые слова: *задача выполнимости, целочисленное программирование, метод перебора L-классов.*

1. Введение и постановка задачи

Значительное число исследований в области дискретной оптимизации посвящено задаче выполнимости логической формулы и ее обобщениям [1, 2, 4, 5, 7—12]. Широкое практическое применение этих задач в экономике, управлении, проектировании и других областях является стимулом для разработки алгоритмов их решения. Обзор алгоритмов для решения задачи выполнимости представлен в работе [11].

В работах [1, 4, 5, 7, 8, 12] проводились исследования задач выполнимости и максимальной выполнимости на основе моделей целочисленного линейного программирования (ЦЛП) и *L*-разбиения. В данной статье описывается алгоритм решения задачи выполнимости, основанный на методе перебора *L*-классов. Приводятся результаты вычислительного эксперимента, выполненного с использованием задач из электронной библиотеки [13].

Рассмотрим постановку *задачи выполнимости* (SAT). Пусть $x_1, ..., x_n$ — переменные, принимающие значение *истина* или *ложь*. Под литералом понимается либо переменная x_j, либо ее отрицание $\bar{x}_j$. Пусть логическая формула F представляет собой конъюнкцию формул (скобок) C_i, $i = 1, ..., m$, каждая из которых является дизъюнкцией литералов. Требуется определить, выполнима ли формула F, т. е. существует ли такой набор значений переменных, при котором F принимает значение *истина*. Известно, что задача SAT является *NP*-полной, а в случае, когда каждая скобка содержит не более двух литералов, задача полиномиально разрешима [10]. Многие известные задачи теории графов, построения расписаний, криптографии могут быть сформулированы как задача выполнимости некоторой логической формулы в конъюнктивной нормальной форме.

Важным обобщением задачи SAT является задача максимальной выполнимости (MAX SAT), состоящая в отыскании набора значений логических переменных, при котором суммарный вес выполненных скобок будет наибольшим (предполагается, что каждой скобке C_i соответствует неотрицательный вес c_i). Задачи оптимизации с логическими ограничениями находят применение при решении многих практических задач, в частности, при проектировании изделий сложной структуры [6].

Сформулируем SAT в виде задачи ЦЛП. Введем булевы переменные $y_1, ..., y_n$ такие, что y_j соответствует переменной x_j, а $(1 - y_j)$ — ее отрицанию, т. е. $y_j = 1$ тогда и только тогда, когда переменная x_j принимает значение *истина*. Рассмотрим пример логической формулы $F = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$ и запишем условие ее выполнимости. Для того чтобы первая скобка $(x_1 \vee x_2)$ была истинной; необходимо и достаточно выполнение неравенства $y_1 + y_2 \geqslant 1$. Аналогично, для второй и третьей скобок получим соответственно неравенства $y_1 + (1 - y_3) \geqslant 1$ и $(1 - y_1) + (1 - y_2) + y_3 \geqslant 1$. После преобразований имеем систему неравенств: $y_1 + y_2 \geqslant 1$, $y_1 - y_3 \geqslant 0$, $-y_1 - y_2 + y_3 \geqslant -1$. Таким образом, существование булевых значений переменных y_j, удовлетворяющих данной системе, равносильно условию выполнимости логической формулы F. В общем случае нетрудно показать, что условие выполнимости логической формулы F эквивалентно существованию решения системы:

$$\sum_{j \in C_i^+} y_j - \sum_{j \in C_i^-} y_j \geqslant 1 - |C_i^-|, \; i = 1, ..., m, \qquad (1)$$

$$0 \leqslant y_j \leqslant 1, \; j = 1, ..., n, \qquad (2)$$

$$y_j \in Z, \; j = 1, ..., n, \qquad (3)$$

где C_i^- и C_i^+ — множества индексов переменных, входящих в скобку C_i с отрицанием и без него соответственно, величина $|C_i^-|$ — мощность множества C_i^- .

Для получения задачи ЦЛП необходимо ввести целевую функцию. В качестве такой функции может быть вы-

брана, например, $f(y) = y_1 \to \max$ или $f(y) = \sum_{j=1}^{n} y_j \to \max$, где $y = (y_1, ..., y_n)$. В данной работе рассматривается лексикографическая постановка задачи ЦЛП, т. е. ищется лексикографически максимальный вектор y, удовлетворяющий системе ограничений (1)—(3).

2. Метод перебора *L*-классов

Введем необходимые определения и обозначения. Пусть $x, y \in R^n$, рассмотрим функцию $\eta(x, y) = \min\{i : x_i \neq y_i, i = 1, ..., n\}$. Точка x *лексикографически больше* точки y ($x > y$), если $x_k > y_k$ для $k = \eta(x, y)$. Запись $x \geqslant y$ означает, что $x > y$ или $x = y$.

Точки $x, y \in R^n$ ($x > y$) называются L-эквивалентными, если не существует отделяющей их точки $z \in Z^n$, т. е. такой, что выполняется $x \geqslant z \geqslant y$. Такое отношение порождает разбиение любого множества $X \subseteq R^n$ на непересекающиеся классы эквивалентности, которые называются *L-классами*. Соответствующее фактор-множество X/L называется *L-разбиением* множества X. Указанное разбиение имеет ряд полезных свойств [3], применяемых при разработке и исследовании алгоритмов целочисленного программирования, в частности, алгоритмов перебора L-классов. Отметим, что каждая точка $z \in Z^n$ образует отдельный L-класс, а остальные классы состоят из нецелочисленных точек и называются *дробными*.

В работе также используется лексикографическое сравнение множеств. Пусть $X, Y \subset R^n$. Будем считать, что $X > Y$, если для любых $x \in X$ и $y \in Y$ выполняется $x > y$.

Рассмотрим следующую лексикографическую постановку задачи ЦЛП:

$$\textit{найти } y^* = \operatorname{lexmax}(M \cap Z^n), \qquad (4)$$

где M — некоторый многогранник. Множество

$$M_* = \{y' \in M: y' > y \text{ для всех } y \in (M \cap Z^n)\}$$

называется *дробным накрытием* задачи (4), а M_*/L — ее *L-накрытием*.

Выделение этой части релаксационного множества задачи связано с тем, что в некоторых методах ЦП происходит последовательное исключение точек из M, т. е. мощность M_*/L можно рассматривать как показатель сложности задачи для данных алгоритмов.

Опишем идею метода перебора L-классов для задачи (4). Пусть L-накрытие задачи состоит из классов $V_1, V_2, ..., V_p$, причем $V_1 > V_2 > ... > V_p > y^*$, где y^* — лексикографически максимальное решение (если оно существует).

Основной шаг метода заключается в переходе от одного класса из M_*/L к другому. Обычно это осуществляется в порядке лексикографического убывания, начиная с лексикографически максимального L-класса. Этот процесс удобно интерпретировать с помощью "ленты", в которой каждая ячейка отвечает отдельному L-классу (рис. 1). Классы, соответствующие целочисленным точкам, показаны штриховкой. Алгоритм порождает последовательность точек $y^{(t)} \in M_*$, обладающую свойствами:

- $y^{(t)} > y^{(t+1)}$, $t = 1, 2, ..., s - 1$; $s \leqslant p$;
- все точки принадлежат различным L-классам.

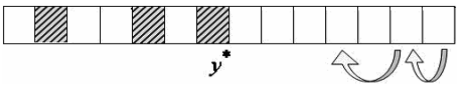


Рис. 1. "Лента" *L*-классов

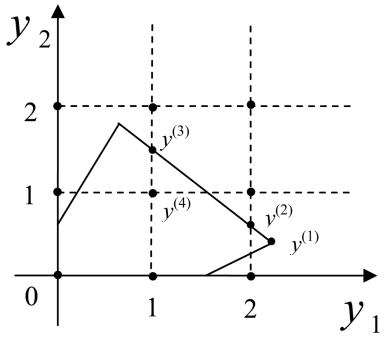


Рис. 2. Метод перебора *L*-классов

Пусть ищется лексикографически максимальная целочисленная точка многогранника M, изображенного на рис. 2. Покажем, как выглядит процесс решения указанной задачи алгоритмом перебора L-классов. L-накрытие задачи состоит из классов:

$$V_1 = \{y \in M : y_1 > 2\};\ V_2 = \{y \in M : y_1 = 2,\ 0 < y_2 < 1\};$$
$$V_3 = \{y \in M : 1 < y_1 < 2\};\ V_4 = \{y \in M : y_1 = 1,\ y_2 > 1\},$$

где $V_1 > V_2 > V_3 > V_4$. В ходе работы алгоритма будет построена последовательность точек $y^{(1)}, ..., y^{(4)}$ (см. рис. 2), где $y^{(1)}, y^{(2)}, y^{(3)}$ принадлежат соответственно элементам L-накрытия V_1, V_2, V_4; $y^{(4)} = (1,1)$ — оптимальное решение.

В общем случае для задачи ЦЛП переход от одного L-класса к другому выполняется путем решения некоторого числа задач линейного программирования. Для решения задачи выполнимости поиск представителей $y^{(t)}$ указанных классов можно осуществлять, не решая задач линейного программирования, что заметно повышает эффективность алгоритма. Это основано на следующем свойстве: в каждом L-классе многогранника задачи SAT содержится *полуцелочисленная точка*, т. е. точка со значениями координат из множества $\{0, \frac{1}{2}, 1\}$.

3. Алгоритм перебора *L*-классов для задачи выполнимости

Пусть M — многогранник задачи выполнимости, определяемый системой ограничений (1)—(2). Приведем описание предлагаемого нами алгоритма перебора L-классов для задачи SAT, который за конечное число шагов либо находит оптимальное решение задачи (4), либо указывает на невыполнимость формулы. Основные процедуры алгоритма: поиск представителей L-классов и переход от текущего L-класса к следующему. Поиск представителей L-классов происходит с помощью специальной комбинаторной *процедуры W*, в процессе выполнения которой строится последовательность недопустимых целочисленных точек с координатами, равны-

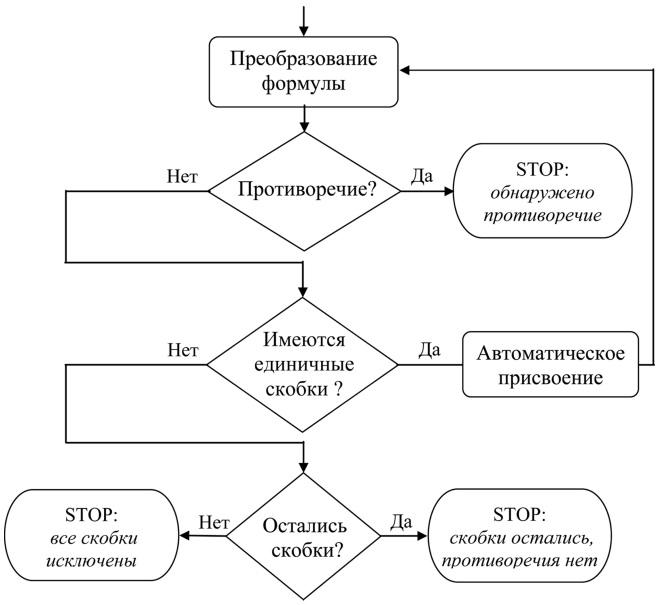


Рис. 3. Блок-схема процедуры упрощения формулы

ми 0, 1 или 2. Отметим, что значение 2 используется для обеспечения лексикографической монотонности указанной последовательности. С этой целью можно также выбрать любое число, большее 1. Переход от одного L-класса к следующему происходит в порядке лексикографического убывания, начиная с лексикографически максимального L-класса.

В данном алгоритме регулярно используется *процедура упрощения формулы*, блок-схема которой изображена на рис. 3. Рассмотрим более подробно некоторые ее этапы.

Процедура упрощения формулы

Блок *"Преобразование формулы"* выполняется после присвоения некоторой переменной y_j значения 0 или 1.

- Если $y_j = 1$, то исключаем из формулы литерал $\overline{x}_j$ и скобки, содержащие литерал x_j;
- Если $y_j = 0$, то исключаем из формулы литерал x_j и скобки, содержащие литерал $\overline{x}_j$.

Блок *"Противоречие?"*. Проверяем, существуют ли одновременно единичные скобки x_j и $\overline{x}_j$ для некоторого j. Наличие такой пары скобок означает, что получено противоречие.

Операция *"Автоматическое присвоение"* выполняется после того как в формуле найдена какая-либо единичная скобка. Возможны следующие случаи:

- если в формуле имеется скобка вида $\overline{x}_j$, то переменной y_j "*автоматически*" присваиваем значение 0;
- если в формуле имеется скобка вида x_j, то переменной y_j "*автоматически*" присваиваем значение 1.

Предполагается, что перед началом работы алгоритма формула не содержит единичных скобок. В противном случае число переменных и скобок можно сократить путем присвоения таким переменным соответствующих значений.

Алгоритм завершает работу в двух случаях: если получено оптимальное решение задачи (4) или если не удается найти очередной L-класс.

Алгоритм LCE

Шаг 0 (поиск представителя первого L-класса). Выполняем *процедуру W*, начиная с недопустимой точки $w^{1(0)} = (2, ..., 2)$. Если находим полуцелочисленную точку $y^{(1)} \in V_1$, то переходим к шагу 1 первой итерации.

Итерация t ($t \geqslant 1$).

Шаг 1. Начинаем с точки $y^{(t)} = (a_1^{(t)}, ..., a_{s-1}^{(t)}, \frac{1}{2}, y_{s+1}^{(t)}, ..., y_n^{(t)})$, где $a_i^{(t)} \in \{0, 1\}$, $i = 1, ..., s-1$; $y_j^{(t)} \in \{0, \frac{1}{2}, 1\}$, $j = s+1, ..., n$. Первую дробную координату $y_s^{(t)}$ округляем до 0. Выполняем *процедуру упрощения формулы*. В ходе этого преобразования некоторые координаты $y_j^{(t)}$ ($j = s+1, ..., n$) "автоматически" примут значения 0 или 1. Обозначим полученную точку $\tilde{y}^{(t)}$. После упрощения формулы возникает один из следующих случаев.

- *Все скобки исключены.* Рассматривая точку $\tilde{y}^{(t)}$, переходим на шаг 4.
- *Скобки остались, противоречие не обнаружено.* Из вектора $\tilde{y}^{(t)}$ строим недопустимую точку $w^{1(t)} = (a_1^{1(t)}, ..., a_s^{1(t)}, 0, w_{s+1}^{1(t)}, ..., w_n^{1(t)})$, где $w_j^{1(t)} \in \{0, 1, 2\}$, причем, если $\tilde{y}_j^{(t)} = \frac{1}{2}$, то $w_j^{1(t)} = 2$, $j = s+1, ..., n$. Переходим на шаг 2.
- *Обнаружено противоречие.* Возвращаемся к точке $y^{(t)}$, переходим на шаг 3.

Шаг 2. Выполняем *процедуру W*, начиная с недопустимой точки $w^{1(t)}$. Если находим полуцелочисленную точку $y^{(t+1)} \in V_{t+1}$, то переходим к шагу 1 итерации $(t+1)$.

Шаг 3. Среди координат вектора $y^{(t)}$ ищем максимальную по номеру координату $y_p^{(t)} = 1$, которой это значение не было присвоено "автоматически" ($p < s$). Возможны два случая.

1. Такая координата *существует*, тогда продолжаем просматривать вектор $y^{(t)}$. Находим все переменные $y_j^{(t)}$ ($p < j \leqslant n$), которым значение 0 или 1 было присвоено после назначения $y_p^{(t)} = 1$. Изменяем их значения на 1/2. Значение координаты с номером p также уменьшаем до 1/2. В результате получаем точку $y^{(t+1)} \in V_{t+1}$ и переходим на шаг 1 итерации $(t+1)$.

2. Такой координаты *не существует*. В этом случае алгоритм завершает работу: *формула невыполнима*.

Шаг 4. Алгоритм завершает работу: *формула выполнима*. Оптимальное решение y^* задачи (4) строим из текущего вектора $\tilde{y}^{(t)}$ следующим образом:

$$y_j^* = \begin{cases} \tilde{y}_j^{(t)}, \text{ если } \tilde{y}_j^{(t)} \in \{0, 1\}, \\ 1, \text{ если } \tilde{y}_j^{(t)} = \dfrac{1}{2}, \end{cases}$$

для $j = 1, ..., n$.

Отметим, что на шаге 4 координатам, равным 1/2, можно произвольным образом присвоить булевы значе-

ния. Это дает множество различных допустимых решений задачи (4).

Процедура *W*

Полагаем $\gamma = 1$.

Итерация γ. Начинаем с вектора $w^{\gamma(t)} = (a_1^{\gamma(t)}, ..., a_{s-1}^{\gamma(t)}, 0, w_{s+1}^{\gamma(t)}, ..., w_n^{\gamma(t)})$, где $a_i^{\gamma(t)} \in \{0, 1\}$, $i = 1, ..., s-1$; $w_j^{\gamma(t)} \in \{0, 1, 2\}$, $j = s+1, ..., n$. Для получения представителя первого L-класса начинаем с вектора $w^{1(0)} = (2, ..., 2)$. Среди координат этого вектора ищем первую координату $w_q^{\gamma(t)}$, значение которой равно 2, и изменяем его на 1. Выполняем *процедуру упрощения формулы*, в ходе которой некоторые компоненты вектора "автоматически" примут значения 0 или 1. Обозначим полученный вектор $\tilde{w}$. После упрощения формулы возникает один из следующих случаев.

- *Все скобки исключены.* В векторе $\tilde{w}$ заменяем все координаты, равные 2, на 1/2. Полученную точку обозначаем $\tilde{y}^{(t)}$ и переходим на шаг 4 алгоритма LCE.
- *Скобки остались, противоречия не обнаружено.* Вектор $\tilde{w}$ обозначаем $w^{\gamma+1(t)}$ и переходим на $(\gamma+1)$-ю итерацию.
- *Обнаружено противоречие.* Из вектора $w^{\gamma(t)}$ строим точку $y^{(t+1)}$: если $w_j^{\gamma(t)} = 2$, то полагаем $y_j^{(t+1)} = \frac{1}{2}$, $j = s+1, ..., n$. Таким образом, значение q-й координаты меняется на 1/2. Возвращаемся на шаг 2 алгоритма LCE.

Нами показано, что число операций, выполняемых алгоритмом при переходе от одного L-класса к другому, не превосходит значения $(2mn + 2m + 1)(n^2 + n) + 2n$. Таким образом, трудоемкость алгоритма LCE составляет $O(n^3 m|M_*/L|)$.

Отметим, что можно использовать и другие схемы перебора L-классов, например, начиная с лексикографически минимального, переходить к следующему в порядке лексикографического возрастания. В этом случае задачу ЦЛП для SAT естественнее формулировать как задачу поиска лексикографически минимального вектора, удовлетворяющего (1)—(3). Кроме этого, изменить процесс решения задачи можно с помощью предварительного переупорядочивания переменных, которое может существенно влиять на L-накрытие задачи и скорость работы алгоритма (см. п. 4).

Нетрудно показать, что решение задачи MAX SAT можно свести к решению конечной последовательности специальных задач SAT. В работах [4, 5, 7] рассматриваются варианты такого сведения для точного и приближенного решения задачи MAX SAT. Для решения задач выполнимости из данной последовательности используется описанный алгоритм перебора L-классов LCE. Проведенный эксперимент на ряде различных семейств логических формул показал эффективность такого подхода для точного и приближенного решения задачи максимальной выполнимости [7]. Кроме того, на основе указанного подхода нами разрабатываются параллельные алгоритмы для решения задач выполнимости и максимальной выполнимости [7].

4. Вычислительный эксперимент

В целях апробации разработанного алгоритма LCE нами проведены экспериментальные исследования на сериях тестовых примеров из известной специализированной библиотеки SATLIB [13], в которой содержатся семейства задач выполнимости, построенные разными авторами.

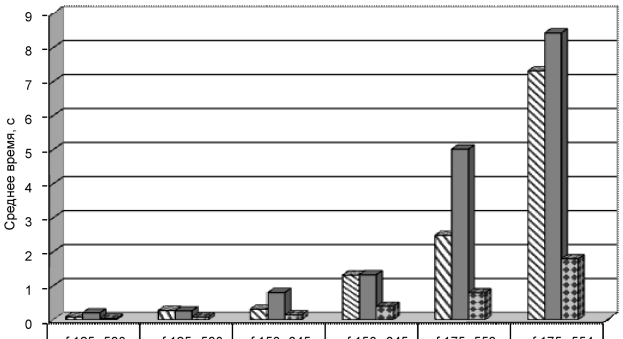
Первая часть экспериментальных расчетов выполнена на задачах из класса *Uniform Random-3-SAT*. Задачи данного класса, содержащие n переменных и m скобок, обладают следующими свойствами. Все скобки состоят из трех литералов, каждый из которых выбран из $2n$ возможных литералов $x_1, \bar{x}_1, ..., x_n, \bar{x}_n$ с вероятностью $1/2n$. В формуле не допускаются скобки, содержащие одинаковые литералы, и тождественно истинные скобки (т. е. включающие переменную и ее отрицание). Отношение числа скобок к числу переменных приблизительно равно 4,3. Такие задачи считаются наиболее сложными. В эксперименте были рассмотрены восемь серий по 100 задач в каждой.

Вторая часть эксперимента проведена на задачах класса "*Flat*" *Graph Colouring*, построенных путем сведения к задаче SAT задачи о вершинной раскраске графа. Все логические формулы задач данного класса получены из трех раскрашиваемых графов, для которых задача о раскраске является трудной для многих эвристических алгоритмов. В эксперименте рассмотрены пять серий по 100 задач; цифры в названии серии обозначают число вершин и ребер исходного графа. Характеристики задач приведены в таблице, указан тип формул: *sat* — формулы выполнимы, *unsat* — невыполнимы. Расчеты, представленные в этом разделе, проводились на компьютере AMD Athlon XP 2500+ (1,84 GHz).

Известно, что упорядочение переменных существенно влияет на мощность L-накрытия и, следовательно, на эффективность алгоритма. Нами предлагаются три варианта предварительной сортировки переменных задачи. Обозначим через α_j (β_j) число вхождений литерала x_j ($\bar{x}_j$) в формулу. Перед началом работы алгоритма переменные упорядочиваются следующим образом:
сортировка S1 — по невозрастанию значений α_j;
сортировка S2 — по невозрастанию значений β_j;
сортировка S3 — по невозрастанию значений $(\alpha_j + \beta_j)$.

Характеристики серий

Название серии	n	m	Тип формул
Класс Uniform Random-3-SAT			
uf 100-430/uuf 100-430	100	430	*sat/unsat*
uf 125-538/uuf 125-538	125	538	*sat/unsat*
uf 150-645/uuf 150-645	150	645	*sat/unsat*
uf 175-753/uuf 175-753	175	753	*sat/unsat*
Класс "Flat" Graph Colouring			
flat 100-239	300	1117	*sat*
flat 125-301	375	1403	*sat*
flat 150-360	450	1680	*sat*
flat 175-417	525	1951	*sat*
flat 200-479	600	2237	*sat*



	uf 125 -538	uuf 125 -538	uf 150 -645	uuf 150 -645	uf 175 -553	uuf 175 -554
S1+LCE	0,08	0,27	0,31	1,31	2,48	7,31
S2+LCE	0,2	0,26	0,8	1,32	5,01	8,43
S3+LCE	0,04	0,08	0,14	0,39	0,79	1,79
LCE	2,52	5,59	20,85	69,77	278,99	859,144

Рис. 4. Сравнение сортировок

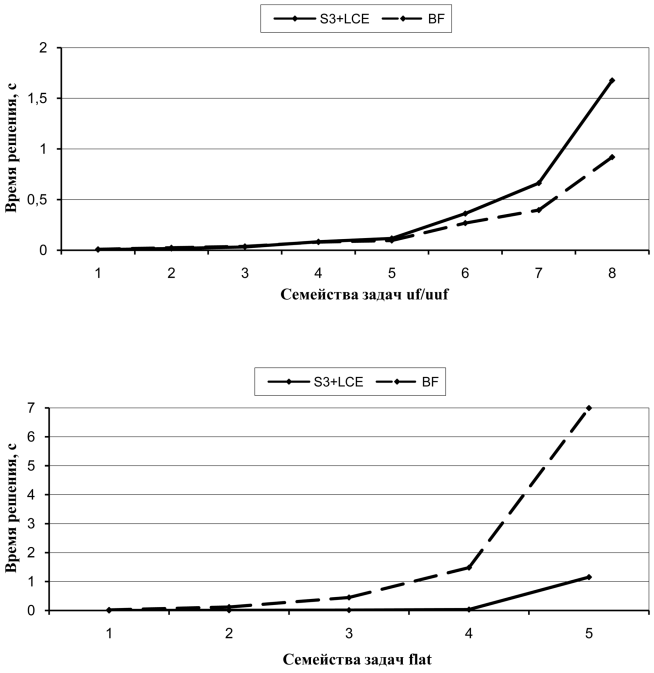


Рис. 5. Сравнение алгоритмов *S3* + LCE и BF

Нами было проведено сравнение среднего времени решения задач из класса *Uniform Random-3-SAT* в зависимости от использования указанных сортировок. На диаграмме приведены результаты для шести серий (рис. 4). Отметим, что все сортировки значительно ускорили базовый алгоритм LCE. Лучшие результаты показала сортировка *S3* и в дальнейшем она применялась перед запуском LCE.

Алгоритм LCE сравнивался также с известным точным гибридным алгоритмом, предложенным авторами работы [9] (обозначим его BF). Этот алгоритм сначала получает "хорошее" решение с помощью эвристики GSAT, а затем применяется процедура ветвей и границ (DPL-схема). На рис. 5 приведены результаты сравнения алгоритмов *S3* + LCE и BF по времени решения на сериях задач их классов *Uniform Random-3-SAT* и "*Flat*" *Graph Colouring*. Отметим, что для алгоритма LCE задачи с невыполнимыми формулами оказываются заметно сложнее, чем задачи с выполнимыми формулами. Это связано с тем, что дробное накрытие задачи для невыполнимой формулы совпадает со всем многогранником.

Пусть σ — процент задач, для которых время работы алгоритма *S3* + LCE не превосходило времени работы алгоритма BF. Заметим, что для большинства приведенных на диаграммах серий, $\sigma > 50$ %. Среднее же время решения для алгоритма LCE во многих сериях из класса *Uniform Random-3-SAT* оказывалось больше данного показателя для алгоритма BF. Это связано с тем, что в каждой серии встречаются примеры с достаточно большими *L*-накрытиями и алгоритм LCE их перебирает. Поэтому в случае алгоритма LCE наблюдается значительный разброс времени решения задач одной размерности. Алгоритм перебора *L*-классов оказался заметно лучше алгоритма BF на всех сериях класса "*Flat*" *Graph Colouring*, несмотря на достаточно большую размерность логических формул.

Кроме того, было проведено сравнение алгоритмов и для некоторых других семейств задач из указанной библиотеки. В целом, для большинства серий задач среднее время работы алгоритмов было либо сопоставимо, либо алгоритм BF незначительно опережал предложенный алгоритм *S3* + LCE. Кроме того, эти алгоритмы сравнивались с пакетом OPL Studio 3.7 для решения задач ЦЛП. На всех рассмотренных семействах данный пакет заметно уступал по времени рассмотренным алгоритмам LCE и BF. Как видно из эксперимента, предложенный алгоритм перебора *L*-классов позволяет находить решение задачи выполнимости достаточно большой размерности за приемлемое время и может быть использован в прикладных исследованиях.

Список литературы

1. **Адельшин А. В.** Исследование задач максимальной и минимальной выполнимости с использованием *L*-разбиения // Автоматика и телемеханика. — 2004. — № 1. — С. 35—42.

2. **Всемирнов М. А., Гирш Э. А., Данцин Е. Я., Иванов С. В.** Алгоритмы для пропозициональной выполнимости и верхние оценки их сложности // Зап. науч. семин. ПОМИ, 2001. — Т. 277. — С. 14—46.

3. **Колоколов А. А.** Регулярные разбиения и отсечения в целочисленном программировании // Сиб. журнал исследования операций. — 1994. — № 2. — С. 18—39.

4. **Колоколов А. А., Адельшин А. В., Ягофарова Д. И.** Алгоритмы точного и приближенного решения задачи максимальной выполнимости // Материалы всероссийской конференции "Проблемы оптимизации и экономические приложения". — Омск, 2006. — С. 42—46.

5. **Колоколов А. А., Адельшин А. В., Ягофарова Д. И.** Решение задач выполнимости и некоторых ее обобщений с использованием метода перебора *L*-классов // Прикладная математика и информационные технологии. Омск. 2005. — С. 68—79.

6. **Колоколов А. А., Нагорная З. Е., Гуселетова О. Н., Ярош А. В.** Математические модели и программный комплекс для проектирования эскизов одежды // Прикладная математика и информационные технологии. Омск, 2005. — С. 80—98.

7. **Колоколов А. А., Тюрюмов А. Н., Ягофарова Д. И.** Разработка и экспериментальное исследование алгоритмов решения задач выполнимости и максимальной выполнимости: Препринт // ОмГУ. Омск, 2006. — 19 с.

8. **Колоколов А. А., Ягофарова Д. И., Тюрюмов А. Н.** Разработка алгоритмов для задачи выполнимости и некоторых ее обобщений с использованием перебора *L*-классов // Омский научный вестник. — 2006. — № 5 (39). — С. 57—61.

9. **Borchers B., Furman J.** A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems // Journal of Combinatorial Optimization. — 1998. — V. 2. — N 4. — P. 299—306.

10. **Cook S. A.** The complexity of theorem-proving procedure // Proc. 3rd Annual ACM Symposium on the Theory of Computing. 1971. — P. 151—159.

11. **Gu J., Purdom P., Franco J., Wah B.** Algorithms for the Satisfiability (SAT) Problem: A Survey // DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1996. — 131 p.

12. **Kolokolov A., Kallrath J., Yagofarova D.** Analysis and Solving the Satisfiability Problem using *L*-partition // Operations Research 2003. Annual International Conference of the German Operations Research Society. University of Heidelberg, 2003. — P. 128.

13. **www.cs.ubc.ca/**~hoos/SATLIB/benchm.html

УДК 004.023

А. С. Филиппова, д-р техн. наук, доц.,
Д. В. Филиппов, аспирант,
Н. А. Гильманова, студентка,
Уфимский государственный авиационный технический университет
E-mail: annamuh@mail.ru

Задачи маршрутизации в транспортных логистических системах: локальный поиск рациональных решений

Рассматривается оптимизационный блок транспортной логистической системы. Для решения задачи построения маршрутов перевозки при условии многономенклатурности продукции предложен трехэтапный метод: построение остовного дерева; декомпозиция дерева на однопродуктовые маршруты; синтез маршрутов. Для поиска альтернативных допустимых маршрутов используются двухуровневые эволюционные метаэвристики с применением алгоритмов на графах. Приведен пример.

Ключевые слова: *транспортная логистика, многопродуктовая модель, маршрутизация, эвристика, метаэвристика, алгоритмы на графах.*

Введение

В деятельности торговых, производственных, складских и других предприятий часто возникает проблема транспортировки грузов. Фирма может осуществлять самовывоз со складов поставщиков, заниматься доставкой товаров своим клиентам, перевозить материалы и продукцию между цехами или отделами. Минимизация затрат на транспортировку грузов — оптимизационная задача транспортной логистики. Постановки данной задачи многочисленны и разнообразны, так же как и подходы к их решению. Однако в большинстве своем они весьма далеки от реальных ситуаций и требуют серьезной адаптации к логистическим системам.

В качестве основного примера приведем систему организации поставок, осуществляемых снабженческой компанией по некоторой сети мелких и средних торговых предприятий. Номенклатура продукции, поставляемой компанией, как правило, очень обширна, число наименований составляет тысячи. Компания может иметь несколько складов в разных частях транспортной сети. Для доставки она использует автомобильные транспортные средства (ТС) различной вместимости. Минимизация издержек должна осуществляться за счет рационального составления маршрутов движения ТС по сети.

В данной статье представлена модель, основанная на описанной ситуации и учитывающая ее специфику. Для решения многопродуктовой задачи маршрутизации предлагается эвристический алгоритм, приведен пример его использования. Прежде чем перейти к описанию модели, рассмотрим наиболее часто встречающиеся модели на графах и укажем их отличие от нашей.

Краткое описание основных задач на графах и сетях

1. Кратчайший путь. На связном ориентированном или неориентированном графе, каждому ребру (дуге) которого соответствует вещественное или целое число — длина, рассматривают следующие типы задач о кратчайшем пути: поиск пути от одной вершины до другой, от одной вершины до всех остальных или между каждой парой вершин графа. Для нахождения кратчайшего пути хорошо зарекомендовал себя алгоритм **Дейкстры** с локальным индексированием [1].

2. Остовное (покрывающее) дерево. *Остовным деревом* графа Г называют его подграф, являющийся деревом и содержащий все вершины графа Г. Применительно к сетям остовным деревом является подмножество дуг сети, в котором существует путь между каждой парой вершин. Остовное дерево с минимальным весом называется *минимальным остовным деревом*. Хорошо известен алгоритм **Прима** для поиска минимального остовного дерева; данный алгоритм наилучшим образом подходит для построения деревьев на полных графах [2].

Обе приведенные задачи часто выполняют вспомогательную роль. Таково их значение в предлагаемом далее методе решения многопродуктовой задачи поставок.

Следующие модели описывают однопродуктовые поставки при различных дополнительных условиях.

3. Задача коммивояжера. На графе требуется найти кратчайший *цикл*, содержащий все вершины. Иногда рассматривается задача нахождения кратчайшего *тура* (простой цепи, начинающейся в определенной вершине и проходящей через все прочие вершины). Задача коммивояжера является *NP-трудной*, для ее решения не известен точный алгоритм полиномиальной сложности. Кроме того, для данной задачи не предусмотрена возможность наличия у предприятия нескольких складов [1, 2].

4. Транспортная задача в сетевой постановке. По сети следует организовать перевозку только одного продукта, чаще всего однородного. Она хорошо изучена и существуют оптимизационные методы ее решения, в частности, *метод потенциалов*, предложенный Л. В. Канторовичем в конце 30-х годов [3]. При решении этой задачи особая роль принадлежит остовным деревьям, отвечающим базисным допустимым решениям, и поиску среди них оптимального.

5. Максимальные потоки в сети. Дугам сети приписываются не веса (стоимости), а *пропускные способности*. Среди узлов выделяются один *источник* и один *сток*. Задача состоит в определении наибольшего потока (объема однородной продукции), который может быть доставлен из источника в сток. Первый алгоритм нахождения максимального потока принадлежит Форду и Фалкерсону [4]. Относительно задачи о многопродуктовых потоках известно, что она является NP-полной [1]. Вместе с тем, имеются прецеденты ее решения методами линейного программирования с неявно заданной информацией, дающими неплохие результаты [5, 6].

Постановка задачи многопродуктовой маршрутизации

Пусть существует p наименований продуктов, которые поставляет компания. Задана транспортная сеть, состоящая из M узлов (перекрестков, значимых объектов, тупиков и т. д.) и N дуг, связывающих пары узлов. Для каждой дуги $j = \overline{1, N}$ известен ее вес d_j, имеющий смысл стоимости проезда по ней одного ТС (в простейшем случае — длина этой дуги). Каждому узлу сети с номером $i = \overline{1, M}$ и продукту $k = \overline{1, p}$ сопоставляется число $v_i^{(k)}$. Если $v_i^{(k)} > 0$, то в i-м узле расположен склад (пункт производства), в котором k-й продукт хранится (производится) в количестве $v_i^{(k)}$. Если же $v_i^{(k)} < 0$, то в i-м узле (магазине) k-й продукт заказан в количестве $|v_i^{(k)}|$. При $v_i^{(k)} = 0$ узел i не производит и не заказывает продукт k. Требуется найти способ перевозки продуктов от складов к клиентам, обеспечивающий наименьшие транспортные издержки.

Приведенная ситуация моделируется на ориентированном графе

$\Gamma = (V, \vec{E})$, где $V = \{v_1, v_2, ..., v_m, ..., v_M\}$ — множество вершин графа,

$\vec{E} = \{e_1, e_2, ..., e_N\}$ — множество ориентированных дуг.

Каждой вершине v_i, $i = \overline{1, M}$, соответствует вектор заказов/запасов продуктов $(v_i^{(1)}, ..., v_i^{(p)})$. Теоретически для любого узла i числа v_{ik} могут быть произвольными, но практически, как правило, возможны только три ситуации:

1) $v_i^{(k)} \geqslant 0$ ($k = \overline{1, p}$) — пункт i является складом (источником);

2) $v_i^{(k)} \leqslant 0$ ($k = \overline{1, p}$) — пункт i является клиентом (стоком);

3) $v_i^{(k)} = 0$ (k = $\overline{1, p}$) — пункт i является некоторым другим узлом на карте.

Будем полагать, что пункты бывают только этих трех типов.

В модели допускается существование нескольких источников и стоков; при этом должно выполняться следующее равенство:

$$\sum_{i=1}^{M} v_i^{(k)} = 0,\ k = \overline{1, p}. \qquad (1)$$

При перечислении узлы упорядочены так, чтобы на первых местах располагались источники и стоки общим количеством m, а далее — транзитные пункты. Каждой дуге e_j, $j = \overline{1, N}$, соответствует вектор (s_j, f_j, d_j), где s_j — номер начальной вершины дуги, f_j — номер конечной дуги, d_j — длина дуги. Будем считать, что граф Γ симметрический, т. е. для каждой дуги $e_j \in E$ существует единственная дуга $e_h \in E$, такая что $s_j = f_h$ и $f_j = s_h$. Примем также допущение, что $d_j = d_h$ для симметричных дуг e_j и e_h.

Искомым в модели является множество маршрутов, каждый из которых предназначен для движения одного или нескольких ТС в целях выполнения заказов клиентов. Маршрут описывается последовательностью связанных дуг, начинается всегда в источнике и заканчивается в стоке. Допустим, у нас нет ограничений на грузоподъемность и вместимость автомобилей, т. е. всегда можно подобрать подходящее ТС для выполнения маршрута. Тогда в качестве целевой функции выступает суммарная длина всех маршрутов конечного плана, которую следует минимизировать.

Таким образом, рассматривается многопродуктовая транспортная модель на взвешенном орграфе $\Gamma = (V, \vec{E})$. Прежде всего можно упростить задачу путем исключения всех промежуточных пунктов. Трансформированная сеть будет иметь меньшую размерность, и на ней можно легко решать однопродуктовые задачи. Что касается многопродуктовой модели, то целесообразно применить эвристический алгоритм конструирования допустимых маршрутов и метаэвристику для поиска рациональных маршрутов.

Алгоритм решения задачи

Этап 0. Применим алгоритм Дейкстры, чтобы найти кратчайшие пути между каждой парой вершин, задействованных в системе поставок (источниками и стоками). В результате исключения транзитных пунктов и расчета кратчайших путей получим полный граф $\Gamma' = (V', \vec{E}')$, где $V' = \{v_1, v_2, ..., v_m\} \subset V$ — множество вершин графа, а $\vec{E}' = \{e'_1, e'_2, ..., e'_n\}$ — множество дуг, каждая из которых соответствует кратчайшему пути между некоторыми двумя вершинами. Граф Γ' также является симметрическим, и для числа вершин m выполняется соотношение (1).

Этап 1. Построим основное дерево на графе, описывающем трансформированную сеть. Поскольку полный

граф в общем случае не является деревом, на нем существует множество различных остовных деревьев. В качестве одного из вариантов можно рассматривать построение минимального остовного дерева. В результате выполнения этапа будем иметь множество дуг $Tree = \{t_1, t_2, ..., t_{m-1}\}$. Каждой дуге дерева t_j, $j = \overline{1, m-1}$, соответствует вектор (s_j, f_j, d_j, x_j), где s_j — номер начальной вершины дуги; f_j — номер конечной вершины; d_j — длина дуги; x_j — объем перевозки некоторого продукта по дуге, который будет рассчитан на следующем этапе.

Этап 2. Выполняется для каждого продукта $k = \overline{1, p}$.

Сначала для продукта с фиксированным номером k проводится расчет объемов перевозок продукта по каждой дуге остовного дерева с помощью решения системы уравнений следующего вида:

$$\sum_{j:f_j = i} x_j - \sum_{j:s_j = i} x_j + v_i^{(k)} = 0, \quad i = \overline{1, m}. \qquad (2)$$

Если объем перевозки x_j окажется отрицательным, то направление дуги j в дереве следует изменить на противоположное (поменять s_j и f_j местами).

Далее для продукта k выполняется процедура *декомпозиции*: на остовном дереве строятся простые цепи (пути), каждая из которых начинается на складе, заканчивается в пункте потребления и может проходить как через пункты-клиенты, так и через другие склады. ТС не может дозагружаться на промежуточных складах. Проезжая через пункты-клиенты, ТС может выгружать некоторое количество продукта. Когда запас продукта становится равным нулю, цепь перевозки заканчивается.

Декомпозиция может выполняться несколькими способами; по сути их различие состоит в разных подходах к построению каждой отдельной цепочки и расчету объемов перевозок по ней. Один из вариантов — построить максимально длинную цепочку, а затем рассчитать объемы перевозки по ней исходя из допустимых объемов, полученных для остовного дерева. Другой алгоритм состоит в "расходовании" продукта по мере построения цепочки (объем перевозки рассчитывается непосредственно после включения дуги в цепь). Ни один из алгоритмов не дает гарантии оптимальности.

Итак, в результате выполнения этапа 2 для каждого продукта k будет получен набор $Chains^{(k)} = \{c_1^{(k)}, c_2^{(k)}, ...\}$, содержащий априори неизвестное число цепочек перевозки $c_q^{(k)} = \{c_{q_1}^{(k)}, c_{q_2}^{(k)}, ...\}$, каждая из которых состоит из априори неизвестного числа звеньев.

Этап 3. Синтез — получение набора маршрутов. Построенные в результате декомпозиции цепочки для разных продуктов объединяются в том случае, если они начинаются на одном складе и проходят через одни и те же пункты. Каждому звену такой объединенной цепи соответствует массив объемов перевозки продуктов. Результатом является множество маршрутов $Routes = \{R_1, R_2, ...\}$; $R_i \sim \{R_{i1}, R_{i2}, ...\}$ — последовательность дуг i-го маршрута; $R_{ij} \sim (s_{ij}, f_{ij}, d_{ij}, x_{ij})$, где s_{ij} — номер начальной вершины j-й дуги i-го маршрута; f_{ij} — номер конечной вершины; d_{ij} — длина дуги; $x_{ij} = \{x_{ij}^{(1)}, x_{ij}^{(2)}, ..., x_{ij}^{(p)}\}$ — вектор объемов перевозок каждого продукта $k = \overline{1, p}$ по j-й дуге i-го маршрута.

Поиск альтернативных решений

Поскольку алгоритм не является точным, то целесообразно применение метаэвристик. Они позволят провести отбор необходимого числа лучших полученных решений, которые могут выступать в качестве альтернативы друг другу. Простейший эволюционный алгоритм — одноточечная метаэвристика (1 + 1)-EA [7]. Организовать пространство поиска можно на следующих двух этапах.

1. *При построении остовного дерева*, если число дуг графа велико, то на нем существует множество различных остовных деревьев (в том числе множество различных минимальных деревьев). Строить их можно, пользуясь алгоритмом Прима, но каждый раз изменяя порядок просмотра дуг графа. На разных деревьях могут быть построены разные наборы маршрутов.

2. *При декомпозиции остовного дерева* длины цепочек и объемы перевозки каждого продукта по их звеньям существенно зависят от порядка просмотра дуг остовного дерева. Если этот порядок менять, то на одном остовном дереве можно получить несколько разных наборов цепочек, которые, складываясь с цепочками других продуктов, будут образовывать разные конечные маршруты.

Пример расчета маршрутов

Пусть транспортная сеть представлена симметрическим графом $\Gamma = (V, \vec{E})$, условно изображенным на рис. 1 (см. третью сторону обложки) как фрагмент карты города с обозначенными складами и пунктами доставки, а также перекрестками. Изначально сеть имеет 63 узла, из которых 1 и 2 являются складами предприятия, 3—12 — клиентами (магазинами), 13—63 — транзитными. Расстояния в метрах между транзитными узлами обозначены на рис. 1, дополнительно расстояния между источниками, стоками и транзитными узлами приводятся в табл. 1. Предположим, имеется два продукта, которые

Таблица 1

Дуги, инцидентные источникам и стокам

Пункты (s_j, f_j) или (f_j, s_j)	Длина дуги d_j	Пункты (s_j, f_j) или (f_j, s_j)	Длина дуги d_j	Пункты (s_j, f_j) или (f_j, s_j)	Длина дуги d_j	Пункты (s_j, f_j) или (f_j, s_j)	Длина дуги d_j	Пункты (s_j, f_j) или (f_j, s_j)	Длина дуги d_j
1, 17	960	3, 16	100	6, 31	100	9, 52	60	12, 60	250
1, 18	0	4, 14	200	7, 40	30	10, 49	150	12, 61	40
1, 25	420	4, 20	220	7, 46	200	10, 60	170		
2, 26	90	5, 36	650	8, 45	110	11, 51	320		
2, 32	100	5, 37	50	8, 52	120	11, 61	410		
3, 15	110	6, 35	100	9, 51	200	11, 62	0		

нужно доставить со складов в магазины. Данные об объемах продукции, соответствующих каждому из источников и стоков, приводятся в табл. 2. (Остальным пунктам соответствуют нулевые объемы продукции.)

Этап 0. Рассчитаем кратчайшие пути между первыми двенадцатью узлами сети с помощью алгоритма Дейкстры. Сохраним полученные последовательности пунктов для того, чтобы в дальнейшем восстановить по ним маршруты, и заменим пути, проходящие через некоторые подмножества транзитных пунктов, на прямые дуги полного симметрического ориентированного графа $\Gamma' = (V', \vec{E}')$. Приведем фрагмент списка дуг данного графа; нумерация дуг произвольная (табл. 3).

Таблица 2

Объемы продукции

Номер пункта i	Запас/ /заказ 1-го продукта $v_i^{(1)}$	Запас/ /заказ 2-го продукта $v_i^{(2)}$	Номер пункта i	Запас/заказ 1-го продукта $v_i^{(1)}$	Запас/заказ 2-го продукта $v_i^{(2)}$
1	310	500	7	−100	−140
2	500	520	8	−90	−150
3	−160	0	9	−30	−150
4	−170	−70	10	−50	−130
5	−40	0	11	−70	−210
6	−20	−70	12	−80	−210

Таблица 3

Список дуг графа Γ' (фрагмент)

Номер дуги j	Пункты (s_j, f_j)	Длина дуги d_j	Номер дуги j	Пункты (s_j, f_j)	Длина дуги d_j
1	8, 9	180	7	6, 7	810
2	9, 11	520	8	2, 7	630
3	11, 12	450	9	2, 4	810
4	10, 12	420	10	3, 4	580
5	5, 8	620	11	1, 5	1120
6	6, 10	750	...	...	...

Таблица 4

Вычисление объемов перевозок первого продукта по дугам остовного дерева

Номер дуги j	Пункты (s_j, f_j)	Объем перевозки x_j	Скорректированные пункты $(s_j, f_j)'$	Скорректированный объем перевозки x_j'
1	8, 9	180	8, 9	180
2	9, 11	150	9, 11	150
3	11, 12	80	11, 12	80
4	10, 12	0	10, 12	0
5	5, 8	270	5, 8	270
6	6, 10	50	6, 10	50
7	6, 7	−70	7, 6	70
8	2, 7	170	2, 7	170
9	2, 4	330	2, 4	330
10	3, 4	−160	4, 3	160
11	1, 5	310	1, 5	310

Таблица 5

Вычисление объемов перевозок второго продукта по дугам остовного дерева

Номер дуги j	Пункты (s_j, f_j)	Объем перевозки x_j	Скорректированные пункты $(s_j, f_j)'$	Скорректированный объем перевозки x_j'
1	8, 9	460	8, 9	460
2	9, 11	310	9, 11	310
3	11, 12	100	11, 12	100
4	10, 12	110	10, 12	110
5	5, 8	610	5, 8	610
6	6, 10	240	6, 10	240
7	6, 7	−310	7, 6	310
8	2, 7	450	2, 7	450
9	2, 4	70	2, 4	70
10	3, 4	0	3, 4	0
11	1, 5	610	1, 5	610

Этап 1. На полученном графе Γ′ построим минимальное остовное дерево с произвольной ориентацией дуг (рис. 2, см. третью сторону обложки). Здесь в него входят именно те дуги, которые перечислены в табл. 3. Суммарная длина дуг дерева 6890.

Этап 2. Уравнения (2) для первого продукта, записанные по остовному дереву, имеют следующий вид:

$i = 1$: $-x_{11} + 310 = 0$;
$i = 2$: $-x_8 - x_9 + 500 = 0$;
$i = 3$: $-x_{10} - 160 = 0$;
$i = 4$: $x_9 + x_{10} - 170 = 0$;
$i = 5$: $-x_5 + x_{11} - 40 = 0$;
$i = 6$: $-x_6 - x_7 - 20 = 0$;
$i = 7$: $x_7 + x_8 - 100 = 0$;
$i = 8$: $-x_1 + x_5 - 90 = 0$;
$i = 9$: $x_1 - x_2 - 30 = 0$;
$i = 10$: $-x_4 + x_6 - 50 = 0$;
$i = 11$: $x_2 - x_3 - 70 = 0$;
$i = 12$: $x_3 + x_4 - 80 = 0$.

Решение системы представлено в табл. 4; в последних двух столбцах приводится скорректированное решение с измененным, если это требовалось, направлением дуги. Результаты декомпозиции дерева приводятся на рис. 3.

Аналогично проведены расчеты для второго продукта, их результаты представлены в табл. 5 и на рис. 4.

Этап 3. Синтезируем маршруты путем сложения цепочек, полученных в результате декомпозиции для разных продуктов. Результат представлен на рис. 5. В скобках указаны объемы каждого из перевозимых продуктов. Суммарная протяженность маршрутов 6890 м. Заметим, что это значение совпадает с суммарной длиной дуг дерева. Однако в большинстве случаев это не так. Общая длина дуг дерева является нижней границей протяженности соответствующих ему маршрутов.

Теперь мы можем записать окончательные планы следования для каждого из ТС с помощью кратчайших путей через промежуточные пункты, рассчитанных на этапе 0. Для магазинов указаны объемы продукции, которые следует в данном пункте выгрузить из ТС (для начальных складов — загрузить в ТС):

1. $\overset{(+310,\ +610)}{1} \to 25 \to 38 \to 37 \to \overset{(-40,0)}{5} \to 37 \to 44 \to 45 \to$
$\to \overset{(-90,\ -150)}{8} \to 52 \to \overset{(-30,\ -150)}{9} \to 51 \to \overset{(-70,\ -210)}{11} \to 61 \to \overset{(-80,\ -100)}{12}$.

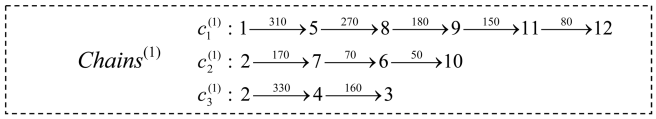


Рис. 3. Результаты декомпозиции остовного дерева для первого продукта

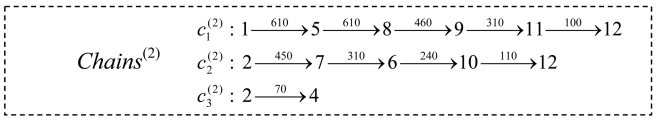


Рис. 4. Результаты декомпозиции остовного дерева для второго продукта

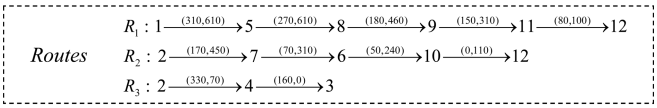


Рис. 5. Результаты синтеза маршрутов

2. $\overset{(+170,\ +450)}{2} \to 32 \to 39 \to 40 \to \overset{(-100,\ -140)}{7} \to 40 \to 33 \to 34 \to$
$\to 35 \to \overset{(-20,\ -70)}{6} \to 41 \to 48 \to 49 \to \overset{(-50,\ -130)}{10} \to 60 \to \overset{(0,\ -110)}{10}$.

3. $\overset{(+330,\ +70)}{2} \to 26 \to 19 \to 20 \to \overset{(-170,\ -70)}{4} \to 14 \to 15 \to$
$\to \overset{(-160,\ 0)}{3}$.

Заключение

Предложены постановка и способ решения многопродуктовой задачи маршрутизации, возникающей в логистических средах. Приведен пример использования трехэтапного алгоритма составления маршрутов. Для поиска рациональных решений предполагается применять двухуровневый эволюционный метод. Данный метод допускает введение дополнительных ограничений и уточнение выбора промежуточного решения на каждом этапе, а также всевозможные модификации и усовершенствования.

Список литературы

1. **Ху Т. Ч., Шинг М. Т.** Комбинаторные алгоритмы / Пер. с англ. Нижний Новгород: Изд-во Нижегородского госуниверситета им. Н. И. Лобачевского, 2004. 330 с.

2. **Кристофидес Н.** Теория графов. Алгоритмический подход / Пер. с англ. М.: Мир, 1979. 432 с.

3. **Мухачева Э. А., Рубинштейн Г. Ш.** Математическое программирование. 2-е изд., перераб. и доп. Новосибирск: Наука, 1987.

4. **Форд Л. Р., Фалкерсон В. Р.** Потоки в сетях. М.: Мир, 1965.

5. **Fulkerson D. R.** Suggested Computation for Maximal Multi-commodity Network Flows // Man. Sci. 1958. Vol. 5. № 1. P. 97—101.

6. **Hu T. C.** Integer Programming and Network Flows. Addison-Wesley, 1969.

7. **Филиппова А. С.** Моделирование эволюционных алгоритмов решения задач прямоугольной упаковки на базе блочных структур // Информационные технологии. 2006. № 6. Приложение. 32 с.

УДК 004.421.5

М. А. Яковлев, аспирант, инженер-программист,
ООО "ДС БАРС",
И. В. Чугунков, канд. техн. наук, доц.,
МИФИ
E-mail: 5may@bk.ru

Повышение эффективности оценочных тестов для псевдослучайных последовательностей

Предлагается новый подход к построению статистических тестов для оценки качества псевдослучайных последовательностей, обеспечивающий сокращение объемов затрачиваемой памяти.

Ключевые слова: *псевдослучайные последовательности, генераторы псевдослучайных последовательностей, оценка качества генераторов псевдослучайных последовательностей, оценочные тесты для псевдослучайных последовательностей, повышение эффективности, оценка качества.*

Одним из основных требований [1], предъявляемых к качественным псевдослучайным последовательностям (ПСП), являются хорошие статистические свойства (т. е. неотличимость от истинно случайных) последних при большом периоде. Для анализа статистических свойств используются различные группы тестов, наиболее известными из которых являются: Руководство *Statistical Test Suit*e [3, 6] Национального института стандартов и технологий (НИСТ) США, система *Diehard* [4] Дж. Марсалья и подборка тестов Д. Кнута [2]. Механизм работы практически всех тестов из вышеупомянутых подборок основан на подсчете числа появлений определенных шаблонов и сравнения полученных значений с теоретическими. При этом для хранения данных о числе появлений шаблонов для последовательности длиной n, состоящей из наборов по k m-разрядных чисел требуется объем памяти, равный $\frac{2^{km}}{k}([\log_2(n-1)]+1)$ бит в случае непересекающихся шаблонов и $2^{km}([\log_2(n-1)]+1)$ в случае пересекающихся. Учитывая, что значение n, исходя из требований к качественным ПСП, должно быть большим, существенно возрастает объем требуемой памяти. Например, для последовательности размером 1 Мбайт анализ наборов по 5 полубайт потребует вспомогательной памяти размером 500 Кбайт в случае непе-

ресекающихся наборов и 1,25 Мбайт в случае пересекающихся. В идеале при длине последовательности, стремящейся к бесконечности, размер вспомогательной памяти также будет стремиться к бесконечности. Таким образом, возникает задача уменьшения затрачиваемой для реализации теста памяти.

В настоящее время большинство разработчиков оценочных тестов (в том числе и авторы наиболее популярного на сегодняшний день Руководства НИСТ США) решают данную проблему путем введения ограничений на размеры анализируемых шаблонов и длину исследуемой последовательности.

Уменьшение размеров шаблонов приводит к существенному ослаблению теста, так как, зная логику работы последнего, можно внести соответствующие изменения в алгоритм работы генератора ПСП или непосредственно в саму последовательность, благодаря чему свойства последней будут неотличимы от свойств истинно случайной последовательности для типа неслучайности, проверяемого заданным тестом. Данное утверждение косвенно подтверждается результатами исследований наиболее эффективных из существующих генераторов ПСП. Все они с легкостью проходят тесты "Проверка серий-пар" и "Проверка серий-троек" (размеры шаблонов — два и три бита соответственно) и полностью проваливают "Посимвольную проверку" (размер шаблона — 8 бит) [1]. И это только для шаблонов, состоящих из одного числа. Можно предположить, что увеличение чисел в шаблоне сделает статистику прохождения еще более удручающей.

Уменьшение длины исследуемой последовательности также снижает качество тестирования. Для примера разработчики набора статистических тестов НИСТ США рекомендуют использовать для анализа последовательности длиной всего лишь 2^{20} бит или 128 Кбайт, что для существующих объемов передаваемой информации, исчисляемой мегабайтами, гигабайтами и даже терабайтами, является просто недопустимым.

Один из вариантов разрешения данной проблемы косвенно предложен в Руководстве НИСТ. Суть его заключается в тестировании не всей последовательности целиком, а подпоследовательностей. Однако данный подход не лишен недостатков. Основной из них связан с выбором размера подпоследовательности. При увеличении размера подпоследовательности опять возникает проблема дополнительной памяти. Уменьшение размера подпоследовательностей приводит к необходимости оценки корреляции между последними для избежания влияния периодичности.

Выход из сложившейся ситуации видится в модификации механизма работы тестов, суть которой заключается в том, чтобы анализировать не число появлений определенных шаблонов, а число отсутствующих шаблонов. В этом случае для хранения информации о шаблоне достаточно будет одного бита, что приведет к уменьшению объема памяти до $2^{km}/k$ для непересекающихся шаблонов и 2^{km} в случае пересекающихся, при этом цель теста не будет изменена, и он продолжит выявлять те же статистические отклонения, что и оригинальный тест. Таким образом, размер требуемой для реализации теста памяти перестает зависеть от длины исследуемой последовательности. Для примера, для реализации тестов НИСТ при рекомендуемой длине в 128 Кбайт объем требуемой памяти уменьшается в 20 раз.

Рассмотрим, как меняется механизм вычисления статистики теста. Для заданной последовательности длиной n, состоящей из наборов по k m-разрядных чисел, подсчитываем число отсутствующих наборов. Дж. Марсалья в своей работе [5] показал, что число отсутствующих наборов аппроксимируется нормальным распределением. Найдем среднее и отклонение.

Для расчета среднего необходимо рассмотреть разложение в ряд Тейлора производящей функции, соответствующей значениям k и m. Очевидно, это не очень удобно, поскольку для различных шаблонов придется заново определять производящую функцию для каждого типа набора и раскладывать ее в ряд Тейлора. Например, для $k = 2$ расчет осуществляется следующим образом:

- вычисляется p_1 — коэффициент при z_n в разложении производящей функции

$$\frac{1}{1 - z + p^2 z^2};$$

- вычисляется p_2 — коэффициент при z_n в разложении производящей функции

$$\frac{1 + pz}{1 - (1 - p)z - (p - p^2)z^2};$$

- вычисляется среднее для числа отсутствующих слов

$$\mu = (2^{km} - 2^m)p_1 + 2^m p_2.$$

С увеличением k увеличивается число производящих функций, а также их сложность. Поэтому для расчета предлагается использовать подход Дж. Марсалья, который показал, что среднее можно вычислить (при условии, что $n > 1000$) по формуле

$$\mu = 2^{km}\mathbf{e}^{-\frac{n}{2^{km}}}.$$

Действительно, для случая $n = 2^{21}$, $k = 2$, $m = 10$ имеем

$$\mu = (2^{2\cdot 10} - 2^{10}) \cdot 0{,}135335283236469 +$$
$$+ 2^m \cdot 0{,}135599351997986596411 \approx 141909{,}60,$$

$$\mu = 2^{2\cdot 10}\mathbf{e}^{-\frac{n}{2^{2\cdot 10}}} = 141909{,}33.$$

Таблица 1

Возможность модификации тестов подборки Д. Кнута

№	Название теста	Возможность модификации
1	Проверка несцепленных серий	+
2	Проверка интервалов	–
3	Проверка комбинаций	+
4	Тест собирателей купонов	+
5	Проверка перестановок	+
6	Проверка на монотонность	–
7	Проверка корреляции	–
	Итого	**4/7**

Таблица 2

Возможность модификации тестов Diehard

№	Название теста	Возможность модификации
1	Промежутки между Днями Рождения	–
2	Проверка пересекающихся перестановок	+
3—5	Проверка рангов матриц	+
6—10	Буквенные ("обезьяньи тесты") (5 тестов)	+
11	Подсчет числа единиц в потоке байтов	+
12	Подсчет числа единиц в определенных байтах	+
13	Тест НОД	–
14	Тест парковки	+
15	Тест минимальных расстояний	–
16	Тест пузырей	–
	Итого	**12/16**

Таблица 3

Возможность модификации тестов Руководства НИСТ

№	Название теста	Возможность модификации
1	Частотный тест	+/–
2	Проверка кумулятивных сумм	–
3	Проверка "дырок" в подпоследовательностях	–
4	Проверка "дырок"	–
5	Проверка рангов матриц	+
6	Спектральный тест	–
7	Проверка непересекающихся шаблонов	+
8	Проверка пересекающихся шаблонов	+
9	Универсальный статический тест Маурера	–
10	Проверка случайных отклонений	–
11	Разновидность проверки случайных отклонений	–
12	Проверка аппроксимированной энтропии	–
13	Проверка серий	+
14	Сжатие с помощью алгоритма Лемпела—Зива	–
15	Линейная сложность	–
	Итого	**5/16**

Расчет отклонения осуществляется по следующей формуле:

$$\sigma = \sqrt{\sum_{i_1=1}^{2^m} \sum_{i_2=1}^{2^m} \dots \sum_{i_k=1}^{2^m} \operatorname{cov}(n_{i_1}, n_{i_2}, \dots, n_{i_k})},$$

где $n_i = \begin{cases} 1, \text{ если буква, равная } 2^i, \text{ присутствует в слове;} \\ 0, \text{ если буква, равная } 2^i, \text{ отсутствует в слове.} \end{cases}$

Примечание. Для вычисления отклонения рекомендуется использовать специализированные программные продукты (MathCAD и др.), в которых присутствует возможность вычисления ковариации нескольких переменных.

В табл. 1—3 показано, какие тесты из наиболее популярных подборок для оценки статистических свойств ПСП могут быть модифицированы. Как можно заметить, улучшить можно больше половины тестов подборки Д. Кнута и системы Diehard, а также почти треть тестов Руководства НИСТ.

Эффективность применения данного подхода обусловлена следующими факторами:

- размер вспомогательной памяти для сбора статистики теста теперь не зависит от длины тестируемой последовательности и определяется только размерами шаблона. Это позволит выделять данную память статически, что ускорит скорость выполнения теста при программной реализации, или прогнозировать объем выделяемой динамической памяти;
- учитывая, что длина последовательности теперь не влияет на объем затрачиваемой памяти, все существующие оценочные тесты можно будет применять не к части последовательности фиксированной длины (ограниченной в существующих системах несколькими мегабайтами), а в идеале ко всему периоду;
- механизм работы модифицированных тестов предполагает досрочное завершение процесса исследования в том случае, если исследуемая статистика вышла за границы доверительного интервала, что существенно сократит время тестирования;
- уменьшение объемов требуемой памяти позволит более свободно варьировать параметрами тестирования, увеличив диапазон используемых значений, что существенно повысит функциональность тестов и качество тестирования.

Список литературы

1. **Иванов М. А., Чугунков И. В.** Теория, применение и оценка качества генераторов псевдослучайных последовательностей. М.: КУДИЦ-ОБРАЗ, 2003. 240 с.

2. **Кнут Д.** Искусство программирования. Т. 2. Получисленные алгоритмы. 3-е изд. / Пер. с англ.: Уч. пос. М.: Издательский дом "Вильямс", 2000. 832 с.

3. **A statistical** test suite for random and pseudorandom number generators for cryptographic applications // NIST Special Publications 800—22. 2001. May 15.

4. **Marsaglia G.** DIEHARD Statistical Tests/ — http://stat.fsu.edu/~geo/diehard.html.

5. **Marsaglia G., Zaman A.** Monkey test for random number generators // Computer & Mathematics with Applications. 1993. V. 9. P. 1—10.

6. **Rukhin A. L.** Testing Randomness: A Suite of Statistical Procedures // Теория вероятностей и ее применения. 2000. Т. 45. Вып. 1. С. 137—162.

УДК 541.182

Б. Г. Кухаренко, канд. физ.-мат. наук, вед. научн. сотр.,
Институт машиноведения РАН, г. Москва
E-mail: kukharenko@imash.ru

Предварительная обработка записей колебаний в технологии спектрального анализа на основе быстрого преобразования Прони

Представлены основные этапы предварительной обработки записей колебаний в технологии спектрального анализа на основе быстрого преобразования Прони. В качестве примера анализируются содержащие шум записи нестационарных колебаний с относительно стабильным спектром собственных частот. Показано, что предварительная обработка записей нестационарных колебаний позволяет надежно определять зависящее от времени демпфирование на собственных частотах, которое играет роль информативных параметров, описывающих эволюцию этих колебаний.

Ключевые слова: *линейные динамические системы, временные ряды, метод Прони, рекурсивные фильтры, дискретное вейвлет-преобразование.*

Спектральный анализ нестационарных колебаний

Для спектрального анализа нестационарных колебаний следует использовать метод, который способен выполнять спектральное оценивание сегментов временных рядов во временных окнах ограниченной продолжительности и использует локальную во времени модель нестационарного временного ряда (например, возрастающего или убывающего в последовательных временных окнах). С этой точки зрения представляет интерес возможность использования метода Прони (Prony) [1]. В методе Прони роль априорного знания играет представление о колебательном изменении временного ряда. Однако главной предпосылкой использования метода Прони для спектрального анализа нестационарных временных рядов является кусочно-линейная модель регистрируемого колебания. Этот метод использует разложение сегментов временных рядов, в результате которого для суммы временных зависимостей вида $A\mathbf{e}^{-\delta t}\cos(2\pi ft + \varphi)$ определяются все четыре независимых параметра этих временных зависимостей: A, φ, δ, f. В результате для каждого последовательного сегмента временного ряда метод Прони определяет его собственный спектр частот и демпфирование на этих собственных частотах. В работе [2] метод Прони используется для спектрального анализа неустойчивых коллективных колебаний (флаттера) лопаток компрессора. В [3] показано, что коллективные колебания лопаток компрессора под воздействием аэроупругих сил описываются линейной моделью. Эти колебания характеризуются относительно стабильным спектром собственных частот. Зависящее от времени демпфирование на собственных частотах позволяет характеризовать эволюцию коллективных колебаний (флаттера) лопаток компрессора. Для этого используется сегментирование записей нестационарных колебаний (многомерных нестационарных временных рядов).

Пусть $x[\overline{1, N}]$, $N \ll N_0$, представляет собой некоторый сегмент полного временного ряда (записи колебания) $x[\overline{1, N_0}]$ (индекс временного ряда опущен, поскольку каждая компонента многомерного временного ряда обрабатывается по отдельности). Например, первый сегмент временного ряда

$$x[k] = x((k-1)\Delta t), \quad k = \overline{1, N}, \tag{1}$$

где Δt — фиксированный шаг дискретизации времени, представляет временную зависимость $x = x(t)$ амплитуды нестационарных колебаний для $t \in [1, t_N]$, $t_N = N\Delta t$. Этот сегмент временного ряда может содержать шум. Для краткости выбран шаг дискретизации времени $\Delta t = 1$. Спектральная декомпозиция Прони сегмента (1) имеет вид

$$x[k] = \sum_{l=1}^{p} r[l]z[l]^{k-1} + n[k], \quad k = \overline{1, N}, \tag{2}$$

здесь p — число полюсов сегмента (1); $z[l] = \exp(\delta[l] + j2\pi f[l])$, $l = \overline{1, p}$, — полюса сегмента, где $\delta[l]$ и $f[l]$ — соответственно фактор демпфирования и частота; $r[l] = A[l]\exp(j\varphi[l])$, $l = \overline{1, p}$, — вычеты в полюсах, где $A[l]$ и $\varphi[l]$ — соответственно амплитуда и фаза; $n[k]$ — аддитивный шум.

Когда определены главные полюса $z[l]$, $l = \overline{1, p}$, вычеты $r[l]$, $l = \overline{1, p}$, в полюсах определяются из формулы (2) по методу наименьших квадратов. Оценка зависящих от времени спектров факторов демпфирования и частот и соответствующих спектров амплитуд и фаз для полного временного ряда (записи колебания) $x[\overline{1, N_0}]$ получается посредством последовательных сдвигов временного окна фиксированной длины $N\Delta t$. В записях нестационарных колебаний, использованных в работе [2], отношение сигнал/шум достаточное, чтобы быстрое преобразование Прони позволило надежно определить собственные частоты $f[l]$ и соответствующие им факторы демпфирования $\delta[l]$. При уменьшении отношения сигнал/шум алгоритм быстрого преобразования Прони продолжает устойчиво определять собственные частоты $f[l]$ сегмента (1), однако точность определения факторов демпфирования $\delta[l]$ может снижаться [4]. Для обеспечения точности определения факторов демпфирования $\delta[l]$ требуется предварительная обработка временного ряда (записи нестационарного колебания) $x[\overline{1, N_0}]$.

Ниже описаны два этапа предварительной обработки записей нестационарных колебаний (нестационарных временных рядов), обеспечивающие точность при обработке их последовательных сегментов посредством алгоритма быстрого преобразования Прони. Как показывают результаты предварительной обработки экспериментальных записей неустойчивых колебаний, эти два этапа предварительной обработки не коммутативны. Первый этап — это линейная полосовая фильтрация записей неустойчивых колебаний на собственных частотах $f[l]$.

Полосовой фильтр для неустойчивого временного ряда

В настоящей работе используется полосовой линейный фильтр. Аппроксимация полного временного ряда $x[\overline{1, N_0}]$ по методу Прони тесно связана с Z-преобразованием этого временного ряда

$$X(z) = \sum_{k=1}^{N} x[k]z^{-k}, \tag{3}$$

где z — комплексная переменная [5]. Формула для обратного преобразования имеет вид

$$x[k] = \sum_{l=1}^{p} \operatorname{Res}_{z=z[l]} [X(z)]z[l]^{k-1}, \tag{4}$$

где $z[l]$, $l = \overline{1, p}$, — полюса временного ряда, а вычеты в этих полюсах

$$r[l] = \operatorname{Res}_{z=z[l]} (X(z)). \tag{5}$$

Члены ряда $x[\overline{1, N_0}]$ возрастают, начиная примерно с нуля при $k = 1$, т. е. ряд является неустойчивым. Линейный фильтр с запаздыванием временного ряда $x[\overline{1, N_0}]$, имеет вид

$$y[k] = \sum_{i=0}^{m} b[i]x[k-i] - \sum_{i=0}^{m} a[i]y[k-i], \quad k = \overline{1, N_0}, \tag{6}$$

где $a[i]$, $b[i]$, $i = \overline{0, m}$, — коэффициенты фильтра, $x[k] = 0$, $k < 1$ [6]. Поэтому начальное условие для решения $y[\overline{1, N_0}]$ разностного уравнения (7), представляющего фильтрованный временной ряд, имеет вид

$$y[k] = 0, \quad k < 1. \tag{7}$$

По определению, передаточная функция

$$H(z) = \Big(\sum_{i=0}^{m} b[i]z^{-i} \Big) \Big(1 + \sum_{i=0}^{n} a[i]z^{-i} \Big)^{-1}. \tag{8}$$

Тогда $Y(z) = H(z)\, X(z)$, где $X(z)$, $Y(z)$ — Z-преобразования временных рядов $x[\overline{1, N_0}]$ и $y[\overline{1, N_0}]$ соответственно. Условие устойчивости фильтра $H(z)$ (8) состоит в том, что в комплексной плоскости все его полюса лежат внутри единичного круга. Вместе с тем, временной ряд $x[\overline{1, N_0}]$ является неустойчивым. Следовательно, все полюса $X(z)$ в комплексной плоскости лежат вне единичного круга. Таким образом, полюса $H(z)$ и $X(z)$ никогда не совпадают, и из формулы (4) для обратного Z-преобразования следует, что общее решение разностного уравнения (6) имеет вид

$$y[k] = y_X[k] + y_H[k], \tag{9}$$

где

$$y_X[k] = \sum_{l=1}^{P} \operatorname{Res}_{z=z[l]} (X(z))H(z[l])z[l]^{k-1};$$

$$y_H[k] = \sum_{l=1}^{Q} X(z[l]) \operatorname{Res}_{z=z[l]} (H(z))z[l]^{k-1}.$$

Таким образом, $y_X[k]$ представляет собой вклад от полюсов $X(z)$, т. е. возрастающее решение уравнения (6), а $y_H[k]$ — вклад от полюсов $H(z)$, т. е. убывающее решение уравнения (6). Поэтому решение уравнения (6), удовлетворяющее начальному условию (7), имеет вид $y[k] = y_X[k]$. Ниже в качестве полосового фильтра используется фильтр второго порядка с $H(z)$ в виде

$$H(z) = (b[0] + b[1]z^{-1} + b[2]z^{-2}) \times (1 + a[1]z^{-1} + a[2]z^{-2})^{-1}. \tag{10}$$

Из условия $H(\exp(j\omega_{-,+})) = 0$, где $\omega_- = 0$, $\omega_+ = \pi$, следует, что $H(z)$ (10) имеет вид

$$H(z) = b[0](1 - z^{-2})(1 + a[1]z^{-1} + a[2]z^{-2})^{-1}. \tag{11}$$

Если

$$H(z_{1,2})^{-1} = 0, \quad z_{1,2} = r\exp(j2\pi f_0),$$

то условие устойчивости фильтра (11) имеет вид $r < 1$. Параметр f_0 — центральная частота фильтра (11). По определению $bw = 1 - r$, где bw — полоса пропускания фильтра (11). Следовательно, $a[1] = -2(1 - bw)\cos(2\pi f_0)$, $a[2] = (1 - bw)^2$, а $b[0]$ в (11) определяется из условия $H(\exp(j2\pi f_0)) = 1$.

Для $H(z)$ (11) разностные уравнения для определения $y[\overline{1, N_0}]$ имеют вид

$$\begin{gathered} y[1] = b[0]x[1]; \\ y[2] = b[0]x[2] - a[1]y[1]; \\ y[k] = b[0](x[k] - x[k-2]) - a[1]y[k-1] - \\ - a[2]y[k-2], \; k = \overline{3, N_0}. \end{gathered} \tag{12}$$

Разностные уравнения (12) обеспечивают предварительную линейную фильтрацию временного ряда, представляющего неустойчивое колебание. Этот временной ряд фильтруется непосредственно во временной области, поэтому перед фильтрацией неустойчивого временного ряда нет необходимости преобразовывать его в частотную область. Является ли стабильной последующая идентификация по методу Прони спектральных параметров фильтрованного временного ряда, зависит от достигнутого значения отношения сигнал/шум. Полосовая фильтрация не затрагивает ту часть шума в записи неустойчивого колебания, частотный спектр которого оказывается в полосе пропускания используемого фильтра (11)—(12), т. е. окрашенный (не белый) шум. Удаление окрашенного шума из фильтрованного неустойчивого временного ряда осуществляется с помощью алгоритма трансляционно-инвариантного удаления шума (*translation-invariant denoising algorithm*) [7, 8].

Удаление окрашенного шума из временного ряда посредством алгоритма трансляционно-инвариантного удаления шума

Алгоритм трансляционно-инвариантного удаления шума формулируется с использованием оператора временного сдвига. Для временного ряда $y[k]$, $k = \overline{1, N_0}$, оператор S_h определяет сдвиг на h значений индекса с повторением:

$$S_h y[k] = y[(k + h) \bmod N_0].$$

Этот оператор унитарный и, следовательно, обратимый:

$$S_{-h} = (S_h)^{-1}.$$

Оператор S_h используется при формулировке идеи сдвига временного ряда: вместо вейвлет-преобразования $W[...]$ рассматриваемого временного ряда используется его сдвинутая версия $W[y;\ S_h] = S_{-h}W[S_h y)]$. Для алгоритма трансляционно-инвариантного удаления шума основная структура данных — это трансляционно-инвариантная таблица.

Пусть рассматриваемый временной ряд y — это вектор длины $N_0 = 2^J$ (J — целое) и пусть $\tau(y)$ — соответствующая ему трансляционно-инвариантная таблица. Таблица $\tau(y)$ обладает тремя основными свойствами:

- для любого целого h вейвлет-коэффициенты сдвига с повторением $W[S_h y]$ содержатся в этой таблице;
- полная трансляционно-инвариантная таблица для y может быть вычислена за время порядка $N_0 \log_2(N_0)$;
- извлечение вейвлет-преобразования любого сдвига временного ряда с повторением $W[S_h y]$ занимает время порядка N_0.

Трансляционно-инвариантная таблица — это массив размерностью $N_0 \times D$, где $0 \leqslant D \leqslant \log_2(N_0)$. d-столбец этой таблицы имеет N_0 строк, разделенных на 2^d "ячеек", т. е. каждая "ячейка" имеет $N_0/2^d$ строк. Интерпретация "ячеек" следующая: они соответствуют 2^d различным наборам вейвлет-коэффициентов, которые появляются в вейвлет-разложении на уровне $(J - d)$ при различных сдвигах исходного временного ряда (вектора) y.

Трансляционно-инвариантная таблица вычисляется в результате последовательности операций прореживания и фильтрации, что обычно при вейвлет-преобразовании, но в ее вычисление введен дополнительный элемент-сдвиг с повторением. Пусть G и H обозначают стандартные для вейвлет-преобразования операции высоко- и низкочастотной фильтрации с прореживанием; а S_h — это определенный выше сдвиг на h значений индекса с повторением; и пусть $\beta[J, 0] = y$. Затем присваиваются

$$\begin{aligned} \alpha[J-1, 0] &= GS_0\beta[J, 0]; \\ \alpha[J-1, 1] &= GS_1\beta[J, 0]; \\ \beta[J-1, 0] &= HS_0\beta[J, 0]; \\ \beta[J-1, 1] &= HS_1\beta[J, 0]. \end{aligned} \tag{13}$$

Далее рекурсивно вычисляются

$$\begin{aligned} \alpha[j, 2k] &= GS_0\beta[j+1, k]; \\ \alpha[j, 2k+1] &= GS_1\beta[j+1, k]; \\ \beta[j, 2k] &= HS_0\beta[j+1, k]; \\ \beta[j, 2k+1] &= HS_1\beta[j+1, k] \end{aligned} \tag{14}$$

и т. д. Трансляционно-инвариантная таблица заполняется, помещая вектор $\alpha[J - d, k]$ в ячейку k столбца d. В дополнительный нулевой столбец таблицы помещаются все $\beta[j_0, k]$, вычисленные на конечном шаге.

Чтобы доказать, что полученная таким образом таблица $\tau(y)$ обладает свойством трансляционной инвариантности, наряду с этой таблицей, соответствующей y, рассмотрим $\tau(S_h y)$ — таблицу, соответствующую сдвигу с повторением $S_h y$. Для каждого сдвига $h \in \overline{0, N_0}$ имеется матрица перестановок N_h, такая что

$$\Pi_h \tau(y) = \tau(S_h y). \tag{15}$$

Стандартное вейвлет-преобразование состоит из несдвинутых данных

$$W[y] = \{\beta[j_0, 0], \alpha[j_0, 0], \alpha[j_0 + 1, 0], ..., \alpha[J - 1, 0]\}. \tag{16}$$

Поскольку $\tau(S_h y)$ содержит все коэффициенты вейвлет-преобразования сдвига $S_h y$, из (15) следует, что эти вейвлет-коэффициенты уже присутствуют в таблице $\tau(y)$; требуется только, чтобы они были корректно распакованы. Более точно, вейвлет-преобразование сдвига с повторением вектора y реализуется как

$$\begin{aligned} W[S_h y] = \{&\beta[j_0, k[j_0]], \alpha[j_0, k[j_0]], \\ &\alpha[j_0 + 1, k[j_0 + 1]], ..., \alpha[J - 1, k[J - 1]]\} \end{aligned} \tag{17}$$

для соответствующей последовательности $\{k[j_0], k[j_0 + 1], ..., k[J - 1]\}$, где каждое $k[j] = 2k[j + 1] + b[j]$ и $b[j] \in \overline{0, 1}$. Биты $b[j]$ кодируют сдвиг h в специальной бинарной нотации. Чтобы извлечь вейвлет-преобразование некоторого сдвига с повторением из трансляционно-инвариантной таблицы, необходимо специфицировать сдвиг h в терминах этого специального кодирования.

Для удаления шума используется порог оставляемых значений вейвлет-коэффициентов. При этом возможны следующие стратегии:

- мягкое применение порога к вейвлет-коэффициентам (за исключением коэффициентов наиболее грубого уровня): $\eta_T(w) = \mathrm{sgn}(w)(w - T)_+$, где функция $(...)_+$ отлична от нуля только для положительных значений аргумента и порог $T = T_{N0} = (2\log_2(N_0))^{1/2}\sigma$;
- жесткое применение порога — это операция применения нелинейности $\nu_T(w) = wI(w > T)$, где $I(...)$ — пороговая функция, ко всем вейвлет-коэффициентам, за исключением коэффициентов наиболее грубой шкалы;
- использование порога SURE (*Stein Unbiased Risk Estimator* — несмещенная оценка риска Стейна). Она наиболее приемлема для рассматриваемого случая аддитивного окрашенного шума в фильтрованном временном ряду. Ее правила описаны в работе [8].

В случае окрашенного шума s можно продолжать считать, что среднее значение шума $E[s]$ равно нулю. Но поскольку шум не белый, вариация

$$\sigma_m^2 = E[|s_{\mathbf{B}}[m]|^2] \tag{18}$$

зависит от каждой функции $g_m(...)$ вейвлет-базиса $\mathbf{B}$, использованного в реализации трансляционно-инвариантного алгоритма удаления шума. В результате применяемый к вейвлет-коэффициентам порог T_m оказывается функцией m [7].

Обратное преобразование от трансляционно-инвариантной таблицы к исходному временному ряду основано на систематическом усреднении: начиная с $j = J - D$ и затем для каждого k в диапазоне $0 < k < 2^j$ вычисляются (с помощью обычных операторов добавления членов ряда G^* и H^*)

$$\begin{aligned} \gamma[k] &= (S_0 G^*\beta[j, 2k] + S_{-1}G^*\beta[j, 2k+1])/2; \\ \theta[k] &= (S_0 H^*\alpha[j, 2k] + S_{-1}H^*\alpha[j, 2k+1])/2; \\ \beta[j+1, k] &= \gamma[k] + \theta[k]. \end{aligned} \tag{19}$$

Исчерпав все значения k на одном уровне, устанавливают $j = j + 1$ и повторяют сначала. Вычисления пре-

кращаются при достижении $j = J$. Пусть $\bar{y} = \beta[J, 0]$. При применении к трансляционно-инвариантной таблице, генерированной из временного ряда y, результат будет $\bar{y} = y$. При применении к трансляционно-инвариантной таблице, сформированной с использованием порога значений вейвлет-коэффициентов, результат будет средним по всем N_0 реконструкциям из всех N_0 сдвигов с повторением. Фактически каждая $\gamma[k]$ и $\theta[k]$ — это среднее из двух возможных реконструкций (одна из несдвинутой последовательности и одна из сдвинутой последовательности), которое является ответственным за этот результат. Этот алгоритм на каждом уровне требует выполнения N_0 арифметических операций и проходит порядка $\log_2(N_0)$ уровней, так что в целом алгоритм требует выполнения $N_0\log_2(N_0)$ арифметических операций. Алгоритм трансляционно-инвариантного удаления шума реализуется дискретным стационарным вейвлет-преобразованием — *Discrete Stationary Wavelet Transform* (DSWT). Этот этап завершает процесс увеличения отношения сигнал/шум в каждой из компоненты анализируемого многомерного временного ряда.

Пример

Как и в [2], рассмотрим каскад лопаток компрессора под воздействием аэроупругих сил как пример распределенной системы, совершающей неустойчивые колебания. При увеличении числа оборотов рассматриваемого компрессора появление флаттера связано с коллективными изгибными колебаниями лопаток. Задача состоит в определении особенностей временной эволюции флаттера по записям неустойчивых колебаний лопаток

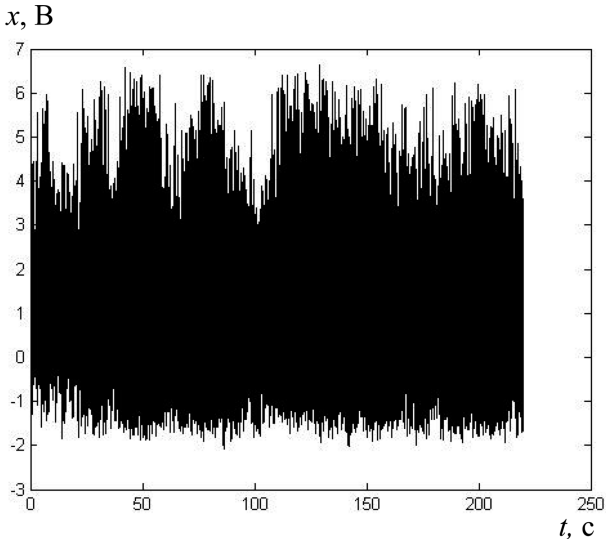


Рис. 1. Запись колебания в локальном максимуме изгибной моды 1-й лопатки

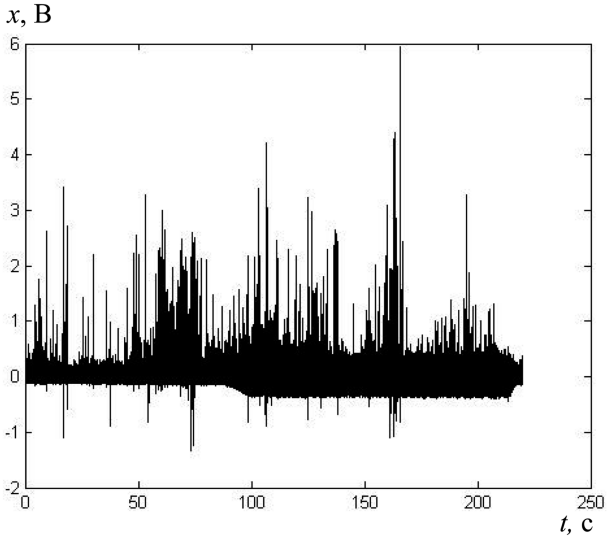


Рис. 2. Запись колебания в локальном максимуме изгибной моды 2-й лопатки

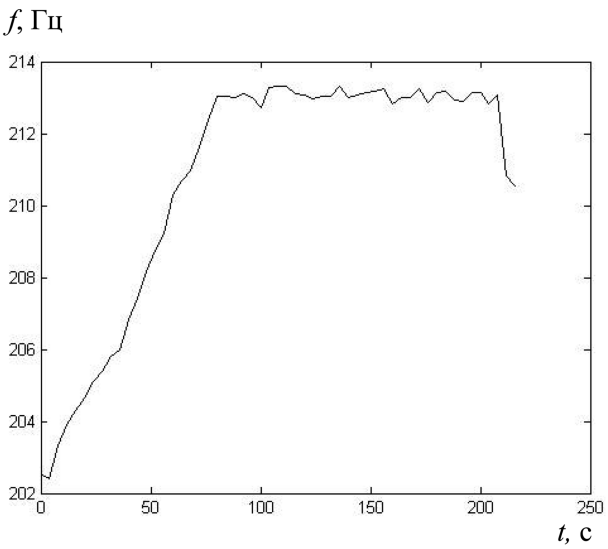


Рис. 3. Зависимость от времени частоты оборотов компрессора

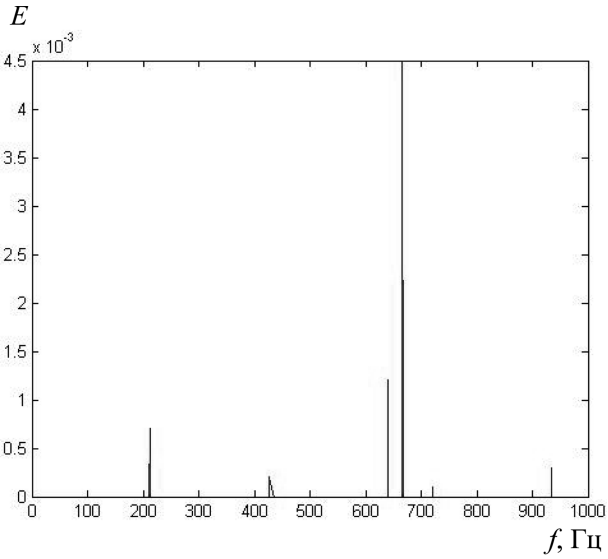


Рис. 4. Оценка энергии частотных составляющих записи на рис. 2

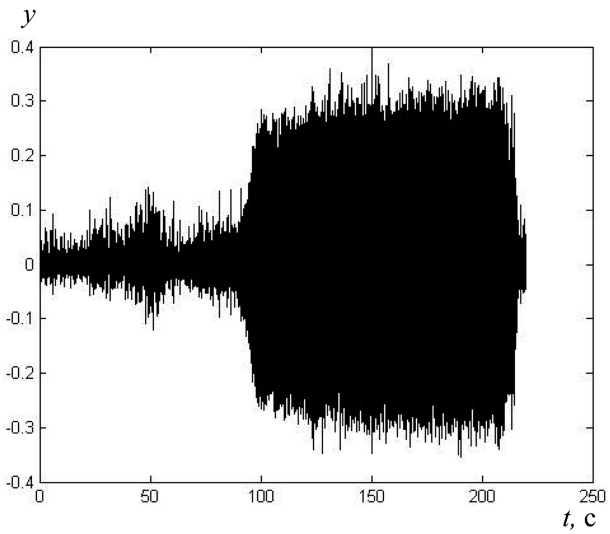


Рис. 5. Результат фильтрации записи колебаний 1-й лопатки

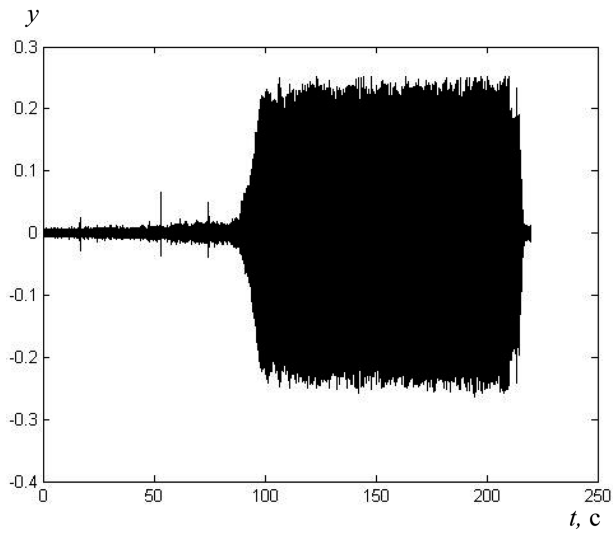


Рис. 6. Результат фильтрации записи колебаний 2-й лопатки

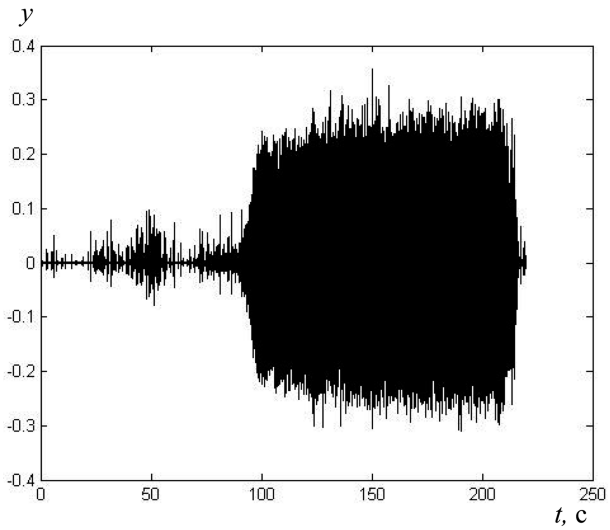


Рис. 7. Результат удаления окрашенного шума из фильтрованной записи колебаний 1-й лопатки

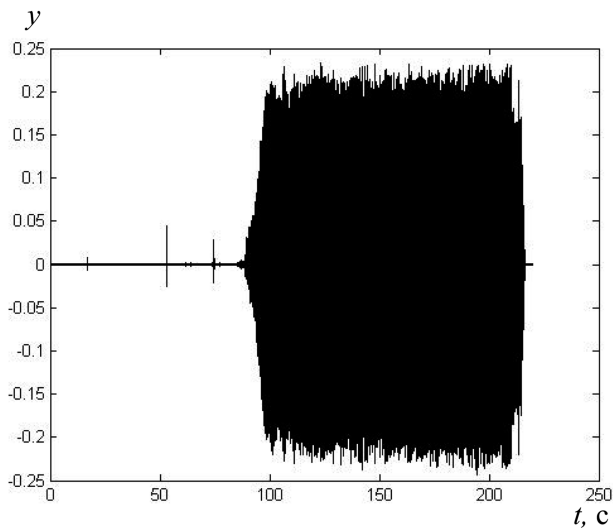


Рис. 8. Результат удаления окрашенного шума из фильтрованной записи колебаний 2-й лопатки

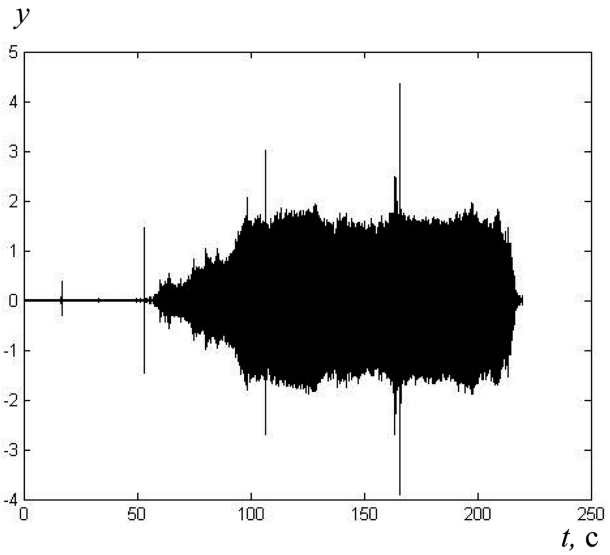


Рис. 9. Компонента колебания на рис. 2 для третьей гармоники оборотной частоты

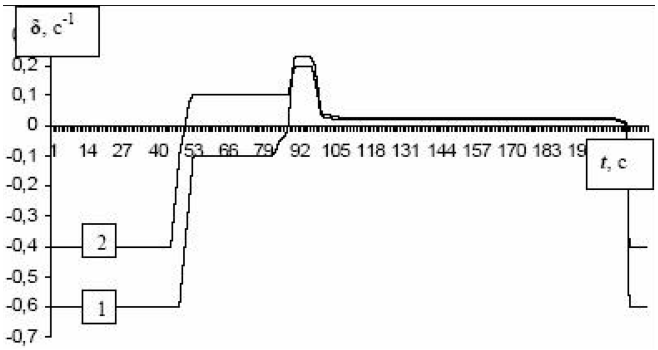


Рис. 10. Временная зависимость демпфирования δ = δ(*t*):
1 — изгибной моды лопатки; *2* — на третьей гармонике оборотной частоты

(рис. 1, 2). Соответствующая зависимость от времени частоты оборотов компрессора приведена на рис. 3.

Для таких распределенных систем важную информацию об изменениях, происходящих в системе, предоставляют временные зависимости демпфирования на собственных частотах системы [2]. Однако для записей на рис. 1, 2 зависящее от времени демпфирование на собственных частотах не может быть непосредственно оценено по методу Прони. Вид записи колебаний лопаток на рис. 1, 2 такой, что только запись на рис. 2 дает некоторое представление о коллективном колебании лопаток. Оценка по методу Прони относительной энергии E частотных составляющих записи на рис. 2 для t = [82...94 с] приведена на рис. 4.

Поскольку записи на рис. 1, 2 представляют две проекции (с шумом) коллективного колебания лопаток компрессора, их узкополосная фильтрация на частотах, соответствующих максимумам на рис. 4, позволяет повысить в них отношение полезный сигнал/шум. Неизменность спектра собственных частот является важным критерием постоянства физических свойств и целостности рассматриваемой распределенной системы. Из рис. 4 следует, что таких частот две: 639 Гц и 665 Гц. Однако сравнение с временной зависимостью оборотной частоты на рис. 3 показывает, что частота 639 Гц — это третья гармоника этой оборотной частоты ≈ 213 Гц. Поэтому именно 665 Гц — это частота изгибной моды лопатки, участвующей в возникновении флаттера. Результаты узкополосной фильтрации записей на рис. 1, 2 с параметрами: центральная частота f_0 = 665 Гц и r = 0,99 представлены на рис. 5, 6 соответственно.

Записи на рис. 5, 6 представляют одно и то же коллективное колебание лопаток, однако в них все еще наблюдаются некоторые различия. Причина не только в том, что неустойчивые колебания различных лопаток во время их возбуждения слегка различаются, но и в том, что после узкополосной фильтрации в этих записях остается окрашенный шум с полосой частот, соответствующей полосе пропускания использованного фильтра. Результаты удаления окрашенного шума в фильтрованных записях колебаний на рис. 5, 6 посредством алгоритма трансляционно-инвариантного удаления шума приведены на рис. 7, 8 соответственно.

Сравнение записей на рис. 7, 8 позволяет уточнить (на 0,8 с в сторону увеличения) момент возбуждения нестабильных коллективных колебаний лопаток. Аналогичным образом последовательность действий по узкополосной фильтрации записей на рис. 1, 2 с параметрами: центральная частота f_0 = 639 Гц и r = 0,99 и удалении из этих записей окрашенного шума посредством алгоритма трансляционно-инвариантного удаления шума позволяет выделить неустойчивую составляющую этих колебаний с частотой третьей гармоники оборотной частоты (рис. 9).

Результаты. Результаты определения временного изменения демпфирования на частоте изгибной моды лопатки и на третьей гармонике по временным рядам, представленным на рис. 7, 8 и рис. 9 соответственно, позволяют охарактеризовать процесс развития флаттера лопаток рассматриваемого компрессора. Временная зависимость $\delta = \delta(t)$ демпфирования на частотах изгибной моды лопатки и третьей гармоники оборотной частоты компрессора показана на рис. 10.

Как показывает спектральный анализ по методу Прони, вплоть до $t \approx 88$ с спектр колебаний на рис. 8 содержит только собственную частоту изгибной моды лопатки и третью гармонику оборотной частоты, т. е. колебания являются линейными. С ростом оборотной частоты компрессора (см. рис. 3) при $t \approx 89$ с демпфирование на частоте изгибной моды становится положительным, и ее амплитуда резко возрастает. Изменение знака и значения демпфирования на собственной частоте означает изменение роли этой частотной составляющей нестационарного колебания и позволяет предсказывать его неустойчивость. При $t \approx 99$ с в спектре нестационарных колебаний появляются гармоники изгибной частоты. В линейной модели колебаний появление гармоник собственных частот показывает, что формы с этими собственными частотами становятся нелинейными. Таким образом, начиная с $t \approx 99$ с, флаттер лопаток развивается как изгибный и нелинейный.

Список литературы

1. **Weiss L., McDonogh R. N.** Prony's method, *Z*-transform, and Pade approximation // SIAM Review. 1963. V. 9. № 2. P. 145—149.

2. **Кухаренко Б. Г.** Технология спектрального анализа на основе быстрого преобразования Прони // Информационные технологии. 2008. № 4. С. 38—42.

3. **Bendiksen O. O., Friedmann P. P.** The effect of bending-torsion coupling on fan and compressor blade flutter // Transactions of the ASME. 1982. V. 104. № 3. P. 617—623.

4. **Kukharenko B. G.** Spectral identification of periodic trajectories for the model of interwell jumps in a bistable oscillator // Fradkov A. L., Kurzhanski A. B., eds. Nonlinear Control Systems 2001. Oxford: Elsevier Science. 2002. V. 2. P. 951—956.

5. **Оппенгейм А. В., Шафер Р. В.** Цифровая обработка сигналов. М.: Связь. 1979. 416 с.

6. **Рабинер Л., Гоулд Б.** Теория и применение цифровой обработки сигналов. М.: Мир. 1978. 848 с.

7. **Mallat S.** A Wavelet Tour of Signal Processing. Academic Press. 1999. 851 p.

8. **Coifman R. R., Donoho D. L.** Translation invariant de-noisng // Lecture Notes in Statistics. Springer Verlag. 1995. V. 103. P. 125—150.

УДК 004 42

О. С. Исаева, канд. техн. наук, науч. сотр.,
Институт вычислительного моделирования
СО РАН, г. Красноярск
E-mail: isaeva@icm.krasn.ru

Унифицированная информационная модель описания медицинских услуг

Рассматривается реализация информационной системы для описания медицинских услуг. Задача потребовала построения формального описания элементов реализации для произвольной настройки визуального отображения данных. Приведены структурная схема предметной области, построенная на ее основе модель данных и пример реализации адаптируемой информационной системы. Полученные результаты могут быть использованы для построения сложно структурированных информационно-справочных систем.

Ключевые слова: *проектирование баз данных, информационные системы, формальная модель визуализации данных, система для описания медицинских услуг.*

Введение

Информационная поддержка управленческих задач по организации медицинской помощи связана с разработкой и совершенствованием критериев оценки, контроля и обеспечения качества и эффективности медицинских услуг. Основой технологии поддержки управления в территориях является создание и ведение базы данных клинико-экономических стандартов (КЭС) оказания медицинской помощи [1]. Под клинико-экономическим стандартом понимается описание медицинской услуги как комплекса мероприятий, направленных на предупреждение заболеваний, диагностику или лечение, имеющих самостоятельное законченное значение и определенную стоимость.

Неустойчивость и недостаточность нормативной базы приводят к неопределенности состава и содержания информационного наполнения медицинских стандартов. Необходимость решения этой проблемы потребовала создания унифицированного и легко адаптируемого описания КЭС, дополненного формальным описанием способов визуализации данных.

Результатом работ стали унифицированная информационная модель и программная система, предназначенная для ведения базы данных клинико-экономических стандартов. Программная система поддерживает манипулирование произвольной медицинской услугой как одним информационным объектом, независимо от физической структуры входящих в нее данных и сложности внутренней организации информации, и поэтому легко адаптируется к изменениям состава и содержания медицинских услуг. Обеспечение информационной совместимости с программными продуктами, функционирующими в системе управления здравоохранением, осуществляется за счет использования единых классификаторов и справочников.

Структурная схема предметной области

Процесс построения информационной модели КЭС включал несколько этапов:

- анализ объекта исследования и нормативно-справочных данных;
- определение целей и задач исследования;
- описание элементов модели;
- создание структуры базы данных по описанию;
- выработку требований к функционированию;
- реализацию.

Основной нормативный документ, на котором строится база данных, — это "Номенклатура работ и услуг в здравоохранении" [2], содержащая перечень медицинских услуг: манипуляции, исследования, процедуры, услуги медицинского сервиса и работы в здравоохранении. В номенклатуре определено деление медицинских услуг на простые, сложные и комплексные (услуги по заболеванию). Такое деление обусловлено особенностями предметной области.

Каждый из видов услуг характеризуется своим набором, составляющим описание элементов. В описании услуг содержатся иерархические связи: простая услуга как неделимый элемент может входить в состав как сложных, так и комплексных медицинских услуг; сложная — в состав сложных медицинских услуг; комплексная услуга может входить в состав только комплексных услуг.

На рис. 1 представлена структурная схема, содержащая основные информационные объекты, входящие в описание медицинских услуг, и показаны отношения информационной зависимости между ними. Данная схема неоднократно модифицировалась в результате итера-

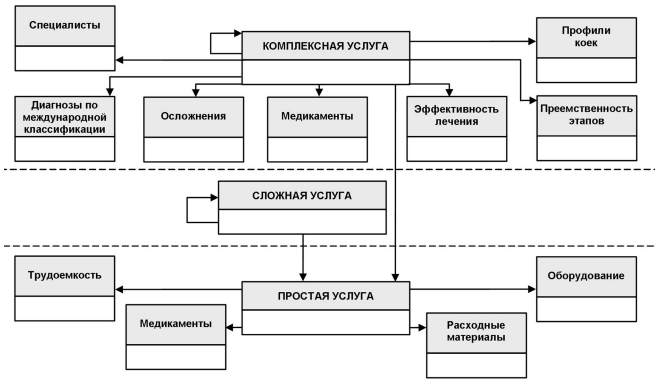


Рис. 1. Основные информационные объекты

ционного процесса постановки задачи и реализации информационной системы. На основе структурной схемы строилась логическая модель базы данных.

Набор итераций при проектировании и разработке информационной модели связан с переработкой структуры базы данных. Это потребовало многократного выбора используемых справочников и классификаторов, смены атрибутов объектов, дополнения и изменения состава информации для описания услуг в соответствии с изменениями нормативной базы и пожеланиями специалистов предметной области.

Логическая модель данных

На этапе проектирования базы данных принимались во внимание следующие основные критерии эффективности структуры данных:

- обеспечение быстрого доступа к данным;
- исключение избыточности данных;
- полнота представления информации;
- обеспечение временной и ссылочной целостности данных.

Формальное описание медицинской услуги состоит из конечного набора отдельных свойств и характеризующих объектов, описываемых классификаторами с помощью таблиц связи. Структура базы данных строилась с использованием справочников и классификаторов, ведением которых занимается "Система ведения справочников" (СВС) [3], и для обеспечения информационной совместимости при создании структуры поддерживались требования и ограничения справочной системы. При разработке логической модели данных учитывалась возможность работы с иерархической организацией справочных данных. Иерархия задается в специальных таблицах, называемых в терминологии СВС оглавлениями.

В базе данных содержатся справочники медицинских услуг, должностей специалистов, медикаментов, оборудования и пр. Для единообразного представления объектов предметной области и связей между ними были введены специальные таблицы связи. Состав КЭС описывался с помощью таблиц, определяющих связь записи об услуге из справочника с записями о конкретных элементах состава услуги, например медикаментах. Таблицы связи содержат набор ссылок с дополнительными числовыми полями, например, дозировкой медикаментов и пр. На рис. 2 представлен фрагмент базы данных, включающий набор таблиц для формирования стандартов простых медицинских услуг.

Для обеспечения временной целостности данных в таблицах содержится набор служебных данных, определяющих даты и основания для введения в действия или прекращения действия конкретных записей справочника.

В структуре содержатся четыре группы таблиц: справочники с оглавлениями, таблицы связи данных для описания стандартов оказания простых медицинских услуг, сложных медицинских услуг и услуг по заболеванию. Каждая из групп таблиц для описания стандартов услуг определенного типа содержит таблицу с основной информацией об услуге, включающую выделенные характеристики, определяющие услугу.

Основное назначение базы данных — информационно-справочное. Для удобства работы с данными требуется строить сложные экранные формы, объединяющие смысловые блоки информации из разных таблиц в общие структуры. Поэтому при разработке информационной модели была выполнена формализация описания способов отображения таблиц, включающая ограничения на возможные значения данных и допустимые состояния визуальных объектов.

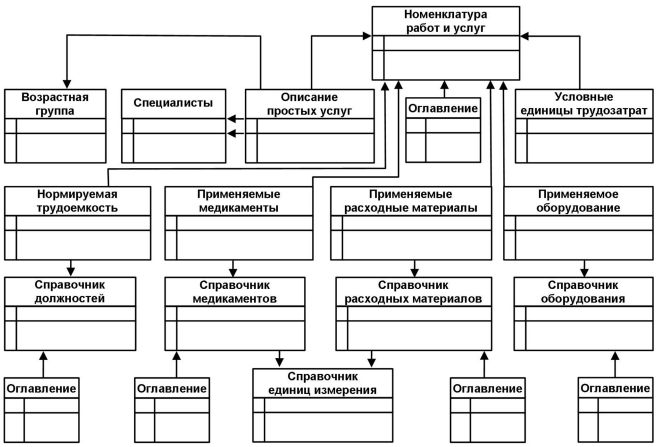


Рис. 2. Фрагмент логической модели базы данных КЭС

Формальное описание модели КЭС

Формальное описание клинико-экономических стандартов оказания медицинской помощи можно представить в виде набора информационных объектов:

$$\text{КЭС} = [U;\ T_0;\ D_1, ..., D_n;\ O_1, ..., O_r;\ T_1, ..., T_m;$$
$$S_1, ..., S_l;\ V_0, ..., V_k;\ F_1, ..., F_m].$$

Здесь:

U — справочник, содержащий перечень медицинских услуг. На основе U строится описание КЭС;

T_0 — таблица связи, содержащая данные, характеризующие стандарт оказания услуги определенного типа, связанная со справочником U связью "один к одному";

$D_1, ..., D_n$ — справочники, предназначенные для описания состава услуг, где n — число справочников, необходимых для задания стандарта оказания услуги определенного типа;

$O_1, ..., O_r$ — оглавления для справочников U, $D_1, ..., D_n$, где r — число оглавлений для справочников. Справочник может не иметь оглавлений или иметь несколько оглавлений, описывающих разные способы иерархической организации данных;

$T_1, ..., T_m$ — таблицы связи для описания стандарта оказания услуги определенного типа, m — число таблиц связи, причем $m \leqslant n$. Таблицы $T_1, ..., T_m$ связывают справочник U с одним или несколькими справочниками из $D_1, ..., D_n$. Таблица связи кроме задания ссылок может содержать дополнительные данные;

$S_1, ..., S_l$ — способы отображения полей справочников U, $D_1, ..., D_n$ и таблиц $T_0, ..., T_m$, причем $l \geqslant (n + + m + 1)$;

$V_0, ..., V_k$ — способы отображения выделенных полей таблиц $T_0, ..., T_m$ в специальных визуальных элементах;

$F_1, ..., F_m$ — правила ограничения множества записей справочников U, $D_1, ..., D_n$ при отображении в таблицах $T_0, ..., T_m$.

Формальное описание справочников U, D_1, ..., D_n, оглавлений O_1, ..., O_r и таблиц T_0, ..., T_m включает уникальный идентификатор, русское наименование таблицы, служебную информацию и признак редактирования.

При описании таблиц T_0, ..., T_m можно задавать в модели виртуальные поля, физически не заданные в структуре данных, а построенные по формальному описанию. Виртуальные поля задаются через поля-ссылки на справочники U, D_1, ..., D_n (справочник указывается как характеристика соответствующего поля). Обозначим в качестве $P(T)$ произвольное поле таблицы T, тогда

$$P(T) = (Id,\ N\_rus,\ type,\ D,\ SP,\ Spec),$$

где Id — уникальный идентификатор; N_rus — текстовое значение с русским наименованием поля; $type$ — тип данных (длина, точность); D — указатель на U, D_1, ..., D_n; SP — строка с перечнем дополнительных полей; $Spec$ — служебная информация (признак уникальности, редактирования).

Описание P является модифицированным описанием формирования дополнительных полей по ссылкам для справочной системы. Строка SP строится из номеров полей справочника и служебных символов "+", "|". По полям справочника, наименования которых входят в описание SP через символ "+", в таблице создаются виртуальные поля, значения которых формируются как конкатенация строк исходных полей. Если два наименования полей в SP связаны символом "|", то в таблице создаются два виртуальных поля, каждое из которых может состоять из одного или нескольких полей справочника.

Способы отображения таблиц S_1, ..., S_l при программной реализации формальной модели переходят во внешний файл настроек и доступны для изменения. Наименование таблицы задает раздел в описании. S_1, ..., S_l представляются строками вида

$$Np + " = " + Vis + Ed + Wh,$$

где Np —уникальное имя поля (для полей, физически существующих в таблице, его значение совпадает с идентификатором поля, для виртуальных полей имя указывает на принадлежность к таблице и к справочнику, по которому оно строится); Vis — признак видимости поля при отображении в визуальном элементе ("+" или "−"); Ed — признак редактируемости поля (0 или 1); Wh — ширина поля при отображении в визуальном элементе (целое число).

При работе со справочником значения полей должны быть ограничены по правилам, заданным специалистом предметной области. Настраиваемые ограничения задаются через элементы модели F_1, ..., F_m в виде

$$"N_f = Filter",$$

где N_f — уникальное наименование, оно строится из идентификатора таблицы и идентификатора справочника, значения которого нужно ограничить для отображения в таблице; $Filter$ — логическое выражение над значениями полей справочника.

Объекты формального описания модели КЭС при реализации отображаются в визуальных элементах. Таблицы T_0, ..., T_m отображаются в стандартных сетках согласно заданным в описании S_1, ..., S_l параметрам. Для удобства работы с информацией часть полей таблиц или справочников требуется отображать в специальных визуальных элементах, содержащих поля для ввода данных или выпадающие списки. Такие выделенные случаи задаются в формальном описании в виде V_0, ..., V_k. Это описание из модели переходит в конфигурационный файл и доступно для редактирования. V_0, ..., V_k содержат уникальные наименования поля, таблицы и визуального элемента для отображения. В этом случае в одном из описаний полей в S_1, ..., S_l указывается дублирующееся поле с признаком видимости "−", т. е. поле в основном визуальном элементе не показывается.

Элементы модели описывают либо объекты базы данных, либо визуальные элементы отображения. Модель КЭС строится отдельно для простой, сложной или комплексной услуги. Формальное описание клинико-экономических стандартов положено в основу базы данных и программной системы.

Программная реализация модели ведения КЭС

Построение информационной системы выполнено на основе программных модулей системы СтатЭкспресс [4]. Основными функциями информационной системы являются наполнение и ведение базы клинико-экономических стандартов оказания медицинской помощи. При этом выполняется поддержка временной и ссылочной целостности данных, проверка корректности, импорт и экспорт базы.

В качестве примера рассмотрим основную таблицу T_0 для простых медицинских услуг (см. таблицу).

Признак редактируемости указывает, что при реализации информационной модели в программной системе часть справочников не редактируется (редактирование справочников выполняется централизованно в единой системе ведения справочников [3]).

Наименование из модели КЭС	Идентификатор	Наименование	Признак редактируемости (0/1)
T_0	SMSERV	Таблица связи "Описание простых медицинских услуг"	1
U	MEDUSL	Справочник "Номенклатура работ и услуг"	0
O_1	_MEDUSL1	Оглавление справочника "Номенклатура работ и услуг"	0
D_1	FIO	Справочник "Специалисты"	1
D_2	DEM_GR	Справочник "Возрастные группы"	0
O_2	_DEM_GR1	Оглавление справочника " Возрастные группы "	0

Рассмотрим основные содержательные поля справочников.

В справочнике U:

$P_1(U)$ = (IDMEDUSL, "Ключевое поле", Целый, _, 0);
$P_2(U)$ = (MEDUSL, "Наименование медицинской услуги", Строка [250],, 0);
$P_3(U)$ = (RAZDEL, "Раздел", Строка [1], _, 0);
$P_4(U)$ = (VKT, "Код услуги", Строка [100], _, 0);
$P_5(U)$ = (UTYPE, "Тип услуги", Строка [250], _, 0);
$P_6(U)$ = (UETVR, "Условные единицы трудозатрат врача", Целый, _, 0);
$P_7(U)$ = (UETMS, "Условные единицы трудозатрат медсестры", Целый, _, 0).

В справочнике D_1:

$P_1(D_1)$ = (IDFIO, "Ключевое поле", Целый, _, 0);
$P_2(D_1)$ = (FIO, "ФИО специалистов", Строка [250], _, 0);
$P_2(D_1)$ = (STATUS, "Должность", Строка [250], _, 0).

В справочнике D_2:

$P_1(D_2)$ = (IDDEM_GR, "Ключевое поле", Целый, _, 0);
$P_2(D_2)$ = (DEM_GR, "Возрастная группа", Строка [250], _, 0).

Таблица T_0 содержит поля $P_1 - P_{20}$. Часть полей физически существует в таблице T_0:

$P_1(T_1)$ = (IDSMSERV, "Ключевое поле", Целый, _, 0);
$P_2(T_1)$ = (IDMEDUSL, "Простая услуга", Целый, U, $P_2(U)|P_3(U) + P_4(U)|P_5(U)|P_6(U)|P_7(U)$, 0);
$P_3(T_0)$ = (TIMEWAIT, "Допустимое время ожидания", Целый, _, _, 1);
$P_4(T_0)$ = (IDDEM_GR, "Возрастная группа", Целый, D_2, $P_2(D_2)$, 0);
$P_5(T_0)$ = (IDIDFIO1, "Автор", Целый, D_1, $P_2(D_1)|P_3(D_1)$, 0);
$P_6(T_0)$ = (DATE1, "Дата заполнения автором", Дата, 1);
$P_7(T_0)$ = (IDIDFIO2, "Эксперт", Целый, D_3, FIO|STATUS, 0);
$P_8(T_0)$ = (DATE2, "Дата заполнения экспертом", Дата, 1);
$P_9(T_0)$ = (MARGIND "Коэффициент для врача", Целый, 1);
$P_{10}(T_0)$ = (MARGINM "Коэффициент для медсестры", Целый, 1).

Остальные поля таблицы T_0 — виртуальные и строятся по описаниям физически заданных в таблице полей. Помимо этого в таблицах содержатся служебные поля, обеспечивающие временную целостность данных.

Например, в поле $P_2(T_0)$ содержится ссылка на справочник U — "Номенклатура работ и услуг", а в визуальном представлении требуется отображать поля из справочника: "*Наименование*", "*Раздел*" + "*Код услуги*", "*Тип услуги*" и "*Трудозатраты на оказание медицинской услуги*". Это задает следующую логику работы: пользователь при заполнении выбирает наименование услуги, с которой будет работать. Тип, код и трудозатраты автоматически отображаются в визуальных элементах, а в невидимое поле P_2 подставляется ссылка на соответствующее выбранной услуге значение идентификатора записи из справочника.

Способ отображения полей задается в описании с указанием типа визуального элемента (сетка таблицы, выпадающий список, поле ввода) и параметров визуализации. Ограничения на значения полей, представленные в модели через F_i, позволяют контролировать заполнение содержания клинико-экономического стандарта. Например, в U задается перечень всех простых, сложных и комплексных услуг, но для заполнения таблицы T_0 перечень услуг должен быть ограничен

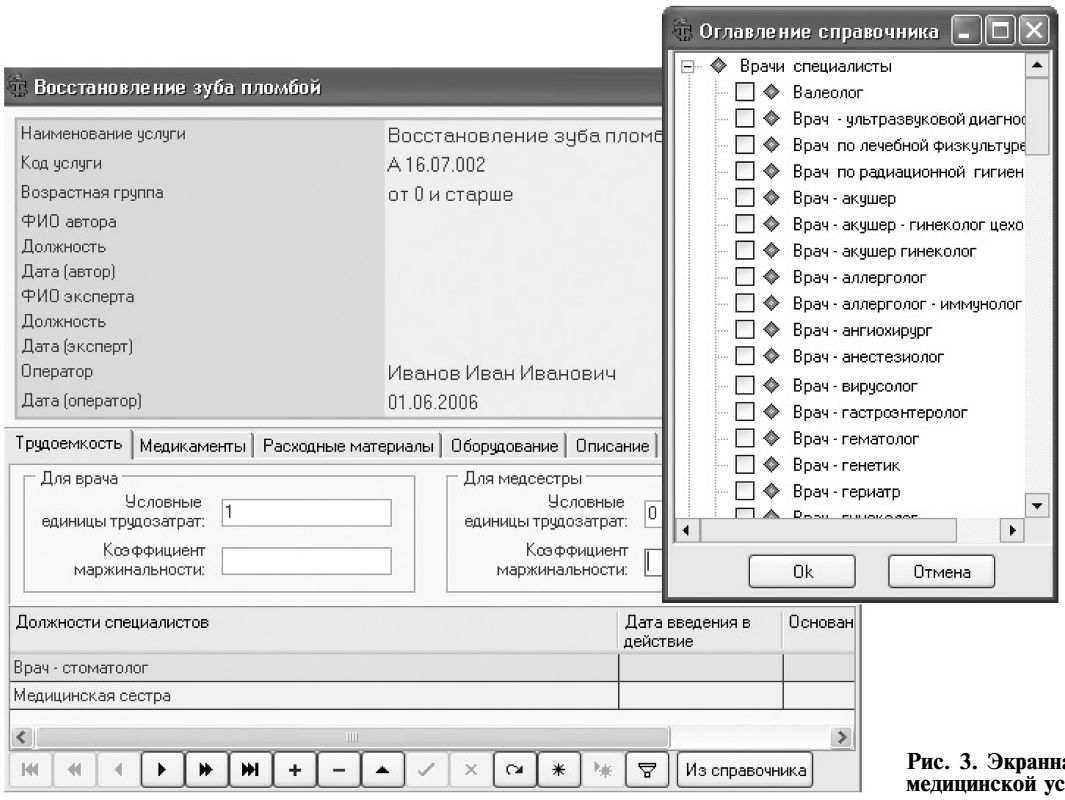


Рис. 3. Экранная форма для ввода простой медицинской услуги

только простыми услугами, поэтому в описании для поля $P_2(T_0)$ задается фильтр.

Для удобства наполнения базы данных в системе развиты функции копирования разделов услуг, функции быстрой навигации по стандартам, поддержки работы со справочниками. Введенное формальное описание позволило настраивать способ отображения данных в соответствии с пожеланиями специалиста предметной области.

Экранная форма для ввода простой медицинской услуги приведена на рис. 3.

Заключение

Построено унифицированное описание информационной модели стандартов оказания медицинских услуг. Информационная модель отражает сложную взаимосвязь параметров оказания медицинских услуг и предназначена для разработки норм, определяющих объем диагностических и лечебных процедур, требований к результатам лечения при определенных заболеваниях.

Обеспечивается возможность быстрой адаптации базы данных и информационной системы к изменениям требований к составу и содержанию медицинских услуг. Модель обладает простотой реализации и хорошей интегрируемостью с другими программными продуктами, функционирующими в системе здравоохранения и обязательного медицинского страхования.

Список литературы

1. **Виноградов К. А., Голубева Т. Н., Денисов В. С., Корчагин Е. Е., Исаева О. С., Никитина М. И.** Разработка и реализация технологий оказания медицинских услуг // Тр. Всероссийской конференции "Информационно-аналитические системы и технологии в здравоохранении и обязательном медицинском страховании". Красноярск: КМИАЦ, 2002. С. 262—266.

2. **Номенклатура** работ и услуг в здравоохранении (Утв. МинЗдравСоцразвития РФ 12.07.2004).

3. **Виноградов К. А., Жучков Д. В., Никитина М. И.** Система ведения базы данных нормативно-справочной информации // Врач и информационные технологии. М.: Менеджер здравоохранения. 2004. № 3.

4. **Евдокимов Д. В.** Средства автоматизации сбора и обработки медицинской статической информации. Автореф. канд. дисс. ... 05.13.01. Красноярск: ИВМ СО РАН. 2004. 24 с.

УДК 61:004.651.2

О. В. Воробейчикова, канд. техн. наук, доц.
Сибирский государственный
медицинский университет, г. Томск
E-mail: mbc@ssmu.tomsk.ru

Технология построения структуры усвоения учебного материала по результатам тестирования

Внедрение ПК в учебный процесс является предпосылкой широкого использования компьютерного тестирования как средства проверки знаний испытуемых.

Предлагается метод, позволяющий по результату тестирования делать вывод не только об уровне знаний испытуемых в целом по тесту, но и построить структуру усвоения ими учебного материала, и тем самым, выявить слабо усвоенные темы учебного материала.

Ключевые слова: *компьютерное тестирование, уровень знаний, структура усвоения учебного материала.*

Введение

В процессе обучения контроль играет важную роль в качестве источника информации об уровне знаний обучаемых. В настоящее время для контроля знаний все чаще используется компьютерное тестирование. Проведение компьютерного тестирования позволяет организовывать опрос в режиме, удобном преподавателю и студенту, в удобное время и так часто, как это необходимо с точки зрения преподавателя. Кроме того, оценка за тестирование, выставленная автоматически, не зависит от мнения преподавателя и поэтому воспринимается студентами как более объективная.

Оценка, полученная одним испытуемым, отражает его уровень знаний в целом по тому учебному материалу, который представлен в тесте. Но, кроме этого, для преподавателя результаты тестирования всей группы предоставляют полезную информацию, которая позволяет, во-первых, проанализировать сами задания теста, например, определить сложность заданий, а во-вторых, изучить структуру усвоения знаний испытуемыми.

Преподаватель и испытуемые, взаимодействуя с одним и тем же тестом, обращают внимание на разные его стороны (рис. 1). Испытуемые видят тест как набор заданий и определяют его, в первую очередь, как содержащий более или менее сложные задания, а только потом, как соотносятся эти задания с элементами предметной области.

При составлении теста преподаватель держит в голове структуру предметной области, и эта структура находит свое отражение в наборе заданий теста. Назовем такую структуру теста **экспертной.**

При этом нужно иметь в виду, что преподаватель всегда опирается на свое понимание предметной области. Привлечение других экспертов к построению структуры даст другой результат, причем может быть принципиально разный. Причины этого состоят не только в том, что у них может быть различная степень

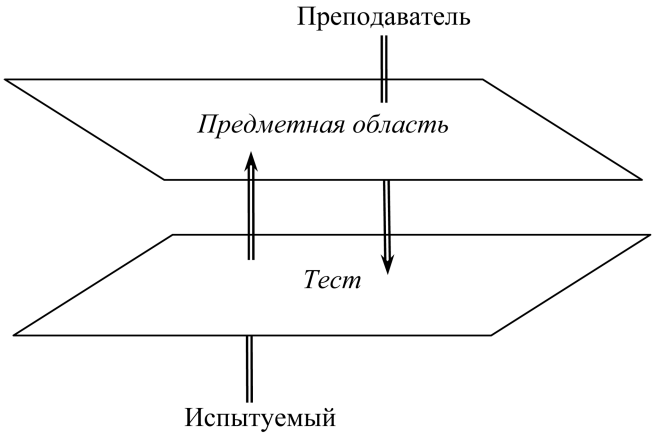


Рис. 1. Взаимодействие испытуемых и преподавателя с тестом и предметной областью

знания предметной области, даже один и тот же эксперт может определить разные структуры в зависимости от уровня преподнесения материала, от разной методики ведения занятий. Конечно, существуют методы согласования экспертных оценок, но все же следует помнить, что преподаватель всегда исходит из своего понимания предметной области, именно это понимание он вкладывает в порядок изложения материала и именно оно берется за основу при построении структуры теста.

Альтернативными подходами к построению структуры теста являются подходы, основанные на анализе экспериментальных данных, полученных при непосредственном тестировании самих испытуемых [1]. Назовем такую структуру **эмпирической.**

Технология построения эмпирической структуры теста

На кафедре медицинской кибернетики СибГМУ разработана технология, позволяющая по результатам тестирования построить эмпирическую структуру теста (рис. 2).

На рис. 2 приняты следующие обозначения: q_{ij} — ответ i-го студента на j-е задание (ответы фиксируются как 0 или 1); r_{ij}^{tet} — тетрахорический коэффициент корреляции между i-м и j-м заданиями теста; z % — шаг диапазона разбиения заданий теста по уровням сложности для построения структуры; l — число групп сложности; P_i — значение i-го диапазона сложности.

На первом этапе группа испытуемых подвергается опросу по всем заданиям. В результате составляется матрица, столбцы которой соответствуют заданиям, а строки — испытуемым. Элемент матрицы представляет собой ответ на соответствующее задание теста и равен единице, если испытуемый выполнил задание правильно, или нулю — в противном случае.

На втором этапе для каждой пары заданий подсчитывается число ответов испытуемых, которые представляют собой следующие пары соответствий:

а) выбраны правильные ответы на оба задания;

б) ответ на первое задание выбран правильный, на второе — неправильный;

в) ответ на первое задание выбран неверно, на второе — правильно;

г) неверно выбраны ответы на оба задания.

Полученные четырехзначные таблицы используются для вычисления меры связи (в виде тетрахорического коэффициента корреляции [2]) между заданиями в паре. В результате образуется квадратная симметричная матрица, строки и столбцы которой соответствуют заданиям, а элементами являются значения коэффициентов корреляции. Два задания считаются взаимосвязанными, если значение коэффициента корреляции между ними удовлетворяет установленному уровню значимости.

Из всех видов связей между двумя заданиями для определения структуры теста важными являются связи: "знает ответ на одно задание" — "знает ответ на второе задание" и "не знает ответ на первое задание" — "не знает ответ на второе задание". Такой тип связи соответствует положительному значению тетрахорического коэффициента корреляции, поэтому для построения эмпирической структуры рассматриваются только такие типы.

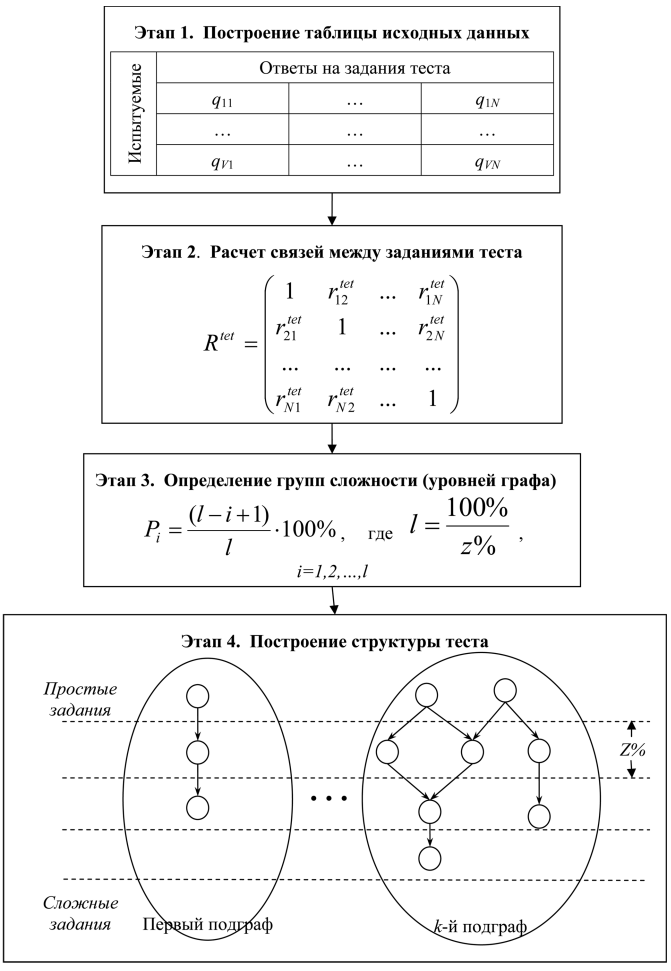


Рис. 2. Этапы формирования структуры теста

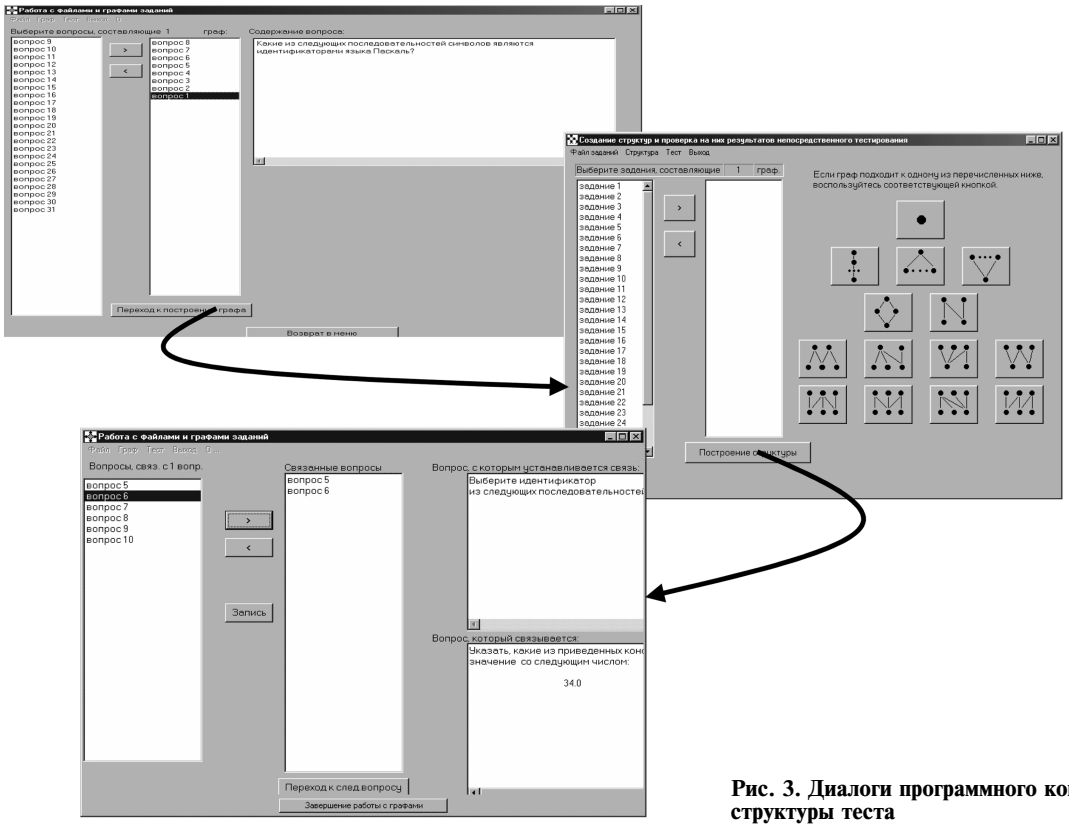


Рис. 3. Диалоги программного комплекса при построении структуры теста

На третьем этапе формируется шкала уровней сложности заданий, так как именно сложность задает направленность связи между заданиями в структуре. Сложность задания оценивается по частоте правильных ответов на это задание, вычисленной в процентном отношении по всем испытуемым. Более сложными считаются задания с меньшей частотой верных ответов. Таким образом, нулевая отметка шкалы соответствует самым сложным заданиям, а максимальная отметка (100 %) — самым простым. Шкала делится на равные интервалы, между которыми распределяются по сложности все задания теста.

Четвертый этап посвящен построению структуры теста в виде ориентированного графа, дуги которого направляются от заданий с меньшим уровнем сложности к более сложным заданиям. Из всех полученных связей нас интересуют только те, которые означают, что два задания связаны тогда, когда увеличение числа верных ответов на более легкое задание сопровождается устойчивым увеличением числа верных ответов на другое более сложное. Поэтому при построении структуры теста рассматриваются только связи между заданиями, расположенными на разных уровнях сложности.

Данная методика реализована на ЭВМ с помощью программного комплекса "ТеСТ", который позволяет выявить связи между заданиями теста и построить структурированный тест (рис. 3).

Анализ эмпирической структуры теста

Описанные две структуры, экспертная и эмпирическая, пересекаются в том смысле, что экспертная связь двух заданий отражает семантическую связь двух соответствующих понятий предметной области, которая подразумевает, что "нельзя усвоить понятие более сложное, если не знаешь понятие более легкое". Можно сказать, что структура теста определяется для него, прежде всего, как **структура обучения.**

Эмпирическая же связь означает, что два задания связаны тогда, когда увеличение числа верных ответов на одно задание (более легкое) сопровождается устойчивым увеличением числа верных ответов на другое (более сложное). При построении эмпирической структуры выявляются, прежде всего, связи самих заданий, а через них — связи соответствующих элементов предметной области. Эти связи показывают, в первую очередь, меру близости заданий между собой, и опосредованно, — меру близости элементов предметной области, хотя, конечно же, структура предметной области оказывает влияние на структуру теста и отражается в ней. Таким образом, можно сказать, что эмпирическая структура отражает **структуру усвоения.**

Понятно, что эти две структуры (эмпирическая и экспертная) будут отличаться, но можно сказать, что они будут тем ближе, чем яснее преподаватель представляет себе структуру усвоения понятий обучаемыми. А это, в свою очередь, можно сделать, изучив эмпирическую структуру, построенную по результатам тестирования.

Каждый преподаватель знает, что для лучшего усвоения знаний необходимо студентам дать представление не только об основных элементах учебной темы, но и показать их взаимосвязь. Следовательно,

можно ожидать, что чем эмпирическая структура будет связанней, тем лучше студенты воспримут эту взаимосвязь и, следовательно, лучше воспримут саму изучаемую тему. Если же структура будет разорванной (или слабоструктурированной), то, скорее всего студенты механически восприняли изучаемую тему, или, другими словами, просто заучили учебный материал, без глубинного понимания.

Для подтверждения этих рассуждений приведем эмпирическую структуру теста по разделам "Синтаксис языка Паскаль" и "Операторы языка Паскаль" курса "Информатика" для студентов III курса специальности "Медицинская кибернетика" СибГМУ. Тестировались две группы. Надо отметить, что студенты обеих групп получили, в целом, неплохие оценки по тесту, средний балл у первой группы — 22 балла, у второй — 16, хотя во второй группе уровень знаний у студентов ниже, чем в первой.

Анализ структуры теста (рис. 4), полученной по результатам тестирования первой группы, позволил сделать следующие выводы. Выделилась группа заданий (16 % от общего числа заданий), не имеющих достоверных связей с остальными заданиями теста. Тип заданий, в данном случае, не имеет значения: задания 4, 11 и 19 — это задания на определение понятий; 21, 30 — это задания на решение задач. По типу ответа: задания 4, 21 — задания с множественным выбором, остальные — с единственным правильным ответом. По сложности: 4 и 19 — легкие задания (более 80 % испытуемых ответили на них верно); 21, 11 — сложные задания (менее 60 % верных ответов); задание 30 оказалось средним по сложности.

И наконец, темы данных заданий: задание 4 относится к теме "Целочисленные константы", задание 11 — к теме "Условные выражения языка", задание 19 относится к теме "Условные операторы", задания 21 и 30 — к теме "Операторы цикла". Если не брать во внимание задание 4, то получается, что остальные задания связаны с темой "Условные операторы" (замечу, что тема "Операторы цикла" тесно связана с этой темой). Поэтому возникло предположение, что данная тема была плохо усвоена студентами, и была проведена дополнительная контрольная работа по теме "Условные операторы", которая подтвердила это предположение. В результате было выделено дополнительное время в учебном плане на закрепление данного учебного материала.

По результатам тестирования второй группы также была построена структура усвоения учебного материала (рис. 5).

Тот факт, что студенты второй группы слабее по уровню знаний, чем студенты первой группы, отражает вид полученных эмпирических структур. Во второй группе оказалось несвязанным большее число заданий, а именно, 25 %, чем в первой группе — 16 %. Если обратить внимание на число полученных подграфов, то в первой группе их — 4, во второй группе — 6, что также свидетельствует о более низком уровне знаний студентов второй группы. Несвязанными во второй группе оказались задания (большинство), относящиеся к теме "Оператор цикла", поэтому со студентами этой группы также было проведено дополнительное занятие по данной теме.

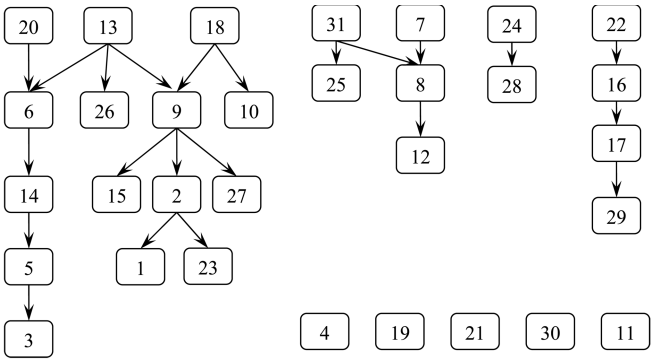


Рис. 4. Эмпирическая структура, построенная по результатам тестирования первой группы

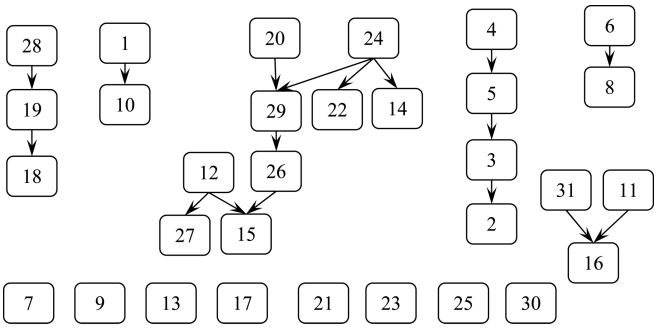


Рис. 5. Структура теста по результатам тестирования второй группы

Таким образом, польза компьютерного тестирования не только в том, что оно позволяет преподавателю получить сведения об уровне знаний каждого из студентов в отдельности. Построив структуру теста на основе расчета связей между ответами испытуемых на задания теста, можно получить информацию об усвоении всей группой учебного материала: чем меньше несвязанных заданий будет выявлено, тем, значит, лучше усвоена тема студентами. Анализ тем, попавших в "несвязанные", служит своеобразным сигналом о том, что на них следует обратить особое внимание.

Список литературы

1. **Воробейчикова О. В.** Структуризация тестов для оценки знаний в системе ДО // Научное и методическое обеспечение системы дистанционного образования. Материалы Международной конференции. — Томск: Изд-во Томского университета, 2000. С. 84—85.

2. **Гласс Дж., Стэнли Дж.** Статистические методы в педагогике и психологии: Пер. с англ. / Общ. ред. Ю. П. Адлера. — М.: Прогресс, 1976. 495 с.

ДИСКУССИОННЫЙ КЛУБ

Письмо в редакцию

УДК 518.5

В. И. Левин, д-р техн. наук,
Пензенская государственная
технологическая академия
E-mail: levin@pgta.ac.ru

Что такое интервальная задача математического программирования

В статье [1] описана некоторая модель оперативного планирования производственной деятельности, отличающаяся от традиционных моделей этого назначения в форме задач математического программирования с точно известными коэффициентами и точно определяемыми переменными тем, что переменные определяются в интервальном виде. Автор статьи считает, что его подход позволяет "отражать в моделях неточность исходных данных, динамику протекающих процессов, непредсказуемые флуктуации мощностей технологических агрегатов... и многое другое". Все эти преимущества подхода, конечно, соблазнительны. Однако существуют ли они в действительности? И вообще, существует ли сам подход?

В статье [1] предлагается искать в интервальной форме решения задач математического программирования, коэффициенты которых известны точно. Однако из теории математического программирования хорошо известно, что такие задачи всегда (за исключением некоторых вырожденных случаев) имеют единственное решение — это обеспечивается соответствующими теоремами существования и единственности задач математического программирования [2]. Так, любая задача линейного программирования с точечными коэффициентами имеет единственное решение, которое достигается в одной из угловых точек многоугольника (многогранника) допустимых решений. Короче, задачи математического программирования с точечными коэффициентами имеют точечные решения, а не интервальные или в форме "многомерного параллелепипеда", как утверждается в статье [1], которая фактически приглашает читателя искать то, чего не существует. Из этого базового положения следует соответствующая оценка и всех остальных построений и утверждений статьи. Сверх того, изложение материала в статье страдает неточностью, что лишь запутывает читателя. Ограничусь двумя примерами.

1. Цитата из [1]: "Кроме условия интервальности переменных, такие задачи внешне ничем не отличаются от задач математического программирования — таким образом, появляется возможность применения уже известных моделей". В этом отрывке из [1], помимо его неточности (следовало сказать "от задач математического программирования с точечными коэффициентами"), содержится неверное утверждение, согласно которому из одинаковости записи двух различных задач следует возможность их решения одинаковыми средствами. Следующий простейший пример опровергает это утверждение: операции умножения вещественных чисел и матриц с вещественными элементами записываются одинаково, но выполняются по совершенно различным правилам. В нашем случае принципиальное различие между точечными и интервальными постановками задач математического программирования, при всем их внешнем сходстве, состоит в том, что правила действий над вещественными интервалами совсем другие, чем правила действия над вещественными числами [3, 4]. Поэтому решать интервальные задачи методами решения точечных задач напрямую нельзя.

2. А вот другой отрывок из [1]: "Однако решение этой (по мнению автора статьи [1], интервальной — В. Л.) задачи представляет не точку, а многомерный параллелепипед, погруженный в область допустимых решений. Допустимым решением считается любая точка этого параллелепипеда, а координаты центра его определяют значение целевой функции". В этом отрывке все неверно. Во-первых, сформулированная в [1] задача математического программирования имеет точечные коэффициенты и потому, в соответствии с теорией математического программирования, имеет решение в виде точки, а не многомерного интервала (параллелепипеда) — см. выше. Во-вторых, даже если предположить, что, вопреки теории указанная задача каким-то чудом получила решение в виде многомерного параллелепипеда, этот параллелепипед находился бы внутри области допустимых решений, и любая его точка была решением задачи, а не ее допустимым решением. Другими словами, в этом случае имелось бы бесконечное множество решений и, соответственно, бесконечное множество экстремальных значений целевой функции. В-третьих, выделение в [1] в качестве решения задачи центра многомерного параллелепипеда решений (в предположении, что последний существует) ни на чем не основано. С точки зрения теории, все точки множества решений задачи математиче-

ского программирования равноправны, и потому выделение одной из них в качестве исключительного решения требует специального обоснования, выходящего за рамки указанной теории.

После сказанного возникает естественный вопрос: а можно ли вообще построить теорию и методы решения интервальных задач математического программирования, которые бы позволяли учитывать неточность исходных данных, динамику изучаемых процессов и другие факторы неопределенности и вместе с тем опирались на уже имеющиеся теорию и методы решения соответствующих точечных задач? Положительный ответ на этот вопрос был получен автором этих строк еще в 1989 году и опубликован в 1990-е годы в ряде изданий, включая журнал "Информационные технологии" ([4]—[22]). Идея заключалась в том, что предлагалась математическая модель проектирования различных объектов и планирования процессов, функционирующих в условиях неопределенности, в форме соответствующей задачи математического программирования с интервальными коэффициентами, которые и аккумулируют в себе указанную неопределенность. Решение всех возникающих интервальных задач математического программирования осуществлялось на базе специально построенной теории сравнения интервальных чисел и выбора большего и меньшего интервальных чисел. Сравнение интервалов, как показано в этой теории, сводится к сравнению их одноименных границ. При этом сравнение некоторых интервалов оказывается невозможным, что можно рассматривать как плату за их неопределенность. Построенная теория сравнения интервальных чисел позволяет сводить решение любой задачи математического программирования с интервальными коэффициентами к совместному решению пары однотипных с исходной задач математического программирования с точечными коэффициентами. Эти задачи называются нижней и верхней граничными задачами исходной задачи. Обе эти задачи формируются из исходной интервальной задачи математического программирования с помощью описанной выше теории сравнения интервальных чисел при использовании общеизвестных правил интервальной алгебры [3, 4]. Заканчивается процедура конструированием решения интервальной задачи по найденным решениям ее нижней и верхней граничных задач. При этом используется следующее доказанное положение: множество решений задачи математического программирования с интервальными коэффициентами равно пересечению множеств решений ее нижней и верхней граничных задач. Поскольку обе граничные задачи имеют точечные коэффициенты, их можно решать известными методами решения задач математического программирования. Тем самым решение интервальных задач математического программирования, а вместе с ним и исследование систем оптимального планирования производства в условиях неопределенности оказываются возможными осуществлять стандартными методами решения задач математического программирования с точно известными коэффициентами.

Список литературы

1. **Поляков В. В.** Планирование производственной деятельности на основе задач математического программирования с интервальными переменными // Информационные технологии. 2006. № 10. С. 40—42.
2. **Браверман Э. М.** Математические модели планирования и управления в экономических системах. М.: Наука, 1976. 375 с.
3. **Алефельд Г., Херцбергер Ю.** Введение в интервальные вычисления. М.: Мир, 1987. 360 с.
4. **Левин В. И.** Интервальная математика и исследование систем в условиях неопределенности. Пенза: Изд-во ПТИ, 1998. 56 с.
5. **Левин В. И.** Интервальный подход к оптимизации в условиях неопределенности // Вопросы управления и проектирования в информационных и кибернетических системах. Уфа: Изд-во УГАТУ, 1990.
6. **Левин В. И.** Дискретная оптимизация в условиях интервальной неопределенности // Автоматика и телемеханика. 1992. № 7.
7. **Левин В. И.** Булево линейное программирование с интервальными коэффициентами // Автоматика и телемеханика. 1994. № 7.
8. **Левин В. И.** Интервальное дискретное программирование // Кибернетика и системный анализ. 1994. № 6.
9. **Левин В. И.** Оптимизация расписаний в системах с неопределенными временами обработки. I, II // Автоматика и телемеханика. 1995. № № 2, 3.
10. **Левин В. И.** Задача трех станков с неопределенными временами обработки // Автоматика и телемеханика. 1996. № 1.
11. **Левин В. И.** Нелинейная оптимизация в условиях неопределенности // Вестник Тамбовского университета. Сер.: естественные и технические науки. 1997. Т. 2. № 3.
12. **Левин В. И.** Игры с интервальными параметрами // Вестник Тамбовского университета. Сер. Естественные и технические науки. 1997. Т. 2. № 3.
13. **Левин В. И**. Математическая теория сравнения интервальных величин // Измерительная техника. 1998. № 5.
14. **Левин В. И.** Интервальная математика и изучение неопределенных систем // Информационные технологии. 1998. № 6.
15. **Левин В. И.** Сравнение интервальных величин и оптимизация неопределенных систем // Информационные технологии. 1998. № 7.
16. **Левин В. И.** Интервальная модель общей задачи линейного программирования // Вестник Тамбовского университета. Сер. Естественные и технические науки. 1998. Т. 3. № 4; 1999. Т. 4. № 1.
17. **Левин В. И.** Интервальный подход к оптимизации в условиях неопределенности // Информационные технологии. 1999. № 1.
18. **Левин В. И.** Задачи непрерывной оптимизации в условиях интервальной неопределенности // Информационные технологии. 1999. № 7.
19. **Левин В. И.** Задачи нелинейной оптимизации с интервальными параметрами // Информационные технологии. 1999. № 12.
20. **Левин В. И.** Нелинейная оптимизация в условиях интервальной неопределенности // Кибернетика и системный анализ. 1999. № 2.
21. **Левин В. И.** Антагонистические игры с интервальными параметрами // Кибернетика и системный анализ. 1999. № 3.
22. **Левин В. И.** Интервальные методы оптимизации систем в условиях неопределенности. Пенза: Изд-во ПТИ, 1999. 95 с.

Уважаемая редакция!

Благодарю Вас за предоставленную возможность ознакомиться с замечаниями на мою статью, сделанными д-ром техн. наук, профессором В. И. Левиным. Его письмо позволило мне лучше понять, какие именно аспекты рассматриваемой задачи вызывают у читателя непонимание и требуют более подробных объяснений.

Хочется поблагодарить и Виталия Ильича за внимание, проявленное к моей работе, и пояснить причины, побудившие меня к рассмотрению затронутых в статье вопросов и задач.

Интерес к подобным задачам имеет практический характер. Он возник на основе опыта, накопленного автором и его коллегами по решению задач оперативного планирования работы оборудования целлюлозно-бумажных предприятий, представляющих сложные технологические системы с непрерывным процессом производства [1, 2]. Целью решения этих задач — задач координации работы технологического оборудования — является поиск таких производительностей агрегатов, которые обеспечивают сбалансированность материальных потоков системы в течение заданного времени (от десятков минут до десятков часов) и, одновременно, достижение наилучшего по некоторому критерию результата производственной деятельности.

При использовании методов математического моделирования на практике для достижения этой цели чаще всего применяют несложные, относительно грубые модели, приводящие к задачам линейного программирования (ЛП) с точечными коэффициентами. Выбор этого класса оптимизационных задач обусловливается, прежде всего, простотой их формулировки (в статье отмечено, что на производстве затруднительно найти специалиста, способного сформулировать даже такие задачи, не говоря о более сложных) и наличием хорошо зарекомендовавших себя методов решения. Не менее важен и тот факт, что данные задачи требуют небольшого объема исходных данных, необходимых для решения. Однако для реальных производственных проблем математические модели, приводящие к задачам ЛП, далеко не всегда оказываются адекватными.

Решение классической задачи ЛП имеет, как справедливо отмечает Виталий Ильич, точечное решение, находящееся в одной из угловых точек многоугольника допустимых решений (хотя возможен и случай альтернативных решений). Однако в условиях реального производства строго реализовать рассчитанное оптимальное решение практически невозможно. Значения параметров, соответствующих переменным (например, производительности агрегатов), из-за влияния различных возмущающих факторов будут изменяться в некоторых пределах, отличаясь вследствие этого от оптимальных значений. Рассмотрим возможные последствия этого на примере стандартной задачи ЛП, в системе ограничений которой присутствуют только неравенства:

$$\sum_{j \in N} c_j x_j \to \max;$$

$$\sum_{j \in N} a_{ij} x_j \leqslant b_i, \quad i \in M;$$

$$x_j \geqslant 0, \quad j \in N.$$

Система ограничений такой задачи задает выпуклое множество (выпуклый многогранник решений), как правило, содержащее внутренние точки (точки, принадлежащие множеству, вместе с некоторой окрестностью). Поскольку точка оптимального решения задачи ЛП всегда находится на границе области допустимых решений, отклонение значения даже одного управляемого параметра — производительности агрегата, от оптимального значения с вероятностью не ниже 0,5 приведет к тому, что реализованное решение станет недопустимым (точка, соответствующая реализованному решению, окажется вне области допустимых решений). С математической точки зрения это означает нарушение одного или нескольких ограничений, а для упомянутых выше задач координации — разбалансировку материальных потоков, что не только нежелательно, но просто недопустимо.

Приведем пример. Рассмотрим систему ограничений, характерных для многих оптимизационных задач, формулируемых как задачи ЛП:

$$b_i' \leqslant \sum_{j \in N} a_{ij} x_j \leqslant b_i'', \quad i \in M, \qquad (*)$$

где величины b_i' и b_i'' определяют, например, допустимые диапазоны запасов некоторых продуктов, которые могут не только потребляться, но и создаваться в процессе производства. Переменные x_j здесь могут рассматриваться как объемы выпуска некоторых видов продукции, а найденное решение — как план работы некоторого производства.

Поскольку оптимальное решение задачи ЛП является крайней точкой допустимого множества решений (за исключением вырожденных случаев), выражения $\sum_{j \in N} a_{ij} x_j$, входящие в ограничения (*), на оптимальном решении часто принимают какие-то из крайних значений b_i' и b_i''. Реализовать оптимальный план работы на практике удается редко, вследствие чего становится возможной ситуация, когда значения $\sum_{j \in N} a_{ij} x_j$ могут стать либо меньше b_i', либо больше b_i''. В то же время, с точки зрения производства, вполне допустимо, если в ходе реализации оптимального решения значения выражений $\sum_{j \in N} a_{ij} x_j$ окажутся либо несколько больше b_i', либо несколько меньше b_i''. Например, в случае задач координации определяющим является

обеспечение сбалансированности технологической системы в целом, даже если не будет достигнут формальный оптимум.

Таким образом, можно сделать вывод, что вполне приемлемым для производства является план, соответствующий некоторой внутренней точке множества допустимых решений задачи ЛП, близкой к точке оптимального решения, который может рассматриваться как субоптимальное решение исходной задачи. Если реализовывать субоптимальное решение задачи, то при реализации становятся допустимыми некоторые отклонения значений объемов выпуска продукции, являющихся переменными, от значений субоптимального решения, не приводящие к нарушениям ограничений. Если задать величины допустимых отклонений, то в качестве окрестности точки субоптимального решения возникает параллелепипед фиксированных размеров, любая точка которого будет также считаться допустимым субоптимальным решением. Поскольку значение целевой функции можно рассчитать только для конкретной точки, представляется разумным выбрать в качестве такой точки центр возникающего параллелепипеда (не исключено, что могут быть и другие соображения по выбору такой точки).

Например, для задачи:

$$x_1 \to \max; \tag{1}$$

$$x_1 + x_2 \leqslant 10; \tag{2}$$

$$x_1 \geqslant 1, \quad x_2 \geqslant 1, \tag{3}$$

если допустить возможность отклонения значения переменной x_1 от субоптимального не более, чем на 1, а для переменной x_2 не более, чем на 2, т. е. разрешить переменным x_1 и x_2 принимать любые значения в диапазонах

$$x_1 \in [w_1 - 1;\ w_1 + 1], \tag{4}$$

$$x_2 \in [w_2 - 2;\ w_2 + 2], \tag{5}$$

где w_1 и w_2 — срединные точки интервалов допустимых значений переменных, решением задачи (1)—(5) будет считаться прямоугольник размерами 2×4 $\{3 \leqslant x_1 \leqslant 5,\ 1 \leqslant x_2 \leqslant 5\}$ со срединной точкой (4, 3), каждая точка которого удовлетворяет ограничениям (1)—(3).

Заметим, что переменными в задаче (1)—(5) можно считать значения w_1 и w_2, зависимыми от которых являются интервалы допустимых значений x_1 и x_2. При этом в целевую функцию должны входить только точечные значения w_1 и w_2, а ограничения задачи должны выполняться для всех значений x_1 и x_2, попадающих в интервалы, соответствующие найденным значениям w_1 и w_2. То есть решение задачи (1)—(5), так же как и задачи, сформулированной в статье [3], является, по сути, точечным, и предложенный в [3] способ преобразования иллюстрирует данное обстоятельство, позволяя перейти от задачи с интервальными переменными к классической задаче нелинейного программирования, имеющей точечное решение, на основе которого легко строится интервальное.

Таким образом, возникающая задача не является классической задачей математического программирования как с точечными, так и с интервальными параметрами (найти в литературе аналогов не удалось). Вероятно, именно это обстоятельство и ввело в заблуждение уважаемого Виталия Ильича, судя по письму, неправильно интерпретировавшего сформулированную в [3] задачу (как показывает опыт — это типичная ошибка тех, кто впервые сталкивается с такой постановкой задачи).

Данная задача существенным образом отличается от задач интервального программирования, которые рассматриваются в работах Виталия Ильича и других авторов, например, в работах [4—8], и вряд ли может быть решена предложенными в этих работах методами, в частности, методом детерминизации [4, 5]. Тем не менее, предлагаемая постановка задачи, на наш взгляд, имеет право на существование, о чем говорилось выше. При этом, безусловно, следует согласиться с замечаниями по поводу ряда терминологических неточностей, замеченных Виталием Ильичем и заметить в оправдание, что пока еще по отношению к предложенному классу задач не сформировалась четкая терминология.

С работами Виталия Ильича я знаком (к сожалению, не со всеми, поэтому хотелось бы поблагодарить Виталия Ильича за приведенную в его письме библиографию) и не счел необходимым сделать на них ссылки в своей статье исключительно ввиду существенного различия рассматриваемых задач. Надеюсь, что Виталия Ильича удовлетворят сделанные здесь разъяснения существа и формулировки представленной в [3] задачи.

В заключение обращаюсь с убедительной просьбой сообщить, сочли ли уважаемые редакция и Виталий Ильич достаточными приведенные разъяснения позиции автора статьи и возможна ли в дальнейшем публикация в Вашем журнале статей, посвященных данной тематике?

Список литературы

1. **Воронин А. В., Кузнецов В. А.** Математические модели и методы в планировании и управлении предприятием ЦБП. Петрозаводск: Изд-во ПетрГУ, 2000.
2. **Поляков В. В.** Оптимизация управления целлюлозно-бумажным производством // Известия высших учебных заведений. Лесной журнал. № 3, 2006.
3. **Поляков В. В.** Планирование производственной деятельности на основе задач математического программирования с интервальными переменными // Информационные технологии. 2006. № 10.
4. **Левин В. И.** Дискретная оптимизация в условиях интервальной неопределенности // Автоматика и телемеханика. 1992. № 7.
5. **Левин В. И.** Интервальное линейное программирование и теория интервальных чисел. [Электронный ресурс]. Режим доступа: http://webservis.ru/it/scm/1999/session1/levin.html
6. **Тимохин С. Г., Шапкин А. В.** О задачах линейного программирования в условиях неточных данных // Экономика и математические методы. 1981. Т. 17. № 5.
7. **Ватолин А. А.** О задачах линейного программирования с интервальными коэффициентами // Журнал вычислительной математики и математической физики. 1984. Т. 24. № 11.
8. **Ащепков Л. Т., Косогорова И. Б.** Интервальная задача линейного программирования // Экономика и математические методы. 2006. Т. 42. № 3.

24.12.2007 В. В. Поляков

УДК 518.5

В. И. Левин, д-р техн. наук,
Пензенская государственная
технологическая академия

Еще раз об интервальных задачах математического программирования

В моем письме в редакцию [1] была дана оценка статьи [2], напечатанной в журнале "Информационные технологии", и показано, что, вопреки названию статьи, рассматриваемая в ней задача не является интервальной задачей математического программирования, а используемый подход, вопреки обещанию автора, не позволяет осуществлять оперативное оптимальное планирование производства, "отражая в моделях неточность исходных данных, динамику протекающих процессов, непредсказуемые флуктуации мощностей агрегатов... и многое другое" (цитата из [2]). Были также приведены примеры ошибочности некоторых утверждений и построений, сделанных в статье [2], а также дана краткая справка об одном, математически корректном построении модели интервальных задач математического программирования.

В своем ответе [3] автор статьи [2] обходит молчанием критику его работы, прозвучавшую в моем письме [1]. Он рассматривает эту критику как результат "непонимания" и "неправильной интерпретации сформулированной в [2] задачи", что является "типичной ошибкой тех, кто впервые сталкивается с такой постановкой задачи", и видит проблему лишь в том, что "требуются более подробные объяснения". Такой подход к делу явно неконструктивен: ведь критика работы совсем не обязательно связана с ее непониманием; так что было бы гораздо лучше, если бы автор статьи [2] дал прямые ответы на прозвучавшие критические замечания. Однако, поскольку ответы так и не были даны, а обсуждаемый вопрос правильного построения математических моделей оперативного планирования и управления производством в условиях неопределенности его параметров весьма важен, считаю нужным еще раз рассмотреть этот вопрос — на этот раз в тех терминах, которые предложил сам автор статьи [2] в своем письме в редакцию [3]. Тем самым теперь исключаются любые разговоры о "непонимании" работы автора.

В письме [3] сказано: "Решение классической задачи линейного программирования (ЛП) есть точечное решение, находящееся в одной из угловых точек многоугольника допустимых решений (МДР). Однако в условиях реального производства строго реализовать рассчитанное оптимальное решение практически невозможно, значения параметров, соответствующих переменным (например, производительности агрегатов), вследствие влияния возмущающих факторов будут изменяться в некоторых пределах, отличаясь вследствие этого от оптимальных значений. Поскольку точка оптимального решения задачи ЛП всегда находится на границе МДР, отклонение значения даже одного управляемого параметра от оптимального значения приведет к тому, что реализованное решение станет недопустимым. Это означает нарушение одного или нескольких ограничений, т. е. разбалансировку материальных потоков, что не только нежелательно, но и недопустимо... Таким образом, можно сделать вывод, что вполне приемлемым для производства является план, соответствующий некоторой внутренней точке МДР задачи ЛП, близкой к точке оптимального решения, который может рассматриваться как субоптимальное решение (СОР) исходной задачи. Если реализовать СОР задачи, то становятся допустимыми некоторые отклонения объемов выпуска (переменных) от значений СОР, не приводящие к нарушениям ограничений. Если задать величины допустимых отклонений, то в качестве окрестности точки СОР возникает параллелепипед фиксированных размеров, любая точка которого будет также допустимым СОР". Как следует из приведенного отрывка, автор статьи [2] предлагает строить математическую модель оперативного планирования производства в условиях неопределенности его параметров в виде интервальной задачи математического программирования, особенности которой в том, что: 1) в интервальных переменных неизвестными считаются лишь центры интервалов, а их ширина задается; 2) ищется не оптимальное решение задачи, а ее СОР. При этом предполагается, что благодаря этим особенностям можно получить решение указанной задачи, обладающее одновременно двумя преимуществами: а) отклонения значений переменных относительно полученного точечного решения, находящиеся в пределах соответственно заданных интервалов, не выводят это решение задачи за пределы МДР; б) получаемое решение является СОР, т. е. близко к оптимальному. Однако такое предположение ошибочно, так как требования а) и б) взаимно противоречивы. Именно, если мы хотим обеспечить преимущество а) при заданных достаточно широких интервалах, то мы не можем обеспечить достаточную близость получаемого решения к оптимальному, и обратно. Проиллюстрируем сказанное на том же примере из письма в редакцию [3], на котором его автор дает "более подробные объяснения" для "непонимающих" и "неправильно интерпретирующих" его статью [2].

"Пример. Решить задачу

$$x_1 \to \max, \quad (1)$$

$$x_1 + x_2 \leqslant 10, \quad (2)$$

$$x_1 \geqslant 1,\ x_2 \geqslant 1, \quad (3)$$

при допустимости отклонений значения переменной x_1 от субоптимального не более чем на 1, а для переменной x_2 не более чем на 2, т. е. разрешении x_1 и x_2 принимать любые значения в диапазонах

$$x_1 \in [w_1 - 1,\ w_1 + 1]; \quad (4)$$

$$x_2 \in [w_2 - 2,\ w_2 + 2], \quad (5)$$

где w_1 и w_2 — срединные точки интервалов допустимых значений переменных x_1, x_2.

Решением задачи (1)—(5) будет прямоугольник размеров 2×4: $\{3 \leqslant x_1 \leqslant 5,\ 1 \leqslant x_2 \leqslant 5\}$, со срединной точкой ($w_1 = 4$, $w_2 = 3$), каждая точка которого удовлетворяет ограничениям (1)—(3)".

Ясно, что полученное решение $I = (w_1 = 4,\ w_2 = 3)$ обладает свойством а), т. е. при отклонении от него переменных x_1, x_2 в пределах интервалов (4), (5) точка (x_1, x_2) не выходит за пределы МДР. Однако обладает ли оно свойством б), т. е. является ли оно СОР-решением, близким к оптимальному? Последнее получается как решение классической задачи ЛП (1)—(3), в которой недопустимы никакие отклонения значений переменных x_1, x_2. Очевидно, что это решение

$$I_{\text{опт}} = (x_1' = 9,\ x_2' = 1).$$

Разница между решениями I и $I_{\text{опт}}$ равна $d(I, I_{\text{опт}}) = x_1' - w_1$ или, в относительных единицах, $\rho(I, I_{\text{опт}}) = d(I, I_{\text{опт}})/x_1' = (x_1' - w_1)/x_1' = (9 - 4)/9 = 0{,}55 = 55\ \%$. Видим, что разница между решением задачи I, полученным по методике [2], и оптимальным решением $I_{\text{опт}}$ очень велика (более чем двукратная), так что решение I не является СОР (субоптимальным решением) задачи (1)—(5).

Итак, решив предложенную в примере задачу оптимизации в условиях интервальной неопределенности переменных по методике статьи [2], мы не получили не только оптимального, но даже и субоптимального решения. Известно, что задачи оптимизации решаются в целях получения строго оптимального или близкого к нему (субоптимального) решения. Таким образом, подход к решению оптимизационных задач некоторого класса, предложенный в [2], несостоятелен.

Вместе с тем, задачи рассмотренного в [2] класса важны для практики. Поэтому разработка эффективных подходов к их решению должна быть продолжена.

Список литературы

1. **Левин В. И.** Что такое интервальная задача математического программирования. Письмо в редакцию "Информационные технологии". 2007.

2. **Поляков В. В.** Планирование производственной деятельности на основе задач математического программирования с интервальными переменными // Информационные технологии. 2006. № 10.

3. **Поляков В. В.** Письмо в редакцию "Информационные технологии". 2007.

От редакции

Задачи принятия оптимальных управленческих и проектных решений в производственной и проектной деятельности продолжают оставаться весьма актуальными. В связи с этим редакция считает, что вопросы, поднятые в письмах В. И. Левина по поводу статьи В. В. Полякова, опубликованной в журнале "Информационные технологии" № 10 2006 г., и ответы автора представляют интерес для читателей журнала. По мнению редакции, возможно продолжение дискуссии с расширением проблемы на решение задач оптимизации с учетом допусков на управляемые параметры.

Представляем книгу

Российская энциклопедия CALS. Авиационно-космическое машиностроение / Гл. ред. А. Г. Братухин. М.: ОАО "НИЦ АСК", 2008, 608 с.: илл.

Состоялась презентация уникального издания, освещающего методы и средства автоматизации всех этапов жизненного цикла сложной наукоемкой продукции авиационно-космической отрасли промышленности.

Как отметил в предисловии к книге генеральный директор НПО "Энергомаш имени академика В. П. Глушко" академик РАН Б. И. Каторгин, "... энциклопедия CALS преследует главную цель — обобщить накопленный опыт, провести первичную систематизацию материалов и наметить пути дальнейшего совершенствования технологий обеспечения разработки, производства и эксплуатации конкурентоспособной машиностроительной продукции в области авиации и космонавтики".

Главы энциклопедии посвящены вопросам стратегии CALS, особенностям жизненного цикла и электронному представлению изделий, конструкторско-технологической подготовке и организации производства, логистике, эксплуатации и экологии авиационно-космической техники, кадровому обеспечению CALS.

В написании книги участвовали более 100 известных ученых НИИ, КБ, предприятий авиакосмической индустрии, технических университетов.

Энциклопедия адресована специалистам не только авиационной, ракетно-космической, но и другой наукоемкой техники, а также преподавателям и студентам высших технических учебных заведений.

CONTENTS

In the paper the realization of the information system for the description of medical services is considered. The problem has demanded construction of the formal description of elements of realization for any adjustment of visual display of data. The block diagram of a subject domain constructed on its basis model of data and an example of realization of adapted information system is resulted. The results can be used for construction of information systems with complex structure.

Keywords: designing of databases, information systems, formal model of visualization of data, system for the description of medical services.

Nowadays computers are active used for student's knowledge testing. Such kind of testing can take place in such a way and so often which is convenient both to students and teacher. Besides the result of computer testing doesn't depend on teacher's meaning. Because of that this result is more objective and receptive for students. As a rule the control process is assumed to be finished when the mark is given. In this paper the method is proposed which could be used for construction a structure of student's learning skills. The testing results show the parts of the course that are learned in unsufficient level.

Keywords: computer testing, student's knowledge level, structure of student's learning skills.

Адрес редакции:

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**

E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Технический редактор *О. А. Ефремова*.

Корректор *Е. В. Комиссарова*

Сдано в набор 09.12.2008. Подписано в печать 20.01.2009. Формат 60×88 1/8. Бумага офсетная. Печать офсетная.

Усл. печ. л.10,78. Уч.-изд. л. 12,22. Заказ 40. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати, телерадиовещания и средств массовых коммуникаций.

Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Отпечатано в ООО "Подольская Периодика"

142110, Московская обл., г. Подольск, ул. Кирова, 15