ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

5(189) 2012

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://novtex.ru/IT.**
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 004.031.43

А. Б. Барский, д-р техн. наук, проф.
МИИТ, Москва,
E-mail: arkbarsk@mail.ru

Алгоритмические, архитектурные и структурные методы организации управляющих процессов в виртуальном пространстве средств Grid-системы

Исследуются проблемы перспективного применения Grid-технологий для построения управляющих систем, включая системы реального времени. Предполагается, что супервизор системы формирует поток запросов хост-процессору Grid-центра для организации распределенных вычислений в вычислительной сети. По каждому запросу решается пакет задач за минимальное время. Обоснован простейший алгоритм компенсации для точного оптимального распределения пакета задач между однородными ресурсными процессорами сети. Для минимизации времени сбора результатов вычислений предлагается параллельная структура памяти базового хост-процессора, блоки которой являются адресуемыми объектами. Их связь с процессорами ресурсного кластера позволяет осуществлять одновременный обмен данными.

Ключевые слова: *Grid-технологии, вычислительная сеть, распределенные вычисления, хост-процессор, алгоритм компенсации, сбор результатов*

Введение

Фантастические замыслы питают идею *Grid*-вычислений, на деле призванную перенести достижения Интернета в область выполнения запросов на решение математических задач. Сегодня нереальными представляются рассуждения о том, что, дескать, пока Австралия спит, ее вычислительные средства недогружены и могут использоваться в Москве. Практически мыслящего разработчика таких "критических" систем, как системы управления, которые работают в условиях задания директивных сроков решения задач, а то и в реальном масштабе времени, интересует вопрос: "Насколько *это* возможно, как *это* сделать и во что *это* обойдется?"

Становится все более очевидным, что накладные расходы на организацию *Grid*-вычислений, в особенности потери времени на обмен данными при распределенных сетевых вычислениях, колоссальны и целиком "съедают" досужие идеи. Однако можно легко себе представить привлекательность сложной системы управления в области обороны с вычислительными средствами, расположенными в виртуальном информационном пространстве. Поэтому необходимо постепенно, поступательно двигаться к намеченной цели от разработки новых методов вычислений [1, 2] на базе критериев оптимального распараллеливания до разработки специализированных архитектур вычислительных средств на базе критериев минимизации времени сетевого обмена данными при распределенных вычислениях в реальном времени.

1. Структура управляющего вычислительного процесса

Отметим [1—4], что *Grid*-система не может быть создана столь "обезличенно", как существующая *WWW*, призванная решать только информационные задачи с помощью поисковых систем, расположенных на *web*-узлах. *Grid*-система выполняет специфические функции обслуживания, такие как прием и первичная обработка запросов, выбор и применение методов обслуживания, назначение и управление ресурсами, сбор и возвращение результатов пользователю. Это требует создания специальных Центров *Grid*-технологий, с которыми и взаимодействует пользователь.

Основным операционным звеном *Grid*-центра является мощный Хост-процессор (один или несколько), который ведает приемом и первичной обработкой запросов, назначает кластер ресурсных процессоров, организует, если необходимо, их взаимный обмен данными, распределяет работы, собирает результаты вычислений и возвращает пользователю.

На рис. 1 изображена схема формирования и обработки пакетов задач пользователя.

Супервизор системы управления формирует поток пакетов решаемых задач в соответствии с временным режимом работы [1]. В системах реального времени возможны как одноциклический режим (все задачи решаются с одинаковой частотой), так и многоциклический (задачи связаны с одной из частот решения). Как показано в работе [1], формирование мультипрограммного и мультипроцессорного режимов решения задач на основе приоритетов (для решения в многоциклическом режиме относительный приоритет задач убывает с ростом длительности

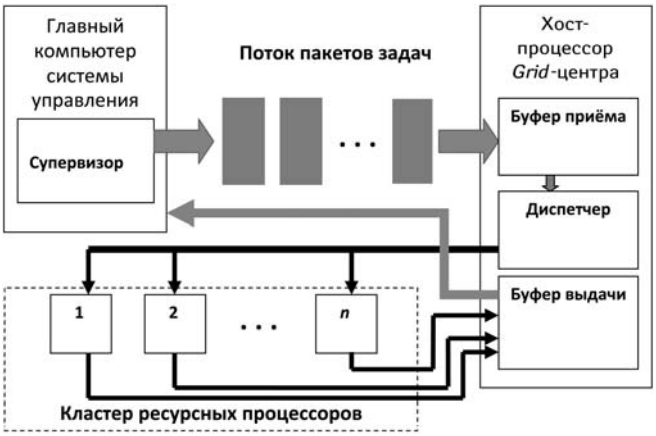


Рис. 1. Структура распределенного управляющего вычислительного процесса

цикла) имеет значительную сложность, требующую оперативного взаимодействия процессоров и применения системы прерывания. Реализация такого режима при распределенных сетевых вычислениях недопустима. Поэтому следует считать, что пакеты задач, решаемых в циклах разной длительности (с разной частотой), назначаются на разные кластеры (подкластеры) ресурсных процессоров. (На рис. 1 отображен одноциклический режим решения пакетов задач.)

В общем случае управляющая система реализует набор функциональных программных модулей — задач, образующих частично упорядоченное множество работ. Для распараллеливания такое множество представляется информационным графом. Алгоритмы распараллеливания информационного графа ориентированы на вычислительные системы с общей оперативной памятью. Они предусматривают значительные возможности оптимизации, но требуют оперативного взаимодействия ресурсных процессоров с Хост-процессором. Эти алгоритмы непригодны для диспетчеров распределенных вычислений.

Следует вернуться к раннему опыту решения задачи распараллеливания [5] на основе ярусно-параллельной формы представления информационного графа.

На рис. 2 показан пример перехода от общего графового описания множества частично упорядоченных работ к ярусно-параллельной форме. Для этого используются максимальные длины путей, ведущих к вершинам.

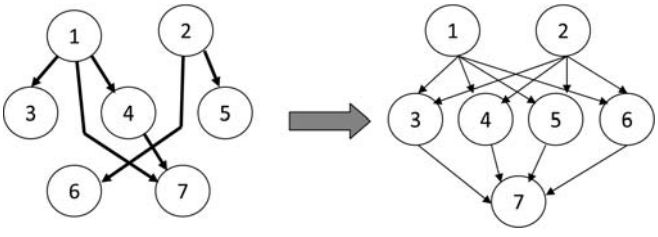


Рис. 2. Переход к ярусно-параллельной форме графа

Ярусы такого графа и отображают пакеты информационно независимых задач, направляемых супервизором в *Grid*-центр. Задачи каждого пакета с помощью *диспетчера* должны распределяться между ресурсными процессорами назначенного *кластера*.

2. Диспетчеризация

Диспетчер Хост-процессора реализует оптимальный план решения каждого пакета задач поступающего потока. Критерием оптимизации является минимум времени решения пакета задач:

$$T_{\text{реш.пак}} \to \min.$$

Пусть m — число задач в пакете; t_i, $i = 1, ..., m$, — время решения i-й задачи пакета; n — число выделенных процессоров для решения задач пакета (размер кластера). Тогда грубая желаемая усредненная оценка минимального времени решения пакета задач достигается при распределении этих задач поровну:

$$T_{\text{ср}} = (1/n) \sum_{i=1} t_i. \qquad (1)$$

Однако дискретность значений времени решения в общем случае не допускает столь усредненного планирования "поровну". Распределение работ между процессорами приводит в общем случае к разному времени T_j их загрузки ($j = 1, ..., n$). Тогда критерий эффективности планирования определяется как

$$|\max_j T_j - T_{\text{ср}}| \to \min. \qquad (2)$$

Для диспетчеризации распараллеливания пакета информационно независимых задач целесообразно использовать алгоритм, описанный в работе [1]. Суть его в следующем.

1. Задачи пакета накапливаются в буфере Хост-процессора и располагаются в очереди *в порядке невозрастания времени решения*.

2. Циклически реализуется решающее правило *"Менее других загруженный процессор берет задачу из головы очереди"*.

Такой диспетчер назовем *компенсирующим* ввиду того, что каждое назначение стремится уравнять текущую занятость процессоров.

Упорядочение задач пакета по невозрастанию времени решения может быть предусмотрено при формировании этих пакетов управляющей системой. Это следует считать целесообразным, ибо ожидание поступления всех задач пакета с последующим их упорядочением может весьма негативно сказаться на производительности *Grid*-вычислений.

Теорема. Распределение между n однородными процессорами пакета m задач, образующих очередь, упорядоченную по невозрастанию времени их выполнения, при последовательном обращении процессоров к очереди по принципу "наименее загруженный процессор берет очередную задачу из головы очереди" (алгоритм компенсации) приводит к оптимальному, в смысле минимума времени выполнения, плану реализации пакета.

Доказательство. Воспользуемся методом математической индукции.

Пусть $m > n$. Легко видеть, что распределение первых n задач выполняется оптимально (наиболее близко к принципу "поровну"), так как первоначальная загрузка процессоров нулевая.

Предположим, что назначение i-й задачи из очереди привело к получению оптимального расписания. Тогда видно, что очередное назначение $(i + 1)$-й задачи максимально компенсирует разницу в загрузке недогруженного по сравнению с другими процессора. Дополненное таким образом расписание выполнения $i + 1$ задач остается оптимальным, так как назначение этой задачи на другой, более загруженный, процессор увеличило бы длину расписания.

При фиксированном n, с ростом числа m распределяемых задач, вследствие их упорядочения по невозрастанию времени выполнения, расписание стремится к такому виду, когда разность загрузки любой пары процессоров не превышает времени решения "самой маленькой" задачи пакета. С учетом дискретности времени решения задач это означает, что полученное расписание является оптимальным в смысле критерия (2), максимально приближенным к распределению задач между процессорами "поровну".

Выполнение критерия (2) подтверждается предельным переходом:

⟨разность загрузки любой пары процессоров⟩ $\to 0$,
при
⟨время выполнения последней задачи в упорядоченной очереди⟩ $\to 0$,
⟨количество задач в пакете, n⟩ $\to \infty$.

Теорема доказана.

Следует еще раз подчеркнуть практическую важность решающего правила, по которому назначение задач проводится в порядке невозрастания времени их выполнения. Ведь разность загрузки любой пары процессоров при таком назначении может только убывать. Это обеспечивает максимально возможное равное распределение конечного множества задач в соответствии с критерием (2).

Если пакет достаточно велик, а время выполнения задач не характеризуется резкими скачками значения времени выполнения, то значения длины расписания выполнения задач отдельными процессорами различаются не более чем на время выполнения самой "маленькой" задачи. Это иллюстрируется рис. 3, где выбранные пакеты распределяются между тремя процессорами. Время решения задачи условно определяется длиной прямоугольника.

На рис. 3, *а* представлен случай, когда время решения задач принадлежит небольшому диапазону изменения, а задач достаточно много. Здесь компенсация эффективна. На рис. 3, *б* при малом числе задач их время решения резко "скачет". Иллюстрируется невозможность достижения компенсации в смысле (1) и (2). Следует учесть, что в примере число процессоров ($n = 3$) задано, и необходимо

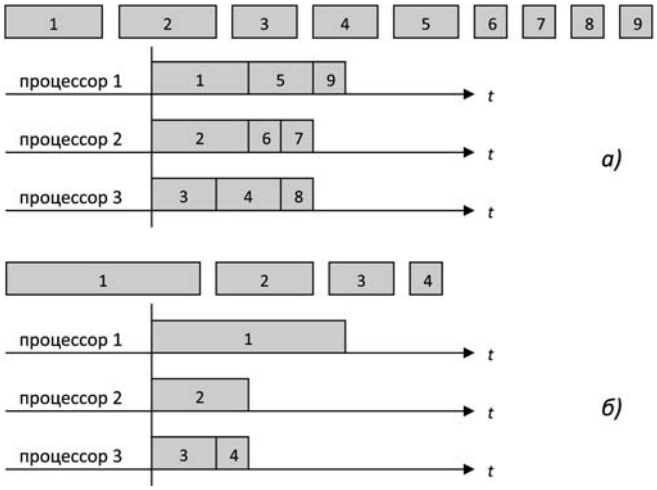


Рис. 3. Распределение задач по алгоритму компенсации

спланировать их работу. Очевидно, что во втором случае n выбрано неправильно. Легко видеть, что при $n = 2$ та же длина оптимального расписания достигается при нулевом пределе (2).

Однако для крайнего упрощения принципов управления распределенным процессом реального времени сознательно откажемся от динамической корректировки уже принятых решений по выбору ресурсных процессоров. Этот выбор мог быть обусловлен некоторыми техническими возможностями и не должен корректироваться динамически, подвергаясь "экономии". Тем более что на самом деле такая экономия приведет к резкому росту "накладных расходов" на управление.

В заключение этого раздела заметим, что специализированные распределенные вычисления, какими являются вычисления в реальном времени, допускают ряд ограничений и соглашений, служащих минимизации "накладных расходов" на управление этими вычислениями (в отличие от общих вопросов оптимального обслуживания случайного потока заданий к Центру *Grid*-технологий, рассмотренных в работе [4]).

Предметом таких ограничений и соглашений, в основном уже отмеченных выше, являются:

- порядок формирования и следования пакетов задач;
- независимое предоставление ресурсов для задач, решаемых с разной частотой (в многоциклическом режиме);
- упорядочение задач внутри пакетов по невозрастанию времени выполнения;
- жесткое количественное закрепление ресурсов.

3. Проблема сбора результатов вычислений на Хост-процессоре

В работе [2] отмечено, что основную схему распределенных *Grid*-вычислений можно представить как "распределение работ Хост-процессором — счет

ресурсными процессорами — возвращение результатов Хост-процессору". Обмен данными здесь определяет основную нагрузку на сетевые технологии.

При раздаче заданий ресурсным процессорам допустим широковещательный обмен, подобный обмену, практикуемому при реализации *SPMD*-технологии [2, 6, 7]. Решая задачу планирования выполнения пакета, Хост-процессор лишь метит задачи номерами тех процессоров, которым они предназначены. Затем он выполняет широковещательный обмен, направляя все задачи с указанными метками всем процессорам выделенного кластера. Каждый процессор решает только "свои" задачи.

Обратный обмен — сбор результатов счета на Хост-процессоре — представляет серьезную проблему. Ведь если не предпринять никаких мер, то каждый ресурсный процессор *последовательно* должен связаться с Хост-процессором.

В работах [6, 7] показано, что уже при малом числе процессоров (порядка 15 в суперкомпьютере Скиф, основанном на сетевой технологии связи процессоров, и столько же в локальной сети) распределенные вычисления, предполагающие сбор данных на головном процессоре, утрачивают эффективность и целесообразность. Время вычислений начинает резко (экспоненциально) расти с увеличением числа используемых процессоров.

В соответствии с проблемой кластеризации ресурсных процессоров для "древовидной" структуры обмена в работе [6] предложен параллельный алгоритм такой кластеризации. Это говорит о том, что кластеризация сама по себе является серьезной задачей, нуждающейся в распараллеливании.

Однако решение задачи кластеризации выделяемых ресурсов следует отнести к универсальному режиму работы *Grid*-центра, обрабатывающему случайный поток заявок. При специализированном применении такая кластеризация может быть выполнена однажды для всех жестко используемых кластеров ресурсных процессоров. (Термин "кластеризация" выше использован в двух смыслах: первое вхождение этого слова традиционно указывает на придание кластеру процессоров специальной "древовидной" структуры связи его подкластеров.) Выполнение алгоритма кластеризации экспериментально позволило автору [6, 7] "отодвинуть" оптимальное число используемых процессоров (в области, где с ростом числа процессоров время вычислений убывает) на значения 80—90.

Однако справедливо считать, что всякое новое применение вычислительных средств предъявляет и новые требования к особенностям их архитектуры.

4. Архитектура и структура оперативной памяти базового Хост-процессора для параллельной сборки результатов счета ресурсных процессоров

Оперативная память (ОП) Хост-процессора (рис. 4) имеет трехуровневую структуру.

Структура нулевого уровня содержит следующие блоки.

1. Блок собственной оперативной памяти, предназначенной только для вычислений этого блока и ограниченно доступной другим блокам нулевого уровня.

2. N блоков — образов памяти ресурсных процессоров. Объем этих блоков может быть значительно меньше объема памяти ресурсного процессора, так как они предназначены лишь для хранения результатов вычислений.

Блоки-образы, хотя и входят в состав Хост-процессора, являются автономными сетевыми объектами, имеющими адреса. Поэтому в соответствии с выполняемыми функциями они, в сущности, являются отдельными процессорами памяти в составе Хост-процессора. На основе сетевой технологии каждый процессор памяти ПП_i связан со своим ресурсным процессором РП_i, $i = 0, \ldots, N-1$. Ресурсные процессоры самостоятельно и независимо (а следовательно, параллельно) дублируют по необходимости информацию в этих процессорах, в том числе результаты вычислений. С помощью процессоров памяти выполняется и широковещательный обмен.

Блоки-образы ввиду сравнительно небольшого объема могут быть выполнены по технологии двухвходовой памяти. Однако даже механическое (релейное) их переключение "центральный процессор — сеть" может быть несоизмеримо более оперативным, чем использование кластеризованного сетевого обмена.

Таким образом, Хост-процессор с помощью процедуры широковещательного обмена способен выдавать задания ресурсным процессорам, а ресурсные процессоры с помощью параллельной связи способны возвращать Хост-процессору результаты вычислений.

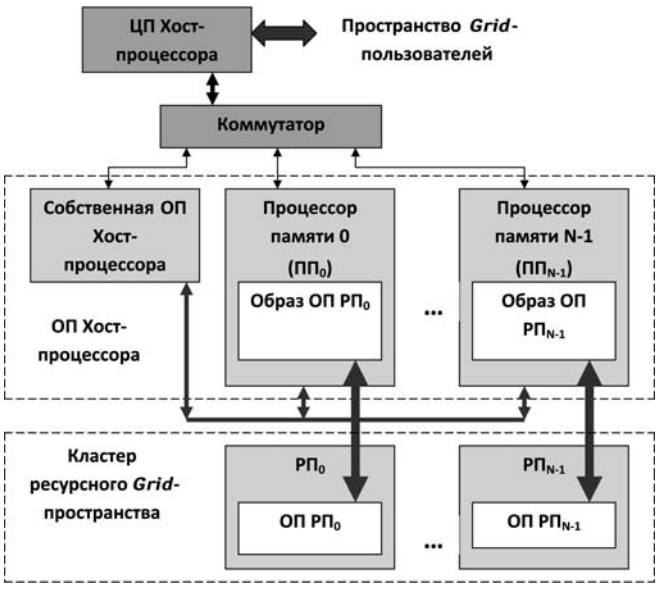


Рис. 4. Структура памяти Хост-процессора

Номер блока нулевого уровня	Номер блока первого уровня	Смещение в модуле	Номер модуля (интерливинг)

Рис. 5. Структура адреса ОП Хост-процессора

Нулевой уровень при указанной структуризации памяти выделен в связи с тем, что в общем случае каждый рассмотренный блок, в свою очередь, реализует традиционную двухуровневую блочно-модульную структуру памяти. Адресация блоков (первый уровень) отражена старшими разрядами адреса, а младшие разряды на втором уровне образуют интерливинг — последовательное отражение модулей внутри блока. Тогда нулевой уровень этой структуры может быть отображен "еще более старшими" разрядами разрядной сетки Хост-процессора, как показано на рис. 5.

Значение N отображает максимальный объем ресурсного кластера. В случае многоциклического режима управления кластер разбивается на подкластеры, закрепленные за циклами. Кластеры ресурсных процессоров свободно формируются и модифицируются в *Grid*-пространстве.

Заключение

Представленный материал следует рассматривать в свете обоснования технического задания на разработку типовых вычислительных средств *Grid*-системы, расширяющей применение *WWW* и опоясывающей земной шар. В результате перспективных исследований методом моделирования применение *Grid*-технологий для решения задач управления в реальном времени, предъявляющих наиболее высокие требования к производительности вычислительных средств, рано или поздно станет успешным.

Список литературы

1. **Барский А. Б.** Параллельные информационные технологии. — М.: ИНТУИТ; БИНОМ. Лаборатория знаний, 2007.
2. **Барский А. Б.** Оптимизационные задачи в основе пакета параллельных прикладных программ и системы информационного обслуживания Центра *GRID*-технологий // Информационные технологии. Приложение. 2010. № 10. 32 с.
3. **Барский А. Б.** Параллельные информационные технологии в основе *GRID*-систем. // Информационные технологии. 2006. № 12. С. 54—60.
4. **Барский А. Б.** Применение логических нейронных сетей для выбора оптимальной стратегии обслуживания потока запросов в системе *GRID*-вычислений // Информационные технологии. 2009. № 1. С. 2—6.
5. **Поспелов Д. А.** Введение в теорию ВС. М.: Советское радио, 1972.
6. **Амиршахи Б.** Кластеризация *GRID*-ресурсов для оптимизации информационного обмена при совместной обработке результатов распределенных вычислений // Информационные технологии. 2011. № 2. С. 22—28.
7. **Амиршахи Б.** *GRID*-технологии решения больших систем линейных уравнений на вычислительной сети и на суперкомпьютере кластерного типа // Информационные технологии. 2011. № 6. С. 17—22.

УДК 004.023

В. М. Захаров, д-р техн. наук, проф.,
С. В. Шалагин, канд. техн. наук, доц.,
e-mail: sshalagin@mail.ru,
ФГБОУ ВПО "Казанский национальный исследовательский технический университет им. А. Н. Туполева — КАИ"

Вычисление нелинейных полиномиальных функций на многопроцессорной системе с программируемой архитектурой

Получены оценки временной и аппаратной сложности вычисления нелинейной полиномиальной функции (НПФ), определенной над полем Галуа вида $GF(2^n)$, на многопроцессорных системах с программируемой архитектурой, элементами которой являются ПЛИС серии Virtex-7. НПФ представима в виде IP-ядер, реализующих определенные функции и выполненных по технологии ПЛИС класса FPGA.

Ключевые слова: *ПЛИС, вычислительная техника, многопроцессорные системы, программируемая архитектура*

Введение

В настоящее время существуют многопроцессорные вычислительные системы с программируемой архитектурой (МВС ПА) различного назначения. Указанные системы выполнены с использованием унифицированных базовых модулей — многопроцессорных реконфигурируемых вычислителей [1—4]. Данные модули реализованы на основе программируемых логических интегральных схем (ПЛИС) класса FPGA [5—7]. МВС ПА позволяют реализовать различные устройства вычислительной техники (УВТ), реконфигурируемые в реальном времени, что находит применение для таких областей, как символьная обработка информации, защита компьютерных сетей, управление в реальном масштабе времени объектами энергетики, летательными и космическими аппаратами и др. [8].

МВС ПА РВС-1Р и РВС-5 реализованы на основе базовых модулей "Триада" и "Алькор", включающих в свой состав 16 ПЛИС семейств Virtex-4 и Virtex-5 соответственно [8]. Современные ПЛИС семейства Virtex-7 позволяют эффективно реализовать булевы функции как мультиплексоры с большим числом входов за счет гибкости ресурсов межсоединений. Гибкость достигается за счет того, что каждая логическая ячейка конфигурируемого ло-

гического блока (КЛБ) имеет автономную связь с переключательной матрицей.

На основе нелинейных полиномиальных функций (НПФ), определяемых над полем Галуа, реализуем широкий класс УВТ [9—11], позволяющих вычислить: произвольную дискретную детерминированную нелинейную функцию, значения дискретных стохастических последовательностей класса марковских и их функций [12, 13] и теоретико-полиномиальных преобразований [14], в частности, дискретных преобразований Фурье, Хартли и цифровых фильтров, рекурсивных и нерекурсивных. При реализации указанных устройств, описываемых НПФ, в качестве IP-ядер (англ. *Intellectual Property*) предложено использовать системы булевых функций, каждая из которых позволяет вычислить заданную НПФ над полем Галуа вида $GF(2)$ [15].

Цель настоящей работы — оценка для достаточно сложных УВТ, позволяющих вычислить НПФ над полем Галуа и реализуемых на основе МВС ПА по технологии ПЛИС семейства Virtex-7 как времени задержки их функционирования, так и числа кристаллов ПЛИС, требуемых для их реализации.

Структурные схемы и сложность вычисления НПФ над $GF(2^n)$

В работах [15—17] предложен и развит метод синтеза НПФ $\psi(x_1, ..., x_m)$ как суммы элементарных полиномов над $GF(2^n)$ вида

$$f(q_1, ..., q_m) = \sum_{i_1 = 0}^{w} ... \sum_{i_m = 0}^{w} a_{i_1 ... i_m} q_1^{i_1} ... q_m^{i_m},$$

$$w = 2^n - 1,\ a_{i_1 ... i_m},\ q_1, ..., q_m \in GF(2^n). \qquad (1)$$

Определены теоретические оценки сложности реализации НПФ вида (1) на основе параллельной, систолической и последовательностной структурных схем (структур) [10, 18] в базисе элементарных схем — вычислителей, реализующих произвольную булеву функцию от двух переменных (БФ(2)) [19].

Определим оценки сложности вышеуказанных структур, реализованных в архитектуре ПЛИС класса FPGA и вычисляющих НПФ вида (1) над $GF(2^n)$. Указанные оценки получены исходя из того, что коэффициенты в формуле (1) $a_{i_1 ... i_m}$, $i_1 = \overline{0, w}$, ..., $i_m = \overline{0, w}$, $w = 2^n - 1$, есть постоянные величины. Примем в качестве меры временной сложности задержку τ функционирования элементарных схем. Мерой аппаратной сложности служит БФ(2). Следует отметить, что аппаратная сложность БФ(z), $z > 2$, определена как $(z - 1)$ БФ(2).

Для вычисления (1) на основе *параллельной структуры* (рис. 1) требуется вычислить в сумме $n(2^{mn} - 1)$ БФ (mn) и $n(2^{mn} - 2)$ БФ(2). Верхняя оценка временной сложности для указанной структурной схемы равна $(\log_2(mn) + mn)\tau$.

Систолическая структура использует схему Горнера для вычисления НПФ вида (1):

$$f(q_1, ..., q_m) = \sum_{i_1 = 0}^{w} f_{i_1}^{(1)}(q_2 ... q_m) q_1^{i_1} =$$

$$= (...(f_w^{(1)}(q_2 ... q_m) q_1 + f_{w-1}^{(1)}(q_2 ... q_m))) q_1 + ... +$$

$$+ f_0^{(1)}(q_2 ... q_m). \qquad (2)$$

В случае, когда для НПФ вида (1) определены переменные $q_1, ..., q_{k-1}$, то

$$f^{(k-1)}(q_k, ..., q_m) = \sum_{i_k = 0}^{w} f_{i_1 ... i_k}^{(k)}(q_{k+1} ... q_m) q_k^{i_k} =$$

$$= (...(f_{i_1 ... i_{k-1} w}^{(k)} q_k + f_{i_1 ... i_{k-1}(w-1)}^{(k)})) q_k + f_{i_1 ... i_{k-1} 0}^{(k)} =$$

$$= (...(f_{i_1 ... i_{k-1} w}^{(k)}(q_{k+1} ... q_m) q_k +$$

$$+ f_{i_1 ... i_{k-1}(w-1)}^{(k)}(q_{k+1} ... q_m))) q_k +$$

$$+ f_{i_1 ... i_{k-1} 0}^{(k)}(q_{k+1} ... q_m),\ 1 < k \leqslant m. \qquad (3)$$

Для вычислений, выполняемых согласно (3), требуется выполнить операцию над $GF(2^n)$ вида

$$g^{(j+1)} = g^{(j)} q_k + b_{w-j}, \qquad (4)$$

где $g^{(1)} = f_{i_1 ... i_{(k-1)} w}^{(k)}$, $f_w^{(m)} = a_{i_1 ... i_{(m-1)} w}$, $b_{w-j} = f_{i_1 ... i_{(k-1)}(w-j)}^{(k)}$, $f_{w-j}^{(m)} = a_{i_1 ... i_{(m-1)}(w-j)}$, $i_k = \overline{0, w}$, $j = \overline{1, w}$, $k = \overline{1, m}$, $f^{(0)} = f(q_1, ..., q_m)$, $w = 2^n - 1$.

Систолическая структура, позволяющая вычислить НПФ вида (1) на основе выражений (2), (3) и (4), приведена на рис. 2 и реализуема $wn2^{n(m-1)}$ БФ $(2n + 1)$, $w = 2^n - 1$. Оценка временной сложности для указанной структурной схемы равна $(mw\log_2(2n + 1))\tau$, $w = 2^n - 1$.

Последовательностная структура, вычисляющая значение НПФ вида (1), представленной на базе схемы Горнера в виде (2), реализуема единственным блоком, который вычисляет выражения вида (4) согласно (2) и (3) одной БФ$(2n + 1)$. Оценка

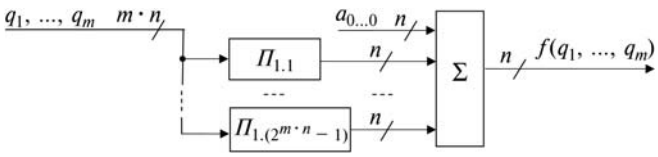


Рис. 1. Параллельная структура, позволяющая вычислить НПФ вида (1)

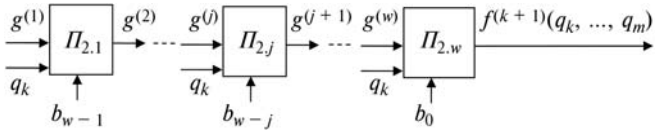


Рис. 2. Систолическая структура, позволяющая вычислить НПФ вида (3)

временной сложности для данной структуры равна $(w \cdot 2^{n(m-1)}\log_2(2n+1))\tau$, $w = 2^n - 1$.

Схемы на рис. 1 и 2 определены как графы алгоритма вычислений, которые определяют функционирование УВТ для вычисления НПФ вида (1) [2, 3].

Метод оценки сложности реализации УВТ в архитектуре ПЛИС семейства Virtex-7

Рассмотрим реализацию произвольной булевой функции от z переменных — БФ(z), на одном кристалле ПЛИС семейства Virtex-7 как IP-ядра, по аналогии с [20]. Один логический элемент (англ. *Look-Up Table, LUT*) данного семейства позволяет реализовать БФ(l), $l \leqslant 6$. Один КЛБ включает в себя две логические ячейки (англ. *Slices*), каждая из которых содержит по четыре логических элемента. Одна логическая ячейка позволяет реализовать одну, а КЛБ — две БФ(l), $l \leqslant 8$ [5, стр. 11—17].

В результате определено число КЛБ, позволяющих реализовать БФ(z) согласно формуле

$$N^{(z)} = N_1^{(z)} + N_2^{(z)}, \qquad (5)$$

где $N_1^{(z)}$ — число КЛБ, требуемых для вычисления БФ(8), который реализует БФ(z) при заданных значениях ($z-8$) переменных; $N_2^{(z)}$ — число КЛБ, используемых для мультиплексирования v значений, вычисляемых БФ(8), т. е. сконфигурированных как мультиплексоры на v входов и один выход (MUX(v:1)), $v = 4, 8, 16$ [5, стр. 16]:

$$N_1^{(z)} = \begin{cases} 2^{z-9}: & z > 8; \\ 1/2: & z = 8; \\ 1/4: & z = 7; \\ 1/8: & z \leqslant 6; \end{cases}$$

$$N_2^{(z)} = \begin{cases} 30^{-1}(16^{(d-1)} - 1) + N_2^{(t)}: & z > 12; \\ 1/2: & z = 12; \\ 1/4: & z = 11; \\ 1/8: & z = \overline{9, 10}; \\ 0: & z \leqslant 8; \end{cases}$$

$$d = \left]\frac{z-8}{4}\right[, \; t = (z-8)\mathrm{mod}4 + 9.$$

Согласно [6, стр. 29] для ПЛИС семейства Virtex-7 время задержки функционирования логического элемента, реализующего БФ(l), $l \leqslant 6$, равно $T_{\mathrm{ILO\_1}}$, время задержки функционирования программируемых элементов внутри КЛБ, реализующих БФ(7), — $T_{\mathrm{ILO\_2}}$, а данное время для элементов внутри КЛБ, реализующих БФ(8), — $T_{\mathrm{ILO\_3}}$. При реализации БФ(z) на основе КЛБ ПЛИС семейства Virtex-7 время задержки вычисления ее значения определено согласно формуле

$$T^{(z)} = T_1^{(z)} + T_2^{(z)} + T_{\mathrm{MC}} + T_{\mathrm{IOB}}, \qquad (6)$$

где $T_1^{(z)}$ — задержка КЛБ, требуемых для вычисления БФ(k), который реализует БФ(z) при заданных значениях ($z-8$) переменных; $T_2^{(z)}$ — задержка КЛБ, используемых как MUX(v:1), $v = 4, 8, 16$, для мультиплексирования значений, вычисляемых БФ(8): T_{MC} и T_{IOB} — времена задержек межсоединений и блоков ввода-вывода (БВВ) ПЛИС Virtex-7:

$$T_1^{(z)} = \begin{cases} T_{\mathrm{ILO\_3}}: & z \geqslant 8; \\ T_{\mathrm{ILO\_2}}: & z = 7; \\ T_{\mathrm{ILO\_1}}: & z \leqslant 6; \end{cases}$$

$$T_2^{(z)} = \begin{cases} (d-1)T_{\mathrm{ILO\_3}} + T_2^{(t)}: & z > 12; \\ T_{\mathrm{ILO\_3}}: & z = 12; \\ T_{\mathrm{ILO\_2}}: & z = 11; \\ T_{\mathrm{ILO\_1}}: & z = \overline{9, 10}; \\ 0: & z \leqslant 8, \end{cases}$$

d и t определены аналогично (5).

При этом $T_{\mathrm{IOB}} = T_{\mathrm{IOPI}} + T_{\mathrm{IOOP}}$, где T_{IOPI} — время задержки прохождения сигнала от вывода ПЛИС до выхода БВВ внутри кристалла (БВВ сконфигурирован как вход); T_{IOOP} — время задержки прохождения сигнала от входа БВВ внутри кристалла до вывода ПЛИС (БВВ сконфигурирован как выход).

Оценки $T_{\mathrm{ILO\_1}}$, $T_{\mathrm{ILO\_2}}$, $T_{\mathrm{ILO\_3}}$, T_{IOPI} и T_{IOOP} (для БВВ стандарта LVCMOS18, Fast, 16 мА, ПЛИС с напряжением питания 1,8 В) составляют 0,05, 0,17, 0,25, 0,5 и 1,37 нс соответственно [6, стр. 19, 29].

Определим вклад величины T_{MC} в общее время задержки функционирования $T^{(z)}$ как переменную γ. Кроме того, если внутри ПЛИС доступно для конфигурирования Q_{LE} КЛБ и Q_{IOB} БВВ, то оценка величины $T^{(z)}$, определенная согласно (6), составляет

$$T^{(z)} \cong \gamma^{-1}(T_1^{(z)} + T_2^{(z)}) + T_{\mathrm{IOB}}. \qquad (7)$$

Оценка аппаратных затрат при реализации произвольной БФ(z) на МВС ПА по числу кристаллов ПЛИС семейства Virtex-7 будет равна

$$\mu = \max(N^{(z)}/(kQ_{\mathrm{LE}}), (z+1)/Q_{\mathrm{IOB}}), \qquad (8)$$

где k — коэффициент задействования КЛБ ПЛИС, $0 < k < 1$, определяемый проектировщиком. Эмпирическим путем установлено, что для оптимального (с точки зрения обеспечения минимального времени задержки функционирования) размещения логических ячеек внутри КЛБ, используемых для реализации УВТ на ПЛИС, $k \in [0{,}5;\ 0{,}7]$ [21, 22].

Предложен метод получения оценок сложности реализации УВТ в архитектуре ПЛИС семейства Virtex-7, включающий три этапа:

1) определение числа БФ, на основе которых проводится вычисление значения НПФ вида (1) согласно заданной структурной схеме;

2) вычисление для одной и более БФ(z) оценок аппаратных затрат по числу кристаллов ПЛИС заданного семейства и времени задержки функционирования согласно (5) и (6) соответственно;

3) вычисление оценок, определенных на этапе 2, для заданной структурной схемы.

Оценка сложности реализации полиномиальной функции над $GF(2^n)$ на ПЛИС Virtex-7

Рассмотрим реализацию БФ(z) как IP-ядра на ПЛИС Virtex-7 типа XC7VX1140T-3FLG1930, которая включает в свой состав 178 тыс. логических ячеек или 89 тыс. КЛБ ($Q_{\text{LE}} = 8{,}9 \cdot 10^4$), что соответствует 1 139 200 эквивалентным вентилям, а также 1100 БВВ ($Q_{\text{IOB}} = 1{,}1 \cdot 10^3$), доступных для программирования [7, стр. 4].

Оценки аппаратных затрат УВТ, вычисляющего БФ(z), определены согласно формуле (5) как функция от z и приведены на рис. 3. Величина z превышает Q_{IOB} на два порядка, поэтому оценка $\mu(z)$ числа ПЛИС семейства Virtex-7, требуемых для вычисления БФ(z), согласно (8) определяема величиной Q_{LE}. Значения $\mu(z)$ и $]\mu(z)[$ как функции от z для $k = 0{,}7$ приведены на рис. 4.

Согласно данным, приведенным на рис. 4, БФ(z) реализуема как IP-ядро на одном кристалле только для $z \leqslant 24$. Оценки времени задержки функционирования при реализации БФ(z), $z \leqslant 24$, вычисленные согласно (7) при $\gamma = 0{,}7$, приведены на рис. 5.

Рассмотрим реализацию УВТ на ПЛИС XC7VX1140T-3FLG1930, позволяющего получить значение НПФ вида (1) над $GF(2^n)$ от m переменных, где n и m — целые положительные числа. При реализации параллельной, систолической и последовательностной структур фактором ограничения служит максимальное значение величины z — числа переменных БФ(z), реализуемой как IP-ядро на одном кристалле ПЛИС.

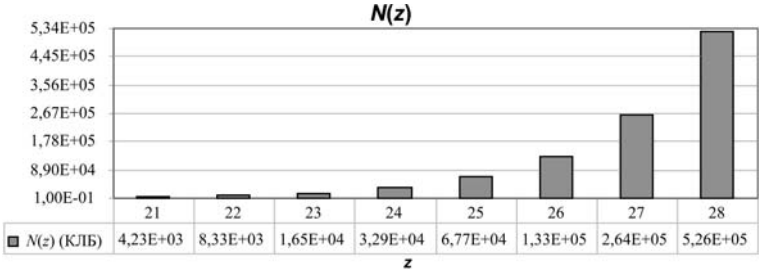


Рис. 3. Оценки аппаратных затрат УВТ, реализующего БФ(z)

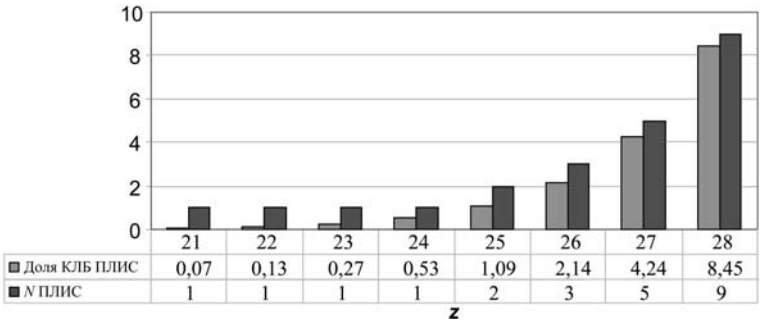


Рис. 4. Оценки доли КЛБ и числа кристаллов ПЛИС, требуемых для синтеза УВТ, реализующего БФ(z)

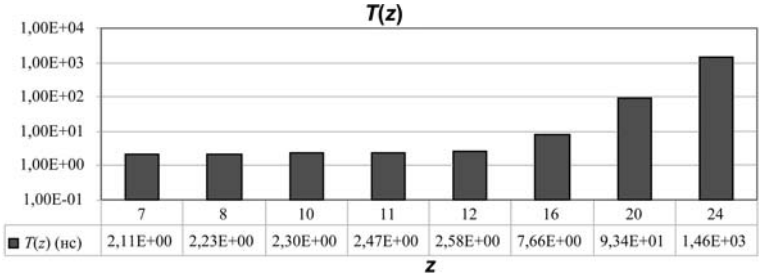


Рис. 5. Оценка времени задержки функционирования УВТ, реализующего БФ(z)

Для параллельной структуры $mn \leqslant 24$. При выполнении данного условия число кристаллов указанной ПЛИС, требуемых для реализации $\Pi_{1.i}$, $i = \overline{1, 2^{mn} - 1}$ (см. рис. 1), не превышает значения $(2^{mn} - 1)$. Число кристаллов, требуемых для реализации блока Σ, вычислим следующим образом. Один логический элемент ПЛИС семейства Virtex-7 позволяет реализовать 16 БФ(2), так как один генератор функции позволяет реализовать две БФ(l), $l \leqslant 5$, а КЛБ содержит две логические ячейки, включающие по четыре логических элемента каждая. Кроме того, для реализации поразрядной суммы по модулю два от y переменных требуется $(y + 1)$ БВВ ПЛИС. В результате аппаратная сложность УВТ, заданного на основе параллельной структуры, определяема по числу кристаллов ПЛИС согласно формуле

$$\mu_{\text{пар}} \leqslant \mu_1 + \mu_\Sigma,$$

где

$$\mu_1 =]n(2^{mn} - 1)\mu(mn)[;$$

$$\mu_\Sigma = \max(]n(2^{mn} - 2)/(16kQ_{\text{LE}})[,]n2^{mn}/Q_{\text{IOB}}[).$$

Оценка времени задержки функционирования указанного УВТ равна сумме задержек вычисления одной БФ(mn) и $]\log(\mu_\Sigma/n)[$ БФ$(n(2^{mn}-1)/(16k\mu_\Sigma Q_{\text{LE}}))$ — $- T^{(mn)} +]\log(\mu_\Sigma/n)[\, T^{(1\,+]n(2^{mn}-1)/(16k\mu_\Sigma Q_{\text{LE}})[)}$, $mn \leqslant 24$. Например, для $n = 16$, $m = 1$ $\mu_1 = 2165$, $\mu_\Sigma = 954$ (определена на основе Q_{IOB}), а $\mu_{\text{пар}} \leqslant 3119$. Оценка времени задержки функционирования составляет 19,3 нс при $\gamma = 0{,}7$.

Для систолической структуры $2n + 1 \leqslant 24$, а следовательно, $n \leqslant 11$. Аппаратная сложность (число кристаллов ПЛИС, которое обозначим как μ) и время задержки функционирования (которое обозначим как T) IP-ядра, реализующего БФ($2n + 1$), не зависит от величины m. Но данная величина влияет на число кристаллов ПЛИС, реализующих УВТ для НПФ вида (1). В результате аппаратная сложность УВТ, заданного на основе систолической структуры, определяема согласно формуле

$$\mu_{\text{сист}} \leqslant wn2^{n(m-1)}/\mu_2,$$

где $\mu_2 = \max([kQ_{\text{LE}}/N^{(n)}], [Q_{\text{IOB}}/(2n + 2)])$, $w = 2^n - 1$.

Например, для $n = 8$ один кристалл позволяет реализовать 61 БФ(17), что определено по числу доступных пользователю ББВ. Кроме того, для $m = 2$ $\mu_{сист} = 8562$. Оценка T указанного УВТ не превышает суммы задержек вычисления mw БФ($2n + 1$), $w = 2^n - 1$. Например, для $n = 8$ и $m = 2$ оценка $T = 46{,}6$ мкс (вычисленная согласно (7) для $\gamma = 0{,}7$). В случае если значение БФ(17), вычисленное внутри каждого кристалла ПЛИС, буферизируется (при использовании D-триггеров внутри логической ячейки), то в случае реализации конвейерной обработки данных среднее время вычисления одного значения НПФ вида (1) составляет примерно 93,4 нс.

Последовательностная структура реализуема на одном кристалле ПЛИС XC7VX1140T-3FLG1930 при условии, что $2n + 1 \leqslant 24$, а следовательно, $n \leqslant 11$. Оценка $T = w2^{n(m-1)}T^{(2n+1)}$, $w = 2n - 1$, для $n \leqslant 11$. Например, для $n = 8$ один кристалл данной ПЛИС позволяет реализовать одну БФ(17) независимо от размерности аргументов НПФ вида (1). Согласно (7) для $\gamma = 0{,}7$ $T^{(17)} = 93{,}4$ нс, поэтому $T = w2^{n(m-1)}T^{(17)} \approx 23{,}8 \cdot 2^{8(m-1)}$ мкс.

Основные выводы

Вычисление НПФ вида (1) на ПЛИС семейства Virtex-7 определено на основе параллельной, систолической и последовательностной структурных схем. Каждая БФ(z), $z \leqslant 24$, реализуема как IP-ядро на кристалле ПЛИС типа XC7VX1140T-3FLG1930, содержащей заданное число доступных пользователю КЛБ и ББВ. В данном случае имеют место следующие ограничения на такие параметры НПФ вида (1), как число ее аргументов m и размерность n двоичных векторов, задающих ее аргументы и значение.

Для параллельной структуры $m \cdot n \leqslant 24$. Величины μ (число кристаллов ПЛИС XC7VX1140T-3FLG1930) и T (время задержки функционирования) для нее имеют порядки $O_\mu(n \cdot 2^{mn})$ и $O_T(m \cdot n)$ соответственно. Для систолической структуры $n \leqslant 11$, а порядки величин μ и T для нее составляют $O_\mu(n \cdot 2^{mn})$ и $O_T(m \cdot 2^n)$ соответственно. Последовательностная структура имеет ограничение: $n \leqslant 11$; порядок величины T для нее равен $O_T(2^{mn})$.

В результате рост значения n для параллельной структуры способствует экспоненциальному росту порядка величины μ и линейному росту порядка значения T; для систолической структуры — экспоненциальному росту μ и T; для последовательностной — экспоненциальному росту T. В том случае, когда возрастает величина m, для параллельной и систолической структурных схем значение μ растет экспоненциально, а T — линейно; для последовательностной структуры T возрастает экспоненциально.

При этом для систолической и последовательностной структур число аргументов НПФ вида (1) — m не ограничивается программируемыми ресурсами ПЛИС, а только числом кристаллов ПЛИС, организованных в МВС ПА.

Например, при моделировании конечных простых однородных цепей Маркова (ЦМ) НПФ вида (1) от двух переменных, степень данной НПФ — 2^n, связана с размером h стохастической матрицы, определяющей ЦМ, соотношением вида [9]

$$2^n \geqslant h^2 - h + 1. \qquad (9)$$

Параллельная структура позволяет реализовать НПФ (1) для $m = 2$ на ПЛИС XC7VX1140T-3FLG1930 при $n \leqslant 12$, а систолическая и последовательностная — при $n \leqslant 11$. Согласно (9) в первом случае $h \leqslant 64$, а во втором — $h \leqslant 45$.

Заключение

Предложен и теоретически обоснован метод предварительной оценки числа кристаллов ПЛИС семейства Virtex-7 и времени задержки функционирования УВТ, описываемого НПФ вида (1) и реализуемого на МВС ПА. Расчет указанных оценок проводится на основе данных, полученных путем анализа сложностных оценок IP-ядер, на базе которых определено данное устройство. Методика открывает возможности для предварительного анализа характеристик реконфигурируемых ресурсов современных ПЛИС, требуемых для синтеза широкого класса УВТ на МВС ПА.

Показаны возможности современных ПЛИС для синтеза УВТ, реализующих НПФ над полем Галуа.

Список источников

1. **Каляев А. В., Левин И. И.** Модульно-наращиваемые многопроцессорные системы со структурно процедурной обработкой вычислений. М.: Янус-К, 2003. 380 с.
2. **Каляев И. А., Левин И. И., Семерников Е. А., Шмойлов В. И.** Реконфигурируемые мультиконвейерные вычислительные структуры. 2-е изд. Ростов н/Д: Изд-во ЮНЦ РАН, 2009. 344 с.
3. **Каляев И. А., Левин И. И.** Реконфигурируемые мультиконвейерные вычислительные системы для решения потоковых задач обработки информации и управления // Параллельные вычисления и задачи управления (PACO'10): пленарные доклады 5-й междунар. конф. М.: ИПУ РАН, 2010. С. 23—37.
4. **Дордопуло А. И., Каляев И. А., Левин И. И., Семерников Е. А.** Высокопроизводительные реконфигурируемые вычислительные системы // Суперкомпьютеры. 2010. № 3 (3). С. 44—48.
5. **7 Series** FPGAs Configurable Logic Block. URL: http://www.xilinx.com/support/documentation/user_guides/ ug474_7Series_CLB.pdf.
6. **Virtex-7 FPGAs** Data Sheet: DC and Switching Characteristics. URL: http://www.xilinx.com/support/documentation/data_sheets/ds183_ Virtex_7_Data_Sheet.pdf.
7. **7 Series FPGAs** Overview. URL: http://www.xilinx.com/support/ documentation/ data_sheets/ds180_7Series_Overview.pdf.
8. **Компьютеры** с реконфигурируемой архитектурой. Лаборатория Параллельных информационных технологий НИВЦ МГУ. 2011. URL: fpga.parallel.ru/areas.html. Яз. рус.
9. **Захаров В. М., Нурутдинов Ш. Р.** Полиномиальные модели вероятностных автоматов и функций цепей Маркова над полем $GF(2^n)$ / Методы моделирования: сб. тр. Казанского городского семинара. Вып. 2. Казань: Фэн (Наука). 2004. С. 51—73.
10. **Шалагин С. В.** Представимость дискретных детерминированных нелинейных функций на основе многочленов над полем Галуа в базисе ПЛИС класса FPGA. Казань: Изд-во КГТУ им. А. Н. Туполева, 2010. 184 с.

11. **Захаров В. М., Эминов Б. Ф.** Методы и алгоритмы построения и анализа полиномиальных функций над конечным полем на основе стохастических матриц. Saarbrucken Germany: LAP — Lambert Academic Publishing GembH & Co KG, 2011. 168 с.
12. **Бухараев Р. Г., Захаров В. М.** Управляемые генераторы случайных кодов. Казань: Изд-во Казанского гос. ун-та, 1978. 160 с.
13. **Бухараев Р. Г.** Основы теории вероятностных автоматов. М.: Наука, 1985. 287 с.
14. **Крот А. М.** Дискретные модели динамических систем на основе полиномиальной алгебры. Минск: Навука і тэхніка, 1990. 312 с.
15. **Шалагин С. В.** О представлении нелинейных полиномов над конечным полем распределенной вычислительной системой // Нелинейный мир. 2009. № 5. С. 376—379.
16. **Лидл Р., Лидл Р., Нидеррайтер Г.** Конечные поля. В 2 т. М.: Мир, 1988.
17. **Нурутдинов Ш. Р.** Основы теории полиномиальных моделей автоматных преобразований над полем Галуа. Казань: Изд-во КГУ им. Н. А. Туполева, 2005. 156 с.
18. **Захаров В. М., Шалагин С. В.** Параллельные марковские модели над полем $GF(2^n)$ // Высокопроизводительные параллельные вычисления на кластерных системах: сб. тр. Восьмой межд. конф. Казань: изд-во КГТУ им. А. Н. Туполева, 2008. С. 155—160.
19. **Ахо А., Хопкрофт Дж., Ульман Дж.** Построение и анализ вычислительных алгоритмов: Пер. с англ. Слисенко А. О. / Под ред. Ю. В. Матиясевича. М.: Мир, 1979. 536 с.
20. **Шалагин С. В.** Реализация устройств вычислительной техники на многопроцессорных системах с программируемой архитектурой // Вестник МарГТУ. 2011. № 1 (11). С. 38—46.
21. **Норенков И. П.** ECAD: автоматизация проектирования в электронике // Приложение. Информационные технологии. 2001. № 8. 24 с.
22. **Пономарев В. И., Шабалин Л. А.** Проектирование реконфигурируемых устройств обработки цифровых потоков данных // Информационные технологии. 1996. № 5. С. 24—28.

УДК 004.728.4

А. О. Крючков, студент магистратуры,
В. А. Крищенко, канд. техн. наук, доц.,
e-mail: kva@bmstu.ru,
Московский государственный
технический университет им. Н. Э. Баумана

Отслеживание жизненного цикла IP-пакетов

Разработан метод, позволяющий построить весь жизненный цикл IP-пакета от его создания до прихода к получателю. Метод позволяет отслеживать фрагментацию и дефрагментацию, многоадресную маршрутизацию, туннелирование и преобразование сетевых адресов для протоколов IPv4 и IPv6. Описывается программная реализация метода для компьютерных сетей с узлами на основе ОС Linux.

Ключевые слова: *анализ сетевого трафика, протокол IP, IP-пакет*

Введение

При изучении работы компьютерной сети, использующей протокол IP [1, 2], и выявлении неисправностей интересен полный жизненный цикл IP-пакета от момента его возникновения до момента уничтожения. Решить эту задачу мог бы такой метод сетевого мониторинга, который бы восстанавливал жизненный цикл пакета на основе фиксации следующих событий: создание нового пакета, прохождение через маршрутизатор, фрагментация и дефрагментация, выполнение многоадресной маршрутизации, туннелирование, преобразование сетевых адресов, доставка пакета, отбрасывание пакета.

Для фиксации сетевой активности на отдельном узле сети используются анализаторы трафика, называемые *снифферами*, такие как Tcpdump [3]. Их возможности ограничены фиксацией прохождения пакета через сетевой интерфейс, поэтому они не дают информации о событиях, случившихся с пакетом в ядре ОС.

Утилита Traceroute, использующая служебный протокол ICMP, может быть применена для решения задачи определения маршрута специально создаваемых IP-пакетов. Основанный на протоколе ICMP механизм нельзя использовать для пакетов обычного TCP-соединения.

Параметр Record Route протокола IP позволяет маршрутизаторам записывать свои IP-адреса в заголовок пакета в процессе его передачи [1]. Этот метод дает возможность получить маршрут при отсутствии фрагментации пакета, поскольку данный параметр не копируется при фрагментации.

Можно сделать вывод, что существующие подходы не позволяют получить информацию о возникновении представляющих интерес событий жизненного цикла IP-пакета.

В настоящий момент существует также учебное программное обеспечение (ПО), такое как Cisco Packet Tracer, которое решает поставленную задачу для модели сети передачи данных. При таком подходе встает вопрос адекватности используемой модели существующим реализациям стека TCP/IP в современных ОС. Таким образом, задача создания и реализации метода отслеживания жизненного цикла IP-пакетов, пригодного для использования в реальных ОС, является достаточно актуальной.

В настоящей статье описывается создание такого метода и его программной реализации для сети, узлы которой реализованы на основе ОС Linux. Предполагается, что необходимое программное обеспечение может быть установлено на каждом из узлов анализируемой сети.

Метод отслеживания жизненного цикла IP-пакета

Для создания метода нужно классифицировать события жизненного цикла, формализовать описание жизненного цикла, разработать алгоритм построения этого описания.

Жизненный цикл IP-пакета. На рис. 1 приведена классификация событий, составляющих жизненный цикл отдельного IP-пакета.

Рис. 1. Классификация событий жизненного цикла IP-пакетов

Структуру данных для хранения информации о жизненном цикле одного IP-пакета будем называть *журналом*. Журнал состоит из *блоков*, которые подразделяются на блоки событий и служебные блоки. *Блоки событий* описывают движение IP-пакета в рамках одного узла сети и содержат список событий. *Служебные блоки* могут быть трех видов:

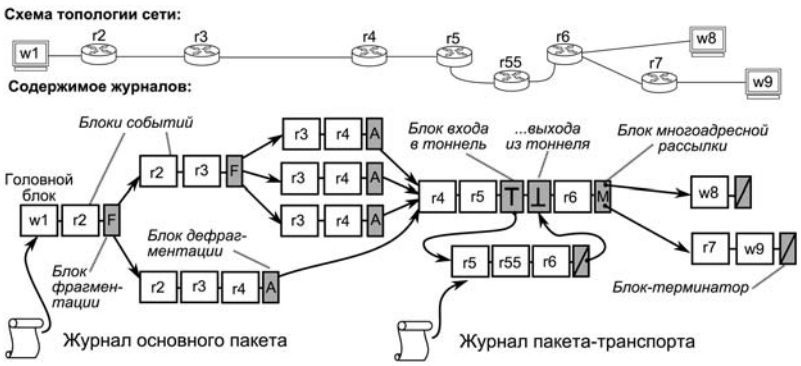


Рис. 2. Пример журнала

1) блоки, связывающие несколько пакетов отношениями "один — ко-многим" — блоки фрагментации, дефрагментации, многоадресной рассылки;

2) блоки входа и выхода из тоннеля, содержащие ссылки на журнал пакета, послуживший транспортом в тоннеле;

3) блоки-терминаторы, завершающие журнал и связанные с событиями завершения жизни пакета.

В журнале блоки связаны между собой дугами, отражающими логическую последовательность событий. Таким образом, журнал представляет собой ациклический направленный граф, где вершинами являются блоки, относящиеся к одному исходному IP-пакету (рис. 2).

Блок событий, соответствующий только что сформированному IP-пакету, называется *головным блоком* и является единственным истоком — вершиной графа, в которую не входит ни одна дуга. Граф может содержать произвольное число стоков — блоков-терминаторов.

В примере на рис. 2 исходный пакет разбивается на два фрагмента на узле **r2**. Затем первый фрагмент разбивается еще на три на узле **r3**. Далее фрагменты собираются на узле **r4**, собранный пакет проходит через тоннель, после чего маршрутизатор **r6** выполняет многоадресную маршрутизацию, передавая две копии пакета узлам **w8** и **r7**.

Этапы метода отслеживания IP-пакетов. В разрабатываемом методе журналы движения пакетов формируются в несколько этапов (рис. 3):

- перехват событий, возникающих в сетевой подсистеме ядра ОС;
- предварительная обработка перехваченных событий и создание блоков событий для сборки журнала;
- соединение блоков в готовые журналы IP-пакетов.

Алгоритм сборки журналов движения пакетов. Для сборки журналов нужно создать алгоритм, соединяющий полученные события в готовый журнал. Сущности, с которыми будет оперировать алгоритм, и отношения между ними показаны на рис. 4.

Эти отношения реализуются в следующих структурах данных (словарях), используемых алгоритмом построения журнала:

1) *сегменты*: id сегмента сети → набор пакетов, находящихся в этом сегменте;

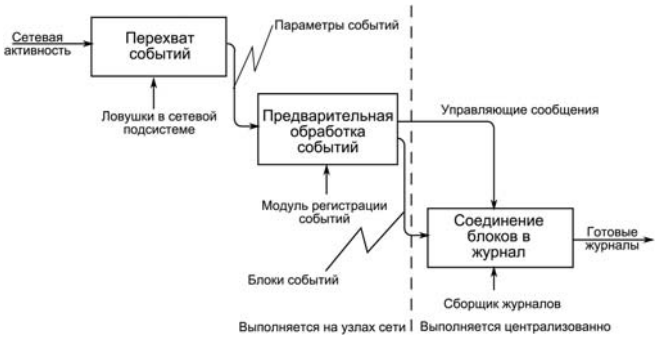


Рис. 3. Функциональная схема разрабатываемого метода

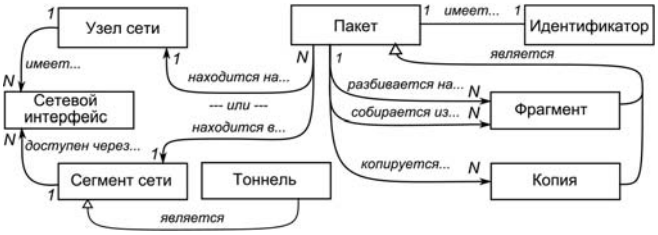


Рис. 4. Инфологическая модель с точки зрения сборщика журналов

2) *узлы*: символьное имя узла сети → структура с полями:

- *интерфейсы*: имя интерфейса → id сегмента сети;
- *пакеты*: id пакета → блок (все пакеты, находящиеся на узле);
- *фрагменты*: id фрагмента → блок целого пакета;
- *копии*: id копии → блок пакета-оригинала;
- *склейка*: id целого пакета → список блоков фрагментов.

Следующий псевдокод описывает действия, выполняемые сборщиком журналов при получении сообщения от узла сети:

```
[от узла H получено сообщение M]
switch (содержание M):
    case уведомление о поступлении пакета id из интерфейса I:
        сегмент сети S : = сегменты[узлы[H].интерфейсы[I]]
        блок : = извлечь S[id]
        узлы[H].пакеты[id] : = блок
    case уведомление о завершении фрагментации пакета id:
        удалить узлы[H].фрагменты[id]
    case уведомление о завершении копирования пакета id:
        удалить узлы[H].копии[id]
    case блок событий E[1]..E[N] для пакета id:
        if E[1] = = "формирование нового пакета":
            журнал : = новый журнал
            журнал.начало : = блок
        else:
            блок0 : = извлечь узлы[H].пакеты[id]
            добавить ссылку блок0 → блок
        обработать первое событие блока
        обработать последнее событие блока
```

При обработке первого события блока событий выполняется добавление связей между блоками по следующему алгоритму:

```
[обработать первое событие блока]
switch (E[1]):
    case фрагментация пакета id0:
        добавить ссылку узлы[H].фрагменты[id0] → блок
    case многоадресная рассылка id0:
        добавить ссылку узлы[H].копии[id0] → блок
    case дефрагментация:
        for each блок0 in узлы[H].склейка[id]:
            добавить ссылку блок0 → блок
        удалить узлы[H].склейка[id]
    default:
        ничего не делать
```

На основании последнего события в блоке выполняется обновление структур данных, его обработка проводится следующим образом:

```
[обработать последнее событие блока]
switch (E[N]):
    case фрагментация:
        узлы[H].фрагменты[id] : = новый служебный блок фрагментации
        добавить ссылку блок → узлы[H].фрагменты[id]
    case многоадресная рассылка:
        узлы[H].копии[id] : = новый служебный блок многоадресной рассылки
        добавить ссылку блок → узлы[H].копии[id]
    case дефрагментация пакета id2:
        блок1 : = новый служебный блок сборки
        добавить ссылку блок → блок1
        if ключ id2 отсутствует в словаре узлы[H].склейка:
            узлы[H].склейка[id2] : = новый список
        добавить блок1 в список узлы[H].склейка[id2]
    case передача в сетевой интерфейс I:
        сегмент сети S : = сегменты[узлы[H].интерфейсы[I]]
        S[P] : = блок
    case доставка по назначению или сброс:
        закрыть журнал пакета
```

Для корректной работы алгоритма сообщения от узлов сети должны поступать на вход сборщика строго в хронологическом порядке.

Программная реализация метода

Для создания программной реализации рассматриваемого метода необходимо разработать архитектуру программного комплекса и реализовать его, включая взаимодействие с сетевой подсистемой ядра ОС.

Архитектура программного комплекса. Архитектура реализующего разработанный метод программного комплекса представлена на рис. 5. Созданная программная реализация использует для выполнения среду Netkit [4] — инструмент, позволяющий связать в сеть виртуальные машины на основе ОС Linux. Детали реализации, специфичные для этой среды, ниже отмечаются особо.

Ловушки событий реализованы в патче к ядру Linux и позволяют вызывать функцию-обработчик событий жизненного цикла IP-пакета.

Модуль регистрации событий представляет собой загружаемый модуль ядра, содержащий саму функцию-обработчик. Специфичным для среды Netkit является способ передачи сообщений сборщику журналов: для приема сообщений от виртуальных машин используются UNIX-сокеты.

Сборщик журналов получает информацию от модулей регистрации событий и реализует описанный алгоритм сборки журналов.

Сборщик журналов использует в работе информацию о топологии сети, она также отображается в пользовательском интерфейсе. При работе со средой Netkit эти данные считываются из конфигурационных файлов виртуальной сети.

Рис. 5. Архитектура программного комплекса

Ловушки в сетевой подсистеме ОС Linux. Для перехвата информации о событиях жизненного цикла IP-пакета в сетевой подсистеме ядра ОС необходимо расположить ловушки. На рис. 6 представлена схема взаимодействия функций ядра, образующих основу реализации протокола IPv4, реализация протокола IPv6 в целом аналогична [5]. Функции с префиксом **NF_** на схеме фактически реализуются функцией **nf_hook_slow** сетевого экрана Netfilter.

В табл. 1 приведено соответствие между событиями жизненного цикла IP-пакета и функциями сетевой подсистемы ядра Linux. Все события, показанные на рис. 1, возникают в единственной функции, за исключением прохождения пакета через IP-тоннель. Поскольку границами IP-тоннеля являются виртуальные сетевые интерфейсы, то движение пакетов через них распознается как вход или выход из тоннеля.

Каждая ловушка сигнализирует о конкретном событии с конкретным IP-пакетом, поэтому в обработчик перехваченных событий передается идентификатор IP-пакета, место возникновения события, код события и зависящие от события параметры. В качестве идентификатора IP-пакета используется адрес экземпляра структуры ядра **sk_buff**, в которой содержится IP-пакет. Событие идентифицируется функцией, где оно возникло, и кодом из множества {GENERIC, BEFORE, AFTER, DROPPED}, поскольку в ряде функций возникает несколько событий.

Ловушка с кодом GENERIC обычно размещается в самом начале каждой функции, показанной на рис 6. Ловушки с кодом DROPPED размещаются в местах отбрасывания пакетов, обычно перед вызовом функции **kfree_skb**. Дополнительная ловушка с кодом DROPPED размещается в функции **kfree_skb** для перехвата всех случаев удаления пакета.

Использование ловушек с кодами BEFORE и AFTER специфично для конкретных функций. В функции **nf_hook_slow** размещаются две ловушки: одна — до применения правил сетевого экрана, вторая — после. Фрагментация пакета (функция **ip_fragment**) также сопровождается двумя ловушками, как показано в следующем псевдокоде:

```
function ip_fragment(struct sk_buff *skb):
    ловушка(skb, ip_fragment, BEFORE, 0)
    while нужно разделять на фрагменты:
        struct sk_buff *skb_frag = создать_очередной_фрагмент(skb)
        ловушка(skb_frag, ip_fragment, GENERIC, skb)
        ip_finish_output2(skb_frag)
```

Аналогично размещаются ловушки в функции многоадресной пересылки **ip_mr_forward**, где создаются копии исходного пакета. В функции дефрагментации **ip_frag_reasm** ловушки размещаются следующим образом:

```
function ip_frag_reasm(struct ipq *qp, struct sk_buff *skb):
    выделить_память_под_собираемый_пакет(skb)
    for each skb_frag in qp:
        ловушка(skb_frag, ip_frag_reasm, BEFORE, skb)
        добавить_данные(skb_frag, skb)
        kfree_skb(skb_frag)
    ловушка (skb, ip_frag_reasm, AFTER, 0)
```

Предварительная обработка перехваченных событий. Модуль регистрации событий, функционирующий на каждом узле сети, обрабатывает полученные от ловушек события. Результатом работы модуля являются сообщения с блоками событий и управляющие сообщения, пересылаемые централизованному сборщику журналов: *уведомления о поступлении пакета* содержат имя сетевого интерфейса, с которого получен пакет, а *уведомления о завершении операции* содержат код только что завершенной операции (фрагментация либо многоадресная рассылка).

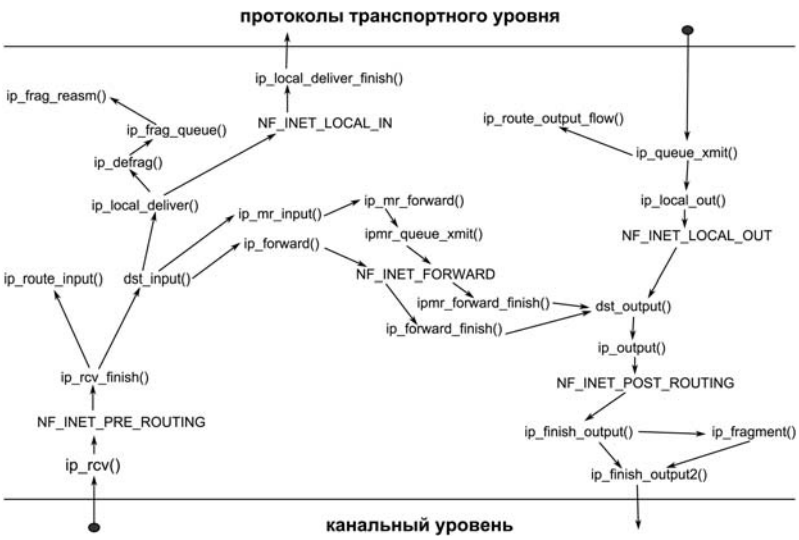


Рис. 6. Функции сетевой подсистемы ядра ОС Linux

Таблица 1

Связь событий жизненного цикла и функций ядра Linux

Событие	Реализация IPv4	Реализация IPv6
Формирование нового IP-пакета узлом-отправителем	ip_local_out	ip6_local_out
Доставка пакета по назначению	ip_local_deliver_finish	ip6_input_finish
Передача пакета сетевому интерфейсу	ip_finish_output2	ip6_output_finish
Поступление пакета от сетевого интерфейса	ip_rcv	ipv6_rcv
Пересылка пакета (простая и многоадресная)	ip_forward, ip_mr_forward	ip6_forward, ip6_mr_forward
Отбрасывание пакета	Любая функция + kfree_skb	
Фрагментация пакета	ip_fragment	ip6_fragment
Дефрагментация пакета	ip_defrag, ip_frag_reasm	ip6_frag_rcv, ip6_frag_reasm
Преобразование сетевых адресов	nf_hook_slow в Netfilter	

Общими для всех сообщений полями являются символьное имя узла и адрес экземпляра **sk_buff**, идентифицирующий IP-пакет в пределах узла. Уведомления о поступлении пакета и сообщения с блоками событий содержат первые 96 байт пакета.

Модуль регистрации событий ведет список незаконченных блоков событий для пакетов, находящихся в данный момент в сетевой подсистеме узла. Вызываемая в ловушке функция ищет в данном списке блок по перехваченному адресу структуры **sk_buff** и добавляет в него информацию о событии. Дальнейшие действия модуля регистрации приведены в табл. 2.

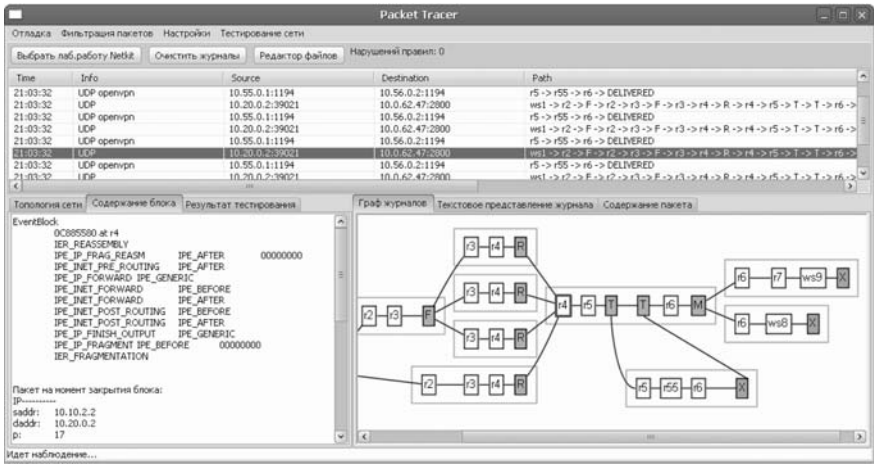


Рис. 7. Главное окно программы

Интерфейс пользователя. Внешний вид пользовательского интерфейса разработанного ПО представлен на рис. 7. В верхней части окна находится список журналов движения IP-пакетов. Правая нижняя часть отображает составленный журнал для выбранного в верхнем списке исходного пакета. Левая нижняя часть окна показывает содержимое выбранного блока событий.

Таблица 2

Правила обработки событий

Источник события	Код события	Действия
ip_rcv	GENERIC	Отправить уведомление о получении пакета; создать новый блок событий
ip_local_out ip_fragment ip_mr_forward ip_frag_reasm	GENERIC GENERIC GENERIC AFTER	Создать новый блок событий для обрабатываемого IP-пакета
ip_fragment ip_mr_forward	AFTER AFTER	Отправить уведомление о завершении операции
ip_local_deliver_finish ip_finish_output2 любой ip_fragment ip_mr_forward ip_frag_reasm	GENERIC GENERIC DROPPED BEFORE BEFORE BEFORE	Закрыть блок событий и отправить его сборщику журналов
NF_INET_PRE_ROUTING NF_INET_LOCAL_OUT NF_INET_POST_ROUTING	BEFORE BEFORE BEFORE	Зафиксировать значения адресов и портов для обнаружения преобразования сетевых адресов
NF_INET_PRE_ROUTING NF_INET_LOCAL_OUT NF_INET_POST_ROUTING	AFTER AFTER AFTER	При отличии текущих значений от зафиксированных добавить в блок событие преобразования адресов

Использование созданного ПО

Рассмотрим простую сеть из трех узлов, показанную на рис. 8.

Выполним на узле **ws1** команду **ping -c 1 -s 2000 10.20.0.2**, при этом будет создан IP-пакет размером 2028 байт. Основываясь на [1], можно предположить следующий исход опыта: узел **ws1** разбивает пакет на два фрагмента, размер которых не превышает значение MTU (1500 байт), которые будут собраны на узле-получателе **ws2**.

Созданная система показывает отличающийся от ожидаемого журнал, показанный на рис. 9 (обозначения блоков: *F* — фрагментация, *R* — дефрагментация, *X* — доставка). Маршрутизатор **r1** собирает поступившие фрагменты и затем заново разбивает результат на две такие же части перед дальнейшей посылкой. Блок событий, предшествующих дефрагментации на маршрутизаторе **r1**, показан ниже.

```
1  0C885880 at r1
2  <--- eth0
3  (RECIEVED)
4  IP_RCV                  GENERIC
5  INET_PRE_ROUTING        BEFORE
6  IP_DEFRAG               BEFORE
7  IP_FRAG_REASM           BEFORE      0C885380
8  (REASSEMBLY)
```

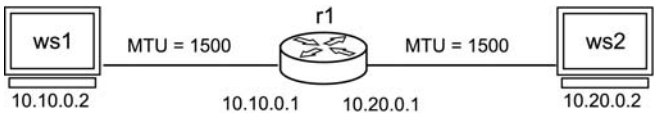


Рис. 8. Топология сети для опыта

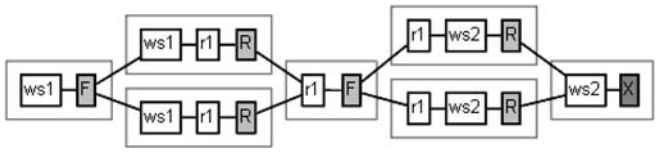


Рис. 9. Журнал пакета

Судя по строкам 5—6, процедура дефрагментации была инициирована после попадания в ловушку сетевого экрана Netfilter, но до выхода из этой ловушки. Изучение исходных кодов ядра Linux объяснят наблюдаемый эффект: принудительная дефрагментация вызывается из функции **ipv4_conntrack_defrag** модуля отслеживания соединений для всех входящих фрагментированных пакетов.

Заключение

В работе был создан метод отслеживания IP-пакетов, позволяющий получить полную информацию о перемещении и трансформациях каждого отдельного пакета. Описана реализация разработанного метода для сети с узлами на основе ОС Linux. Приведен опыт с программной реализацией метода, показывающий неочевидную особенность сетевой подсистемы ОС Linux.

Список литературы

1. **Postel J.** Internet Protocol. RFC 791 (Standard). 1981.
2. **Deering S**. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard). 1998.
3. **Mccanne S., Jacobson V.** The BSD Packet Filter: A New Architecture for User-level Packet Capture // Proc. 1993 Winter USENIX Conference. 1993. P. 259—269.
4. **Pizzonia M., Rimondini M.** Easy Emulation of Complex Networks on Inexpensive Hardware // Proc. 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM 2008), Innsbruck. 2008.
5. **Wehrle K., Pahlke F., Ritter H.** et al. Linux Network Architecture. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004. 648 p.

УДК 004.896

О. Ф. Немолочнов, д-р техн. наук, проф., зав. каф.,
e-mail: nemolochnov_o_f@mail.ru,
А. Г. Зыков, канд. техн. наук, доц.,
e-mail: zykov_a_g@mail.ru,
В. И. Поляков, канд. техн. наук, доц.,
А. В. Меженин, канд. техн. наук, доц.,
Санкт-Петербургский национальный
исследовательский университет
информационных технологий,
механики и оптики

Неравенства-отношения и выбор альтернативных решений управления вычислительными процессами

Исследуются вопросы поиска и построения альтернативных решений управления вычислительными процессами, заданного системой неравенств. Определяется общее число неравенств в системе, число реально существующих отношений и число неравенств, не имеющих решений. Рассмотрено множество решений и предикаты от одной до трех переменных, приведена их графическая интерпретация.

Ключевые слова: *вычислительный процесс, неравенства-отношения, don't care, частично определенные функции*

Постановка задачи

Решением любого неравенства является множество отношений между его переменными, а простые предикаты на этих множествах образуют булевы переменные. Решения системы неравенств образуют сложные предикаты в виде булевых функций *f*. Системы неравенств могут иметь или не иметь решений, и, следовательно, булевы функции *f* являются частично определенными. В области, где система неравенств не имеет решений, значение функции $f = d$ (*don't care*), а в области, где система имеет решение, значение булевой функции является определенным и полагается $f = p$, где $p \in \{0, 1\}$. В этом случае функция может быть задана в виде дизъюнкции конъюнкций простых предикатов как альтернативные решения.

Любое управление, в том числе и управление вычислительным процессом, состоит, по своей сути, в принятии решений. Для того чтобы был выбор, должно существовать некоторое множество альтернатив и множество условий их инициализации, которое, в общем случае, не зависит от физической реализации процесса, например, в виде схемного или программного продукта. Вычисление альтернатив осуществляется условными операторами *if* <*условие* > *then* S_1 *else* S_2 и *if* <*условие* > *then S*. В первом случае обязательно исполняется либо S_1, либо S_2, что и создает альтернативу. Во втором случае оператор *S* либо исполняется, либо не исполняется, что также является альтернативой.

Условия есть не что иное, как логические (булевы) функции $f = f(a, b, c, ...)$ в виде конъюнкций и дизъюнкций переменных, на которых она задана. Значение функции *f* может быть определенным: $f = p$, $p \in \{0, 1\}$, безразличным: $f = \times$, но вычислимым произвольно в $\times = 0$ или $\times = 1$, и не определенным $f = d$, т. е. невычислимым из-за ограничений на типы и области изменения величин переменных, на которых функция *f* задана при проектировании.

Пусть M_p есть множество наборов аргументов, на которых f вычислима, знак "×" используется для компактного задания подмножеств из M_p, а M_d есть множество наборов аргументов, на которых функция f из-за тех или иных ограничений не определена. Если $M_d \neq \varnothing$, то функция f является частично определенной. Универсальное множество булевой функции от n переменных задается объединением двух множеств $U = M_p \cup M_d$, а комплекс функции на кубе $E_2^n = (e_1, e_2, ..., e_n)$ [1—3] задается объединением подкомплексов $K(f^p)$ и $K(f^d)$, т. е. $K(f) = K(f^1) \cup K(f^0) \cup K(f^d)$.

Если природа формирования подкомплекса $K(f^p)$ понятна, то интерпретация подкомплекса $K(f^d)$ как при формировании проектного задания, так и при его анализе вызывает определенные затруднения [4]. Источником наборов аргументов булевой функции, входящих в M_d, могут являться:

- избыточное кодирование переменных (входных и выходных) и состояний вычислительного процесса двоичными значениями;
- ограничения на типы и диапазоны изменения переменных;
- отсутствие решения системы неравенств, которые задаются условиями (и, соответственно, предикатами от них) выбора альтернативного решения.

Поиск решений систем неравенств условий-предикатов и будет рассматриваться в дальнейшем.

Неравенства-отношения и предикаты от *n* переменных

Для перехода от систем неравенств, задающих отношение между переменными в виде множеств, необходимо взять предикаты на этих множествах. Множества могут быть пустыми, если система не имеет решения, и не пустыми, если решение есть. Простой предикат от одного неравенства порождает логическую переменную, а сложный предикат от системы неравенств — булеву функцию. Другими словами, необходимо установить общие закономерности между неравенствами и законами булевой алгебры для того, чтобы, формулируя задание на проектное решение альтернативных вычислений в предметной области, в целях упрощения их синтеза и анализа перейти к формальной логике.

Последовательно рассмотрим неравенства от $n = 0, 1, 2, 3, ...$ переменных. Их сочетание каждого с каждым дает общее число неравенств в системе $2^{C_n^2}$, а число реально существующих отношений равняется $n!$, и, следовательно, число неравенств, не имеющих решений, $m_d = 2^{C_n^2} - n!$.

При $n = 0$ имеем $m_d = 2^0 - 0! = 0$. В этом случае неравенство может быть тождественно истинным

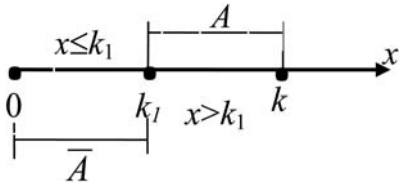


Рис. 1. Множества отношений-неравенств от одной переменной

или тождественно ложным. Например, неравенство $4 > 3$ истинно, и, обозначив $A = M(4 > 3)$, получим предикат $P(A) \equiv 1$, а неравенство $4 \leqslant 3$ ложно, и, обозначив $\overline{A} = M(4 \leqslant 3)$, получим предикат $P(\overline{A}) \equiv 0$, что соответствует понятиям единицы и нуля в булевой алгебре.

Положив $n = 1$, получим $m_d = 2^0 - 1! = 0$. Это означает, что неравенства с одной переменной и некоторым числом констант всегда имеют решение, ограниченное типом переменной и диапазоном ее изменения. Множество отношений вырождается в линию и может быть изображено на числовой оси по Ламберту [5]. Например, пусть заданы неравенства: $x > k_1$ и $x \leqslant k_1$ в некотором диапазоне $x[0...k]$ и $k > k_1$. Тогда, обозначив $A = M(x > k_1)$ и $\overline{A} = M(x \leqslant k_1)$, получим предикаты $P(A) = 1$ и $P(\overline{A}) = 0$. Указанные отношения показаны на рис. 1 в виде линии Ламберта.

Особо рассмотрим системы неравенств от $n = 2$ и $n = 3$ булевых переменных как предельные случаи. Заметим, что системы неравенств от $n \geqslant 4$ переменных можно свести к сочетаниям из них по три и тиражированию систем неравенств, имеющих и не имеющих решения при безразличных значениях остальных булевых переменных.

Множество решений неравенств и предикаты от двух переменных

Решить неравенство — значит, найти такие значения переменных, при которых оно бы выполнялось. Неравенства задаются отношениями, которые образуют взаимно дополняющие друг друга до универсума U пары: ($>$, $\leqslant$), ($=$, $\neq$) и ($<$, $\geqslant$). Эти пары, в свою очередь, являются взаимно исключающими, что полностью соответствует понятию логических переменных и функций в булевой алгебре.

При $n = 2$ имеем $m_d = 2^1 - 2! = 0$, т. е. все неравенства системы имеют решения.

На рис. 2, *а*, на универсальном множестве U показаны множества решений для двух переменных x_1 и x_2 в диапазоне $[0...k]$. Здесь множество, обозначенное как $M(x_1 < x_2)$, есть решение неравенства $x_1 < x_2$, а $M(x_1 > x_2)$ и $M(x_1 = x_2)$ — неравенства $x_1 > x_2$ и равенства $x_1 = x_2$ соответственно. Таким образом, универсальное множество образуется объединением

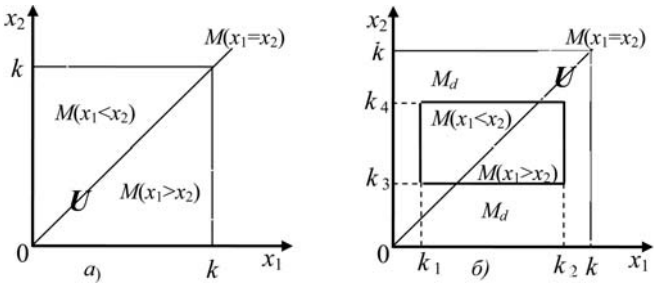


Рис. 2. Диаграммы множеств решений системы неравенств для двух переменных

указанных множеств $U = M(x_1 < x_2) \cup M(x_1 > x_2) \cup \cup M(x_1 = x_2)$, а их пересечение пусто. Попарное объединение данных множеств дает решение остальных трех неравенств: $M(x_1 < x_2) \cup M(x_1 > x_2)$ есть решение равенства $x_1 = x_2$, а $M(x_1 > x_2) \cup M(x_1 = x_2)$ — неравенства $x_1 \geqslant x_2$ и $M(x_1 < x_2) \cup M(x_1 = x_2)$ — неравенства $x_1 \leqslant x_2$ соответственно.

На рис. 2, *б* приведены те же множества с учетом ограничения диапазонов изменения переменных: x_1 $[k_1, k_2]$ и x_2 $[k_3, k_4]$. На рисунке также показано множество M_d — потенциально возможных, но не существующих решений из-за ограничений на диапазоны изменения переменных x_1 и x_2. Заметим, что ограничение диапазонов в значительной степени усложняет поиск решений систем неравенств.

Неравенства и предикаты от трех переменных

Для трех переменных x_1, x_2 и x_3 при установлении отношений каждого с каждым число систем неравенств равно $2^{C_3^2} = 2^3 = 8$, а число размещений — $3! = 6$, поэтому $m_d = 2$, и для двух систем решение должно отсутствовать. Универсум U можно изобразить в виде трехмерного куба $E_2^3 = (x_1, x_2, x_3)$, что и показано на рис. 3, *а*, где диапазон изменения величин x_1, x_2 и x_3 [0, 1]. Такой диапазон задан для удобства кодирования вершин, ребер, диагоналей и плоскостей, так как рассматривается топологическое пространство, и замена k на 1 является нормированием и не меняет сути.

На рис. 3, *а* диагональная плоскость $M(x_1 = x_2)$ обозначена вершинами: 000, 001, 111 и 110, плоскость $M(x_1 = x_3)$ — вершинами 000, 101, 111 и 010 и плоскость $M(x_2 = x_3)$ — вершинами 000, 100, 111 и 011. Каждая плоскость делит куб на две призмы. На рис. 3, *б* показаны призмы, разделенные плоскостью $M(x_1 = x_2)$. Эта плоскость разделяет множества $A = M(x_1 < x_2)$ и $\overline{A} = M(x_1 \geqslant x_2)$, плоскость $M(x_1 = x_3)$ — множества $B = M(x_1 < x_3)$ и $\overline{B} = M(x_1 \geqslant x_3)$, а плоскость $M(x_2 = x_3)$ — множества $C = M(x_2 < x_3)$ и $\overline{C} = M(x_2 \geqslant x_3)$. Для этих множеств $A \cup \overline{A} = B \cup \overline{B} = C \cup \overline{C} = U$, а $A \cap \overline{A} = B \cap \overline{B} = C \cap \overline{C} = \phi$. Пересечение плоскостей в кубе и, соответственно, пересечение множеств $(A \cup \overline{A}) \cap (B \cup \overline{B}) \cap (C \cup \overline{C})$ порождает шесть прямоугольных тетраэдромов (рис. 4), каждый из которых содержит в себе все множество решений соответствующей ему системы неравенств.

Как было сказано выше, для двух систем решения отсутствуют. И, действительно, нельзя построить пирамиды для пересечений $\overline{A} \cap B \cap \overline{C} = \phi$ и $A \cap \overline{B} \cap C = \phi$. Следовательно, системы неравенств, для которых существуют пирамиды, имеют множество решений, которое не пусто. От них можно взять сложные предикаты в виде конъюнкций простых предикатов: $P(A) = a$, $P(\overline{A}) = \overline{a}$, $P(B) = b$, $P(\overline{B}) = \overline{b}$ и $P(C) = c$, $P(\overline{C}) = \overline{c}$.

Далее, на тех конъюнкциях, которые существуют, можно образовать булевы функции $f = p$, где p можно произвольно положить равным 0 или 1 в соответствии с требованиями вырабатывания альтернативных решений, а для несуществующих конъ-

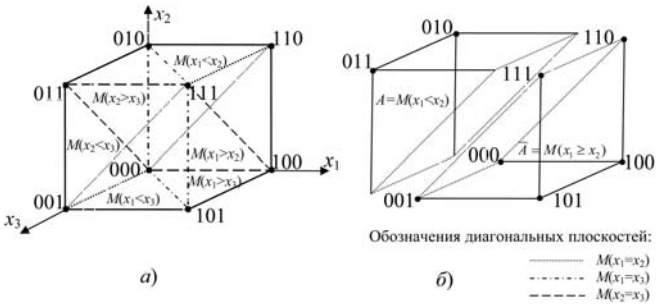


Рис. 3. Универсальный куб $E_2^3 = (x_1, x_2, x_3)$ (*а*) и куб, разделенный плоскостью $M(x_1 = x_2)$ (*б*)

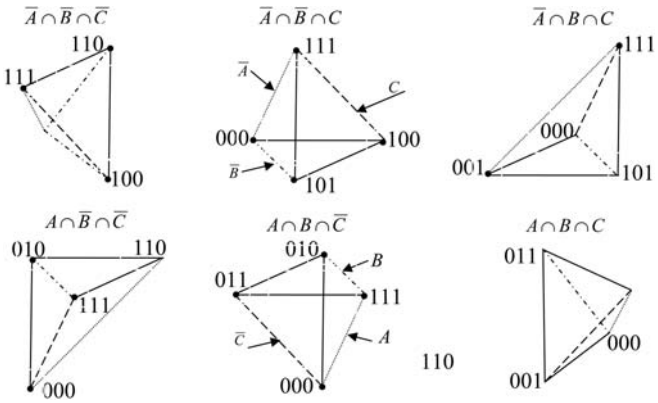


Рис. 4. Тетраэдромы, содержащие множество решений систем неравенств

Альтернативные решения системы неравенств от трех переменных

Системы неравенств	Множества решений	Предикаты $a\ b\ c$	Функция $f(a,b,c)$	Признак ∃ решения	Примечание
$x_1 \geqslant x_2$, $x_1 \geqslant x_3$, $x_2 \geqslant x_3$	$\overline{A} \cap \overline{B} \cap \overline{C} \neq \phi$	0 0 0	p	∃	
$x_1 \geqslant x_2$, $x_1 \geqslant x_3$, $x_2 < x_3$	$\overline{A} \cap \overline{B} \cap C \neq \phi$	0 0 1	p	∃	
$x_1 \geqslant x_2$, $x_1 < x_3$, $x_2 \geqslant x_3$	$\overline{A} \cap B \cap \overline{C} = \phi$ $\overline{A} \cap B \subset C$	0 1 0	d	∄	$b \to c$ при $a = 0$
$x_1 \geqslant x_2$, $x_1 < x_3$, $x_2 < x_3$	$\overline{A} \cap B \cap C \neq \phi$	0 1 1	p	∃	
$x_1 < x_2$, $x_1 \geqslant x_3$, $x_2 \geqslant x_3$	$A \cap \overline{B} \cap \overline{C} \neq \phi$	1 0 0	p	∃	
$x_1 < x_2$, $x_1 \geqslant x_3$, $x_2 < x_3$	$A \cap \overline{B} \cap C = \phi$ $A \cap C \subset B$	1 0 1	d	∄	$\overline{b} \to \overline{c}$ при $a = 1$
$x_1 < x_2$, $x_1 < x_3$, $x_2 \geqslant x_3$	$A \cap B \cap \overline{C} \neq \phi$	1 1 0	p	∃	
$x_1 < x_2$, $x_1 < x_3$, $x_2 < x_3$	$A \cap B \cap C \neq \phi$	1 1 1	p	∃	

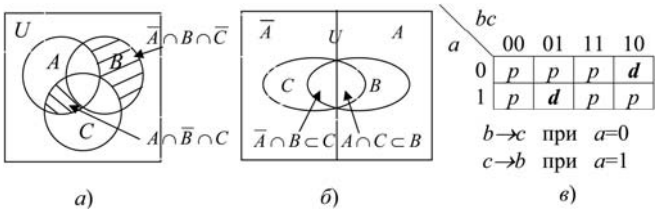


Рис. 5. Различные отображения частично определенной функции $f(a, b, c)$

юнкций значение f приравнивается d. Это значение, как правило, используется для упрощения логических выражений.

Соотношения между системами неравенств, множествами их решений, предикатами и булевыми функциями сведены в таблицу. Таблица является полной и фактически является таблицей истинности в формальной логике. Значение $f = d$ получается, если какое-то пересечение двух множеств из трех содержится в третьем. Это является признаком импликации, который не зависит от порядка пересечений и вычислений условий [6].

Функция $f = f(a, b, c)$ является частично определенной. Множества решений систем неравенств можно изобразить на диаграмме Эйлера—Венна (рис. 5, *а*). Недостатком такого изображения является возможность отображения несуществующих решений (заштрихованные области рис. 5, *а*), так как диаграммы Эйлера—Венна задают множества полностью определенных функций. Этого недостатка лишено изображение множеств на стилизованной диаграмме (рис. 5, *б*). На рис. 5, *в* показан обобщенный аналог карты Карно. Если на карте p заменить на 1, а d на 0, то получатся карты Карно функции импликации при $a = 0$ и $a = 1$. Заметим, что на карте можно задать до шести альтернативных решений.

Заключение

Исследование систем неравенств альтернативных решений управления вычислительным процессом и, в свою очередь, систем управления на его основе является необходимым условием проектирования любого устройства и не зависит от физической реализации. Эти исследования должны быть полными и входить в любую систему автоматизации проектирования и как составная часть — в любую систему защиты информации на функциональном уровне.

Рассмотренный выбор альтернативных решений управления вычислительными проектами является методологической основой вычисления переменных в условных вершинах графоаналитических моделей вычислительных процессов в целях синтеза тестов для их программных реализаций.

Список литературы

1. **Roth J. P**. Diagnosis of automata failures: a calculus and a method // IBM Journal of Research and Development. 1966. N 7. P. 18—32.

2. **Миллер Р.** Теория переключательных схем. Т. 1. М.: Наука, 1970. 416 с.

3. **Немолочнов О. Ф., Зыков А. Г., Поляков В. И.** Комплексные кубические покрытия и графо-аналитические модели как средство описания вычислительных процессов программ // Тр. Междунар. научно-техн. конф. "Интеллектуальные системы" (AIS'06) и "Интеллектуальные САПР" (CAD-2006). В 3 т. М.: Физматлит. 2006. Т. 2. С. 3—7.

4. **Немолочнов О. Ф., Зыков А. Г., Кулагин В. С., Осовецкий Л. Г., Поляков В. И., Суханов А. В.** Метод обнаружения недекларированных возможностей и значений ***DON'T CARE*** вычислительного процесса // Изв. вузов. Приборостроение, 2009. Т. 52. № 12. С. 32—39.

5. **Кондаков Н. И.** Логический словарь. М.: Наука. 1971. 656 с.

6. **Немолочнов О. Ф., Зыков А. Г., Поляков В. И.** Импликация и эквивалентность как основа верификации // Научно-технический вестник СПбГУ ИТМО. СПб: СПбГУ ИТМО, 2010. № 4 (68). С. 122.

УДК 519.857

В. И. Струченков, д-р техн. наук, проф.,
МГТУ МИРЭА, г. Москва,
e-mail: str1942@mail.ru

Устаревшие стереотипы и новые алгоритмы решения прикладных задач дискретной оптимизации

Рассмотрены различные алгоритмы решения ряда задач дискретной оптимизации. Показано, что известные алгоритмы могут быть существенно улучшены при использовании комбинированных методов: динамическое программирование с отбраковкой не только путей, приводящих в одно и то же состояние, но и бесперспективных (доминируемых) состояний, плюс метод ветвей и границ для вычисления двусторонних прогнозных границ оптимума для дополнительной отбраковки состояний. Для вычисления границ предложен новый алгоритм "спуска — подъема". Приведены результаты сопоставительных расчетов по различным алгоритмам применительно к задаче оптимального распределения ресурса.

Ключевые слова: *динамическое программирование, множества Парето, метод ветвей и границ*

Введение

При создании сложных систем в многократно повторяющийся цикл расчетов часто приходится встраивать решение задачи оптимизации. От алгоритма оптимизации зависит быстродействие системы. Известно, что при использовании традиционных алгоритмов, в частности метода динамического программирования, с ростом размерности задачи резко растет объем вычислений, что получило название "проклятие размерности". В этой связи, несмотря на успехи вычислительной техники, разработка новых быстродействующих алгоритмов остается актуальной.

1. Постановка задачи

Рассмотрим задачу: найти максимум (или минимум)

$$\sum_{i=1}^{n} g_i(x_i) \qquad (1)$$

при ограничениях

$$\sum_{i=1}^{n} f_i(x_i) \leqslant R,\ x_i \in X_i,\ i = 1, 2, ..., n, \qquad (2)$$

где X_i — конечные множества, $f_i(x_i) \geqslant 0$, $g_i(x_i) \geqslant 0$ и $R > 0$. Целочисленность функций $f_i(x_i)$ и $g_i(x_i)$ необязательна. Предполагается, что множество допустимых решений не пусто.

В таком виде могут быть записаны различные задачи, сводящиеся к распределению заданного ресурса R между n потребителями. В задаче на максимум функции $f_i(x_i)$ характеризуют ресурс, а $g_i(x_i)$ — эффективность его использования. При $f_i(x_i) = p_i x_i$ и $g_i(x_i) = c_i x_i$ получается задача оптимальной загрузки транспортного средства предметами, масса которых p_i, стоимость c_i, а их число x_i ($x_i = 0, 1, 2, ...$) [1], если при этом $x_i = \{0, 1\}$, получается известная задача "о ранце"[2].

В задаче на минимум функции $f_i(x_i)$ — ресурс, а $g_i(x_i)$ — затраты. Например, $f_i(x_i)$ — ресурс времени, а $g_i(x_i)$ — затраты на осуществление некоторого проекта.

Или как при проектировании оптимальной защиты поверхности [3] $f_i(x_i)$ — допускаемый ущерб от неполной защиты i-го элемента поверхности, а $g_i(x_i)$ — соответствующие затраты.

К задаче (1), (2) сводится поиск оптимальных характеристик надежности системы, состоящей из n приборов, каждый из которых имеет несколько вариантов технической реализации [4]. В этом случае $g_i(x_i)$ — вероятности отказа ($g_i(x_i) \ll 1$), а $f_i(x_i)$ — стоимость, вес или другая характеристика i-го прибора ($i = 1, ..., n$). К этой же задаче сводится и выбор параметров для контроля работоспособности систем при ограничении на суммарное время контроля, а также задача оптимального резервирования [4].

Возможно наличие нескольких ограничений вида (2), например ограничения на массу и объем. Для решения этой более сложной задачи в работе [4] предложены различные способы, основанные на многократном решении задачи (1), (2) с одним ограничением. Разработка быстродействующих алгоритмов решения этой задачи открывает перспективы создания быстродействующих алгоритмов решения задач с несколькими ограничениями, в частности задач целочисленного линейного программирования.

Далее используются обозначения: x_i^j ($i = 1, ..., n$; $j = 1, ..., k_i$) — значения $x_i \in X_i$; $f_i^j = f_i(x_i^j)$ и $g_i^j = g_i(x_i^j)$, соответствующие им дискретные значения функций $f_i(x_i)$ и $g_i(x_i)$.

2. Классические алгоритмы динамического программирования

Для решения задачи (1), (2) применяется метод динамического программирования [1, 2, 4, 5, 6, 7]. При его изложении в отечественной учебной литературе для технических и экономических вузов [2, 5, 6] в последние 30—40 лет сложился стереотип, в соответствии с которым задача сводится к поиску оптимальной траектории, соединяющей узлы регулярной сетки. При этом считается несущественным выбор направления поиска (от начальной точки к конечной или наоборот); более того, в некоторых учебниках [6, с. 204] утверждается: "В большинстве приложений динамическое программирование получает оптимальное решение путем движения в обратном направлении — от конца задачи к началу".

Чтобы понять неэффективность такой реализации метода динамического программирования и наличие новых возможностей, рассмотрим решение задачи о загрузке автомашины, приведенное в учебном пособии [5, с. 104—106], которое в последние 40 лет выдержало более 14 изданий. Задача состоит в следующем: имеется автомашина грузоподъемностью $Q = 35$ единиц массы и шесть предметов, масса и стоимость которых указаны в табл. 1.

Суммарная масса предметов превышает грузоподъемность машины, поэтому нужно взять такие предметы, суммарная масса которых не превышает $Q = 35$, а суммарная стоимость максимальна.

Согласно [5, с. 104] рассматривается шесть шагов. Номер шага соответствует номеру предмета в табл. 1. На каждом шаге всего лишь два возможных решения: 0 — не брать соответствующий предмет и 1 — брать. Состояние системы S перед очередным шагом характеризуется ресурсом грузоподъемности, который еще остался до полной загрузки машины после того, как предыдущие шаги выполнены, т. е. решен вопрос, брать или не брать предметы с меньшими номерами, и какие-то из них погружены в машину. Для каждого состояния S предлагается найти $W_i(S)$ — суммарную максимальную стоимость предметов, которыми можно "догрузить" машину при данном значении S, и положить $x_i(S) = 1$, если мы берем данный (i-й) предмет, и $x_i(S) = 0$, если не берем. Последовательность величин $x_i(S)$ ($i = 1, 2, ..., 6$) определяет набор взятых предметов.

Решение начинается с анализа шестого шага. Вначале рассматриваются все переходы на шестом шаге из 36 возможных состояний в конечное состояние, затем переходы пятого шага из тех же 36 возможных состояний и т. д.

Таблица 1

Предмет $П_i$	$П_1$	$П_2$	$П_3$	$П_4$	$П_5$	$П_6$
Масса q_i	4	7	11	12	16	20
Стоимость c_i	7	10	15	20	27	34

Характерно, что даже на втором шаге рассматриваются все те же 36 состояний, хотя очевидно, что после первого шага есть только два состояния, после второго — четыре, после третьего — восемь и т. д., так что большинство состояний вообще можно было бы не рассматривать.

Вообще при полном переборе имеем всего 64 варианта (траектории), включая несколько недопустимых. Получается, что такая реализация динамического программирования может быть менее эффективна, чем полный перебор. Это становится очевидным, если решать ту же задачу при несколько иных исходных данных. Сохраняя то же значение ресурса $Q = 35$, массу отдельных предметов примем нецелой (например 3,9997 вместо 4; 7,003 вместо 7 и т. д.). Другими словами, в рассматриваемом примере ресурс составляет целое число тонн, а масса отдельных предметов выражается целыми числами не тонн, а килограммов. Тогда в целочисленном представлении $Q = 35000$ при массе отдельных предметов 3997,7003 и т. д.

Следуя стереотипу, после разбиения регулярной сетки состояний придется рассматривать на отдельных шагах алгоритма оптимизации 35001 состояние вместо 36, тогда как при полном переборе остаются все те же 64 варианта.

И вообще при любом заданном n существует ресурс Q, такой, что число рассматриваемых состояний по данному алгоритму регулярной сетки больше, чем при полном переборе ($Q > 2^n/n$).

Если решать ту же задачу в прямом направлении и рассматривать только реально достижимые состояния вместо узлов регулярной сетки, то число состояний будет такое же, как при полном переборе, но при наличии комбинаций предметов с равными массами (то есть различных путей, приводящих в некоторое состояние) в соответствии с принципом оптимальности Р. Беллмана можно оставить только одну комбинацию (путь), которой соответствует большая стоимость, что сокращает перебор. Например, в исходной задаче для первых трех предметов возможны две комбинации с суммарной массой 11, но с разными стоимостями. Первая комбинация: взять только третий предмет массой 11 и стоимостью 15, вторая комбинация: взять первые два предмета с общей массой 11 и стоимостью 17, а третий не брать. Первая комбинация (путь) и все ее продолжения отбраковываются. Такая реализация динамического программирования существенно более эффективна, но она возможна только при отказе от стереотипов разбиения регулярной сетки и поиска в обратном направлении. Допустимые состояния следует формировать в процессе счета, а не задавать с равным шагом.

При решении этой и аналогичных задач имеется возможность существенного повышения эффективности динамического программирования за счет отбраковки не только путей, приводящих в заданное

состояние, но и собственно некоторых бесперспективных состояний. Так, в рассматриваемом примере после четырех шагов есть два пути:

1) 1001 — берем только первый и четвертый предметы общей массой 16 и стоимостью 27;

2) 0110 — берем только второй и третий предметы общей массой 18 и стоимостью 25.

Получаем два различных состояния, причем первое предпочтительнее второго и по массе (она меньше), и по стоимости (она больше). Поскольку возможности дальнейшего выбора для первого состояния те же, что и для второго, второе состояние отбраковывается, и все его продолжения не рассматриваются. Отбраковка бесперспективных состояний существенно сокращает счет; так, после пятого шага останется для дальнейшего анализа только 15 (а не 32 и не 36) состояний.

Отбраковка не только путей, но и бесперспективных состояний реализована в новых алгоритмах, использующих множества Парето [8, 9].

3. Алгоритмы с отбраковкой состояний

Для задачи "о ранце" эта идея отбраковки состояний реализована в работе [8]. Алгоритмы, реализующие эту идею для решения более общей задачи (1), (2), рассмотрим на примере задачи на минимум.

Переменные x_i^j ($i = 1, ..., n$) пронумеруем так, что $f_i^1 < f_i^2 < ... < f_i^{ki}$. Если $f_i^j = f_i^{j+1}$, то исключаем f_i^{j+1} при $g_i^j < g_i^{j+1}$ и f_i^j в противном случае. Такая отбраковка предложена в работе [7]. Однако и при $f_i^j < f_i^{j+1}$ и $g_i^j \leqslant g_i^{j+1}$, можно исключить f_i^{j+1}, так как в задаче на минимум большему ресурсу должны соответствовать меньшие затраты, то есть в этом случае f_i^j всегда будет использоваться вместо f_i^{j+1}. После отбраковки получим $g_i^1 > g_i^2 > ... > g_i^{ki}$ ($i = 1, ..., n$). Следовательно, при заданном i оставшиеся точки с координатами (f_i^j, g_i^j) образуют множество Парето двухкритериальной задачи (минимум затрат при минимуме ресурса), остальные варианты бесперспективны. Для каждого $i = 1, 2, ..., n$ отложим по оси абсцисс значения f_i^j, а по оси ординат g_i^j и получим n строго монотонно убывающих ломаных линий.

В схеме динамического программирования под состоянием на каждом шаге будем понимать суммарное значение уже использованного ресурса. Соответственно, после первого шага алгоритма динамического программирования множество состояний f_1^j таково, что пары (f_1^j, g_1^j) образуют множество Парето.

На каждом следующем шаге $m = 2, ..., n$ рассмотрим множество точек (F_m, G_m) вида

$$F_m = \sum_{i=1}^{m} f_i(x_i),\ G_m = \sum_{i=1}^{m} g_i(x_i),$$

где вектор $(x_1, ..., x_m)$ пробегает все значения, удовлетворяющие условиям

$$\sum_{i=1}^{m} f_i(x_i) \leqslant R,\ x_i \in X_i,\ i = 1, ..., m.$$

Пусть для двух точек (F'_m, G'_m) и (F''_m, G''_m), где нижний индекс — номер шага, а верхний — номер точки, выполнено $F''_m \geqslant F'_m$ и $G''_m > G'_m$. Точке (F'_m, G'_m) соответствует путь $(x'_1, x'_2, ..., x'_m)$, а точке (F''_m, G''_m) — путь $(x''_1, x''_2, ..., x''_m)$.

Точку (F''_m, G''_m) можно исключить из рассмотрения, так как приводящий в нее путь не может быть частью оптимальной траектории. Действительно, пусть $(x''_{m+1}, x''_{m+2}, ..., x''_n)$ — оптимальное продолжение из (F''_m, G''_m), которому соответствуют затраты G^*. Тогда для $(x'_1, x'_2, ..., x'_m, x''_{m+1}, x''_{m+2}, ..., x''_n)$ — затраты $G'_m + G^* < G''_m + G^*$.

При $F''_m \geqslant F'_m$ и $G''_m = G'_m$ точку (F''_m, G''_m) также можно исключить. Это означает, что из множества точек (F_m, G_m) на каждом шаге m можно оставить только паретовское подмножество, причем из двух совпадающих точек остается только одна. Это правило включает и отбраковку неэффективных путей, приводящих в заданное состояние (при $F'_m = F''_m$), и отбраковку собственно бесперспективных состояний, которым соответствуют непаретовские точки.

Совокупность паретовских точек множества (F_m, G_m) обозначим через $S_m = \{(F_m^k, G_m^k) \mid k = 1, ..., K_m\}$, нумеруя их по возрастанию ресурса, т. е. $F_m^1 < F_m^2 < ... < F_m^{Km}$, $G_m^1 > G_m^2 > ... > G_m^{Km}$. Формирование пошаговых упорядоченных паретовских множеств S_m возможно различными способами. Вариант, при котором сначала формируется все множество допустимых состояний, а потом из него удаляются непаретовские точки, оказался неэффективным. Был реализован алгоритм:

$S_0 = \{(0, 0)\}$; $S_1 = \{(F_1^j = f_1^j,\ G_1^j = g_1^j) \mid j = 1, 2, ..., k_1\}$.

Отбраковки нет.

Общий шаг. Пусть уже построено множество $S_{m-1} = \{(F_{m-1}^k, G_{m-1}^k) \mid k = 1, ..., K_{m-1}\}$. На первом этапе ($k = 1$) вычисляем $\{(F_m^j = F_{m-1}^1 + f_m^j,\ G_m^j = G_{m-1}^1 + g_m^j) \mid j = 1, ..., k_m\}$ без отбраковки.

На втором этапе для $k = 2, ..., K_{m-1}$ (внешний цикл) последовательно вычисляем $P(F_{m-1}^k + f_m^j, G_{m-1}^k + g_m^j)$ $j = 1, ..., k_m$ (внутренний цикл). Для каждой вычисленной точки P после сравнения с уже имеющимися (ближайшими по значению ресурса) возможно три результата:

1) точка P не включается в формируемое множество, так как в нем есть доминирующая точка;

2) точка P включается в формируемое множество (с сохранением упорядоченности), так как нет доминирующих по отношению к ней точек и она не является доминирующей;

3) новая точка P включается в имеющееся паретовское множество, а одна (возможно, и несколько) точка, по отношению к которым она является доминирующей, из него исключается.

В силу упорядоченности паретовского множества и $f_m^j \geqslant 0$ для поиска доминирующей точки нет необходимости в переборе всех его точек.

При $k = 2$ и $j = 1$ перебор начинается с 1, т. е. точки с F_m^1. При $k = 3, ..., K_{m-1}$ и $j = 1$ перебор начинается с номера, полученного точкой с $F_{m-1}^{k-1} + f_m^1$ или доминирующей над ней. При $k = 3, ..., K_{m-1}$ и $j = 2, ..., k_m$ во внутреннем цикле перебор начинается с номера точки в формируемом множестве, полученного точкой с $F_{m-1}^k + f_m^{j-1}$ или доминирующей над ней. Перебор идет всегда с увеличением ресурса и заканчивается при достижении $F_{m-1}^k + f_m^j$, т. е. ресурса новой точки — "кандидата". В п. 3 поиск доминируемых точек начинается с $N + 1$, где N — номер включенной новой точки, и прекращается при первой неудачной попытке.

В итоге получаем паретовское множество точек, упорядоченных по возрастанию ресурса.

Для задач большой размерности, особенно при нецелых значениях f_i^j ($i = 1, 2, ..., n$; $j = 1, 2, ..., k_i$), число паретовских точек может быть велико, что потребовало разработки новых алгоритмов, позволяющих отбраковывать и часть бесперспективных паретовских точек.

Это возможно в процессе формирования множества состояний на всех шагах динамического программирования, если использовать приближенное решение задачи (начальное приближение) как верхнюю оценку оптимума с возможностью ее уточнения в процессе счета или строить двусторонние прогностические оценки оптимума. В частности, на каждом шаге в паретовском множестве содержатся точки, которым соответствует минимальный израсходованный ресурс и, соответственно, максимальные затраты, которые уже через несколько шагов могут превысить суммарные затраты на всех шагах, соответствующие начальному приближению. Такая ситуация наступает за меньшее число шагов при наличии хорошего начального приближения.

Предположим, что вычислен некоторый допустимый вектор $x^*(x_1^*, x_2^*, ..., x_n^*)$ и соответствующее значение целевой функции $G_n^* = \sum_{i=1} g_i(x_i^*)$.

При решении задачи (1), (2) на минимум на шаге m паретовская точка $(F_m^j, G_m^j) \in S_m$ отбраковывается, если $G_m^j > G_n^*$, так как она не может принадлежать оптимальной траектории, ибо на последующих шагах целевая функция может только возрастать в силу $g_i(x_i) \geqslant 0$.

Если для $j_k = \max j$: $F_m^j \leqslant F_m^*$, $\delta = G_m^* - G_m^{jk} > 0$, то точка (F_m^*, G_m^*) не является паретовской, она заменяется на (F_m^{jk}, G_m^{jk}); для $i > m$ сохраняются x_i^* и, соответственно, f_i^* и g_i^*, поэтому все G_i^* ($m < i <= n$) уменьшаются на δ, а F_i^* — на $F_m^* - F_m^{jk}$.

На последующих шагах ($m < i <= n$) при отбраковке паретовских точек используются скорректированные значения F_i^* и G_i^*. Затраты на оставшуюся часть траектории и, соответственно, верхнюю границу суммарных затрат можно приближенно вычислять для каждой паретовской точки, чтобы, используя минимальное из полученных значений суммарных затрат, уменьшить G_n^* и получить возможность дополнительной отбраковки паретовских точек. Соответствующие алгоритмы используют уточняемые значения G_n^* как верхние оценки оптимума, а нижняя оценка равна нулю и может быть далека от оптимума.

Для построения улучшенных двусторонних оценок оптимума используем комбинацию динамического программирования и метода ветвей и границ [4]. При этом каждое паретовское состояние на каждом шаге рассматривается как точка ветвления с построением двусторонних оценок оптимума (нижней и верхней границы). Эффективность такого метода зависит от числа состояний, вычислительных затрат на определение границ и их близости к оптимуму.

Обозначим затраты на всю траекторию, соответствующие начальному приближению, через E (в методе ветвей и границ они называются рекордом), а их нижнюю границу через H. Затраты на оставшуюся часть траектории из точки (F_m^j, G_m^j), соответствующие некоторому допустимому приближению, обозначим через E_m^j, их нижнюю границу через H_m^j. Тогда условие отбраковки паретовской точки (F_m^j, G_m^j) примет вид $G_m^j + H_m^j \geqslant E$.

Если же $G_m^j + E_m^j < E$, то новое значение рекорда равно $G_m^j + E_m^j$. Соответственно, изменяется и само начальное приближение.

Если для некоторых m и j $E_m^j = H_m^j$, то нет смысла рассматривать (F_m^j, G_m^j) как точку ветвления. Если при этом $G_m^j + E_m^j < E$, то эта точка и соответствующая траектория запоминаются и хранятся до тех пор, пока не будет получено значение рекорда меньшее, чем $G_m^j + E_m^j$. Если "рекорд устоит", то соответствующее ему решение и является оптимальным.

Если на некотором шаге не останется ни одной паретовской точки, то рекорд является искомым решением. На каждом шаге можно корректировать нижнюю границу, заменяя H на $h = \min(G_m^j + H_m^j)$ ($j = 1, 2, ..., K_m$), если $H < h$, и прекращать расчет при $E - H < \varepsilon H$, где ε определяется требуемой точностью решения задачи.

Начальное приближение можно построить различными способами. Простой алгоритм состоит в равномерном распределении ресурса между потребителями с последующим округлением до ближайшего (слева) заданного дискретного значения f_i^* и, соответственно, g_i^*.

Начальное приближение (траектория)

$$F_m = \sum_{i=1}^{m} f_i^*, \; G_m = \sum_{i=1}^{m} g_i^* \; m = 1, ..., n.$$

Аналогично можно строить начальное приближение в процессе динамического программирования для оставшейся части траектории, исходящей из любой паретовской точки.

4. Вычисление двусторонних оценок оптимума

Для построения двусторонних оценок оптимума используем кусочно-линейные функции $g_i(f_i)$, получаемые как описано в разд. 3. Те из них, которые не являются выпуклыми, заменим их выпуклыми оболочками $w_i(z_i)$ ($i = 1, 2, ..., n$) (рис. 1).

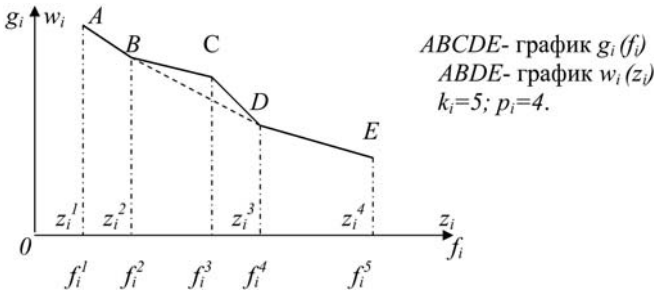


Рис. 1. Исходная функция $g_i(f_i)$ и ее выпуклая оболочка $w_i(z_i)$

В результате получаем непрерывную оценочную задачу: найти минимум

$$W(z) = \sum_{i=1}^{n} w_i(z_i), \qquad (3)$$

где $z(z_1, ..., z_n)$ — вектор неизвестных,

$$\text{при} \sum_{i=1}^{n} z_i \leqslant R, \; f_i^1 \leqslant z_i \leqslant f_i^{ki}, \; i = 1, ..., n. \qquad (4)$$

Оптимум непрерывной задачи (3), (4), который, очевидно, не более оптимума задачи (1), (2), примем в качестве искомой нижней границы.

В этой задаче нелинейного программирования целевая функция и система ограничений имеют существенные особенности, которые будут использованы для ее решения простыми алгоритмами.

Концы звеньев ломаных $w_i(z_i)$ обозначим через z_i^j ($j = 1, ..., p_i$).

Имеем $z_i^1 = f_i^1$, $z_i^{pi} = f_i^{ki}$, $p_i = k_i$, если $g_i(f_i)$ — выпуклая функция, иначе $p_i < k_i$.

Величины $u_i^j = |(w_i^{j+1} - w_i^j)/(z_i^{j+1} - z_i^j)|$ ($i = 1, ..., n; j = 1, ..., p_i - 1$) будем называть уклонами. В силу монотонности и выпуклости $w_i(z_i)$ последовательности уклонов u_i^j ($i = 1, ..., n$) строго монотонно убывающие.

Задача (3), (4) имеет простой смысл: насколько нужно спуститься по каждой ломаной $w_i(z_i)$ из начальной точки z_i^1 ($i = 1, ..., n$) и соответствующим значениям целевой функции, чтобы, не нарушая ограничение на сумму абсцисс, получить минимальную сумму ординат? Простой и достаточно очевидный ответ состоит в том, что на каждом шаге надо спускаться по звену с максимальным уклоном до исчерпания ресурса R. Приведем формальное обоснование этого утверждения.

Пусть $z^*(z_1^*, ..., z_n^*)$ — решение задачи (3), (4). Обозначим через $M = \{u_i^j \mid i = 1, ..., n; j = 1, ..., p_i - 1\}$ — множество всех уклонов; $M_1(z^*) = \{u_i^j: \; z_i^{j+1} \leqslant z_i^* \mid i = 1, ..., n; j = 1, ..., p_i - 1\}$; $M_2(z^*) = \{u_i^j: z_i^j < z_i^* < z_i^{j+1} \mid i = 1, ..., n; j = 1, ..., p_i - 1\}$ и $M_3(z^*) = \{u_i^j: z_i^* \leq z_i^j \mid i = 1, ..., n; j = 1, ..., p_i - 1\}$ — соответственно, множества уклонов, полностью использованных для спуска, используемых частично и не используемых при спуске; $M = M_1 \cup M_2 \cup M_3$.

$$W(z^*) = \sum_{i=1}^{n} (g_i^1 - \sum_{j=1}^{p_i} u_i^j \delta_i^j) = \sum_{i=1}^{n} g_i^1 - \sum_{i=1}^{n} \sum_{j=1}^{p_i} u_i^j \delta_i^j, \qquad (5)$$

где $\delta_i^j = z_i^{j+1} - z_i^j$, если $u_i^j \in M_1$; $\delta_i^j = z_i^* - z_i^j$, если $u_i^j \in M_2$ и $\delta_i^j = 0$, если $u_i^j \in M_3$.

Пусть $u_m^1 = \max u_i^1$ $(i = 1, ..., n)$. Докажем, что $u_m^1 \in (M_1 \cup M_2)$, то есть $z_m^* > z_m^1$.

Предположим, что это утверждение не верно. Возьмем минимальный уклон $u_k^r \in (M_1 \cup M_2)$, ему соответствует $\delta_k^r > 0$. Уменьшив δ_k^r на некоторую Δ, получаем увеличение целевой функции на $u_k^r \Delta$ и возможность ее уменьшения на бо́льшую величину $u_m^1 \Delta$ без нарушения ограничения по ресурсу, так как $u_m^1 > u_k^r$. Следовательно, z^* не оптимум. Полученное противоречие доказывает необходимость использования максимального уклона. Аналогично доказывается необходимость максимального использования максимального из еще неиспользованных уклонов, пока не будет исчерпан ресурс.

В итоге получаем следующий алгоритм спуска.

1. Начальной точке $z(z_1^1, z_2^1, ..., z_n)$ соответствует максимальное значение целевой функции

$$\sum_{i=1}^{n} w_i(f_i^1) = \sum_{i=1}^{n} g_i^1 .$$

Фиксируем остаток ресурса $T = R - \sum_{i=1}^{n} f_i^1$.

2. Из всех звеньев всех ломаных берем звено с максимальным уклоном. Пусть это будет звено с номером j ломаной с номером m. На первом шаге $j = 1$ в силу выпуклости и монотонности ломаных. Будем менять только переменную z_m. Ее увеличение дает уменьшение целевой функции с наибольшей скоростью. Вычисляем шаг движения $c = \min(z_s^{j+1} - z_m^j, T)$ и меняем z_m^j на $z_m^j + c$. Целевая функция уменьшится на $u_m^j c$, а оставшийся ресурс — на c.

3. Из всех оставшихся звеньев всех ломаных снова выбираем звено с максимальным уклоном и повторяем п. 2, пока оставшийся ресурс T не будет исчерпан.

Если при выборе звена с максимальным уклоном таких звеньев было несколько, то приоритет отдается звену с максимальной длиной, которое оставшийся ресурс позволяет использовать полностью ($z_s^{j+1} - z_s^j \leqslant T$). Если для полного использования ни одного из таких звеньев ресурса не достаточно, то используется любое из них.

В работе [4] предложено использовать алгоритм спуска из точки с максимальными затратами для построения границ для каждой точки на каждом шаге. Далее излагается усовершенствованный алгоритм.

Отметим, что в точке минимума только один уклон, используемый последним, может использоваться частично. Если же и он используется полностью, т. е. на последнем шаге $z_s^{j+1} - z_s^j = T$, то полученное решение непрерывной задачи совпадает с решением исходной дискретной задачи и является окончательным. В противном случае оно является начальным приближением, а значение целевой функции в точке минимума является искомой нижней границей H_0. Если последней рассматривалась ломаная, которая изначально была выпуклой ($k_s = p_s$), то приближенное решение исходной дискретной задачи и, соответственно, рекорд E получаем, аннулируя последний шаг (этим объясняется стремление сделать его меньше). В противном случае по оптимальному значению z_s^* определяется абсцисса f_s^k ближайшей слева вершины исходной s-й ломаной. Ординату этой вершины обозначим через g_s^k. Нижняя граница H_0 не изменяется, а рекорд $E = H_0 + g_s^k - w_s(z_s^*)$ (рис. 2).

Отметим, что выпуклые оболочки используются только для вычисления границ, а при формировании множества состояний на каждом шаге рассматриваются все допустимые состояния, т. е. исходные ломаные.

Для эффективной реализации алгоритма существенное значение имеет способ поиска наибольших уклонов. Простой способ состоит в сортировке всех уклонов всех звеньев ломаных. Но затраты оперативной памяти на сортировку могут быть излишними, так как существенная часть уклонов вообще может не потребоваться при расчете нижней границы и рекорда. Вместо этого упорядочиваются в порядке невозрастания только уклоны первых элементов ломаных. С каждым из них связывается номер ломаной и номер ее звена, начальное значение которого равно 1. При наличии равных уклонов приоритет отдается звену с наибольшей длиной. Полученный массив уклонов обозначим через $U(u_1, u_2, ..., u_n)$. В соответствии с изложенным алгоритмом процесс спуска начинается с использования u_1. Далее, если этот уклон используется полностью, он заменяется на уклон u_1' следующего

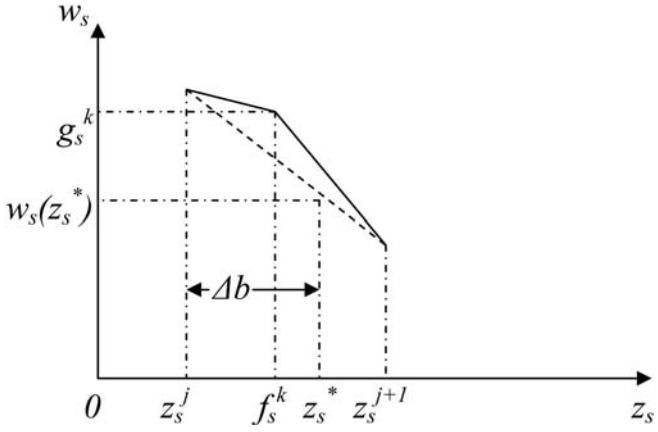


Рис. 2. Последний шаг алгоритма спуска при отсутствии выпуклости

звена соответствующей ему ломаной, иначе спуск закончен. На первом шаге $u_1 = u_m^1$ и $u_1' = u_m^2$, где $u_m^1 = \max u_i^1$ $(i = 1, ..., n)$. Уклон u_1' помещается в массив так, чтобы выполнялось условие невозрастания его элементов. Если $u_1' < u_2$, элементы массива большие, чем u_1', сдвигаются влево, так что на первом месте всегда наибольший уклон. Если для некоторой ломаной s все уклоны использованы, то вводится фиктивное звено с номером k_s и нулевым уклоном, и процесс продолжается до исчерпания ресурса. Запоминается итоговый массив уклонов, а также $\Delta b = z_s^* - z_s^j$, т. е. часть звена ломаной, которое использовалось последним при спуске.

Альтернативой алгоритму спуска является подъем из точки $(f_1^{k1}, f_2^{k2}, ..., f_n^{kn})$ с минимальным значением целевой функции $\sum_{i=1}^{n} w_i(f_i^{ki})$, которое на каждом шаге с использованием очередного минимального уклона увеличивается, а используемый ресурс уменьшается с $R_{\max}$ до R. Здесь $R_{\max} = \sum_{i=1}^{n} f_i^{ki}$.

В этом алгоритме в случае исчерпания всех уклонов (звеньев) некоторой ломаной вводится фиктивное звено с номером ноль, уклон которого равен большому числу.

В итоге получается и запоминается новый массив уклонов $V(v_1, ..., v_n)$, а также $\Delta b_1 = z_s^{j+1} - z_s^*$ — часть звена ломаной, которое использовалось последним при подъеме. Оба алгоритма дают одно и то же решение $(z_1^*, z_2^*, ..., z_n^*)$ и значение целевой функции (нижнюю границу) H_0, но массивы уклонов U и V разные. В массиве U — уклоны звеньев — "претендентов на дальнейший спуск", а в массиве V — на подъем. Каждая i-я ломаная представлена уклоном одного звена, но номер этого звена в массиве U на единицу больше, так как z_i^* равна абсциссе левого конца звена, представленного в массиве U, или правого конца в массиве V. Исключение составляют ломаные, представленные уклонами, оказавшимися в итоге на первом месте, u_1^* и v_1^*. Всегда $u_1^* = v_1^*$, с ними связано одно и то же j-е звено ломаной с номером s, уклон которого использовался последним. При этом справедливо равенство $z_s^{j+1} - z_s^j = \Delta b + \Delta b_1$.

Алгоритм подъема обосновывается аналогично изложенному выше алгоритму спуска. Он начинает работу с упорядоченных в порядке неубывания уклонов последних звеньев всех ломаных.

Алгоритмы спуска и подъема можно использовать и при вычислении границ затрат на оставшейся части траектории для каждой паретовской точки.

Отметим две особенности алгоритма спуска.

1. Получаемый в итоге массив из n уклонов и связанные с ним данные о номерах ломаных и их звеньев позволяют найти z^* и оптимальную траекторию для непрерывной задачи как последовательности значений использованного ресурса z_1^*, $z_1^* + z_2^*$, ..., $z_1^* + z_2^* + ... + z_n^*$ и соответствующих значений целевой функции.

2. Если вместо R имеем $R' > R$, то нет смысла начинать спуск заново, его можно продолжить из полученной для R оптимальной точки, используя итоговый массив и оставшиеся уклоны. Однако при $R' < R$ для корректировки полученного решения пришлось бы осуществлять спуск заново.

3. Если из итогового массива исключить все уклоны, относящиеся к первой ломаной, то полученному массиву будет соответствовать решение $(z_2^*, ..., z_n^*)$ задачи:

$$\min W(z) = \sum_{i=2}^{n} w_i(z_i),$$

где $z(z_2, ..., z_n)$ — вектор неизвестных, при ограничениях

$$\sum_{i=2}^{n} z_i \leqslant R - w_1(z_1^*),\ f_i^1 \leqslant z_i \leqslant f_i^{ki},\ i = 2, ..., n,$$

так как для решения этой задачи пришлось бы использовать максимальные уклоны из всех ломаных, кроме первой. Другими словами, полученный таким образом массив уклонов соответствует оптимальной траектории из получаемого после первого шага алгоритма динамического программирования состояния $(z_1^*, w_1(z_1^*))$ до конца.

Аналогично, переходя к очередному r-му шагу алгоритма динамического программирования и исключая из массива U уклоны u_r^j $(j = 1, ..., p_r - 1)$, получаем массив уклонов, который соответствует z_r^* и, следовательно, состоянию (F_r^*, G_r^*) на оптимальной траектории.

Аналогично можно использовать и алгоритм подъема.

Существенными недостатками изложенных алгоритмов спуска и подъема являются следующие:

- начальные точки, которые соответствуют максимальным и минимальным затратам, как правило, далеки от оптимума;
- информация об оптимальной траектории, полученной на первом шаге при решении непрерывной задачи, не используется;
- как правило, новые значения рекорда получаются для точек, расположенных вблизи точки на оптимальной траектории, определенной при решении непрерывной задачи, но эти алгоритмы находят новое значение рекорда уже после того, как оставлены точки, которые могли бы быть отбракованы при движении из точки на оптимальной траектории.

От этих недостатков свободен новый алгоритм расчета границ, который назовем "спуск — подъем". Приведем его основные пункты.

1. Исходная оценочная задача (3), (4) решается и методом спуска, и методом подъема. Запоминаются массивы U и V как эталонные, а также Δb и Δb_1.

С каждым элементом каждого массива связан номер ломаной и номер звена, которому он соответствует, что дает возможность восстановить точку оптимума z^*.

2. Запоминается оптимальная траектория (F_i^*, G_i^*) ($i = 1, ..., n$) и соответствующие ей границы E_0 и H_0. Это решение непрерывной оценочной задачи, и отклонение от оптимума исходной задачи не превышает $E_0 - H_0$.

3. Строится паретовское множество первого шага динамического программирования.

4. Из массивов U и V исключаются уклоны, соответствующие первой ломаной.

5. Начиная с состояния F_1^*, G_1^*, для состояний с $F_1^k < F_1^*$ осуществляется последовательно спуск, а для состояний с $F_1^k > F_1^*$ — подъем для определения нижней и верхней границ и возможной отбраковки состояния или корректировки рекорда.

6. На последующих шагах динамического программирования границы определяются аналогично.

Особым является s-й шаг алгоритма динамического программирования, где s — номер ломаной, которой принадлежит звено, использованное последним при построении оптимальной траектории задачи (3), (4). Его уклон использовался частично, поэтому при исключении этого уклона из эталонных массивов уклонов Δb и Δb_1 обнуляются.

Задачу (1), (2) на максимум можно преобразовать в эквивалентную ей задачу на минимум, взяв в качестве целевой функции вместо $\sum_{i=1}^{n} g_i(x_i)$ функцию $\sum_{i=1}^{n} (g_{i\max} - g_i(x_i))$, где $g_{i\max}$ — максимальная эффективность для i-го потребителя ресурса.

В вышеупомянутых частных случаях рассматриваемой задачи вычисления существенно упрощаются, так как при $f_i(x_i) = p_i x_i$ и $g_i(x_i) = c_i x_i$ для каждой ломаной все звенья имеют один уклон, а при $x_i = \{0, 1\}$ каждая ломаная состоит из одного звена.

Покажем, как работает алгоритм спуска, на примере рассмотренной выше задачи о загрузке машины после преобразования ее в задачу на минимум.

В этой задаче $f_1^1 = 0$ и $g_1^1 = 7$ (не брать первый предмет), $f_1^2 = 4$ и $g_1^2 = 0$ (взять первый предмет). Аналогично по остальным предметам. После упорядочивания уклонов по убыванию получаем массу и стоимость (указана в скобках): 4 (7); 20 (34); 16 (27); 12 (20); 7 (10); 11 (15). $R = 35$, $E_0 = 72$, $H_0 = 72 - 27/16 \cdot 11$, так как $\sum_{i=1}^{n} g_i^1 = 113$, а спуститься, не превышая ресурс, можно на $7 + 34 + 27/16 \cdot 11$.

После первого шага имеем два состояния $f_2^1 = 0$, $g_2^1 = 7$ и $f_2^2 = 4$, $g_2^2 = 0$, которые не меняют рекорд и нижнюю границу. После второго шага имеем четыре состояния: (0,41); (4,34); (20,7) и (24,0). В состоянии (0,41) для спуска остается $R = 35$, что дает возможность спуститься на $27 + 20 + 10 = 57$ и получить $E = H = 56$. Новый рекорд равен 56, и ни одно из трех других состояний не дает меньшего значения рекорда. Но нижняя граница для состояния (4,34) равна $34 + 72 - (27 + 20 + 3 \cdot 10/7)$, что меньше 56, как и для остальных состояний, поэтому процесс продолжается.

Состояние (0,41) не подвергается ветвлению, и после третьего шага имеем только три допустимых состояния: (4,61); (20,34) и (24,27). Для первого из них нижняя граница очевидно выше рекорда, а для остальных ниже. После четвертого шага имеем три состояния (20,54); (24,47) и (32,34), из которых только последнее дает нижнюю границу меньше 56 ($34 + 25 - 3 \cdot 10/7 < 56$). После пятого шага имеем одно состояние (32,44), из него есть только одно продолжение (32,59). В итоге рекорд равен 56, что соответствует значению 57 для исходной задачи на максимум и набору предметов в 16, 12 и 7 единиц веса. Всего рассматривалось только 14 состояний, тогда как традиционный алгоритм требует рассматривать по 36 состояний на каждом шаге, а алгоритм с отбраковкой только доминируемых (непаретовских) точек требует запоминания 42 состояний суммарно.

Поскольку алгоритм спуска-подъема требует значительного объема вычислений и сокращение числа состояний не гарантирует сокращения общего времени счета, для выявления эффективности нового алгоритма в сравнении с более простыми алгоритмами были выполнены экспериментальные расчеты.

Экспериментальные расчеты

Для сравнительной оценки различные алгоритмы были реализованы в следующих компьютерных программах.

П1. Динамическое программирование без разбиения регулярной сетки состояний.

П2. Отбраковка только непаретовских состояний.

П3. Дополнительная отбраковка части бесперспективных паретовских состояний с использованием начальных приближений.

П4. Дополнительная отбраковка бесперспективных паретовских состояний с вычислением нижней и верхней границ оптимума по алгоритму спуска-подъема.

Расчеты выполнялись на персональном компьютере Intel Pentium 4, CPU 3.0 GHz, 512 Мбайт ОЗУ.

Во всех расчетах абсциссы и ординаты ломаных — это псевдослучайные вещественные числа из [1,100], но число вершин $K_i = K$ не зависит от i.

Время счета зависит не только от числа шагов n, значения K и от заданного значения ресурса R, но и от конкретных значений f_i^j и g_i^j.

В первом расчете задавались малые значения $n = 50$ и $K = 10$. Результаты представлены в табл. 2. Обозначения: *sum* — суммарное по всем шагам

Таблица 2

	$R = 1000$			$R = 2000$			$R = 3000$			$R = 4000$		
N	sum	max	T	sum	max	T	sum	max	T	sum	max	T
П1	4913186	125963	1485	8573228	285081	2499	9774235	325130	2767			> 3600
П2	749454	36890	49	1205318	63635	63	1426221	89532	67	1452752	93875	67
П3	639420	31212	36	752111	23399	23	690340	29526	14	528481	19492	8
П4	9786	436	0	1705	150	0	1748	117	0	78	8	0

число запоминаемых состояний; *max* — максимальное число состояний на отдельном шаге; T — время счета, округленное до целого числа секунд.

Расчеты по П1 выполнялись при дополнительном условии: состояния на каждом шаге считаются совпадающими, если они отличаются (по ресурсу) менее чем на заданное число d.

Попытка получить результат с $d = 0{,}001$ была неудачной ввиду чрезмерных затрат машинного времени. В табл. 2 представлен результат, полученный при $d = 0{,}005$. При $d = 0{,}01$ время счета существенно меньше, но точность расчета может оказаться недостаточной, так как по целевой функции отклонения превышали 0,1.

Расчеты по другим программам выполнялись без этого дополнительного условия, но при сравнении вещественных чисел использовалась константа 10^{-9}.

Время счета по П4 было меньше 0,1 с, а при $R = 4000$ счет завершился на 30-м шаге. Характерно уменьшение времени счета по П3 с ростом R. В данном расчете равномерное распределение ресурса между потребителями оказывается ближе к оптимуму с ростом значения распределяемого ресурса, так как максимальные потребности в ресурсе у потребителей различаются незначительно.

В дальнейших расчетах программа П1 не использовалась ввиду бесперспективности алгоритма для рассматриваемого класса задач. Тем более бесперспективен классический алгоритм динамического программирования (метод регулярной сетки) при шаге сетки d, равном значению, использованному при работе с П1.

Существенное влияние на рост времени счета оказывает рост K. Так, при $n = 40$ и $K = 20$ уже при $R = 2500$ и использовании П2 sum = 2748038, max = 200040, $T = 809$ с, а при использовании П3 sum = 1433853, max = 81889, $T = 241$ с. При увеличении R число остающихся паретовских точек становится неприемлемо большим. Возникает та же ситуация, что и с алгоритмом П1: оперативная память исчерпана, а обмен с подключаемой внешней памятью медленный. Характерно, что в этом же расчете по П4 при $R = 2500$ sum = 2886, max = 152, $T < 0{,}5$ с, но при $R = 1000$, П4 дает sum = 6855, max = 500 и $T = 1{,}2$ с.

Аналогичные результаты были получены при тех же условиях, но с $n = 100$ и $K = 40$. Решение по П2 и П3 в приемлемое время удалось получить только при задании $d = 0{,}002$ и более. Так, при $R = 2000$, $d = 0{,}005$ П3 дает sum = 7367886, max = 98087 и $T = 3208$ с. А при использовании в этом расчете П4 увеличение R давало как увеличение, так и уменьшение числа оставшихся после отбраковки точек и, соответственно, времени счета. Результаты представлены в табл. 3.

Характерно, что в этом расчете малое изменение R существенно повлияло на время счета. Это видно из табл. 4.

Полученные результаты объясняются тем, что при $R = 2077$ начальное значение рекорда отличается от оптимума на 0,29, а при $R = 2078$ на 0,34. Как следует из приведенного выше алгоритма, с увеличением R значение уклона u_s^j, который при вычислении начальных значений нижней границы H_0 и рекорда E_0 используется последним, может только уменьшаться, что дает меньшее значение $E_0 - H_0$, если $z_s^* - z_s^j$ постоянно, так как $E_0 - H_0 = u_s^j(z_s^* - z_s^j)$. Здесь z_s^* — абсцисса точки, полученной при спуске по звену ломаной с номером s при вычислении H_0, а z_s^j — абсцисса ближайшей слева вершины этой ломаной, использованная при вычислении E_0, т. е. при "округлении" до решения дискретной задачи. Но уменьшение u_s^j может компенсироваться увеличением разности $z_s^* - z_s^j$, которая может как возрастать, так и убывать с ростом R, что зависит от исходных данных $(f_i^j,\ g_i^j)$.

Таблица 3

	$R = 2000$			$R = 2100$			$R = 2500$			$R = 4000$		
N	sum	max	T	sum	max	T	sum	max	T	sum	max	T
П4	75663	2192	91	98270	3856	153	17191	420	11	16479	736	6

Таблица 4

	$R = 2077$			$R = 2078$			$R = 2079$			$R = 2080$		
N	sum	max	T	sum	max	T	sum	max	T	sum	max	T
П4	20177	475	12	99514	2274	179	44056	920	49	54973	1399	74

Для выявления возможностей использования П4 для решения задач большой размерности были выполнены расчеты при $K = 20$ и $n = 100, 200, 300, 400, 500$. Установлено, что число запоминаемых состояний и, соответственно, время счета существенно зависят от R. Так, при $n = 400$ и $R = 28000$ sum = 108893, max = 1099 и $T = 2$ с. А при существенно меньшем $R = 2000$ в тех же условиях sum = 544554, max = 3335 и $T = 1189$ с. При $n = 500$ и $R = 35000$ sum = 22475, max = 1740 и $T = 12$ с. Вообще увеличение n и K не означает обязательный рост времени счета, так как при "удачном" задании R, т. е. при малом $z_s^* - z_s^j$, это время может быть мало и в задачах большой размерности. При любых исходных данных существует R, при котором начальное приближение оказывается окончательным решением ($z_s^* - z_s^j = 0$). Для этого достаточно повторить расчет, уменьшив R на $z_s^* - z_s^j$ или увеличив на $z_s^{j+1} - z_s^*$.

Если при использовании П4 ограничиться приближенными решениями, например вместо $d = 10^{-9}$ использовать $d = 10^{-5}$, то в этих же условиях число состояний и время счета существенно снижаются. Однако более перспективным оказывается не увеличение d, а прекращение счета при выполнении условия $(E - H)/H < \varepsilon$.

При $\varepsilon = 10^{-5}$ ни в одном из расчетов, о которых шла речь выше, время счета по П4 не превышало 0,5 с. В этой связи были решены задачи с $n = 4000$ и $K = 40$, а затем с $n = 5000$ и $K = 50$ в тех же условиях выбора f_i^j, g_i^j. При $R = 10^5$, $2 \cdot 10^5$ и $3 \cdot 10^5$ и $\varepsilon = 10^{-5}$ счет прекращался после третьего шага, а время счета не превышало 8 с. Аналогичные результаты были получены и при случайном выборе f_i^j из [1, 1000], сохранении всех прочих параметров расчета и при различных значениях R.

Характерно, что во всех решенных задачах алгоритм спуска-подъема давал существенное сокращение времени счета по сравнению с алгоритмом спуска. Более того, в некоторых задачах алгоритм спуска вследствие больших затрат времени на вычисление границ оказывался менее эффективным, чем простой алгоритм П3, использующий начальные приближения (только верхнюю границу затрат, получаемую при равномерном распределении оставшегося ресурса).

Отметим, что при реализации комбинированного алгоритма, в отличие от рекомендаций [4], не рассматриваются и не решаются уравнения Р. Беллмана, а весь процесс можно трактовать как пошаговое построение дерева состояний с отсечением недопустимых (по ресурсу) вершин, доминируемых вершин и дополнительным отсечением части недоминируемых вершин при вычислении верхней и нижних границ оптимума для каждой из уже построенных вершин. Это вычисление с использованием алгоритма спуска-подъема существенно повышает быстродействие аналогичного комбинированного алгоритма, предложенного в работе [4], в котором для этой цели используется алгоритм спуска.

Заключение

В итоге можно констатировать следующее.

1. Для решения задач рассматриваемого класса классические алгоритмы динамического программирования можно считать устаревшими.

2. Наиболее перспективным из рассмотренных является комбинированный алгоритм (П4).

3. При использовании комбинированного алгоритма целесообразно искать приближенные решения, прекращая счет при малости относительной ошибки поиска минимума ε. Для этого можно задать ее приемлемое значение, но вместо этого для решения задач большой размерности можно задать заведомо малое ε, вывести на экран пошаговые значения E, H и $(E - H)/H$ и прекращать счет с учетом текущих результатов и затраченного времени.

4. При росте R и n возможно как увеличение, так и уменьшение времени счета. Фактически алгоритм П4 если и не преодолевает полностью "проклятие размерности", то делает его действие избирательным.

Задача (1), (2) рассмотрена как пример, но алгоритмы, использующие множества Парето, не универсальные, как и динамическое программирование в целом, применимы и для решения других задач: различного вида двухпараметрических задач распределения ресурсов, управления запасами, расчета планов замены оборудования, выбора поставщиков. Алгоритм спуска-подъема можно использовать вместо алгоритма спуска в различных схемах, предложенных в работе [4] для решения задачи (1), (2) при наличии нескольких ограничений вида (2). Рассмотрение этих задач выходит за рамки настоящей статьи.

Список литературы

1. **Беллман Р., Дрейфус С.** Прикладные задачи динамического программирования. М.: Наука, 1965. 458 с.

2. **Сигал И. Х., Иванова А. П.** Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы: учеб. пособие. Изд. 2-е, испр. М.: ФИЗМАТЛИТ, 2003. 240 с.

3. **Струченков В. И.** Методы оптимизации в прикладных задачах. М.: Солон-Пресс, 2009. 310 с.

4. **Алексеев О. Г.** Комплексное применение методов дискретной оптимизации. М: Наука, 1987. 248 с.

5. **Вентцель Е. С.** Исследование операций: задачи, принципы, методология. М.: КноРус, 2010. 208 с.

6. **Косоруков О. А., Мищенко А. В.** Исследование операций: учебник для студентов вузов, обучающихся по специальности "Математические методы в экономике". М.: Экзамен, 2003. 270 с.

7. **Корбут А. А., Финкельштейн Ю. Ю**. Дискретное программирование. М.: Наука, 1969. 369 с.

8. **Martello S., Toth P.** Knapsack problems: algorithms and Computer implementation. DEIS, University of Bologna. Chichester, England: John Willey & Sons Ltd. 1990.

9. **Struchenkov V. I.** Dynamic Programming with Pareto Sets // Journal of Applied and Industrial Mathematics. Springer. 2010. Vol 4, N 3.

10. **Struchenkov V. I.** Combined Algorithms of Optimal Resource Allocation // Applied Mathematics. Scientific Research Published, USA. 2012. Vol. 3, N 1.

УДК 004.383.3

П. В. Скрибцов, канд. техн. наук, ген. директор,
А. В. Долгополов, вед. программист,
e-mail: avdol@udm.net
ООО "Павлин Технологии",
www.pawlin.ru

Применение GPU для решения задач математического моделирования в области гидродинамики и гидрологии

Анализируется мировой опыт использования многоядерных графических процессоров (GPU) для ускорения расчетов по базовым уравнениям гидродинамики, таким как уравнения Навье—Стокса и Сен-Венана, а также по методу сглаженных частиц. Ускорение расчетов с применением GPU измеряется относительно современных центральных процессоров (CPU). Рассматриваются причины получаемого увеличения быстродействия.

Ключевые слова: *гидрология, гидродинамика, графические процессоры, Сен-Венан, Навье—Стокс*

Введение

Необходимость средств обработки больших объемов информации в режиме реального времени привела к тому, что графические процессорные устройства (GPU) превратились в высокопараллельные, многопоточные, многоядерные процессоры с огромной вычислительной мощностью и очень высокой пропускной способностью памяти.

В отличие от современных универсальных центральных процессоров (CPU) GPU предназначены для параллельных вычислений с большим числом арифметических операций. Значительно большее число транзисторов GPU работает по прямому назначению — обработке массивов данных, а не управляет исполнением (*flow control*) немногочисленных последовательных вычислительных потоков. Этим объясняется то, что теоретическая производительность GPU (видеочипов) значительно превосходит производительность CPU.

Компания NVIDIA приводит графики роста производительности и пропускной способности памяти CPU и GPU за последние несколько лет (рис. 1), по которым видно, что в настоящее время рост производительности CPU ограничен.

Современные GPU являются мощными специализированными вычислительными устройствами, которые могут быть использованы для ускорения решения целого ряда вычислительно сложных задач. Помимо высокой производительности, GPU имеют ряд других преимуществ: низкое энергопотребление, низкая стоимость, высокая доступность. Причина такой производительности видеочипов заключается в том, что они изначально были сконструированы для высокопроизводительных параллельных вычислений.

Достаточно простым и удобным инструментом для разработки приложений, использующих вычислительные мощности GPU, является технология NVIDIA CUDA [19].

Вычислительные задачи, реализованные на CUDA, получают ускорение в десятки и сотни раз в различных областях.

Потенциальный недостаток использования GPU для расчета — ограниченный объем памяти на видеокарте. Например, C1060 имеет 4 Гбайт памяти на 240 процессоров (~16 Мбайт/процессор), что ограничивает круг задач, для которых могут быть использованы GPU.

В последней версии CUDA 4.0 появилась новая особенность — UVA (*Unified Virtual Addressing* — единая виртуальная адресация) [24], которая по-

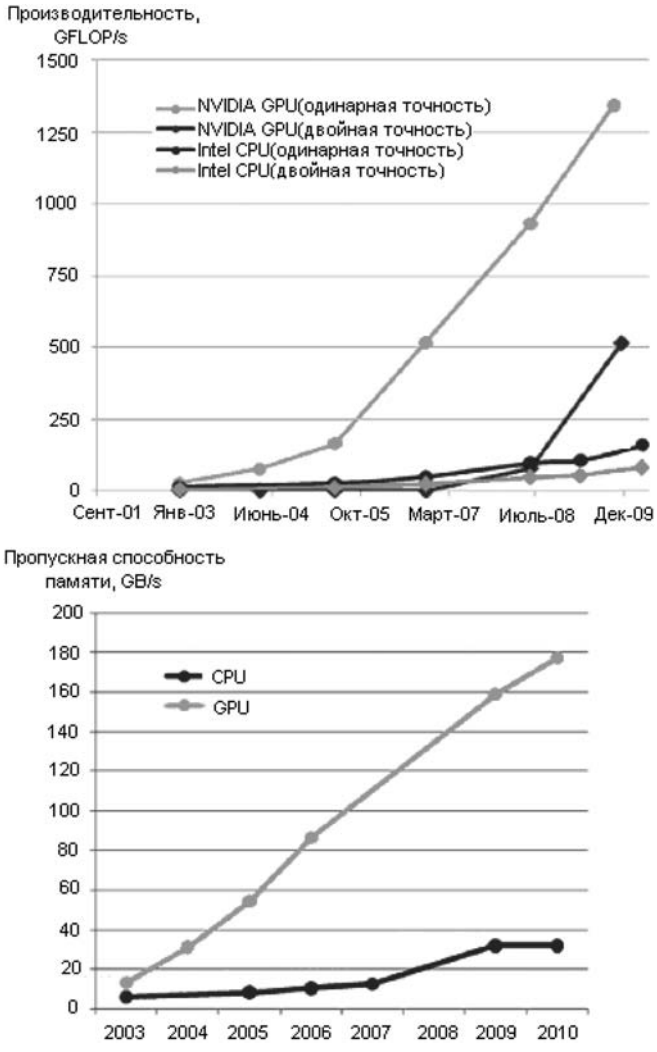


Рис. 1. Динамика роста производительности для CPU и GPU

зволяет лучше использовать большое число GPU и строить высокопроизводительные системы с большим объемом памяти (рис. 2, см. вторую сторону обложки).

Ниже проводится обзор моделей гидродинамики/гидрологии, расчет которых был эффективно ускорен на GPU.

Многие процессы течения жидкости в открытых каналах могут быть описаны на основе уравнения Бернулли. К ним относятся: протекание жидкости через насадки; равномерное движение жидкости в открытых руслах и безнапорных трубах; течение жидкости через различные водосливы; течение жидкости при наличии гидравлических прыжков и др. Но уже при рассмотрении более сложных процессов, например неустановившегося течения жидкости, необходимо применять более сложные уравнения. Наиболее общая математическая модель основывается на уравнениях Навье—Стокса [18].

Для моделирования гидрологических процессов в реках чаще используют уравнения Сен-Венана [6, 7], которые выводятся из уравнений Навье—Стокса.

Другой известный метод, используемый для моделирования поведения жидкости, — метод SPH (*smoothed particle hydrodynamics — гидродинамика сглаженных частиц*).

Так как при моделировании гидродинамических процессов приходится сталкиваться с решением систем дифференциальных уравнений, то целесообразно упомянуть работы, в которых GPU используются для ускорения методов их решения на основе конечных разностных схем, при этом достигаются ускорения в десятки и сотни раз относительно CPU [21, 22, 23].

Уравнения Навье—Стокса

Уравнения Навье—Стокса являются одними из важнейших в гидродинамике и применяются в математическом моделировании многих природных явлений и технических задач.

Система уравнений Навье—Стокса состоит из двух уравнений:

- уравнения движения,
- уравнения неразрывности.

В векторном виде для несжимаемой жидкости они записываются следующим образом:

$$\frac{\partial v}{\partial t} = -(\mathbf{v}\nabla)\mathbf{v} + \nu\Delta\mathbf{v} - \frac{1}{\rho}\nabla p + f;\ \nabla\mathbf{v} = 0, \qquad (1)$$

где ∇ — оператор Гамильтона; Δ — оператор Лапласа; t — время; ν — коэффициент кинематической вязкости; ρ — плотность; p — давление; u — векторное поле скоростей; f — векторное поле массовых сил. Неизвестные ρ и $\mathbf{v}$ являются функциями времени t и координаты $x \in \Omega$, где $\Omega \subset R^n$, $n = 2,3$ — плоская или трехмерная область, в которой движется жидкость.

В работе [1] исследовалась методика моделирования движения вязкой несжимаемой жидкости, описываемой уравнением Навье—Стокса в переменных "функция тока — вихрь", на графических ускорителях NVIDIA с использованием технологии CUDA.

Для нахождения решения уравнения вычислялся набор трехдиагональных систем. Трехдиагональные системы, в последовательном варианте обычно решаемые с использованием алгоритма прогонки, вычисляются с помощью параллельной циклической редукции, так как данный алгоритм эффективно может быть реализован для массивно-параллельных систем с общей памятью [3, 4].

В работе [5] авторам удалось достичь высокого коэффициента ускорения вычислений на GPU относительно CPU при решении уравнений Навье—Стокса для несжимаемой жидкости.

В данном исследовании 3D-область течения $NX \times NY \times NZ$ представляется в виде 2D-матрицы шириной NX и высотой $NY \times NZ$ на стороне CPU, как показано на рис. 3. На GPU используется то же представление данных, что и на CPU. Используется несколько матриц, необходимых для представления давления и компонент скорости на разных уровнях времени.

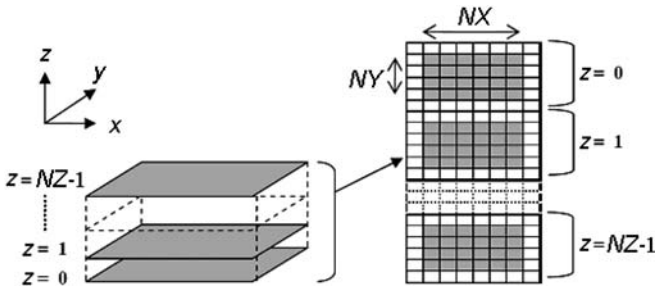


Рис. 3. Отображение 3D-области в 2D-матрицу

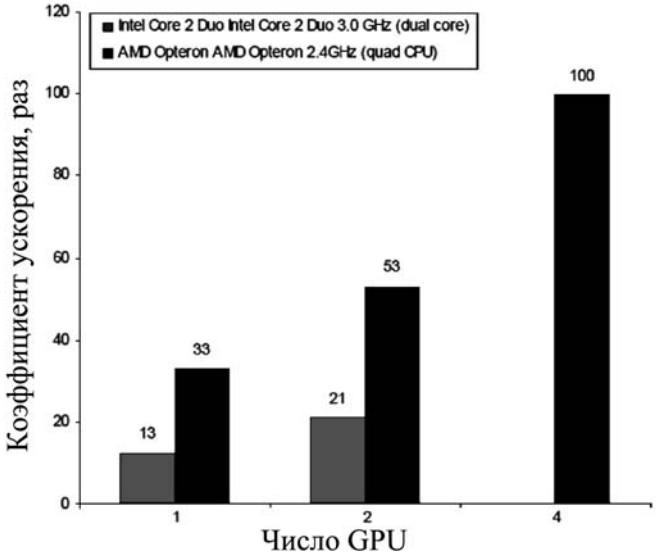


Рис. 4. Коэффициент ускорения GPU относительно CPU на различных аппаратных платформах для сетки 1024 × 32 × 1024

Для тестирования использовались следующие аппаратные платформы (рис. 4):

1) Intel Core 2 Duo 3.GHz (E8400) CPU, ОЗУ 4GB, два ускорителя Tesla C870;

2) восемь процессоров AMD Opteron 2.4GHz (8216) dual-core CPUs, четыре ускорителя Tesla.

Видно, что для платформы с AMD коэффициент ускорения выше, чем для Intel. Это связано с тем, что производительность процессора AMD Opteron 2.4GHz (8216) на данной задаче ниже, чем производительность Intel Core 2 Duo 3.GHz(E8400). При увеличении числа GPU происходит практически прямо пропорциональный рост производительности расчетов.

Уравнения Сен-Венана

Для моделирования одномерного течения жидкости в открытых каналах обычно используются уравнения Сен-Венана [6, 7]. Они описывают постепенно меняющееся течение жидкости в каналах. Уравнения Сен-Венана могут быть выведены из уравнений Навье—Стокса [20].

При выводе этих уравнений делается ряд предположений:

1) длина водотока намного больше его глубины и ширины;

2) русло вытянуто в прямую линию, т. е. центробежные силы отсутствуют;

3) давление внутри потока подчинено гидростатическому закону;

4) зеркало воды в поперечном сечении горизонтально;

5) поток плавно изменяющийся, докритический;

6) уклон дна $I(x)$ мал, так что $I = I(x) = \mathrm{tg}x$;

7) расход $Q(x, t)$ и глубина $z(x, t)$ осреднены по ширине и глубине потока, причем $Q = VW$, где $V(x, t)$ — скорость течения; W — площадь поперечного сечения потока.

Наиболее часто уравнения Сен-Венана используют в следующей форме [13]:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0; \qquad (2)$$

$$\frac{\partial Q}{\partial t} + \alpha \frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right) + gA\frac{\partial h}{\partial x} + gAS_0 - gAS_f = 0. \qquad (3)$$

Здесь A — площадь поперечного сечения канала; $Q = UA$ — расход жидкости; U — осредненная по сечению скорость потока; g — ускорение силы тяжести; h — глубина потока; α — коэффициент, учитывающий неравномерность распределения скорости по сечению потока. В уравнении (3) третий член учитывает действие сил давления, четвертый — вклад гравитационных сил (S_0 представляет собой наклон дна). Влияние сил трения описывается пятым членом, который выражен по аналогии с четвертым членом в виде наклона S_f [6]. В установившихся потоках $S_0 = S_f$. Основные формы уравнений Сен-Венана, которые наиболее часто используются для практических расчетов, рассмотрены в работе [14].

В работе [12] реализован программный пакет FrameWork для симуляции двумерных потоков с использованием технологии CUDA, основанный на реализации выражений Сен-Венана.

Расчетная область разбивается на множество блоков. 2D-массив разделяемой памяти выделяется для каждого блока и имеет дополнительный слой клеток в обоих направлениях для учета ореолов (рис. 5, см. вторую сторону обложки). Сначала заполняются расширенные участки сверху и снизу, затем слева и справа. После этого заполняется центральная часть ячеек 2D-массива [17].

В конце вычислительной итерации обновленные значения ячейки записываются обратно в глобальную память.

Обновление дополнительных ячеек на краях расчетной области осуществляется в два этапа. Первый этап включает в себя обновление двух строк, а следующий этап — обновление двух столбцов (или наоборот). Два этапа выполнены в виде двух разных вызовов ядра, и размеры блоков для ядер равны размеру дополнительной строки/столбца (рис. 6).

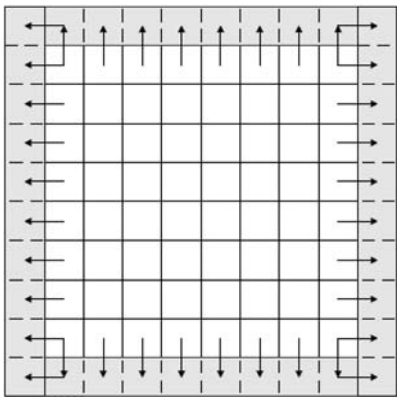


Рис. 6. Обновление дополнительных строк/столбцов

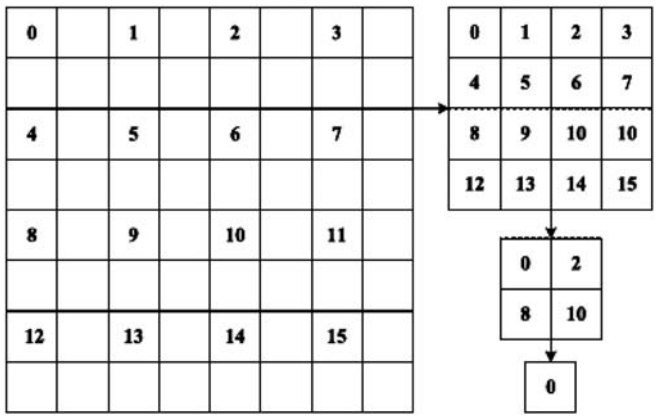


Рис. 7. Параллельная редукция с блоком размера 2 × 2

Как только значение ячеек обновится, выполняется ядро параллельной редукции [16], которое определяет максимальные значения высоты и скорости. Максимальные значения используются в условии Куранта [15] в качестве входных данных, чтобы найти следующий стабильный прирост времени Δt. Размер расчетной области уменьшается на каждой итерации, пока она не достигнет размеров 1×1 (рис. 7).

Проверка проводилась с помощью видеокарт GeForce 8400 GS и Tesla T10. Модель была протестирована на двух примерах. Ускорение GPU составило от 3 до 6 раз на GeForce 8400GS и от 50 до 135 раз на Tesla T10. Время симуляции наводнения при прорыве плотины снизилось с 2,9 ч до 2 мин (в 87 раз).

Отметим, что на данный момент пакет FrameWork в свободном доступе отсутствует.

Гидродинамика сглаженных частиц

Гидродинамика сглаженных частиц (SPH) — вычислительный метод для симуляции жидкостей и газов. Он используется во многих областях исследований, включая астрофизику, баллистику, вулканологию и океанографию. Метод SPH является несеточным (*mesh-free*) лагранжевым методом (то есть координаты движутся вместе с жидкостью). Разрешающая способность метода относительно переменных, таких как плотность, может быть легко отрегулирована.

По сути, SPH — это метод численного решения системы уравнений Навье—Стокса, описывающих движение жидкости. Основной идеей метода SPH можно назвать положение, что каждая частица в некоторой степени "заимствует" физические характеристики у своих ближайших соседей.

При использовании метода SPH жидкость делится на дискретные элементы, называемые частицами. Эти частицы имеют пространственное расстояние (известное как "длина сглаживания", обычно представляемая в уравнениях как h), на котором их свойства "сглаживаются" *функцией ядра*. Это значит, что любая физическая величина любой частицы может быть получена путем суммирования значений соответствующих величин всех частиц, которые находятся в пределах двух сглаженных длин. Например, температура в точке $\boldsymbol{r}$ зависит от температуры всех частиц на расстоянии $2h$ от $\boldsymbol{r}$.

Влияние каждой частицы на свойства оценивается в соответствии с ее плотностью и расстоянием до интересующей частицы. Математически это описывается функцией ядра (обозначается W). В качестве функции ядра часто используют функцию Гаусса (функция нормального распределения) или кубический сплайн. Последняя функция равна нулю для частиц, находящихся дальше, чем две сглаженные длины (в отличие от функции Гаусса, где имеется небольшое влияние на любом конечном расстоянии). Это позволяет экономить вычислительные ресурсы, исключая относительно малое влияние отдаленных частиц.

Значение любой физической величины A в точке $\boldsymbol{r}$ задается формулой

$$A(\boldsymbol{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(|\boldsymbol{r} - \boldsymbol{r}_j|, h). \qquad (4)$$

Гидродинамика сглаженных частиц все чаще используется для моделирования движения жидкостей. Это происходит вследствие некоторых преимуществ метода SPH по сравнению с традиционными методами, основанными на сетке. Благодаря SPH можно моделировать движение жидкости в режиме реального времени. Однако и SPH, и основанные на сетке методы все еще нуждаются в визуализируемой свободной поверхностной геометрии и используют полигонизационные методики, такие как *metaballs*, *marching cubes*, *point splatting* или "ковровую" визуализацию ("*carpet*" *visualization*).

Единственный недостаток SPH по сравнению с основанными на сетке методиками состоит в том, что необходимо большое число частиц для создания симуляции с эквивалентной разрешающей способностью. В типичной реализации основанных на сетке методов и SPH много вокселей или частиц будут находиться под поверхностью воды, в глубине водяного объема, и никогда не будут визуализированы. Однако точность может быть значительно увеличена со сложными, основанными на сетке методами, особенно с теми, которые используются совместно с методами частиц (такими, как наборы уровней частиц).

В работе [8] авторам удалось достичь ускорения SPH на GPU на 1—2 порядка по сравнению с CPU. Для реализации использовалась технология CUDA. Сам расчет состоит из трех основных этапов, которые хорошо ускоряются с помощью GPU: построение списка соседей, расчет сил и интегрирование уравнения движения. На рис. 8 (см. вторую сторону обложки) показан пример моделирования прорыва плотины с препятствием, для моделирования использовался ускоритель TESLA C1060.

В работе [9] была разработана библиотека для расчета SPH. На видеокарте NVIDIA GeForce 260GTX производительность системы составляет 38 кадр/с с 256000 частиц, 63 кадр/с с 128000 частиц и 99 кадр/с с 64000 частиц.

Для сложных моделей SPH использовались следующие шаги.

1. Обновление хэша, radix-сортировка.
2. Расчет SPH плотности.
3. Расчет SPH тензора скорости.
4. Расчет SPH давления и SPH тензора напряжений.
5. Интеграция во времени.

На рис. 9 представлены графики зависимости производительности от числа частиц в модели на различных видеоустройствах.

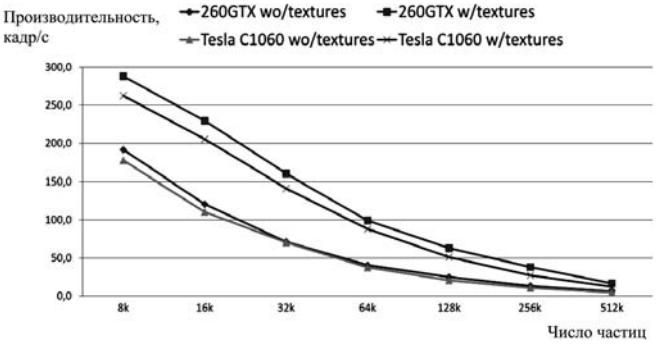


Рис. 9. Зависимость производительности от числа частиц в модели на различных видеоустройствах

Для сравнения в других работах достигнутый уровень производительности ниже, чем при использовании CUDA. На процессоре Intel Pentium 4 1,8 ГГц производительность составляла 20 кадр/с для 2200 частиц [11]. На видеокарте NVIDIA GeForce 8800GTX с использованием OpenGL и CG была достигнута производительность 17 кадр/с для 60000 частиц [10].

Заключение

Из проведенного анализа следует, что в представленных выше моделях благодаря распараллеливанию алгоритмов гидродинамики достигается ускорение на GPU относительно CPU.

Таким образом, при решении задач гидродинамики целесообразно использование видеокарты мощностью в сотни потоковых процессоров. Например, карта Tesla C1060 от NVIDIA содержит 240 потоковых процессоров, Tesla C 2050 содержит 480 процессоров. Используя для расчета GPU такой мощности, можно решать вычислительные задачи, расчет которых на CPU был затруднен ввиду длительного времени обработки.

Работа выполнена при поддержке Министерства образования и науки РФ, ГК № 16.515.11.5004 от 29 апреля 2011 г.

Список литературы

1. **Ясинский Ф. Н., Евсеев А. В.** О решении уравнения Навье—Стокса в переменных "функция тока — вихрь" на многопроцессорной вычислительной машине с использованием системы CUDA // Вестник ИГЭУ. 2010. Вып. 3.

2. **Lai C., Baltzer R. A., Schafranek R. W.** Conservation form equations of unsteady open-channel flow // J. Hydraulic Res. 2001. Vol. 40. P. 321—330.

3. **Zhang Y., Cohen J., Owens J. D.** Fast tridiagonal solvers on the GPU // Principles and Practice of Parallel Programming. 2010. P. 127—136.

4. **Sakharnykh N.** Tridiagonal solvers on the GPU and applications to fluid simulation // Proc. of GPU Technology Conference (GTC). 2009.

5. **Thibault J., Senocak I.** CUDA Implementation of a Navier-Stokes Solver on Multi-GPU Desktop Platforms for Incompressible Flows // Proc. of 47th AIAA Aerospace Sciences Meeting, Orlando, FL. 2009.

6. **Чоу В. Т.** Гидравлика открытых каналов. М.: Изд-во лит-ры по строительству, 1969. 464 с.

7. **Гришанин К. В.** Динамика русловых потоков. Л.: Гидрометеоиздат, 1979. 312 с.

8. **Herault A., Bilotta G., Dalrymple R.** SPH on GPU with CUDA // Journal of Hydraulic Research. 2010. Vol. 48. Extra iss. P. 74—79.

9. **Krog O. E., Elster A. C.** Fast GPU-based Fluid Simulations Using SPH. // Para 2010 — State of the Art in Scientific and Parallel Computing. University of Iceland, Reykjavik. June 6—9. 2010. URL: http://vefir.hi.is/para10/extab/para10-paper-139.pdf

10. **Harada T., Koshizuka S., Kawaguchi Y.** Smoothed particle hydrodynamics on GPUs. 2007.

11. **Müller M., Charypar D., Gross M.** Particle-based fluid simulation for interactive applications. In SCA '03 // Proc. of the ACM SIGGRAPH / Eurographics symposium on Computer animation. 2003. Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. P. 154—159.

12. **Shankar S., Kalyanapu A., Hansen C., Burian S.** A GPU-based Flood Simulation Framework // Proc. of Symposium on Application Accelerators in High Performance Computing. 2010. URL: http://saahpc.ncsa.illinois.edu/10/papers/paper_9.pdf

13. **Sanders B. F.** High-resolution and non-oscillatory solution of the St. Venant equations in nonrectangular and non-prismatic channels // J. Hydraulic Res. 2001. Vol. 39. P. 321—330.

14. **Lai C., Baltzer R. A., Schafranek R. W.** Conservationform equations of unsteady open-channel flow // J. Hydraulic Res. 2001. Vol. 40. P. 321—330.

15. **Liggett J. A., Woolhiser D. A.** Difference solutions of the shallowwater equations // Journal of the Engineering Mechanics Division. 1967. Vol. ASCE 93, no. EM2. P. 39—71.

16. **Harris M.** Optimizing parallel reduction in cuda. 2007 // URL: http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/reduction/doc/reduction.pdf

17. **Micikevicius P.** 3d finite difference computation on gpus using cuda // Proc. of 2nd Workshop on General Purpose Processing on Graphics Processing Units, GPGPU-2,Washington, D.C. US, 2009. P. 79—84.

18. **Ландау Л. Д., Лифшиц Е. М.** Гидродинамика. М.: Наука, 1988. 736 с.

19. **http**://www.nvidia.ru/object/what_is_cuda_new_ru.html

20. **Абдураимов М. Г., Музафаров Х. А., Путтиев А. А.** Движение вод в открытых руслах (уравнения Сен-Венана) // Матем. моделирование. 1998. Т. 10, № 6. С. 97—106.

21. **Egloff D.** High Performance Finite Difference PDE Solvers on GPUs // Social Science Research Network. 2010. P. 1—28.

22. **Матвеева Н. О.** Распараллеливание решения методом конечных разностей эллиптического дифференциального уравнения в частных производных на графическом процессоре. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей Всерос. научно-техн. конф. — Пенза: ПДЗ, 2008. С. 86—89.

23. **Cohen J.** OpenCurrent: Solving PDEs on Structured Grids with CUDA / URL: http: //www.nvidia.com/content/GTC-2010/pdfs/2022_GTC_2010.pdf

24. **NVIDIA CUDA** Toolkit 4.0 Overview // URL: http: //developer.download.nvidia.com/compute/cuda/4_0/CUDA_Toolkit_4.0_Overview.pdf

УДК 519.863

Н. И. Юсупова, д-р техн. наук, проф.,
А. Ф. Валеева, д-р техн. наук, проф.,
Р. И. Файзрахманов, канд. техн. наук,
Уфимский государственный авиационный технический университет,
e-mail: aida_val2004@mail.ru

Вероятностный алгоритм муравьиной колонии для решения задач раскроя промышленных материалов на заготовки различных геометрических форм

Рассматриваются задачи раскроя промышленных материалов на заготовки различных геометрических форм при наличии технологических ограничений (гильотинность реза, направление волокон материала, обход дефектных областей материала и др.), ориентированные на единичное производство. Подобные задачи раскроя относятся к классу NP-трудных, что означает отсутствие в настоящее время алгоритмов полиномиальной сложности, находящих решения за приемлемое на практике время. В связи с этим в статье предлагается вероятностный алгоритм муравьиной колонии, позволяющий получать приближенное решение поставленных задач. Проведены численные эксперименты на случайно сгенерированных и известных тестовых примерах, которые показали эффективность разработанного алгоритма, а также практические примеры.

***Ключевые слова**: задачи раскроя промышленных материалов, заготовки круглой и прямоугольной формы, алгоритм муравьиной колонии, популяция*

Введение

Задачи раскроя промышленных материалов на заготовки различных форм, ориентированные на единичное производство, возникают при изготовлении разнообразной продукции на заказ. В статье рассматриваются задачи раскроя листового и рулонного материала на заготовки круглой и прямоугольной формы при наличии технологических ограничений, возникающих в реальном производстве: припуски на окантовку листа материала; припуски на механическую обработку; припуски на выполнение резов; припуски между заготовками для фиксации материала; учет дефектных областей материала; условие гильотинности. При этом известны значения ширины и длины прямоугольных предметов, радиусы кругов, размеры материала (длина, ширина), а также значения припусков. Требуется раскроить заданные предметы без перекрытия друг с другом и с гранями материала, учитывая перечисленные технологические ограничения таким образом, чтобы расход материала был минимальным.

В работе [1] показано, что уже задача одномерной упаковки в контейнеры принадлежит классу *NP*-трудных задач. Большинство работ посвящено задачам прямоугольного раскроя/упаковки [2—6]. Задачи раскроя/упаковки предметов произвольной формы встречаются реже. В связи с этим основное внимание здесь уделяется именно этим задачам. Для их решения разрабатываются как точные, так и эвристические методы.

В работе [7] рассматривалась задача гильотинного раскроя прямоугольных листов на круги одинаковых размеров. В работе [8] речь идет об оптимизационном методе решения задачи упаковки кругов и прямоугольников в полосу на базе метода ветвей и границ.

В работе [9] описан точный метод решения для задачи упаковки различных кругов в полубесконечную полосу. В работе [10] рассматривается оптимизационный метод решения задачи упаковки кругов и выпуклых многоугольников в минимальное число прямоугольников на базе метода ветвей и границ. При этом разрешены повороты многоугольников на произвольный угол.

Поскольку точные методы не позволяют решать задачи упаковки большой размерности за полиномиальное время, во многих работах уделяется большое внимание разработке приближенных и эвристических методов. Так, в работе [11] приведены приближенные алгоритмы упаковки кругов в прямоугольники. Авторами предложена процедура *ABLP* (*Adapted Best Local Position*), используемая при конструировании рационального размещения множества различных кругов. Для поиска решения использовался генетический алгоритм.

В работе [12] предложены жадные алгоритмы для задачи упаковки кругов в прямоугольный контейнер, а в работе [13] приведены новые эвристические алгоритмы для упаковки различных кругов в контейнер круглой формы. В работах [14, 15] рассматривается метод решения задачи упаковки кругов и прямоугольников в полосу на базе вероятностного алгоритма поиска с запретами и алгоритма имитации отжига. Задача упаковки различных кругов с помощью динамического алгоритма локального поиска рассмотрена в работе [16]. В работе [17] приведена задача упаковки в полосу различных фигур сложной формы с помощью генетического алгоритма. На базе алгоритма муравьиной колонии в работе [18] решалась задача упаковки выпуклых многоугольников в полосу, в работе [19] — задача прямоугольной упаковки в полосу и в прямоугольные листы. В данной статье для решения задач раскроя/упаковки прямоугольников и кругов разработан алгоритм муравьиной колонии, предложенный

в работе [20] для решения задачи о коммивояжере. Для формирования рационального размещения предметов была разработана модифицированная процедура *ABLP*, учитывающая требуемые технологические ограничения.

1. Постановка задач раскроя промышленного материала при наличии технологических ограничений

При решении практических задач оптимизации процесса раскроя промышленных материалов необходимо соблюдать ряд технологических ограничений, которые связаны с существующими технологиями раскройно-заготовительного производства. В настоящее время существуют следующие технологические способы, позволяющие осуществлять раскрой промышленного материала: механические, термические, гидрообразивные. Каждый из них применим в определенном диапазоне толщин и вида раскраиваемого материала. При этом должны учитываться следующие технологические ограничения: припуски на окантовку листа материала; припуски на механическую обработку — зон термического воздействия, неперпендикулярности реза, шероховатость; припуски на выполнение резов; припуски между заготовками для фиксации материала; учет дефектных областей материала; условие гильотинности; припуски на окантовку заготовок.

Задача 1 (раскрой листового/рулонного материала на круглые и прямоугольные заготовки с учетом дефектных областей материала)**.**

Исходная информация задач может быть задана наборами следующего вида: $<W; L; n_c; n_r; R_c; W_r; L_r; \Delta; \sigma; \varphi>$, где W, L — ширина и длина листового/рулонного материала, в случае рулона $L = \infty$ (рис. 1, *а*, *б*); n_c — число круглых предметов; $R_c = (r_{1c}, r_{2c}, ..., r_{ic}, ..., r_{n_c c})$ — вектор радиусов круглых заготовок, r_i — радиус i-й круглой заготовки, $i \in I_c = (1, 2, ..., n_c)$; n_r — число прямоугольных заготовок; $W_r = (w_{1r}, w_{2r}, ..., w_{jr}, ..., w_{n_r r})$, $L_r = (l_{1r}, l_{2r}, ..., l_{jr}, ..., l_{n_r r})$ — векторы ширины и длины прямоугольных заготовок, w_{ir}, l_{ir} — ширина и длина j-й прямоугольной заготовки соответственно, $j \in J_r = (1, 2, ..., n_r)$; Δ — необходимый суммарный припуск между заготовками; σ — необходимый суммарный припуск между верхней/нижней стороной промышленного материала и заготовкой; φ — необходимый суммарный припуск между боковыми сторонами промышленного материала и заготовкой. Введем прямоугольную систему координат *XOY*, у которой оси *OX* и *OY* совпадают соответственно с верхней и левой боковой сторонами области. Решение задачи представляется в виде наборов элементов: $<X_c, Y_c>$ и $<X_r, Y_r>$ — для задачи раскроя рулонного материала, $<X_c, Y_c, K_c>$ и $<X_r, Y_r, K_r>$ — для задачи раскроя листового материала, где $X_c = (x_{1c}, x_{2c}, ..., x_{jc}, ..., x_{n_c c})$, $Y_c = (y_{1c}, y_{2c}, ..., y_{jc}, ..., y_{n_c c})$ — векторы координат круглых заготовок; (x_{ic}, y_{ic}) — координаты центра i-й круглой заготовки соответственно по оси X и Y; $X_r = (x_{1r}, x_{2r}, ..., x_{jr}, ..., x_{n_r r})$, $Y_r = (y_{1r}, y_{2r}, ..., y_{jr}, ..., y_{n_r r})$ — векторы координат прямоугольных заготовок, (x_{jr}, y_{jr}) — координаты верхнего левого угла j-й прямоугольной заготовки по оси X и Y. В случае задачи раскроя листового материала положение i-й круглой и j-й прямоугольной заготовки задается наборами элементов (k_{ic}, x_{ic}, y_{ic}) и (k_{jr}, x_{jr}, y_{jr}), где $k_{ic} \in K_c = (k_{1c}, k_{2c}, ..., k_{1c}, ..., k_{n_c c})$, $k_{jr} \in K_r = (k_{1r}, k_{2r}, ..., k_{jr}, ..., k_{n_r r})$ — номер листа, на котором размещена i-я круглая и j-я прямоугольная заготовка соответственно. Введем множество $D = D_r \cup D_c$, где $D_r = (d_{1r}, d_{2r}, ..., d_{mr}, ..., d_{tr})$ — список дефектных областей, $m = \overline{1, t}$, аппроксимируемых наборами прямоугольников; $D_c = (d_{1c}, d_{2c}, ..., d_{sc}, ..., d_{gc})$ — список дефектных областей, $s = \overline{1, g}$, аппроксимируемых наборами кругов. Дефектные области D_r аппроксимируются наборами прямоугольников d_{mr}, положение d_{mr} определяется набором $<x^d_{mr}, y^d_{mr}, l^d_{mr}, w^d_{mr}, n^d_{mr}>$, где x^d_{mr}, y^d_{mr} — координаты верхнего левого угла дефектной области, l^d_{mr}, w^d_{mr} — длина и ширина дефектной области, n^d_{mr} — номер листа, который содержит дефектную область. Дефектные области D_c аппроксимируются наборами кругов d_{sc}, положение d_{sc} определяется набором $<x^d_{sc}, y^d_{sc}, r^d_{sc}, n^d_{sc}>$, где x^d_{sc}, y^d_{sc} — координаты центра дефектной области, r^d_{sc} — радиус

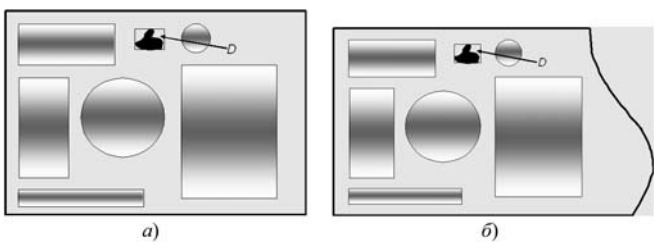


Рис. 1. Раскрой промышленного материала с учетом дефектных областей:
а — раскрой листового материала; *б* — раскрой рулонного материала

дефектной области, n_{sc}^{d} — номер листа, который содержит дефектную область.

Наборы элементов $<X_c, Y_c>$ и $<X_r, Y_r>$, или $<X_c, Y_c, K_c>$ и $<X_r, Y_r, K_r>$, называются допустимым раскроем, если выполнены следующие условия:

1) стороны прямоугольных заготовок параллельны сторонам листового или рулонного материала;

2) прямоугольные заготовки j, $s \in J_r(\forall s \neq j)$ *не перекрывают друг друга и не выходят за границы раскраиваемого материала;*

3) круглые заготовки $i \in I_c$ *не выходят за границы раскраиваемого материала;*

4) круглые заготовки i, $k \in I_c(\forall i \neq k)$ *не перекрывают друг друга* (*формула* (1)):

$$(x_{ic} - x_{kc})^2 + (y_{ic} - y_{kc})^2 \geqslant (r_{ic} + r_{kc} + \Delta)^2; \quad (1)$$

5) круглые $i \in I_c$ *и прямоугольные* $j \in J_r$ *заготовки не перекрывают друг друга:*

$$[((x_{ic} + r_{ic} + \Delta) \leqslant x_{jr}) \cup ((x_{jr} + l_{jr} + \Delta) \leqslant (x_{ic} - r_{ic})) \cup \cup ((y_{jr} + w_{jr} + \Delta) \leqslant (y_{ic} - r_{ic})) \cup ((y_{ic} + r_{ic} + \Delta) \leqslant y_{jr})] \cup \cup [(x_{ic} \leqslant x_{jr}) \cap (y_{ic} \leqslant y_{jr}) \cap ((x_{ic} - x_{jr})^2 + (y_{ic} - y_{jr})^2 \geqslant \geqslant (r_{ic} + \Delta)^2)] \cup (x_{ic} \geqslant (x_{jr} + l_{jr})) \cap (y_{ic} \leqslant y_{jr}) \cap \cap (x_{ic} - x_{jr} - l_{jr})^2 + (y_{ic} - y_{jr})^2 \geqslant (r_{ic} + \Delta)^2)] \cup \cup (x_{ic} \leqslant x_{jr}) \cap (y_{ic} \geqslant (y_{jr} + w_{jr}) \cap ((x_{ic} - x_{jr})^2 + + (y_{ic} - y_{jr} - w_{jr})^2 \geqslant (r_{ic} + \Delta)^2)] \cup [(x_{ic} \geqslant (x_{jr} + l_{jr})) \cap \cap (y_{ic} \geqslant (y_{jr} + w_{jr})) \cap ((x_{ic} - x_{jr} - l_{jr})^2 + + (y_{ic} - y_{jr} - w_{jr})^2 \geqslant (r_{ic} + \Delta)^2)]; \quad (2)$$

6) прямоугольные $j \in J_r$ и круглые $i \in I_c$ заготовки не пересекаются с дефектными областями $m \in D_r$ и $s \in D_c$.

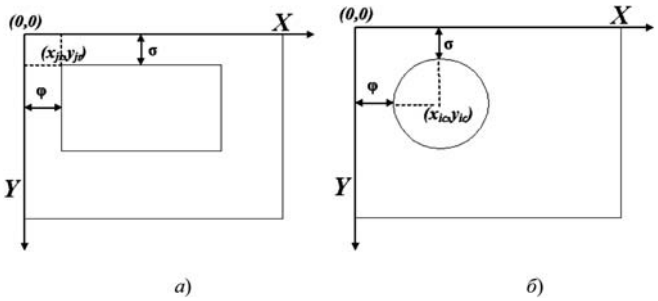
Требуется найти:

1) раскрой рулонного материала на круглые и прямоугольные заготовки при наличии дефектных областей минимальной длины $L' \to \min$;

2) совокупность раскроев листов, минимизирующих их суммарное количество, $N \to \min$.

2. Алгоритм размещения прямоугольников и кругов

В данной работе для рационального размещения прямоугольных и круглых предметов с учетом технологических ограничений предлагается метод $ABLP^+$, являющийся модификацией метода $ABLP$ [16]. Рассмотрим работу метода $ABLP^+$. Пусть имеется множество кругов $I_c \in I_c' \cup I_c''$ и множество прямоугольников $I_r \in I_r' \cup I_r''$, где I_c' — множество неразмещенных кругов, I_c'' — множество размещенных кругов, I_r' — множество неразмещенных прямоугольников, I_r'' — множество размещенных прямоугольников. Тогда общую схему метода $ABLP^+$ можно представить следующим образом:



а) *б*)

Рис. 2. Начальные позиции:
а — размещение прямоугольника; *б* — размещение круга

1) размещение первого предмета из множества $I_c \cup I_r$ в верхний левый угол области размещения (рис. 2);

2) формирование списка $П$ возможных позиций для размещения следующего компонента из множества $I_c' \cup I_r'$ и удаление из него тех позиций, которые не обеспечивают допустимое размещение;

3) выбор из списка $П$ лучшей позиции для размещения следующего предмета.

В алгоритме $ABLP^+$ формирование списка *доступных позиций* происходит следующим образом.

1. Как и в алгоритме $ABLP$, каждый круг P_i радиусом r_{ic} и координатами размещения (x_{ic}, y_{ic}) описывается кругом радиусом $r_{ic} + \Delta$, который очерчивается горизонтальными и вертикальными линиями, позволяющими определить четыре позиции с координатами (x_{jc}^1, y_{jc}^1), (x_{jc}^2, y_{jc}^2), (x_{jc}^3, y_{jc}^3), (x_{jc}^4, y_{jc}^4), приведенные на рис. 3, для размещения следующего круга P_j.

2. Каждый размещенный прямоугольник R_k шириной w_{kr} и длиной l_{kr} описывается прямоугольником шириной $w_{kr} + 2\Delta$ и длиной $l_{kr} + 2\Delta$, который очерчивается горизонтальными и вертикальными линиями, позволяющими определить четыре позиции, приведенные на рис. 4, для размещения

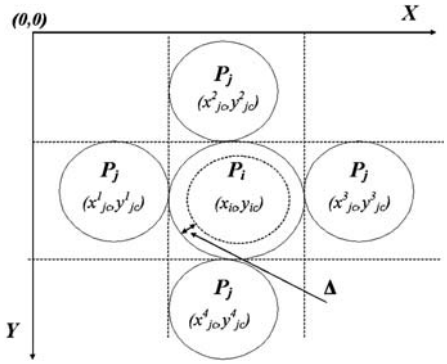


Рис. 3. Позиции, генерируемые кругом P_i для размещения следующего круга P_j

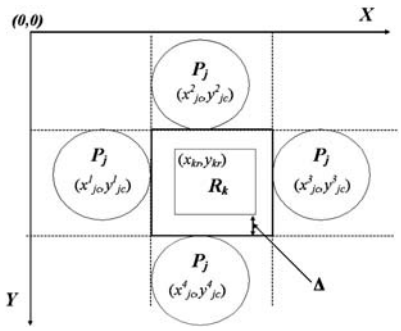


Рис. 4. Позиции, генерируемые прямоугольником R_k для размещения следующего круга P_j

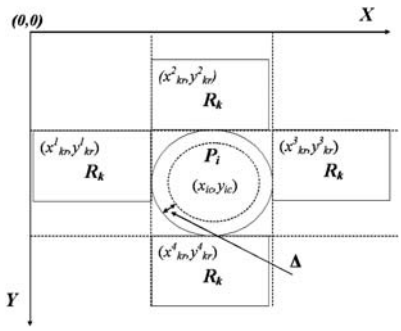


Рис. 5. Позиции, генерируемые кругом P_i для размещения следующего прямоугольника R_k

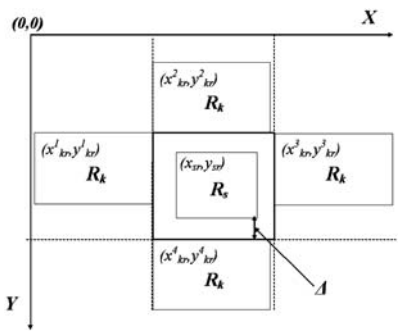


Рис. 6. Позиции, генерируемые прямоугольником R_s для размещения следующего прямоугольника R_k

следующего круга P_j, с координатами $(x^1_{jc},\ y^1_{jc})$, $(x^2_{jc},\ y^2_{jc})$, $(x^3_{jc},\ y^3_{jc})$, $(x^4_{jc},\ y^4_{jc})$.

3. Каждый размещенный круг P_i с радиусом r_i описывается кругом радиусом $r_i + \Delta$, который очерчивается горизонтальными и вертикальными линиями, позволяющими определить четыре позиции, приведенные на рис. 5, для размещения следующего прямоугольника R_k, с координатами $(x^1_{kr},\ y^1_{kr})$, $(x^2_{kr},\ y^2_{kr})$, $(x^3_{kr},\ y^3_{kr})$, $(x^4_{kr},\ y^4_{kr})$.

4. Каждый размещенный прямоугольник R_s шириной w_s и длиной l_s описывается прямоугольником шириной $w_s + \Delta$ и длиной $l_s + \Delta$, который очерчивается горизонтальными и вертикальными линиями, позволяющими определить четыре позиции, приведенные на рис. 6, для размещения следующего прямоугольника R_k, с координатами $(x^1_{kr},\ y^1_{kr})$, $(x^2_{kr},\ y^2_{kr})$, $(x^3_{kr},\ y^3_{kr})$, $(x^4_{kr},\ y^4_{kr})$.

5. Углы области образуют дополнительные позиции для размещения следующего предмета. На рис. 7, *а* представлены дополнительные две позиции с координатами $(x^1_{ic},\ y^1_{ic})$, $(x^2_{ic},\ y^2_{ic})$ для размещения круга в случае полубесконечной области. На рис. 7, *б*, представлены четыре дополнительные позиции с координатами $(x^1_{ic},\ y^1_{ic})$, $(x^2_{ic},\ y^2_{ic})$, $(x^3_{ic},\ y^3_{ic})$, $(x^4_{ic},\ y^4_{ic})$ для размещения круга в прямоугольную область. Аналогично и для прямоугольников.

6. Два смежных круга P_i и P_s генерируют дополнительные две позиции:

- для размещения следующего круга P_j формируются позиции, приведенные на рис. 8, *а*, с координатами $(x^1_{jc},\ y^1_{jc})$, $(x^2_{jc},\ y^2_{jc})$;
- для размещения следующего прямоугольника R_j формируются позиции, приведенные на рис. 8, *б*, с координатами $(x^1_{jr},\ y^1_{jr})$, $(x^2_{jr},\ y^2_{jr})$. Координаты позиций находятся таким образом, чтобы раз-

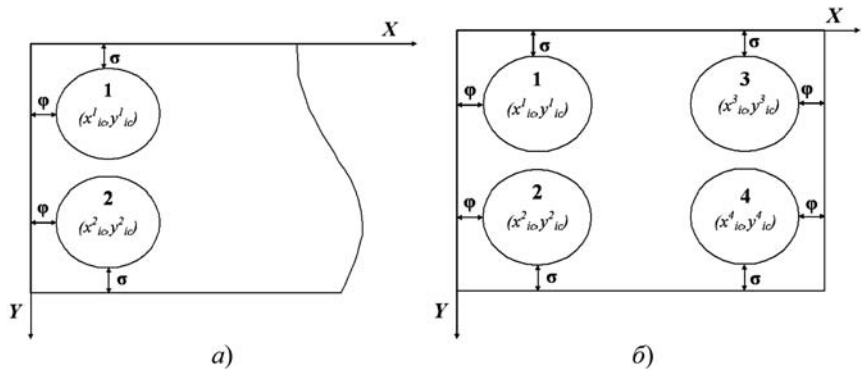


Рис. 7. Размещение кругов с помощью алгоритма *ABLP*$^+$:
а — размещение в полубесконечную область; *б* — размещение в прямоугольную область

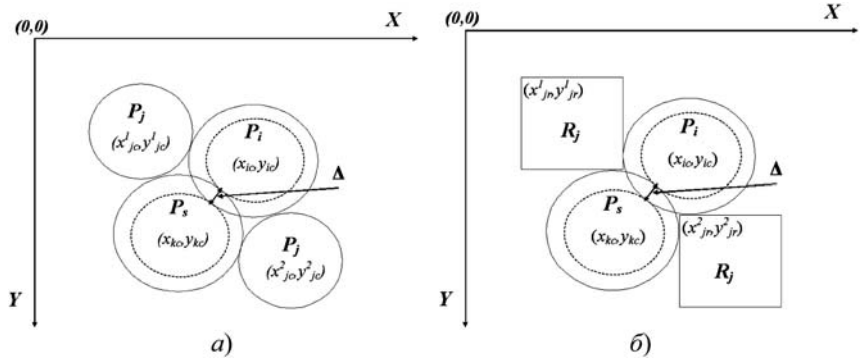


Рис. 8. Позиции, генерируемые двумя кругами P_i и P_s для размещения следующего предмета:
а — для размещения круга P_j; *б* — для размещения прямоугольника R_j

мещаемый прямоугольник R_j касался двух смежных кругов P_i и P_s.

7. Пара — круг P_i и прямоугольник R_k, генерируют дополнительные две позиции:

- для размещения следующего круга P_j формируются позиции, приведенные на рис. 9, *а*, с координатами (x^1_{jc}, y^1_{jc}), (x^2_{jc}, y^2_{jc});
- для размещения следующего прямоугольника R_j формируются позиции, приведенные на рис. 9, *б*, с координатами (x^1_{jr}, y^1_{jr}), (x^2_{jr}, y^2_{jr}).

Проиллюстрируем работу алгоритма $ABLP^+$ на следующем примере. Пусть имеется множество кругов $I_c = \{C_{1c}, C_{2c}, C_{3c}\}$ с заданными радиусами r_{1c}, r_{2c}, r_{3c} и множество прямоугольников $I_r = \{P_{1r}\}$ с заданной длиной l_{1r} и шириной w_{1r}. Требуется разместить заданные предметы в полубесконечную область при соблюдении условий допустимости.

Общая схема $ABLP^+$

1. Пусть первым предметом для размещения выбран круг C_{1c}, который размещается в верхний левый угол с координатами $(x_{1c} = r_{1c} + \varphi, y_{1c} = r_{1c} + \sigma)$.

2. Очередной предмет — круг C_{2c}, для размещения его формируется список позиций следующим образом:

а) круг C_{1c} радиуса r_{1c} описывается кругом радиусом $r_{1c} + \Delta$, который очерчивается горизонтальными и вертикальными линиями, позволяющими определить возможные позиции размещения $П = \{1, 2, 3, 4, 5, 6\}$ (рис. 10, *а*);

б) углы области размещения формируют позиции 3, 6.

Позиции 1, 2, 5 из $П$ не удовлетворяют условиям допустимости, следовательно, эти позиции удаляются из списка $П$. Оставшиеся позиции $П = \{3, 4, 6\}$ позволяют получить допустимое размещение. Из позиций 3, 4, 6 выбирается лучшая — самая верхняя левая позиция 4 (рис. 10, *б*).

Продолжаем размещать предметы. Пусть последним предметом для размещения остался круг C_{3c}.

3. Следующим предметом для размещения выбран круг C_{3c} радиусом r_{3c}. Список позиций для размещения круга C_{3c} формируется следующим образом:

а) каждый размещенный круг C_{1c} радиусом r_{1c} и круг C_{2c} радиусом r_{2c} описываются соответствующими кругами с радиусами $r_{1c} + \Delta$ и $r_{2c} + \Delta$, которые очерчиваются горизонтальными и вертикальными линиями, позволяющими определить возможные позиции для размещения $П = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}$ (рис. 11, *а*, *б*);

б) размещенный прямоугольник P_{1r} длиной l_{1r} и шириной w_{1r} описывается прямоугольником длиной $l_{1r} + 2\Delta$ и шириной $w_{1r} + 2\Delta$, который очерчивается горизонтальными и вертикальными линиями, позволяющими определить четыре позиции 9—12 (рис. 11, *а*);

в) два смежных круга C_{1c} и C_{2c} формируют две позиции размещения 13 и 14 (рис. 11, *б*);

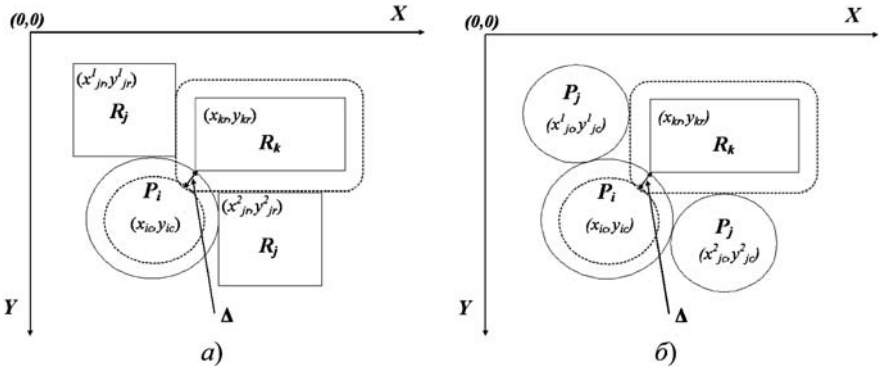


Рис. 9. Позиции, генерируемые парой (P_i, R_k) для размещения следующего предмета:
а — для размещения круга P_j; *б* — для размещения прямоугольника R_j

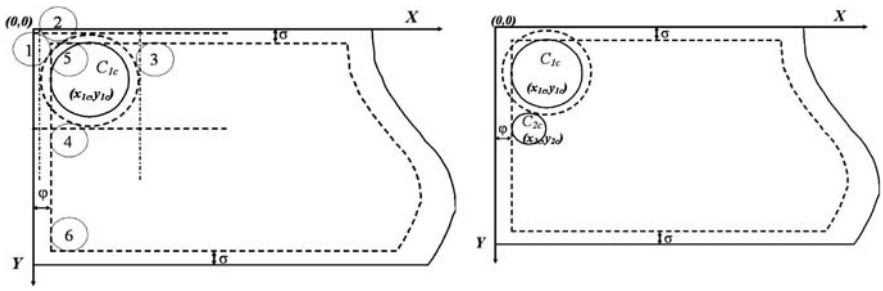


Рис. 10. Пример размещения второго круга C_{2c}:
а — список возможных размещений круга C_{2c}; *б* — размещение круга C_{2c}

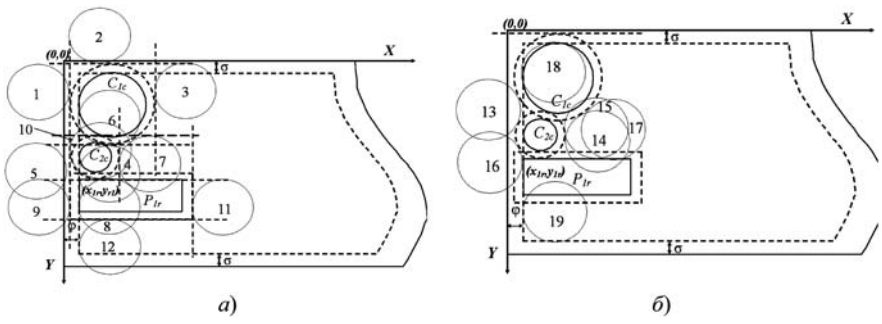


Рис. 11. Пример размещения круга C_{3c}:
а — список позиций для размещения круга C_{3c}; *б* — список позиций для размещения круга C_{3c}, формируемые парами круг и прямоугольник

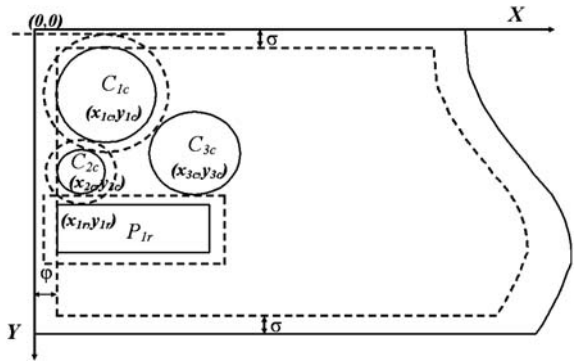


Рис. 12. Пример конечного размещения

г) пара круг C_{2c} и прямоугольник P_{1r} формирует две позиции 15 и 16 (рис. 11, *б*);

д) круг C_{1c} и прямоугольник P_{1r} формируют одну позицию 17 (рис. 11, *б*);

е) углы области размещения формируют дополнительные две позиции 18 и 19 (рис. 11, *б*).

Из списка возможных позиций $\Pi = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}$ для размещения круга C_{3c} относительно уже размещенных кругов C_{1c} и C_{2c} и прямоугольника P_{1r} позиции 1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19 не удовлетворяют условиям допустимости, следовательно, эти позиции удаляются из списка П. Оставшиеся позиции 3, 11, 17 позволяют получить допустимое размещение. Из них выбирается лучшая — самая верхняя левая позиция 17 (рис. 12).

3. Алгоритм муравьиной колонии для задачи оптимизации процесса раскроя промышленных материалов при наличии технологических ограничений

Для получения приближенного решения задачи раскроя прямоугольников и кругов разработан вероятностный алгоритм муравьиной колонии *ACOGA*. Этот алгоритм основан на идее алгоритма муравьиной колонии с некоторыми процедурами генетического алгоритма *P-ACO* и предложен в работе [20] для решения задачи коммивояжера.

Алгоритм муравьиной колонии относится к классу метаэвристик. Под метаэвристикой понимают вероятностные итерационные алгоритмы, допускающие абстрактный уровень описания, что позволяет применять их к любым задачам комбинаторной оптимизации [21]. Наиболее известными метаэвристиками являются: генетический алгоритм; итерационный локальный поиск; имитация отжига; поиск с запретами; поиск с переменной окрестностью; вероятностный жадный алгоритм и другие. Для ряда метаэвристик установлена сходимость по вероятности наилучшего найденного решения к оптимальному. Алгоритм муравьиной колонии относится к конструктивным вероятностным алгоритмам, реализующим мультиагентный подход. При этом решение задачи строится покомпонентно путем добавления нового компонента к частично построенному решению до тех пор, пока оно не будет построено полностью.

Для применения алгоритма муравьиной колонии к задаче комбинаторной оптимизации необходимо определить его основные характеристики: компонент решения, эвристическую информацию (число, не зависящее от решений, найденных на предыдущих итерациях, и отражающее желательность принятия того или иного компонента решения); правило изменения феромона (число, показывающее, насколько часто агентами использовались компоненты решения на предыдущих итерациях). Решение задачи, построенное каждым агентом, формируется из компонентов на основе накопления и использования статистической информации — искусственных следов феромона и специфичной для задачи эвристической информации.

Для задачи размещения кругов и прямоугольников компонентом решения является предмет (прямоугольник или круг), феромон τ_{ij} наносится на последовательности из двух предметов (i, j), где $i, j \in I_c \cup I_r$, $I_c \in I_c' \cup I_c''$ (I_c' — множество неразмещенных кругов, I_c'' — множество размещенных упакованных кругов), $J_r \in J_r' \cup J_r''$ (J_r' — множество неразмещенных прямоугольников, J_r'' — множество размещенных прямоугольников). Эвристическая информация η_j определяется как площадь предмета: $\eta_j = (w_{jr} \times l_{jr})\gamma + (\pi \times r_{jc})(1 - \gamma)$ — численная характеристика полезности предмета j для построения решения;

$$\gamma = \begin{cases} 1, \text{ если предмет } j \in J_r \\ 0, \text{ если предмет } j \in I. \end{cases}$$

Выбор предмета для размещения определяется, как в работе [22]:

$$\forall i, j \in I_c' \cup I_r' : p = \frac{\tau_{ij}^{\alpha} \eta_j^{\beta}}{\sum\limits_{k \in I_c' \cup I_r'} \tau_{ik}^{\alpha} \eta_k^{\beta}}. \qquad (3)$$

Ниже приведена общая схема алгоритма *ACOGA* для решения задачи оптимизации процесса раскроя промышленных материалов на круглые и прямоугольные заготовки, в котором для обновления популяции используется известная стратегия *Age*, в ней из всех решений популяции удаляется самое "старое" решение, а вместо него добавляется новое, лучшее, решение.

Алгоритм *ACOGA*

Вход: W, L — ширина и длина листа области размещения; n_c — число кругов; n_r — число прямоугольников; w_i, l_i — ширина и длина j-го прямо-

угольника соответственно, r_i — радиус i-го круга, $i = 1, 2, ..., n_r$; $j = 1, 2, ..., n_c$, Δ, σ, φ, δ — требуемые припуски; $D = (d_1, d_2, ..., d_s)$ — список дефектных областей, стратегии для обновления популяции *Age*.

Выход: лучшее найденное решение (*карта раскроя*).

1. Инициализация параметров алгоритма: k — размерность популяции; m — число агентов; α — коэффициент влияния феромона (численная характеристика предмета, показывающая, насколько часто данный предмет входил в лучшие решения на предыдущих итерациях алгоритма); β — коэффициент влияния эвристической информации; τ_{init} — начальное значение феромона; $\tau_{\max}$ — максимальное значение феромона; ρ — коэффициент испарения феромона, размещение дефектных областей $D = (d_1, d_2, ..., d_s)$.

2. Повторять, пока не выполнен критерий остановки:

2.1. Выбрать предмет из множества $I_c \cup I_r$ с некоторой вероятностью по формуле (20) и поместить его в верхний левый угол области с помощью метода $ABLP^+$.

2.2. Для каждого из m агентов, не завершивших построение решений, выполнить:

2.2.1. Выбрать следующий предмет из множества $I_c' \cup J_r'$ с некоторой вероятностью, определяемой формулой (3).

2.2.2. Добавить выбранный компонент к частично построенному решению:

2.2.2.1. С помощью метода $ABLP^+$ построить список позиций, удовлетворяющих условиям допустимости;

2.2.2.2. Удалить из списка позиций позиции, не удовлетворяющие условию непересечения с дефектными областями;

2.2.2.3. Поместить компонент в лучшую — верхнюю левую — найденную позицию.

2.3. Определить среди решений, построенных агентами, лучшее решение — карту раскроя с наилучшей целевой функцией.

2.4. Если $|P| = k$, то удалить из популяции решение согласно выбранной стратегии обновления популяции:

2.4.1. Выполнить обновление феромона.

2.5. Добавить лучшее решение в популяцию P:

2.5.1. Выполнить обновление феромона.

3. Выдать результат — лучшее найденное решение.

В начале алгоритма *ACOGA* популяция пуста. В течение первых k итераций в популяцию P добавляется лучшее решение, найденное на каждой итерации, полученное агентами. Начиная с $(k + 1)$-й итерации популяция полностью сформирована, и для добавления следующего решения требуется удалить одно из лучших решений (применение стратегии *Age*), т. е. лучшее решение этой итерации является кандидатом на добавление в популяцию. Обновление феромона происходит лишь для решений популяции [20]: если решение добавляется в популяцию, к значению феромона добавляется величина θ; если решение удаляется из популяции, значение феромона уменьшается на величину θ, где $\theta = \dfrac{Q}{f(a)}$, $f(a)$ — значение целевой функции лучшего найденного решения; Q — некоторая постоянная величина. Критерием остановки алгоритма выступает либо число итераций, либо время работы алгоритма.

4. Численные эксперименты

Разработанный алгоритм *ACOGA* был реализован на языке *C*++. Были проведены численные эксперименты на вычислительной машине Intel Quadra 2,4 GHz.

Эксперимент 1. Определение влияние параметров алгоритма *ACOGA* на качество решений

Для задач размещения прямоугольников в полубесконечную область рассмотрены примеры из международной библиотеки *OR-library* для задач исследования операций (http://people.brunel.ac.uk/~mastjjb/jeb/info.html). На рис. 13, *а* (см. третью сторону обложки) приведены значения целевых функций при некоторых значениях коэффициентов α и β на тестовых данных класса *N1*, содержащего пять примеров *N*11, *N*12, *N*13, *N*14, *N*15, в каждом из которых содержится по 17 прямоугольников. На рис. 13, *б* приведены значения целевых функций при некоторых значениях коэффициентов α и β на тестовых данных класса *N*7, содержащего пять примеров *N*71, *N*72, *N*73, *N*74, *N*75, в каждом из которых содержится по 75 прямоугольников. При этом были получены параметры: коэффициенты α и β изменялись на интервале от 0,4 до 2 с шагом 0,1; число агентов $m = 12$; размер популяции $k = 10$; начальное значение феромона $\tau_{init} = 0{,}05$.

Для задачи размещения кругов в полубесконечную область рассмотрены тестовые примеры, в которых радиусы кругов были случайно сгенерированы в интервале $r_{ic} \in [0{,}1W, ..., 0{,}5W]$, где r_{ic} — радиус i-го круга. На рис. 13, *в* приведены значения целевых функций при некоторых значениях коэффициентов α и β на тестовых данных класса *C*1, содержащего три примера *C*11, *C*12, *C*13, в каждом из которых содержится по 100 кругов. На рис. 13, *г* приведены значения целевых функций при некоторых значениях коэффициентов α и β на тестовых данных класса *C*2, содержащего три примера *C*21, *C*22, *C*23, в каждом из которых содержится по 500 кругов.

Эксперимент 2. Решалась задача размещения кругов

Эксперимент проводился на известных классах примеров SY1—SY6 [Hopper E.]. Для сравнения были также представлены решения задачи размещения кругов, полученные генетическим алгоритмом *CAGA* [Hifi M., M'Hallah R.], методом ветвей и границ *S. Y.*

[Stoyan Y. G., Yaskov G. N.], жадной эвристической процедурой *B1.5* [Huang W. Q., Akeb H., Li Y., Li C. M.], алгоритмом вероятностного поиска с запретами *TABU SEARCH (TS)* [Ю. А. Кочетов, А. С. Руднев] и решениями, найденными с помощью коммерческого пакета *GAMS* [А. С. Руднев] (http://www.gams.com/). В качестве оценки применялась нижняя граница оптимальной длины полубесконечной области LB, $LB = \frac{\sum_{i=1}^{n} S_i}{W}$ (S_i — площадь i-го круга, W — ширина полубесконечной области). В качестве критерия оценки выступает длина занятой части полубесконечной области L (рис. 14, см. третью сторону обложки).

Эксперимент № 3. Решалась задача одновременного размещения кругов и прямоугольников

В эксперименте проводилось сравнение работы алгоритмов *TS*, *ACOGA* и коммерческого пакета *GAMS* (рис. 15, см. третью сторону обложки). *LB*-нижняя и *UB*-верхняя оценки [Руднев А. С.] для алгоритмов *TS*, *ACOGA* приведены результаты работы алгоритмов. При решении задач одновременного размещения кругов и прямоугольников в полубесконечную область алгоритму *ACOGA* удалось найти новые рекордные решения на классах задач: CR3P02; CR5P01; CR5P02; CR5P03; CR6P03 [Руднев А. С.].

Рассмотрим практическое применение на примере металлообрабатывающей промышленности. При изготовлении различных видов продукции: печей, камер сгорания и др. требуется выполнить раскрой листового материала на различные виды заготовок, в частности круглые и прямоугольные (стенки, лопатки, мембраны, обтекатели, сильфоны и др.).

Заключение

Рассмотрена задача раскроя прямоугольников и кругов в полосу, для решения которой разработан алгоритм муравьиной колонии. При этом для формирования частично построенного решения задачи модифицирована известная конструктивная процедура *ABLP*. Проведенные численные эксперименты показали, что с помощью алгоритма найдены хорошие решения на различных тестовых примерах, в том числе оптимальные на тестовых примерах небольшой размерности. На некоторых тестовых примерах удалось найти новые рекордные решения. Кроме того, для задач раскроя кругов и одновременного раскроя кругов и прямоугольников удалось получить решения для тестовых примеров большой размерности ($n \geqslant 100$). В дальнейшем планируется применить разработанный алгоритм для задач трехмерного раскроя.

Список литературы

1. **Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М.: Мир. 1982. 416 с.

2. **Валеева А. Ф.** Методы частичного перебора локального поиска оптимума в задаче двухмерной упаковки // Информационные технологии. 2004. № 1. С. 36—43.

3. **Мухачева Э. А., Валеева А. Ф., Картак В. М.** Модели и методы решения задач ортогонального раскроя и упаковки: аналитический обзор и новая технология блочных структур // Информационные технологии. Приложение. 2004. № 5. 32 с.

4. **Стоян Ю. Г., Яковлев С. В.** Математические модели и оптимизационные методы геометрического проектирования. Киев: Наукова думка. 1986. 268 с.

5. **Dousland K. A., Dousland W. B.** Packing problems // European Journal of Operational Research. 1992. N 56. P. 2—14.

6. **Lodi A., Martello S., Vigo D.** Recent advances on two-dimensional bin packing problems // Discrete Applied Mathematics 123. 2002. P. 379—396.

7. **Канторович Л. В., Залгаллер В. А.** Рациональный раскрой промышленных материалов. Изд. второе. Новосибирск. 1971. 299 с.

8. **Stoyan Y. G., Yaskov G. N.** Mathematical model and solution of optimization problem of placement of rectangles and circles taking into account special constraints // International Transaction in Operational Research. 1998. N 5 (1). P. 45—57.

9. **Stoyan Y. G., Yaskov G. N.** A mathematical model and solution for the problem of placing various-sized into a strip // European Journal of Operational Research. 2004. N 156. P. 590—600.

10. **Kallrath J.** Cutting circles and polygons from area-minimizing rectangles // Journal of Global Optimization. 2009. N 43. P. 299—328.

11. **Hifi M., M'Hallah R.** Approximate algorithms for constrained circular cutting problems. // Computers and Operations Research. 2004. N 31. P. 675—694.

12. **Huang W. Q., Li Y., Aked H., Li C. M.** Greedy algorithms for packing unequal circles into a rectangular container // European Journal of Operational Research. 2005. N 56. P. 539—548.

13. **Huang W. Q., Li Y., Li C. M., Xu R. C.** New heuristic for packing unequal circles into a circular container // Computers and Operations Research. 2006. N 33. P. 2125—2142.

14. **Руднев А. С.** Вероятностный поиск с запретами для задачи раскроя кругов и прямоугольников в полосу // Дискретный анализ и исследование операций. 2009. Т. 16, N 4. С. 61—86.

15. **Kallrath J., Kochetov Y., Rudnev A.** Strip packing problem for circles and rectangles // 4th ESICUP Meeting, 2007, March 25—27, Tokyo, Japan. P. 20.

16. **Hifi M., M'Hallah R.** A dynamic adaptive local search algorithm for the circular packing problem // European Journal of Operational Research. 2007. N 183. P. 1280—1294.

17. **Hifi M., M'Hallah R.** A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes // International Transaction in Operational Research. 2003. N 10. P. 195—216.

18. **Burke E., Kendall G.** Applying Ant Algorithms and the No Fit Polygon to the Nesting Problem // Accepted for the 1999 International Conference on Artificial Intelligence. Las Vegas. Nevada. USA. 1999.

19. **Валеева А. Ф.** Применение метаэвристики муравьиной колонии к задачам двухмерной упаковки // Информационные технологии. 2005. № 10. С. 36—43.

20. **Guntssch M., Middendorf M.** Applying population based ACO for dynamic optimization problem // M. Dorigo et al., (eds.) Ant Algorithms. — Third International Workshop ANT2002, Lecture Note in Computer Science. N 2463. Springer Verlag, 2002. P. 111—122.

21. **Blum A. Roli.** Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison // TR/IRIDIA/2001. 37 p.

22. **Burke E. K., Kendall G.** Search methodologies: tutorials in optimization and decisions. Support Techniques. Springer Science + Business Media, LCC. 2005. 600 p.

УДК. 621.391

И. Н. Лысенков[1],
канд. военных наук, нач. учебного отдела,
С. В. Дворников[1], д-р техн. наук, доц., проф.,
С. С. Дворников[2], студент,
Д. М. Ишин[1], курсант,
А. А. Устинов,
д-р техн. наук, проф., зам. нач. каф.
[1] Военная академия связи, г. Санкт-Петербург
[2] Санкт-Петербургский государственный политехнический институт,
e-mail: practicdsv@yandex.ru

Пороговое декодирование самоортогональных кодов

Представляются результаты исследования методов декодирования самоортогональных кодов. Анализируются практические схемы кодеров и декодеров самоортогональных кодов. Исследуются возможности применения их в системах передачи телевизионной информации.

Ключевые слова: *самоортогональные коды, пороговое декодирование, синдромный регистр, дефинитный декодер*

Введение

Одна из проблем проектирования современных систем телевизионного вещания связана с обеспечением высокой достоверности передачи данных. Теоретически можно создать систему связи со сколь угодно малой вероятностью ошибки на выходе декодера канала. При этом адекватная система без корректирующего кодирования будет более сложной, дорогой и энергоемкой. Отсюда вывод: цифровая система телевизионного вещания работает в режиме с достаточно высокой частотой ошибок в потоке на входе декодера, а декодированный поток должен иметь крайне малую вероятность ошибки на бит [1].

В настоящее время в современных системах телевизионного вещания широкое распространение получили линейные недвоичные систематические блоковые коды Рида — Соломона [2], являющиеся подклассом циклических кодов Боуза — Чоудхури — Хоквингема [3]. В частности, для цифрового наземного телевизионного вещания на базе MPEG-2 в разработанной системе ISDB-T рекомендовано в качестве внешнего кода использовать код Рида — Соломона длиной $n = 204$ символа (188 информационных), исправляющий 8 байтовых ошибок [1].

Между тем возрастающий объем медеоуслуг предъявляет все более жесткие требования к качеству телевизионной информации в условиях ограниченной пропускной способности каналов связи. Следовательно, в сложившейся ситуации необходим поиск новых подходов к решению данной проблемы, в частности, возможности применения в качестве корректирующего кода самоортогональных кодов (СОК) [4, 5]. Данные вопросы довольно глубоко исследованы в работе [6], где в контексте общей теории рассмотрены возможности различных кодовых конструкций. В настоящей статье основное внимание акцентировано только на СОК. Подробно рассмотрен их генезис в рамках обобщающих линейных мажоритарно декодируемых кодов (ЛМДК), описана работа формирующих их кодера и различных вариантов декодирующих устройств, детально исследован алгоритм синтеза СОК согласно предложенной структурной схеме кодирования. Приведены результаты эксперимента по оценке эффективности предложенных технических решений. Авторы надеются, что данная работа будет интересна специалистам, занимающимся вопросами повышения помехоустойчивости передачи телевизионной информации.

Характеристика самоортогональных кодов

В классе блоковых кодов особое место отводится ЛМДК, образующим ортогональные проверки и допускающим принятие решения по критерию максимального правдоподобия [4]. Принцип декодирования ЛМДК основан на свойствах ортогональности их проверочных уравнений по отношению к возможным ошибкам. Одной из разновидностей ЛМДК являются самоортогональные коды [5], допускающие многопороговое декодирование. Важная особенность СОК в том, что система всех проверок, контролирующих произвольный символ e_x, где $x \in [0; n]$, а n — длина кода, сама является ортогональной относительно данного символа.

Как правило, СОК формируются на основе образующих полиномов $g(x)$, разностные треугольники которых не содержат одинаковых элементов. Так, для кодера СОК (рис. 1) длиной $n = 26$, содержащего $k = 13$ информационных символов с кодовым расстоянием $d = 5$, формируемого полиномом $g(x) = x^0 + x^1 + x^4 + x^6$, разностный треугольник имеет вид

$$\begin{array}{r} 1-0;\ 4-0;\ 6-0 \\ 4-1;\ 6-1 \\ 6-4 \end{array} \quad \Rightarrow \quad \begin{array}{r} 1;\ 4;\ 6 \\ 3;\ 5 \\ 2 \end{array}$$

Работает указанный декодер следующим образом. В первоначальный момент последовательность информационных символов u_i поступает на регистры сдвига, а все коммутаторы K находятся в положении 1.

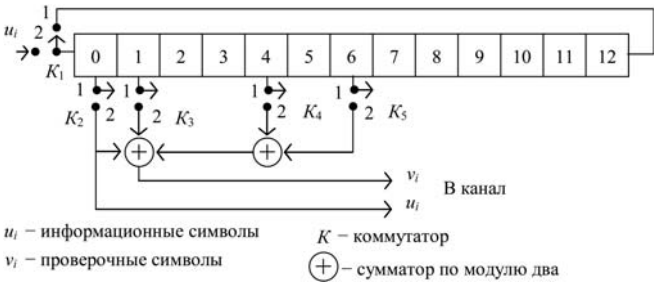


Рис. 1. Структурная схема самоортогонального кодера длиной (26, 13)

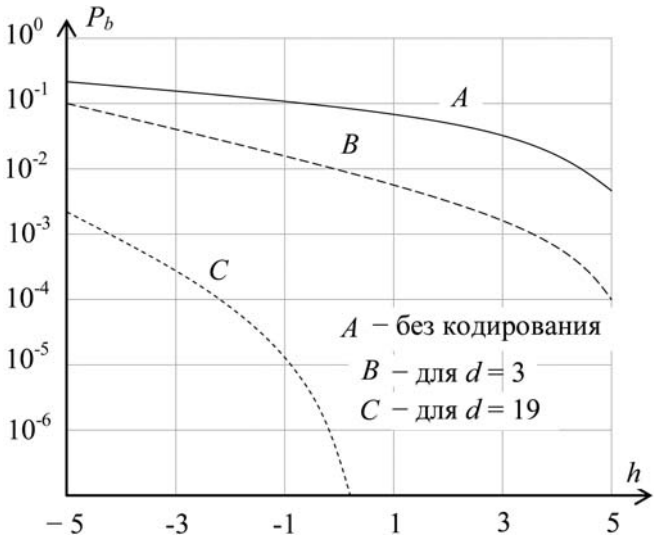


Рис. 2. Оценка характеристик оптимального декодирования СОК в ДСК для различных значений *d*

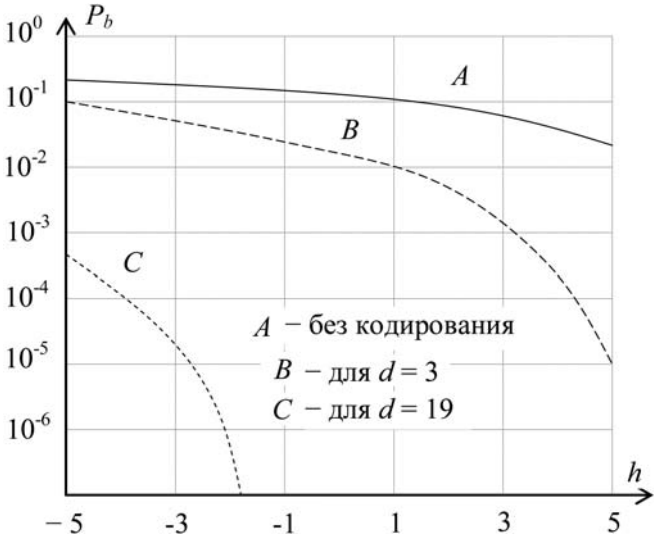


Рис. 3. Оценка характеристик оптимального декодирования СОК в КГШ для различных значений *d*

Затем коммутаторы переводятся в положение 2 и начинается этап формирования кодовой комбинации. На каждый такт с выхода кодера в канал параллельно поступают информационные символы и комбинация проверочных символов, формируемая на сумматорах с выходов 0-, 1-, 4- и 6-го регистров кодера.

Проверочная матрица $\mathbf{H} = [\mathbf{P}^{\mathrm{T}}\mathbf{I}_{n\ -\ k}]$ для СОК формируется на основе порождающего полинома $g(x)$. Здесь $\mathbf{I}$ — единичная матрица размерности $(n - k) \times (n - k)$; $\mathbf{P}$ — проверочная матрица размерности $k \times (n - k)$; т — знак транспонирования. Заметим, что для рассматриваемого случая размерность матрицы $\mathbf{H}$ будет (13×26).

Вероятность ошибки на бит P_b при оптимальном декодировании СОК для двоичного симметричного канала (ДСК) и канала с гауссовым шумом (КГШ) оценивается согласно [6]:

$$P_b(j) = \begin{cases} \sum\limits_{i=(d+1)/2}^{d} C_d^i p^i (1-p)^{d-1} - \\ \text{для ДСК при нечетном } d; \\ \frac{1}{2} C_d^{d/2} p^{d/2} (1-p)^{d/2} + \\ + \sum\limits_{i=(d/2)+1}^{j} C_d^i p^i (1-p)^{d-1} - \\ \text{для ДСК при четном } d; \\ Q\left(\sqrt{2dE_s/N_0}\right) \text{ — для КГШ}, \end{cases} \quad (1)$$

где p — вероятность ошибки на входе декодера; $E_s/N_0 = h$ — отношение сигнал/шум (ОСШ) в канале; $Q(x) = \frac{1}{\sqrt{2\pi}} \int\limits_x^\infty \mathbf{e}^{-t/2} dt.$

На рис. 2 и 3 показаны графики зависимости P_b от ОСШ при работе в ДСК и КГШ.

Анализ представленных результатов показал, что переход к декодированию на основе мягкого принятия решения обеспечивает выигрыш в 2 дБ.

Способы декодирования самоортогональных кодов

Традиционно для блоковых кодов, к которым относятся ЛМДК, декодирование осуществляется с использованием подхода, предложенного Меггитом [3], исправляющего пакет ошибок простым алгебраическим методом на основе критерия максимального правдоподобия. Однако его применение в каналах передачи телевизионной информации нецелесообразно, поскольку реализационная сложность декодера Меггита экспоненциально растет с числом исправляемых ошибок.

Интересным решением при работе в КГШ является применение алгоритмов Чейза [7]. Они базируются на принципе жесткого декодирования нескольких пробных последовательностей, специально

формируемых внесением в вектор жестких решений относительно мягких решений демодулятора, различных комбинаций ошибок. Следует заметить, что необходимость так называемого предварительного обучения такого декодера снижает его практическую ценность для систем передачи телевизионной информации, поскольку применение длинных кодов в этом случае заметно снижает общую производительность системы декодирования.

Между тем, в работе [5] Дж. Месси обоснована целесообразность применения для ЛМДК принципа порогового (мажоритарного) декодирования (ПДК). Сущность данного подхода заключается в следующем. В принимаемой кодовой комбинации выделяют информационную часть, которую подают на декодер, где из нее формируют проверочную комбинацию символов, аналогично тому, как это делается на передающем конце. Затем сформированные проверочные символы складывают по модулю 2 с проверочными символами принятой реализации и заполняют ими синдромный регистр. Затем проверочные элементы синдромного регистра суммируются и поступают на пороговый элемент, где сравниваются с априори установленным значением. И в случае его превышения инвертируют соответствующий им информационный символ принятой кодовой комбинации.

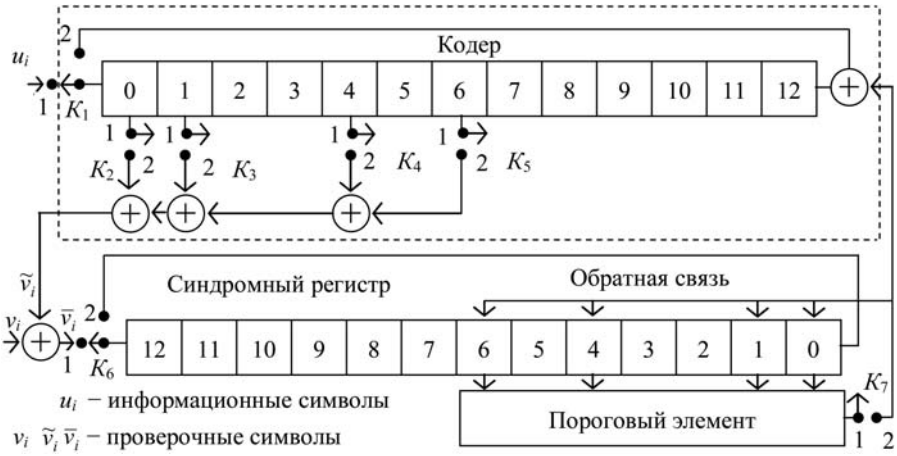


Рис. 4. Структурная схема ПДК для СОК длиной (26, 13)

В целях пояснения особенностей работы указанного принципа рассмотрим пример реализации ПДК (рис. 4) для СОК (26, 13), представленного на рис. 1.

Вначале из последовательности u_i формируются проверочные символы $\tilde{v}_i$, которые по модулю 2 складываются с принятыми проверочными символами v_i на коммутаторе K_6 (в начальный момент все ключи в положении 1). Затем рассчитанные значения $\bar{v}_i = v_i + \tilde{v}_i$ поступают в синдромный регистр. После его заполнения все коммутаторы переводятся в положение 2 и начинается процесс декодирования информационного символа u_{12}. С этой целью в пороговом элементе суммируются элементы синдромного регистра, соответствующие декодированному символу (для u_{12} имеем сумму $v_{\Sigma 12} = \bar{v}_0 + \bar{v}_1 + \bar{v}_4 + \bar{v}_6$), и сравниваются с априорно установленным пороговым значением $v_{\text{пор}}$. Таким образом, выходной символ с пороговым элементом y_i будет определяться формулой

$$y_i = \begin{cases} 1, \text{ если } \Sigma v_i^0 + v_i^1 + v_i^4 + v_i^6 \geqslant v_{\text{пор}}; \\ 0, \text{ если } \Sigma v_i^0 + v_i^1 + v_i^4 + v_i^6 < v_{\text{пор}}. \end{cases}$$

Здесь верхний символ у v обозначает позицию ячейки синдромного регистра, участвующую в суммировании, а нижний индекс — текущее значение корректированного символа.

Из синдромного регистра выходной символ y_i поступает на входной сумматор кодера (см. рис. 4). Если значение y_i будет равно единице, то соответствующий информационный символ будет инвертирован.

Затем выполняется циклический сдвиг содержимого регистров и происходит декодирование следующего символа. Учитывая, что рассмотренное ПДК осуществляется устройством с обратной связью, то исправляемая в информационном регистре ошибка удаляется и из синдромного регистра. Следует заметить, что наличие обратной связи может привести к размножению ошибок, поскольку в синдром по цепи обратной связи дополнительно поступают $d - 1$ собственных ошибок.

Для устранения рассмотренного явления в работе [8] предложено применять СОК с низкой долей общих проверок или использовать дефинитное декодирование без обратной связи [5].

Так, дефинитный декодер при работе с СОК в ДСК обеспечивает вероятность ошибки в информационном бите P_b при скоростях $r = (n - 1)/n$, определяемую выражением

$$P_b = (1 - p) \sum_{i = T+1}^{J} C_J^i P^i (1 - P)^{J - i} + p \sum_{i = T+1}^{J} C_J^i P^i (1 - P)^{J - i}, \quad (2)$$

где p — вероятность искажения бита в канале; J — число слагаемых образующего полинома $J = d - 1$; $P = \dfrac{1 - (1 - 2p)^{J(n-1)}}{2}$ — вероятность ошибки проверочного символа; $T = (d - 1)/2$ — величина порога; n — длина кода.

На рис. 5 представлены зависимости оценки (2) от ОСШ для СОК с различными значениями d.

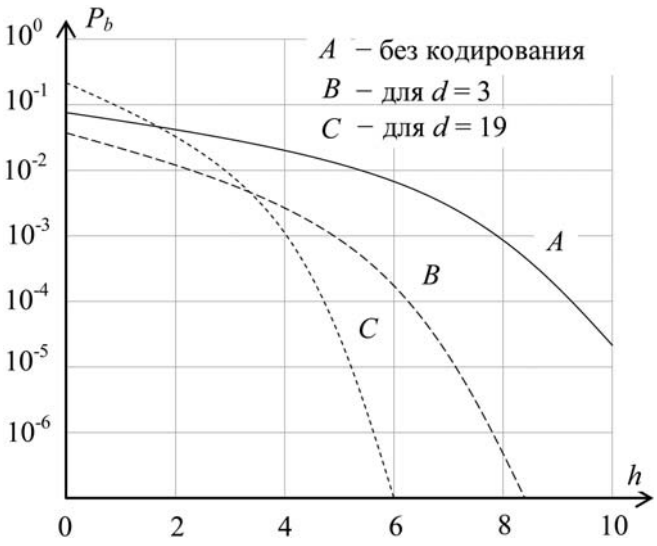


Рис. 5. Оценка эффективности дефинитного ПДК для СОК с $r = 1/2$ в ДСК

Анализ компьютерного моделирования показал следующее. При ОСШ ниже 3,5 дБ эффективность дефинитного декодера снижается с увеличением значения d.

Реализационная сложность такого декодера экспоненциально растет с увеличением d. При фиксированном значении кодового расстояния лучшими характеристиками ОСШ, приходящихся на бит, обладают СОК с более высокой кодовой скоростью. Согласно [6] характеристики дефинитных декодеров при $P_b = 10^{-5}$ оказываются на 0,25 дБ хуже, чем у ПДК с обратной связью. Переход к мягким решениям в ПДК позволяет повысить их эффективность приблизительно на 0,3 дБ. В этом случае в пороговом элементе происходит суммирование элементов синдрома с весом, зависящим от надежности символов, участвующих в формировании элемента синдрома. Кроме того, эффективность декодирования можно повысить и за счет использования так называемых длинных кодов.

Сложность программной реализации ПДК рассчитывается исходя из необходимости реализации $d - 1$ операций сложения целых чисел (подсчет суммы в пороговом элементе), операции сравнения с порогом, d операций сложения по модулю 2 (при коррекции информационного символа), приходящихся на один декодируемый информационный бит. При этом для вычисления синдрома необходимо еще осуществить $d - 1$ операций сложения по модулю 2. Таким образом, среднее число операций, необходимых для декодирования одного информационного бита, составит $2d$.

Заключение

Полученные теоретические и практические результаты показали общую эффективность применения дефинитных ПДК для обработки самоортогональных кодов. Следовательно, можно предположить, что применение данной технологии в системах передачи телевизионной информации позволит снизить предъявляемые требования по ОСШ в канале и тем самым повысить их производительность.

Дальнейшее повышение эффективности каналов передачи телевизионной информации авторы связывают с применением многопороговых декодеров и турбокодирования. А учитывая их эффективное использование в телекоммуникационных системах, можно надеяться, что именно рассмотренные технологии обеспечат новому стандарту телевидения высокую помехоустойчивость.

Список литературы

1. **Зубарев Ю. Б., Кривошеев М. И., Красносельский И. Н.** Цифровое телевизионное вещание. Основы, методы, системы. М.: Изд. Научно-исслед. института радио (НИИР), 2001. 568 с.

2. **Рид И., Соломон Г.** Полиноминальные коды над некоторыми конечными полями // Кибернетический сборник. 1983. Вып. 7. С. 74—79.

3. **Боуз Р. К., Рой-Чоудхури Д. К.** Об одном классе двоичных групповых кодов с исправлением ошибок // Кибернетический сборник. 1961. Вып. 2. С. 83—94.

4. **Берлекэмп Э. Р.** Техника кодирования с исправлением ошибок // ТИИЭР. 1980. Т. 68, № 5. С. 24—58.

5. **Месси Дж.** Пороговое декодирование: Пер. с англ. / Под ред. Э. Л. Блоха. М.: Мир, 1966. 208 с.

6. **Золотарев В. В., Овечкин Г. В.** Помехоустойчивое кодирование. Методы и алгоритмы: Справочник / Под ред. Ю. Б. Зубарева. М.: Горячая линия — Телеком, 2004. 126 с.

7. **Кларк Дж, Кейн Дж.** Кодирование с исправлением ошибок в системах цифровой связи: Пер. с англ. / Под ред. Б. С. Цыбакова. М.: Радио и связь, 1987. 392 с.

8. **Золотарев В. В.** Коды и кодирование. М.: Знание, 1990. 64 с.

УДК 621.391

С. Н. Агиевич, канд. техн. наук, ст. науч. сотр.,
вед. специалист,
ООО "Специальный технологический центр",
г. Санкт-Петербург,
e-mail: aserzhnic@mail.ru

Обработка сигналов методами сплайн-алгебраического гармонического анализа

На основе элементов теории сплайн-алгебраического гармонического анализа разработаны методы и реализующие их алгоритмы модуляции, демодуляции сигналов, отличающиеся повышенной скрытностью, помехозащищенностью, скоростью реализации. Представляются аналитические модели сигналов в базисах функций сплайн-характеров. Приводятся результаты анализа сравнительной вычислительной эффективности предложенных методов и алгоритмов. Обосновывается их помехозащищенность и даются предложения по их практическому использованию.

Ключевые слова: *сплайн-алгебраический гармонический анализ, алгоритмы модуляции и демодуляции сигналов, функции сплайн-характеров*

Введение

Класс дискретных функций значительно шире экспоненциального, который в настоящее время активно используется для формирования и обработки радиосигналов. Однако дискретная природа указанных функций ограничивает их практическое применение в прикладных задачах радиотехники. Указанное обстоятельство стимулировало поиск новых подходов к синтезу сигналов, объединяющих в себе преимущества их дискретного и непрерывного представлений. В настоящей работе рассматриваются вопросы обработки сигналов, сформированных в базисах функций сплайн-характеров (БФСХ). Предлагаемые подходы базируются на методах теории сплайн-алгебраического гармонического анализа, находящего все более широкое практическое применение.

Определение базисных функций сплайн-характеров

Пусть имеется пространство функций $L(H, K)$, областью определения которых является абелева группа H, а областью значений — кольцо K. В данном пространстве аналогами комплексных экспонент выступают характеры $\chi(n, k)$, которые образуют ортонормированный базис [1]. Характеры $\chi(n, k)$, определенные на конечном отрезке, называются χ-функциями. Здесь n — номер функции в базисе характеров (аналог порядкового номера гармоники в гармоническом базисе), k — текущее значение дискретного отсчета функции характера (аналог временных отсчетов для гармонических функций).

Дискретная природа характеров $\chi(n, k)$ изначально ограничивала их применение в интересах синтеза радиосигналов. Однако предложенный в работе [2] подход, основанный на применении сплайнов для сглаживания разрывных функций, позволил принципиально изменить взгляды на данный вопрос. Действительно, в пространстве $L(H, K)$ периодических сплайнов G_n^p, где p — порядок сплайна, сигнал $S^p(t)$ можно разложить следующим образом [3]:

$$S^p(t) = \frac{1}{N}\sum_k q_k M^p(t \underset{\mu}{\Theta} t_k) = \sum_n c_n^* \hat{\chi}_n^p(t), \qquad (1)$$

где N — объем выборки (длина функции); q_k — коэффициенты сплайна (здесь и далее k — текущий номер); $M^p(t)$ — базисный B-сплайн; t — текущая координата времени; t_k — дискретные отсчеты времени; μ — модуль представления чисел; c_n^* — спектральные коэффициенты в базисе $\hat{\chi}_n^p(t)$, которые будем называть ортонормированными сплайн-характерами; * — указывает на принадлежность к ортонормированному базису; $\underset{\mu}{\Theta}$ — сдвиг по модулю μ.

Значение c_n^* будем рассчитывать по формуле

$$c_n^* = F_n(z)\sqrt{u_n^{2p}}/u_n^p, \qquad (2)$$

где $F_n(z)$ — преобразование Фурье (ПФ) в базисе характеров над множеством z отсчетов сигнала; u_n^{2p} — дискретное ПФ от базисного B-сплайна $M^p(t)$; $2p$ — порядок B-сплайна, в 2 раза больший, чем p, причем коэффициент 2 указывает превышение порядка числителя над знаменателем.

Здесь $\hat{\chi}_n^p(t)$ представляют ортонормированный базис пространства G_n^p:

$$\hat{\chi}_n^p(t) = m_n^p(t)/\sqrt{u_n^{2p}}, \qquad (3)$$

где $m_n^p(t)$ — базисные функции в пространстве G_n^p.

Базисные функции $m_n^p(t)$ будем определять следующим образом:

$$m_n^p(t) = \frac{1}{N}\sum_k \overline{\chi}(n, k) M^p(t \underset{\mu}{\Theta} t_k), \qquad (4)$$

где $\chi(n, k)$ — характеры группы H.

Дискретное ПФ от базисного B-сплайна определим как

$$u_n^p = F_n(M^p) = \frac{1}{N}\sum_k \overline{\chi}(n, k) M^p(t_k), \qquad (5)$$

где $t_k = \left(\frac{p}{2} + k\right)/N$; $\overline{\chi}(n, k)$ — значение, комплексно-сопряженное с $\chi(n, k)$.

Аналитические модели радиосигналов в БФСХ

Рассмотрим аналитические модели радиосигналов в БФСХ на примере сигнала с квадратурной амплитудной манипуляцией (КАМ). Пусть $\text{ъ}_n^p(t)$ — манипулируемое колебание. Тогда аналитическую модель сигнала КАМ представим как

$$S_i^p(t) = A_i(t)\text{ъ}_n^p(t, \phi_i(t)),$$
$$i = 1, ..., R,\ 0 \leqslant t \leqslant T,\ T \leqslant \tau(H, K) \leqslant T_c. \qquad (6)$$

Здесь $A_i(t)$ — амплитуда сигнала; $\text{ъ}_n^p(t, \phi_i(t))$ — базисная функция из пространства G_n^p с номером n (для синтезируемых сигналов физический смысл номера базисной функции — частота в БФСХ); R — размерность сигнала (число значений фаз или амплитуд, используемых для его формирования); T — длительность сигнального символа; $\phi_i = 2\pi i/R$ — фаза.

Синтез сигналов в пространстве $L(H, K)$ предполагает, что $\tau(H, K)$ — длительность передачи информации в пределах фиксированного базиса, а T_c — длительность передачи информации.

Поскольку в сигналах КАМ информация заложена одновременно в изменении значений амплитуды и фазы, то, приняв их в качестве константы в формуле (6), можно получить, соответственно, аналитические модели для сигналов фазовой (ФМ) и амплитудной манипуляции (АМ) в БФСХ:

$$S_i^p(t) = A\text{ъ}_n^p(t, \phi_i(t)); \qquad (7)$$

$$S_i^p(t) = A_i(t)\text{ъ}_n^p(t, \phi). \qquad (8)$$

При фиксировании и амплитуды, и фазы приходим к аналитической модели сигнала частотной манипуляцией (ЧМ)

$$S_n^p(t) = A\text{ъ}_n^p(t, \phi). \qquad (9)$$

Аналогичным образом путем обобщения можно получить аналитические описания в БФСХ и для других типов сигналов. В частности, аналитическая модель сигналов OFDM в БФСХ будет иметь следующий вид:

$$S^p(t) = \sum_{i=0}^{R-1} \mathrm{Re}\left[\sqrt{\frac{2}{T}}\, A_i \text{ъ}_n^p(t, \phi_i)\right]. \qquad (10)$$

Следует заметить, что соответствующий выбор H, K и p позволяет получить из обобщенных моделей (6)—(10) аналитические описания сигналов в базисе экспоненциальных функций (ЭФ).

Методы модуляции радиосигналов в БФСХ

Важной составной частью модулятора является генератор, поскольку именно он определяет базис формирования радиосигналов. Следовательно, если использовать генератор сплайн-характеров, то можно синтезировать сигналы в БФСХ. В частности, на рис. 1 представлена структурная схема формирования сигналов КАМ в БФСХ.

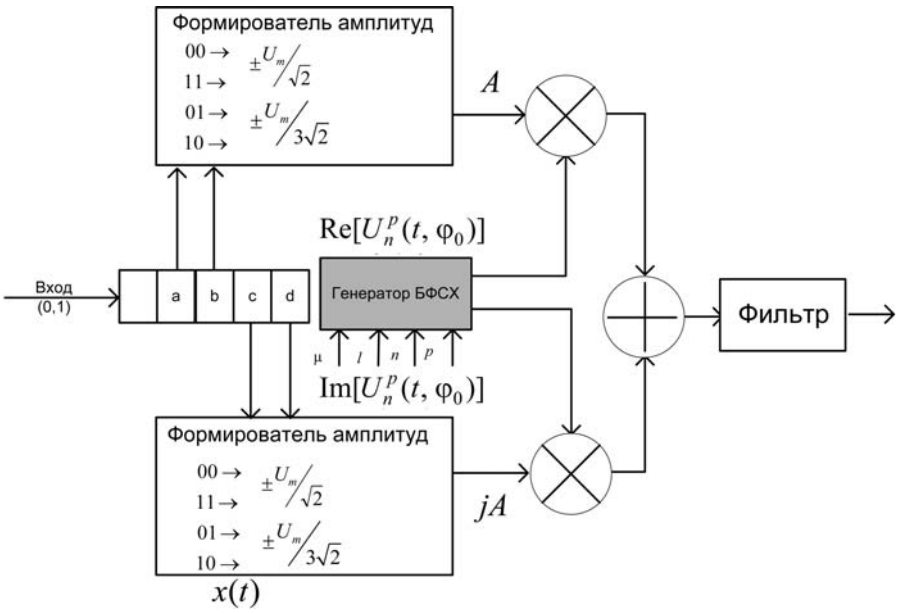


Рис. 1. Структурная схема модулятора сигнала КАМ на основе БФСХ

На рис. 2—4 показан принцип синтеза сигнала КАМ-16 на основе предложенного модулятора. Здесь в качестве манипулированного колебания использовался базис сплайн-Понтрягина — Виленкина — Крестенсона (СПВКФ), являющийся частным случаем БФСХ [4, 5].

На рис. 2 изображен манипулированный сигнал на выходе генератора БФСХ с параметрами $\mu = 4$, $l = 2$, $p = 4$, $n = 7$. На рис. 3 показан модулирующий первичный сигнал (пунктиром показана его мнимая часть). На рис. 4 представлен результирующий сигнал КАМ-16, сформированный в БФСХ.

Учитывая, что для синтеза сигналов используется генератор БФСХ, в качестве параметров,

определяющих их свойства, следует выделить: $L(H, K)$ — пространство, определяемое типом группы и кольца; μ — модуль представления чисел; l — число разрядов представления числа степени по модулю μ; p — порядок сплайна, используемого для построения БФСХ.

В зависимости от значений входных параметров задающий генератор формирует одну из заданных функций выбранного базиса.

Так, в табл. 1 и 2 представлены результаты, характеризующие вычислительные затраты (число процедур) на реализацию операций цифровой обработки сигналов (ЦОС) при расчете быстрого преобразования Фурье (БПФ) и фильтрации (параметры СПВКФ: $\mu = 2$ и $\mu = 4$).

Анализ представленных результатов позволяет сделать вывод о достаточной вычислительной эффективности предложенного подхода. Более того, с увеличением длины обрабатываемого сигнала быстродействие операций цифровой обработки в базисах СПВКФ будет возрастать.

Модулятор (см. рис. 1) можно рассматривать как обобщающий для формирования сигналов ФМ, ЧМ и АМ. Так, для синтеза сигналов АМ (рис. 5) достаточно использовать только одну из ветвей

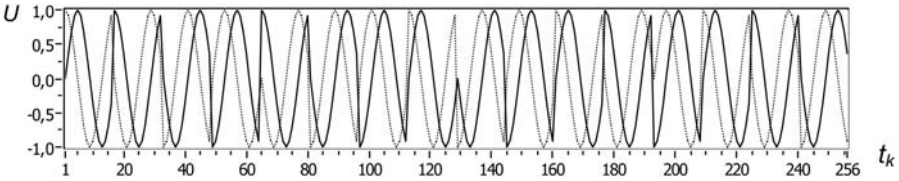


Рис. 2. Манипулируемый сигнал на выходе генератора СВКФ

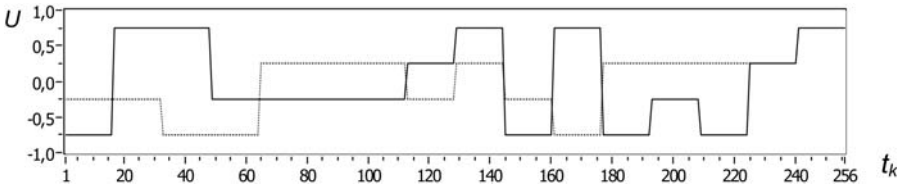


Рис. 3. Модулирующий сигнал на входе устройства

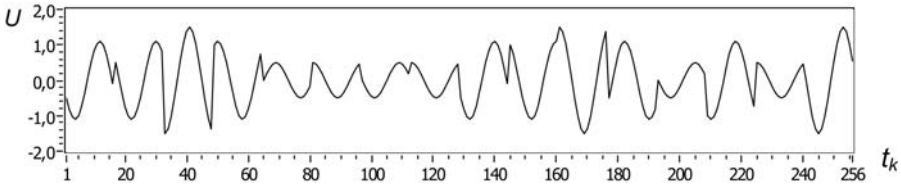


Рис. 4. Результирующий сигнал КАМ-16 на выходе модулятора

Таблица 1

Число процедур в базисах ЭФ и СВКФ при $\mu = 4$

N	Число операций ($\mu = 4$)				Выигрыш
	БПФ в ЭФ	БПФ в СПВКФ	Фильтрация в ЭФ	Фильтрация в СВКФ	
64	384	256	832	576	1,4444444
128	896	535	1920	1198	1,6026711
256	1024	552	2304	1360	1,6941176
512	4608	2355	9728	5222	1,8628878
1024	10240	4681	21504	10386	2,0704795
2048	22528	9557	47104	21162	2,2258766
4096	49152	19456	102400	43008	2,3809524

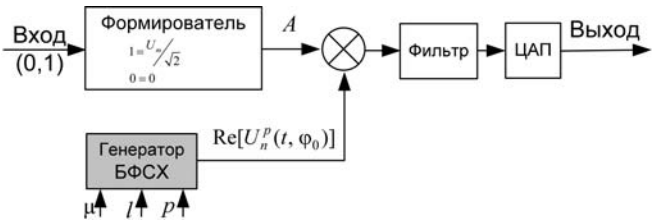


Рис. 5. Структурная схема модулятора сигнала АМ в БФСХ

Таблица 2

Число операций ЦОС в базисах ЭФ и СВКФ при $\mu = 2$

N	Число операций ($\mu = 2$)				Выигрыш
	БПФ в ЭФ	БПФ в СВКФ	Фильтрация в ЭФ	Фильтрация в СВКФ	
64	384	192	832	448	1,857143
128	896	407	1920	943	2,037037
256	1024	410	2304	1075	2,142857
512	4608	1440	9728	3392	2,867925
1024	10240	2340	21504	5704	3,769938
2048	22528	5120	47104	12288	3,833333
4096	49152	10923	102400	25941	3,947368

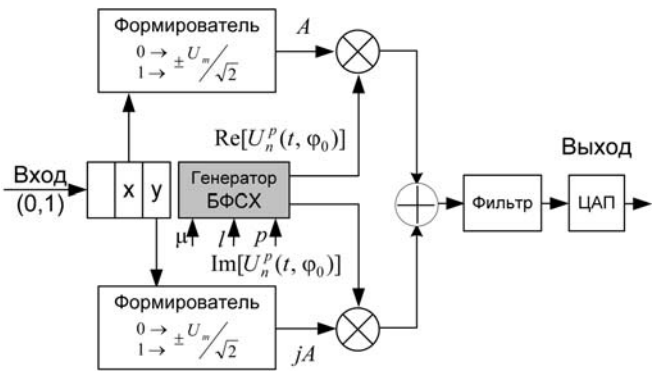


Рис. 6. Структурная схема модулятора сигналов ФМ в БФСХ

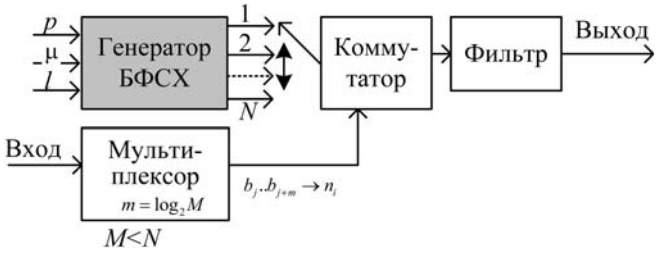


Рис. 7. Структурная схема модулятора сигнала ЧМ в БФСХ

структурной схемы, предназначенной для модуляции сигналов КАМ (здесь ЦАП — цифро-аналоговый преобразователь).

Основное отличие принципа синтеза сигналов АМ от КАМ связано с определением числа значений формируемых уровней.

Очевидна и общность принципов построения модуляторов сигналов КАМ и ФМ (рис. 6). Отличия лишь в том, что для сигналов ФМ значения уровней формируются в соответствии с позиционностью фазы.

Анализ подхода к формированию сигналов ЧМ в СВКФ [5] показывает, что его тоже можно рассматривать с позиций синтеза сигналов КАМ (рис. 7), только для многопозиционного случая (блок мультиплексора в структурной схеме).

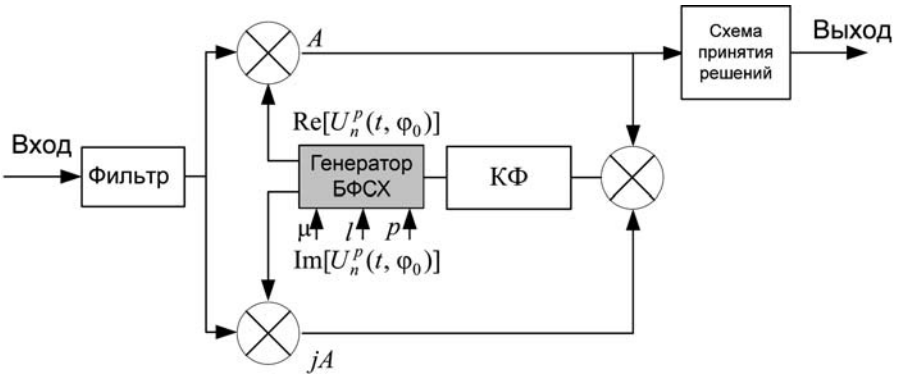


Рис. 8. Структурная схема демодулятора сигналов КАМ в БФСХ

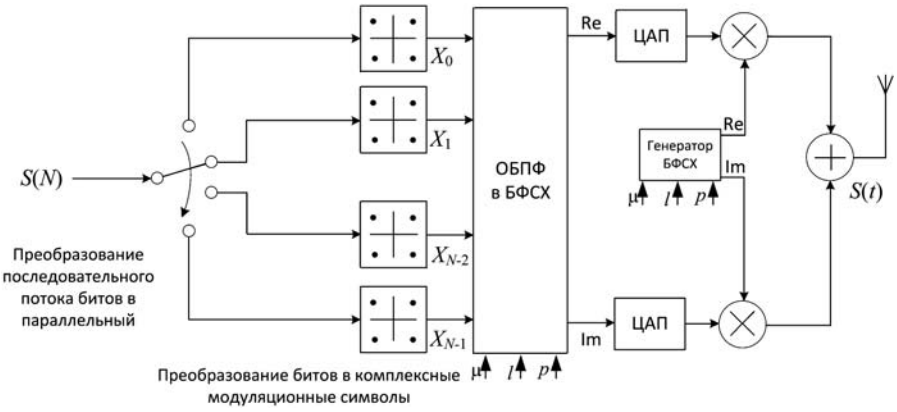


Рис. 9. Структурная схема модулятора сигнала OFDM с использованием БФСХ

Методы демодуляции радиосигналов в БФСХ

Процедуры демодуляции сигналов в БФСХ реализуются в обратном модуляции порядке. Вариант структурной схемы демодулятора сигналов КАМ в БФСХ представлен на рис. 8. В его основу положена синфазно-квадратурная схема.

Основой предложенной схемы является генератор БФСХ. Наиболее вычислительно емкими являются процедуры фильтрации и корреляции с интерполяцией. В частности, переход к частному случаю БФСХ — базису СПВКФ — позволяет существенно сократить объем вычислений. Важную роль в алгоритме играет процедура нахождения корреляционной функции. Это связано с тем, что при решении задачи обработки сигналов необходим поиск компромисса между значением частоты дискретизации и качеством демодуляции. Повышение частоты дискретизации позволяет с большой точностью определять интервал оптимальной выборки, но ведет к существенному увеличению объемов вычислений. Снижение качества может составлять до 3...5 дБ [6].

В то же время предлагаемый подход позволяет без повышения частоты дискретизации обеспечить требуемую точность определения начала уникального слова и при этом не потерять в помехоустойчивости демодуляции (выигрыш может достигать до 26...43,8 % потребного вычислительного ресурса).

Как и в случае модуляции, данный алгоритм является обобщенным для процедур демодуляции сигналов АМ, ФМ и ЧМ, что делает его универсальным по отношению к широкому классу узкополосных сигналов в БФСХ.

Модуляция и демодуляция сигналов OFDM в БФСХ

Рассмотрим возможность реализации методов модуляции и демодуляции сигналов OFDM в БФСХ. Если придерживаться подхода в замене генератора несущей частоты на источник колебаний БФСХ в классической смене на основе ЭФ, то результирующее устройство будет иметь вид, представленный на рис. 9.

Аналогичным образом можно построить и устройство демодуляции (рис. 10).

Согласно предложенным схемам при соответствующем выборе базиса можно получить существенный выигрыш вычислительного ресурса. Он определяется

числом операций, необходимых для реализации прямого и обратного ПФ в выбранных базисах.

Оценка помехоустойчивости и скрытности сигналов в БФСХ

Для оценки помехоустойчивости разработанных методов модуляции и демодуляции сигналов было проведено имитационное моделирование. В эксперименте использовались четырехпозиционные сигналы ФМ, сформированные в базисах СПВКФ как частного случая БФСХ, и дискретных экспоненциальных функций. В качестве деструктивного воздействия рассматривалась репитерная помеха. В результате было определено, что выигрыш в помехозащищенности для ФМ сигнала в базисе СПВКФ может достигать 2...3 дБ. Физическая сущность выигрыша заключается в том, что в отличие экспоненциального базиса помеха, полученная путем классического сдвига сигнала ФМ в базисе СПВКФ, не является для него оптимальной.

Кроме того, анализ полученных выражений для аналитических моделей (6)—(10) показывает, что число БФСХ теоретически бесконечно, поэтому всегда имеется возможность в изменении базиса в процессе работы. В результате возрастает структурная скрытность процесса передачи информации по сравнению с классическим подходом. Изучения данных свойств определяет перспективу дальнейших исследований.

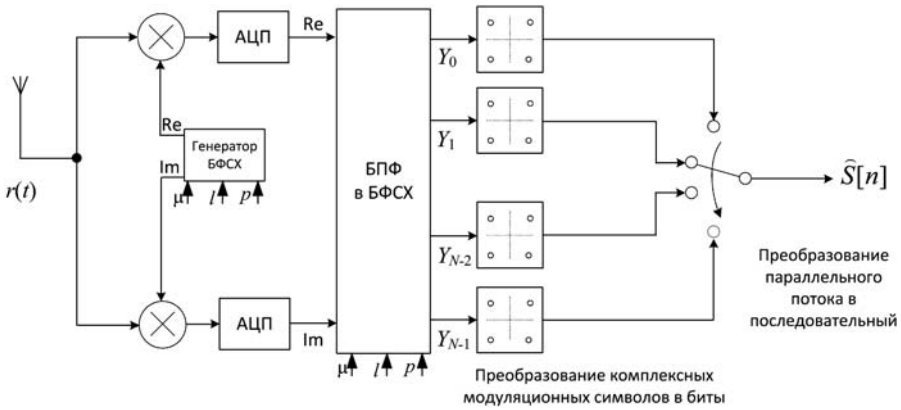


Рис. 10. Структурная схема демодулятора сигнала OFDM с использованием БФСХ

Заключение

Таким образом, разработанные методы и реализующие их алгоритмы модуляции, демодуляции радиосигналов в БФСХ отличаются повышенной скрытностью, помехозащищенностью и быстродействием реализации. Это открывает новые перспективы в области цифровой обработки.

Полученные результаты анализа сравнительной вычислительной эффективности предложенных подходов позволяют заключить, что в зависимости от длины реализации, выбранного базиса и модуля представления чисел общий выигрыш в снижении потребного числа операций может достигать до 4 раз. А отказ от необходимости использования более высокой частоты дискретизации в целях снижения ошибки определения длительности оптимальной выборки на интервале символа позволяет не снижать качество демодуляции.

Результаты анализа помехозащищенности предложенных методов при воздействии репитерной помехи также указывают на их эффективность до 2...3 дБ по отношению к классической обработке в экспоненциальном базисе.

Следовательно, практическая реализация процедур обработки сигналов в БФСХ открывает новые перспективы в построении средств связи.

Список литературы

1. **Вариченко Л. В., Лабунец В. Г., Раков М. А.** Абстрактные алгебраические системы и цифровая обработка сигналов. Киев: Наукова Думка, 1986. 247 с.
2. **Завьялов Ю. С., Квасов Б. И., Мирошниченко В. Л.** Методы сплайн-функций. М.: Наука, 1980. 352 с.
3. **Zheludev V. A.** Periodic splines, harmonic analysis and wavelets in Signal and image representation in combined spaces, wavelet // Eds Y. Y. Zeevi and R. Coifman. San Diego; Academic Press. 1998. P. 477—509.
4. **Агиевич С. Н.** Сплайн-Виленкина—Крестенсона функции в представлении сигналов // Научное приборостроение. 2002. Т. 12, № 1. С. 79—89.
5. **Агиевич С. Н., Дворников С. В., Гусельников А. С.** Описание сигналов в базисах функций сплайн-Виленкина—Крестенсона // Контроль-Диагностика. 2009. № 3. С. 52—57.
6. **Proakis J. G.** Digital communication. NY.: McGrawHill. 2008. 1150 с.

УДК 004.912: 004.023

Р. М. Алгулиев[1], член-корр. НАН Азербайджана, д-р техн. наук, проф.,
Р. М. Алыгулиев[2], д-р техн. наук,
Ф. С. Гянджалиев[2], докторант,
Институт Информационных Технологий НАН Азербайджана
Азербайджан, Баку,
[1]rasim@science.az;
[2]r.aliguliyev@gmail.com

Извлечение социальных сетей научных исследователей в Веб с использованием информации из нескольких источников

В настоящее время большинство научных исследователей размещают свои работы в Веб. Это может быть использовано для извлечения полезной информации об их прошлой, настоящей и будущей деятельности. По этой причине социальная сеть научных исследователей имеет важное значение для научного общества. Социальная сеть может быть построена на основе информации, полученной из разных источников в Веб. В настоящей работе предложен метод извлечения социальных сетей научных исследователей на основе информации, полученной из разных источников.

***Ключевые слова**: неоднородная социальная сеть, социальная сеть научных исследователей, многокритериальный выбор*

Введение

Появление в конце XX столетия Всемирной паутины (Веб) поставило множество новых задач перед исследователями социальных сетей. Прежде всего, это послужило причиной того, что все традиционные методы работы с социальными сетями были пересмотрены. Поскольку анализ социальных сетей в основном проводился на множестве малой группы узлов, применение тех же самых методов в Веб оказалось не столь простым. В большинстве случаев было практически невозможно анализировать сеть, состоящую из миллионов пользователей, учитывая, что тщательный анализ требует построения хотя не всей, но большей части сети. Другой трудной задачей был сбор информации о большой группе участников. Кроме того, в немалом числе случаев существуют участники сети, которые не публикуют о себе информацию. Такие виды социальных сетей оказались наиболее сложными для анализа [1].

Социальная сеть представляет собой набор узлов и отношений, соединяющих их. Узлами могут быть люди, группы людей, какие-либо объекты, ресурсы и т. д. Отношениями могут быть любые связи, соединяющие их. При анализе социальных сетей на Веб в большинстве случаев узлами являются пользователи какого-либо рода.

Существуют разные методы для построения социальных сетей в Веб. Был предложен метод для построения социальной сети на основе информации, полученной из новостных статей, публикуемых на разных языках [2]. В другой работе описывается механизм построения социальной сети из информации, извлеченной из журнальных файлов, так называемых разделяемых рабочих пространств [3]. Существует метод построения социальной сети исторических персонажей с помощью информации, полученной из неструктурированных данных [4]. Был также предложен метод извлечения социальной сети на основе информации, сгенерированной случайными коммуникациями [5]. Метод *Referral Web* считается одним из первых инструментов, позволяющих построение социальной сети в Веб [6]. *Flink* представляет собой другую, более позднюю, систему построения социальной сети [7]. Некоторые исследователи описывают метод вычисления степени связей между пользователями из входящих сообщений электронных адресов пользователей [8]. В настоящее время *Polyphonet* представляет собой мощный инструмент для визуализации социальных сетей научных исследователей [9]. В работе [10] представлен метод для построения социальной сети научных исследователей. Существует также метод, выявляющий связи между пользователями на основе информации, полученной с помощью коммуникаций через электронные почтовые адреса [12]. В последней работе, после извлечения сети, также выявляются клики с помощью метода, описанного в работе [13].

Одной из известных областей в анализе социальных сетей являются сети людей, у которых совпадают сферы интересов. Например, некоторые пользователи в Веб желают найти пользователей, которые опубликовали некоторое число статей по определенной теме, или желают найти наиболее популярного актера в какой-либо области. Авторы работы [14] предлагают подход для построения социальной

сети людей с совпадающими сферами интересов, представляя пользователя с помощью контента его сайта. Некоторые работы посвящены выявлению связей между пользователями из FOAF (Friend Of A Friend) документов [15—17]. Выбор меры сходства может оказывать влияние на результат ранжирования узлов сети, что продемонстрировано в работе [18]. Проблемы выявления сообществ, являющиеся одной из важных областей в анализе социальных сетей, также рассмотрены в некоторых работах [20—22].

На данный момент существует немалое число методов построения социальных сетей, но многие из них предполагают, что между узлами существует одна единственная связь. На самом деле во многих группах существует больше одной связи между ее участниками. Во-первых, каждая из связей может быть ключевой в определенной задаче, и при предположении, что в данной группе существует только одна связь, важная информация может быть утеряна. Во-вторых, большинство алгоритмов применимо в сетях с малым числом участников.

Социальная сеть научных исследователей представляет собой другую интересную область на Веб. Научные исследователи могут иметь разные связи между собой: они могут совместно опубликовать одну или несколько статей, могут участвовать в одной конференции, быть членами одной научной организации, участвовать в одном проекте и т. д. Основная проблема с методами в этой области состоит в том, что они не охватывают всю научную сеть [9, 10].

1. Построение неоднородной социальной сети исследователей на основе информации из нескольких источников

В теории социальных сетей существуют два вида сетей по отношению к связям между их узлами: однородные и многородные. Однородными называют такие социальные сети, в которых существует только одна связь между узлами; в многородных сетях между узлами сети присутствуют несколько отношений. Например, социальная сеть студентов какого-либо факультета может быть многородной, если рассматривать присутствие в сети связи как студентов одного и того же факультета, так и семейных связей, если таковые есть, например двое студентов имеют родственные связи. В таком случае построенная социальная сеть будет многородной, поскольку студентов будут связывать несколько отношений. Очевидно, что анализ многородных социальных сетей является более сложным, чем однородных сетей.

Объем информации в Веб дает основание полагать, что данные, полученные для одного и того же множества участников из разных источников, очень редко могут обеспечить схожую картину. Предположим, что для данной группы участников мы построили социальную сеть из информации, полученной только из одного источника. В таком случае может быть, что, например, какие-то два участника не имеют общей связи. Однако, построив сеть тех же самых участников на основе информации из другого источника, мы можем обнаружить, что те же самые участники имеют между собой связь. Это, в свою очередь, означает, что чем больше источников мы используем для построения социальной сети, тем более реальной у нас будет картина.

В данном случае мы рассматриваем только ненаправленные сети. Предполагается, что участники сети заранее заданы. Хотя число источников информации и участников сети может быть любым, здесь мы рассматриваем случай трех источников данных: s_1, s_2 и s_3, из которых можно извлечь информацию для построения сети для заданного множества участников; также предположим, что имеется множество участников a_1, a_2, a_3, a_4 и a_5.

Предположим, что из заданных трех источников мы извлекли информацию, используя которую построили следующие социальные сети (рис. 1, 2, 3 соответственно).

Ниже представлены веса ребер для каждой сети (табл. 1).

После того как мы построили три однородные сети, наша цель состоит в том, чтобы объединить эти сети в одну неоднородную сеть с помощью метода, описанного в работе [19]. Другими словами, наша цель — для заданных участников построить единую неоднородную сеть. Вес ребра в результирующей сети есть сумма весов соответствующих ребер в сетях, умноженных на коэффициент, показывающий важность источника. Сумма этих ко-

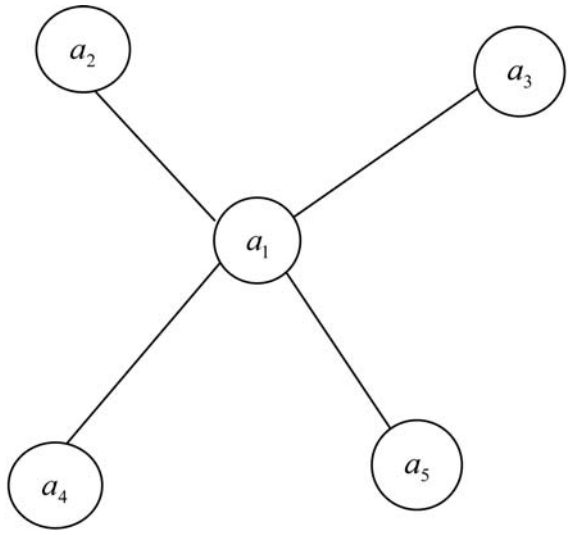


Рис. 1. Социальная сеть участников из первого источника

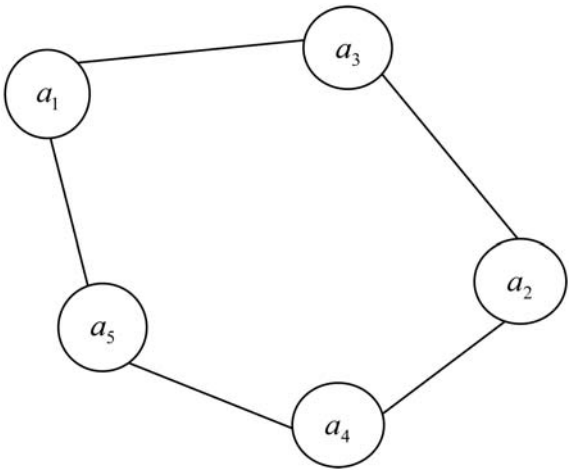


Рис. 2. Социальная сеть участников из второго источника

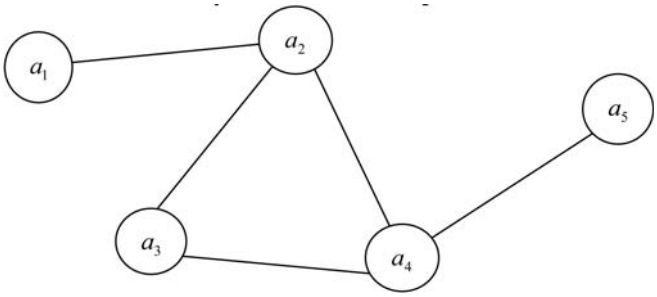


Рис. 3. Социальная сеть участников из третьего источника

Таблица 1

Веса ребер для каждой из сетей

Ребро	Вес в источнике данных		
	s_1	s_2	s_3
w_{12}	0,2	0,0	0,2
w_{13}	0,1	0,5	0,0
w_{14}	0,3	0,0	0,0
w_{15}	0,3	0,8	0,0
w_{23}	0,0	0,7	0,3
w_{24}	0,0	0,4	0,1
w_{25}	0,0	0,0	0,0
w_{34}	0,0	0,0	0,8
w_{35}	0,0	0,0	0,0
w_{45}	0,0	0,9	0,9

эффициентов равна единице. Это отношение запишем следующим образом:

$$w^{ij} = z_1 w_1^{ij} + z_2 w_2^{ij} + z_3 w_3^{ij}, \qquad (1)$$

где w^{ij} — вес ребра, соединяющего участников i и j результирующей сети, z_k — коэффициент, показывающий важность источника k; w_1^{ij}, w_2^{ij}, w_3^{ij} — веса ребер, соединяющих участников i и j из трех источников. Кроме того, $z_1 + z_2 + z_3 = 1$.

2. Вычисление весов источников информации

На данном этапе мы построили единую неоднородную социальную сеть, которая объединяет в себе несколько однородных сетей. Мы также определили формулу для нахождения весов ребер этой сети. Сейчас нам необходимо найти неизвестные коэффициенты в формуле (1). Эти коэффициенты содержат информацию о том, какая часть от каждой из сетей будет результирующей.

Для нахождения этих коэффициентов мы используем метод, описанный в работе [23]. В этой работе предлагается метод нахождения лучшей альтернативы из нескольких возможных на основе того, что каждый критерий альтернатив представляется в виде нечеткого множества. А именно, предполагается, что на заданном множестве альтернатив $S = \{s_1, s_2, ..., s_n\}$ задано множество критериев $C = \{c_1, c_2, ..., c_m\}$. Каждый критерий представляется в виде следующего нечеткого множества:

$$c_j = \left\{ \frac{w_1^{(j)}}{s_1}, \frac{w_2^{(j)}}{s_2}, \ldots, \frac{w_n^{(j)}}{s_n} \right\}, j = 1, 2, ..., m.$$

Элементы $w_i^{(j)}$ представляют собой числа в интервале [0, 1], которые можно учитывать как веса альтернатив относительно критериев c_j, сумма которых по каждому из критериев равна единице, т. е.

$$w_1^{(j)} + w_2^{(j)} + ... + w_n^{(j)} = 1, j = 1, 2, ..., m.$$

Согласно принципу Белмана—Заде наилучшую альтернативу S_{opt} следует искать в пересечении нечетких множеств-критериев, т. е. в множестве $D = c_1 \cap c_2 \cap ... \cap c_m$. Согласно [23], как наилучшую альтернативу S_{opt} необходимо выбирать альтернативу $S_{\text{opt}} \in D$ с максимальным весом

$$w(s_{\text{opt}}) = \max_{i = 1, 2, ..., n} \min\{w_i^{(1)}, w_i^{(2)}, ..., w_i^{(m)}\},$$

так как в теории нечетких множеств можно воспользоваться заменой $\cap \to \min$, из которой, в свою очередь, вытекает, что множество хороших решений можно представить так:

$$D = \left\{ \frac{\min\left\{w_1^{(1)}, \ldots, w_1^{(m)}\right\}}{s_1}, \frac{\min\left\{w_2^{(1)}, \ldots, w_2^{(m)}\right\}}{s_2}, \ldots, \frac{\min\left\{w_n^{(1)}, \ldots, w_n^{(m)}\right\}}{s_n} \right\}.$$

Для нахождения весов автор пользуется рангом, который тем выше, чем выше надежность альтернативы. Поэтому имеет место следующее отношение:

$$\frac{w_1}{r_1} = \frac{w_2}{r_2} = \ldots = \frac{w_l}{r_l} = \ldots = \frac{w_n}{r_n}.$$

Если s_l — наихудшая альтернатива (по критерию c_j) с весом w_l и рангом r_l, то вышеуказанное соотношение можно записать так:

$$w_1 = r_1\frac{w_l}{r_l},\ w_2 = r_2\frac{w_l}{r_l},\ \ldots,\ w_n = r_n\frac{w_l}{r_l},$$

и его можно переписать в следующем виде, учитывая, что сумма весов альтернатив по определенному критерию равна единице:

$$w_l = \frac{1}{\frac{r_1}{r_l} + \frac{r_2}{r_l} + \ldots + \frac{r_n}{r_l}} = \frac{1}{\sum\limits_{i=1}\frac{r_i}{r_l}}.$$

Здесь, как также указывает автор, соотношение $\frac{r_i}{r_j}$ берется из 9-балльной шкалы Саати, в которой это отношение равно одному из чисел в интервале [1, 8] в зависимости от того, насколько альтернатива s_i лучше альтернативы s_j. Используя последнее соотношение для w_l, можно определить веса альтернатив с помощью отношения ранга альтернативы к рангу наихудшей альтернативы.

Следовательно, для использования этого метода нам необходимо предположить существование еще одного условия: построенные социальные сети должны иметь какие-либо критерии. Предположим, что сети имеют следующие критерии: средний вес ребра, плотность и среднее расстояние. В следующей табл. 2 представлены значения критериев для каждой из сетей.

Таблица 2

Значения критериев для каждой из сетей

Источник	Значения критерия		
	Средний вес ребра	Плотность	Среднее расстояние
s_1	0,225	0,660	0,460
s_2	0,660	0,480	0,460
s_3	0,460	0,985	0,712

Далее мы определяем веса источников согласно указанному методу.

1. Критерий *средний вес ребра* (ATS). Наихудшая альтернатива s_1. Вес наихудшей альтернативы

$$z_1 = \frac{1}{\frac{r_1}{r_1} + \frac{r_2}{r_1} + \frac{r_3}{r_1}} = \frac{1}{1+5+3} = \frac{1}{9}.$$

Веса остальных альтернатив:

$$z_2 = \frac{r_2}{r_1}z_1 = \frac{5}{9},\ z_3 = \frac{r_3}{r_1}z_1 = \frac{1}{3}.$$

1. Критерий *плотность* (D). Наихудшая альтернатива s_2. Вес наихудшей альтернативы:

$$z_2 = \frac{1}{\frac{r_2}{r_2} + \frac{r_1}{r_2} + \frac{r_3}{r_2}} = \frac{1}{1+3+5} = \frac{1}{9}.$$

Веса остальных альтернатив:

$$z_1 = \frac{r_1}{r_2}z_2 = \frac{1}{3},\ z_3 = \frac{r_3}{r_2}z_2 = \frac{5}{9}.$$

3. Критерий *среднее расстояние* (ADS). Наихудшая альтернатива $s_2(s_1)$. Вес наихудшей альтернативы:

$$z_2 = \frac{1}{\frac{r_2}{r_2} + \frac{r_1}{r_2} + \frac{r_3}{r_2}} = \frac{1}{1+1+3} = \frac{1}{5}.$$

Веса остальных альтернатив:

$$z_1 = \frac{r_1}{r_2}z_2 = \frac{1}{5},\ z_3 = \frac{r_3}{r_2}z_2 = \frac{3}{5}.$$

Так же, как и в указанном методе, полученные веса альтернатив по каждому из критериев позволяют записать критерии в виде следующего нечеткого множества:

$$ATS = \left\{\frac{1/9}{s_1}, \frac{5/9}{s_2}, \frac{1/3}{s_3}\right\},$$

$$D = \left\{\frac{1/3}{s_1}, \frac{1/9}{s_2}, \frac{5/9}{s_3}\right\},$$

$$ADS = \left\{\frac{1/5}{s_1}, \frac{1/5}{s_2}, \frac{3/5}{s_3}\right\}.$$

Максимум из минимальных значений для каждого из источников, соответственно, показывает,

какая альтернатива лучшая по каждому из критериев, т. е. множество

$$R = \left\{ \frac{1/9}{s_1}, \frac{1/9}{s_2}, \frac{1/3}{s_3} \right\}$$

показывает, какой источник лучший, какой худший и какой наихудший: лучший тот, у которого числитель больше (в дробях со знаменателями s_1, s_2 и s_3).

Как было указано выше, наша цель состоит не в нахождении лучшего источника, а в объединении данных из каждого источника. Полученные выше значения мы могли бы использовать в качестве значений искомых коэффициентов, но в таком случае будет нарушено условие $z_1 + z_2 + z_3 = 1$. Поэтому для выполнения этого условия мы нормализуем элементы множества *R*. Таким образом, мы получим $z_1 = \frac{1}{5}$, $z_2 = \frac{1}{5}$ и $z_3 = \frac{3}{5}$, которые будут удовлетворять требуемому условию.

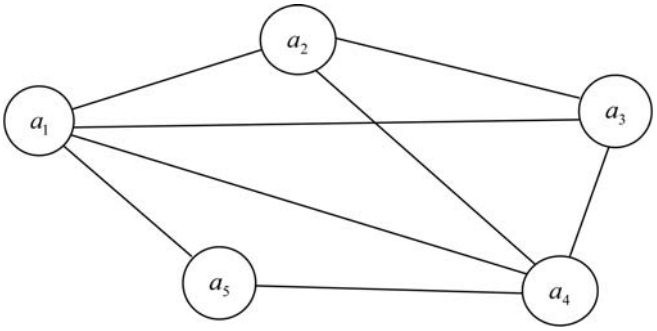


Рис. 4. Результирующая социальная сеть с весами ребер, показанных в табл. 3

Таблица 3

Веса ребер результирующей сети

Ребро	Значение веса
w_{12}	0,16
w_{13}	0,12
w_{14}	0,06
w_{15}	0,22
w_{23}	0,32
w_{24}	0,14
w_{25}	0,000
w_{34}	1,2
w_{35}	0,000
w_{45}	0,36

Таблица 4

Значения критериев для результирующей сети

Средний вес ребра	Плотность	Среднее расстояние
0,32	0,72	0,198

Далее мы используем формулу (1) для вычисления весов ребер результирующей сети (табл. 3). В результате получаем сеть, приведенную на рис. 4.

Значения критериев для результирующей сети даны в табл. 4.

Из полученных выше результатов интересно отметить некоторые факты. Участники a_2 и a_4 не имеют общей связи в первой и второй сетях, но соединены ребром в третьей сети. Также они соединены и в результирующей сети. Аналогично участники a_4 и a_5 не имеют прямой связи в первой сети, но имеют связь во второй и третьей сетях. Наш метод показал, что эти участники также обладают связью в результирующей сети. Это дает основание полагать, что наш метод не претерпевает информационной потери. Другим интересным фактом является то, что поскольку $z_3 = \frac{1}{3}$, то это, в свою очередь, означает, что третья сеть должна быть лучшей, в том смысле, что в ней должны быть скомбинированы все лучшие значения критериев. Как видно из табл. 2, для второго критерия третья сеть действительно является наилучшей, для первого критерия эта сеть не является ни наилучшей, ни наихудшей и, наконец, для третьего критерия сеть является наихудшей, что не соответствует полученному значению коэффициента.

Заключение

Разные виды социальных сетей научных исследователей могут быть извлечены с помощью информации, доступной в Веб. Большинство современных методов для построения социальных сетей исследователей применимо к специфичным типам данных. Пренебрежение одной из сетей, полученных таким образом, может привести к потере важной информации. Мы представили метод, который можно использовать для объединения нескольких однородных сетей в одну неоднородную сеть. В будущем мы покажем некоторые применения описанного метода.

Список литературы

1. **Golbeck J.** Web-based social networks: a survey and future directions // Technical report. University of Maryland at College Park. 2005. 14 p. URL: http://www.cs.duke.edu/courses/spring06/cps018s/class/SocialNetworksOverviewTR.pdf

2. **Poliqean B., Tanev H., Atkinson M.** Extracting and learning social networks out of multilingual news // Proc. of the Social Networks and Application Tools Workshop (SocNet-08). Skalica. Slovakia. 2008. P. 19—21.

3. **Nasirifard P., Peristeras V., Hayes C., Decker S.** Extracting and utilizing social networks from log files of shared workspaces // Proc. of the 10th IFIP Working Conference on Virtual Enterprises. Thessaloniki. Greece. 2009. P. 7—9.

4. **Geleijnse G., Korst J.** Creating a dead poets society: extracting a social network of historical persons from the web // Proc. of the 6th International and 2nd Asian Conference on Asian Semantic Web Conference. Busan. South Korea. 2007. P. 155—168.

5. **Nishimura T., Matsuo Y.** A method of social network extraction via internet and networked Sensing // Proc. of the 3rd International Conference on Networked Sensing Systems. Chicago. USA. 2006. URL: http://www-kasm.nii.ac.jp/papers/takeda/06/nishimura06inss.pdf

6. **Kautz H., Selman B., Shah M.** The hidden web // AI Magazine. 1997. V. 18, N 2. P. 27—35.

7. **Mika P.** Flink: semantic web technology for extraction and analysis of social networks // Journal of Web Semantics. 2005. V. 3, N 2. P. 211—223.

8. **Culotta A., Bekkerman R., McCallum A.** Extracting social networks and contact information from email and the web // Proc. of the 1st Conference on Email and Anti-Spam. California. USA. 2004. URL: http://ceas.cc/2004/176.pdf

9. **Matsuo Y., Mori J., Hamasaki M., Nishimura T., Takeda H., Hasida K., Ishizuka M.** POLYPHONET: an advanced social network extraction system from the web // Web Semantics: Science, Services and Agents on the World Wide Web. 2007. V. 5, N 4. P. 262—278.

10. **Tang J., Zang D., Yao L.** Social network extraction of academic researchers // Proceedings of the 7th IEEE International Conference on Data Mining. Omaha. USA. 2007. P. 292—301.

11. **Stroele V., Oliveira J., Zimbrao G., Souza J. M.** Mining and analyzing multi-relational social networks // Proc. of the 12th International Conference on Computational Science and Engineering. Vancouver. Canada. 2009. P. 711—716.

12. **Rowe R., Creamer G., Hershkop S., Stoflo S. J.** Automated social hierarchy detection through email and network analysis // Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Jose. USA. 2007. P. 109—117.

13. **Bron C., Kerbosch J.** Finding all cliques of an undirected graph // Communications of the ACM. 1973. V. 16, N 9. P. 575—577.

14. **Li Q., Wu Y. B.** People search: searching people sharing similar interests from the web // Journal of the American Society for Information Science and Technology. 2008. V. 59, N 1. P. 111—125.

15. **Finin T., Ding L., Zou L.** Social networking on the semantic web // The Learning Organization. 2005. V. 12, N 5. P. 418—435.

16. **Aleman-Meza B., Nagarajan M., Ramakrishnan C., Ding L., Kolari P., Sheth A., Arpinar I. B., Joshi A., Finin T.** Semantic analytics on social networks: experiences in addressing the problem of conflict of interest detection // Proc. of the 15th International World Wide Web Conference. Edinburgh. Scotland. 2006. P. 407—416.

17. **Goldbeck J., Rothstein M.** Linking social networks on the web with FOAF // Proceedings of the 23rd National Conference on Artificial Intelligence. Chicago. USA. 2008. P. 1138—1143.

18. **Alguliev R., Aliguliyev R., Ganjaliyev F.** Investigation the role of similarity measure and ranking algorithm in mining social network // Journal of Information Science. 2011. V. 37, N 3. P. 229—234.

19. **Ganjaliyev F.** Building a heterogeneous social network of academic researchers // Proc. of the Third International Conference of Problems of Cybernetics and Informatics. Baku. Azerbaijan. 2010. P. 179—182.

20. **Du N., Wu B., Pei X., Wang B., Xu L.** Community detection in large-scale social networks // Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Jose. USA. 2007. P. 16—25.

21. **Baumes J., Goldberg M., Magdon-Ismail M., Wallace W.** Discovering hidden groups in communication networks // Proc. of the 2nd NSF/NIJ Symposium on Intelligence and Security Informatics. Tucson. USA. 2004. URL: http://www.cs.rpi.edu/research/pdf/04-04.pdf

22. **Cai D., Shao Z., He X., Yan X., Han J.** Mining hidden community in heterogeneous social networks // Proc. of the 3rd International Workshop on Link Discovery. Chicago. USA. 2005. P. 58—65.

23. **Ротштейн А. П.** Нечеткий многокритериальный выбор альтернатив: метод наихудшего случая // Известия РАН. Теория и системы управления. 2010. № 3. С. 51—55.

Новые книги

Липаев В. В.
Очерки истории отечественной программной инженерии 1940-е — 80-е годы. — М.: СИНТЕГ. 2012. 245 с.

Описание отечественной программной инженерии начинается с появления в нашей стране электронных вычислительных машин (ЭВМ) и программирования в 1940-е — 60-е годы. Далее изложена история проектирования и производства отечественных ЭВМ, а также средств и систем автоматизации технологических процессов производства программных продуктов в 1960-е — 80-е годы. Подробно представлена история формирования основных компонентов программной инженерии в 1960-е — 70-е годы. Внимание акцентируется на особенностях решения сложных задач по государственным заказам и на создании программных продуктов для мобильных и бортовых ЭВМ реального времени. Особое внимание уделяется истории разработки методов моделирования динамических объектов и стендов для тестирования и испытаний комплексов программ в реальном времени. Изложены методы оценивания качества программных продуктов, рисков, дефектов и ошибок при их разработке, а также история формирования требований к профессиям и квалификации специалистов программной инженерии в 1970-е — 80-е годы. Рассмотрен анализ сложности программных комплексов реального времени и распределение ресурсов ЭВМ для таких комплексов, характеристики и методы оценивания качества их компонентов. Один из разделов посвящен истории формирования в 1980-е годы экономики программной инженерии, созданию средств технико-экономический анализа и экономическому обоснованию планов разработки крупных программных продуктов. Представлены реальные примеры их создания в 1960-е — 80-е годы для оборонных систем на основе методов программной инженерии.

Книга предназначена для специалистов по вычислительной технике и программной инженерии, программистов, студентов и аспирантов, интересующихся историей развития, успехами и проблемами отечественной науки и техники в этой области.

УДК 004.05; 004.052.3; 004.056

С. М. Авдошин, канд. техн. наук, проф.,
e-mail: savdoshin@hse.ru,
А. А. Савельева, канд. техн. наук, доц.,
Национальный исследовательский университет
"Высшая школа экономики"

Междисциплинарный подход к изучению информационной безопасности на основе анализа конкретных ситуаций[1]

Предлагается подход к преподаванию информационной безопасности, основанный на использовании метода кейс-стади для проведения практических занятий. Обосновывается целесообразность применения метода кейс-стади на примере его использования в рамках курсов "Организация и технология защиты информации" и "Технологии обеспечения информационной безопасности", читаемых студентам магистратуры и бакалавриата отделения программной инженерии НИУ "ВШЭ" с 2008 г. Данная работа восполняет отсутствие методических рекомендаций по использованию метода кейс-стади при преподавании в высших учебных заведениях дисциплин, связанных с защитой информации.

Ключевые слова: *информационная безопасность, кейс-стади, управление рисками, методические рекомендации*

Введение

Метод kase-stady (кейс-стади, метод конкретных ситуаций) представляет собой интерактивную технологию для обучения на основе реальных или вымышленных бизнес-ситуаций, способствующую не только усвоению знаний, но и формированию у слушателей аналитических навыков и умений разрешения проблемных ситуаций. Словарь [17] описывает kase-stady как "исследовательский проект, в котором в качестве предмета исследования выбирается единичный случай или несколько избранных примеров ... и определяется совокупность методов их изучения". Менее формальное определение предложено в работе К. Ф. Хэррайда [4]: "Case study — это история с образовательным подтекстом".

Метод конкретных ситуаций возник в начале XX века в Школе бизнеса Гарвардского университета, известной своими инновациями [13]. Распространение метода в мире началось в 70—80-е годы, тогда же метод получил известность и в СССР. Анализ ситуаций начал использоваться при обучении управленцев, в основном на экономических специальностях вузов, в первую очередь как метод обучения принятию решений. Значительный вклад в разработку и внедрение этого метода внесли Г. А. Брянский, Ю. Ю. Екатеринославский, О. В. Козлова, Ю. Д. Красовский, В. Я. Платов, Д. А. Поспелов, О. А. Овсянников, В. С. Рапоппорт и др.

Создание учебной среды, способствующей развитию навыков анализа информации, обнаружения связей между фактами и формирования гипотез, было признано особенно актуальным после публикаций западными исследователями статей о несоответствии уровня образования потребностям современного общества [3, 2, 6, 7].

Новая волна интереса к методу кейс-стади в России началась в 90-е годы в связи с ростом спроса на специалистов, умеющих действовать в ситуациях, связанных с риском или неопределенностью, анализировать проблемы и принимать обоснованные решения. Это привело к распространению практики использования метода кейс-стади в программах гуманитарных и экономических дисциплин, таких как политология, менеджмент, маркетинг, социология и т. д. (см., например, [12, 14, 15, 17]).

Одной из главных тенденций образования в области защиты информации является осознание необходимости использования принципов управления рисками при решении проблем, связанных с обеспечением информационной безопасности [1, 5]. Эта тенденция особенно ясно прослеживалась в условиях мирового финансового кризиса, так как с ростом числа "обиженных" в результате сокращений сотрудников увеличивалась вероятность совершения преступлений в отношении информационных ресурсов предприятий. Наиболее востребованными становятся специалисты по обеспечению информационной безопасности (ИБ), умеющие учитывать не только технические, но и организационные аспекты обеспечения ИБ. Это делает целесообразным применение для подготовки специалистов по защите информации метода кейс-стади, показавшего свою эффективность для развития навыков анализа

[1] Работа выполнена в рамках исследовательского проекта "Исследование и разработка методов оценки эффективности использования криптографических средств защиты информации в сфере бизнеса и финансов", поддержанного государственным грантом № П965 от 27 мая 2010 г.

ситуаций и принятия решений в условиях реального мира. Однако, как показал анализ российских и англоязычных публикаций, до сих пор отсутствуют методические разработки по использованию метода кейс-стади при преподавании в высших учебных заведениях дисциплин, связанных с защитой информации.

В данной работе предлагается подход к преподаванию информационной безопасности, основанный на использовании метода кейс-стади для проведения практических занятий. Описывается методика разработки кейс-стади по информационной безопасности и приводятся примеры, используемые при чтении курсов "Организация и технология защиты информации" и "Технологии обеспечения информационной безопасности" студентам магистратуры и бакалавриата отделения программной инженерии ГУ-ВШЭ. В заключении приводятся выводы о влиянии метода кейс-стади на ход учебного процесса и освоение студентами материалов курса, сделанные на основе опыта применения данного подхода.

Принципы использования кейс-стади

Использование метода кейс-стади позволяет увидеть неоднозначность решения проблем в реальной жизни. Цель метода кейс-стади [18] — научить студентов самостоятельно и в составе группы:

- анализировать информацию,
- сортировать ее для решения поставленной задачи,
- выявлять ключевые проблемы,
- генерировать альтернативные пути решения и оценивать их,
- выбирать оптимальное решение и формировать программы действий.

Для того чтобы учебный процесс на основе метода кейс-стади был эффективным, необходимы два условия: методика использования кейса в учебном процессе и хороший кейс. Согласно приведенным в работе [18] результатам опроса преподавателей, целью которого было выяснение основных проблем при внедрении метода кейсов (*case*) в учебный процесс, наибольшие трудности обычно вызывает *подбор подходящего кейса* и *недостаток методических разработок по использованию метода кейсов* при обучении конкретным дисциплинам. В частности, при попытке внедрить метод кейс-стади в учебную программу дисциплин по информационной безопасности мы столкнулись со следующими проблемами:

- отсутствие готовых кейсов в открытом доступе,
- невозможность использования материалов из консалтинговой практики,
- отсутствие методических рекомендаций по разработке.

В следующих разделах будет показано, как общие принципы построения и использования можно адаптировать при преподавании дисциплин, связанных с защитой информации. Мы приведем разработанные нами рекомендации по разработке кейсов по информационной безопасности, а также расскажем о выявленных нами особенностях использования кейс-стади в учебном процессе.

Структура кейс-стади

Существуют общепринятые правила построения кейс-стади. Для кейс-стади, разработанного в Гарварде, типичной чертой является сознательная перегруженность информацией. Западноевропейские школы придерживаются объема в 1—2 страницы печатного текста.

Структура кейс-стади, которой мы придерживаемся при разработке материалов для нашего курса, выглядит следующим образом.

1. Название.
2. Краткая аннотация.
3. Ключевые слова.
4. Основная часть.
5. Вопросы и задания.
6. Анализ ситуации/решение.
7. Методические указания.
8. Список использованных источников.

Для самостоятельной проработки студент получает сокращенную версию кейс-стади: разделы 1—5 и 8. Разделы 1—2 и 4—6 являются стандартными. Наличие раздела 3, на наш взгляд, серьезно облегчает процесс подбора кейса для закрепления определенной темы. Раздел 7 является руководством по использованию кейс-стади для преподавателя и не всегда публикуется в западных сборниках. В случае сложных кейсов его наличие оправдано, но зачастую для проведения практического занятия достаточно знакомства с разделом 6.

Процесс разработки кейс-стади

Основные источники идеи для сюжета кейс-стади перечислены ниже [13]:

1) информация, полученная в ходе исследовательского или консалтингового проекта или целенаправленного сбора информации;

2) воображение автора;

3) средства массовой информации, специализированные журналы и буклеты, распространяемые на выставках, презентациях и т. д.

Особенностью консалтинговых проектов, связанных с защитой информации, является обязательство исполнителя не разглашать полученные об организации-заказчике сведения (что формально закрепляется в документе под названием "Соглашение о неразглашении" — "Nondisclosure agreement", подписываемом обеими сторонами). Особенно актуальным это является для российского (традиционно закрытого) бизнеса. Таким образом, вариант (1)

для разработки кейс-стади по информационной безопасности практически неприменим.

Недостатком подхода (2) является отстраненность от реального бизнеса, что противоречит самой сути метода конкретных ситуаций.

В случае применения подхода (3) для выбора темы и сбора информации могут использоваться:

- новостные порталы по информационной безопасности (http://www.itsec.ru/, http://infosecurity.report.ru/, http://pd.rsoc.ru/ и др.);
- сайты компаний, занятых в области защиты информации (http://www.kaspersky.ru/, http://www.infowatch.ru/, http://www.securitylab.ru/news/ и др.);
- профессиональные сообщества (RISSPA — Лента инцидентов информационной безопасности http://www.linkedin.com/groups?mostPopular = &gid = 3796607, Информационная безопасность http://professionali.ru/GroupInfo/636).

Преимуществом такого подхода является то, что информация уже частично систематизирована, а недостатками — сложность получения деталей, необходимых для полноценного анализа ситуации, и субъективная оценка ситуации авторами используемых публикаций. Тем не менее, сделать историю интересной и избежать предвзятого отношения можно за счет изменения имен ключевых фигурантов и "обогащения" ситуации подробностями на основе аналогичных ситуаций.

Наиболее продуктивным, по нашему мнению, является описанный выше "комбинированный" метод, основанный на творческой переработке материала из открытой прессы с добавлением авторских деталей, позволяющих адаптировать ситуацию с учетом уровня подготовки слушателей и сделать кейс-стади целостной историей, интересной для проведения анализа.

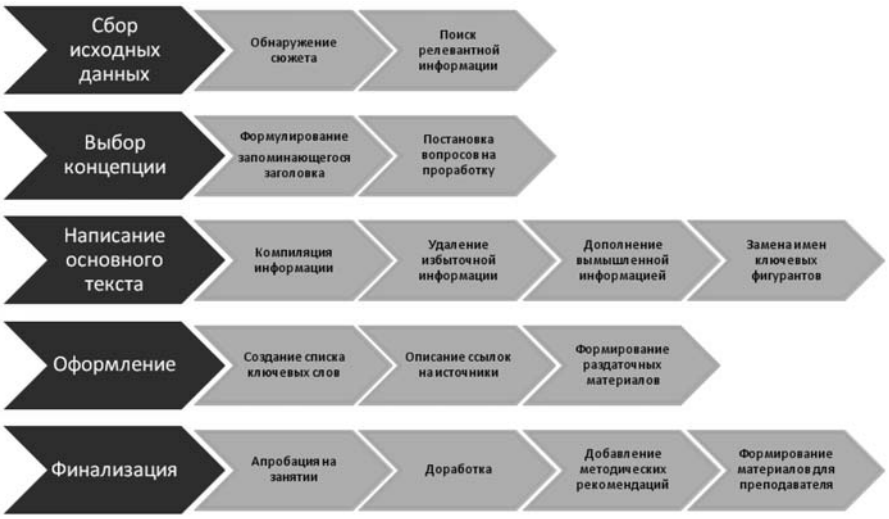


Рис. 1. Процесс создания кейс-стади

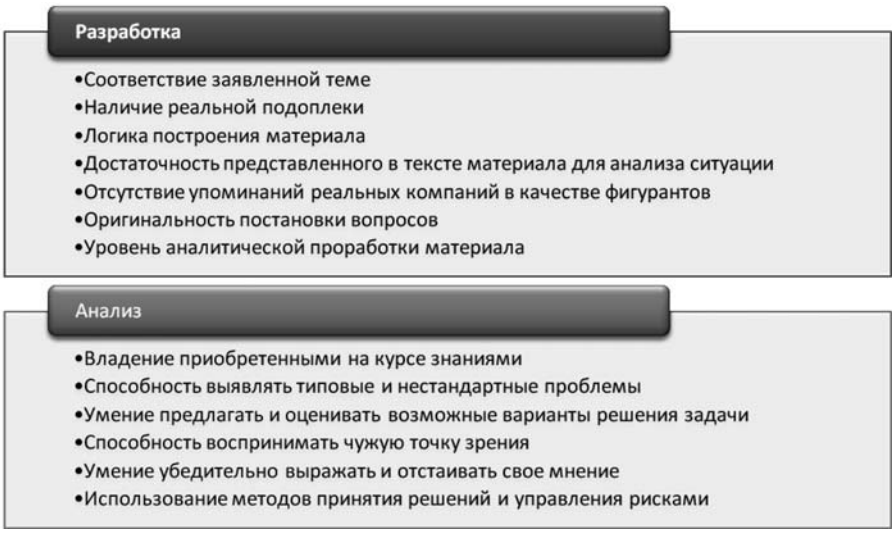


Рис. 2. Критерии оценки работы над кейс-стади

Методика использования кейс-стади в учебном процессе

Процесс создания кейс-стади представлен на рис. 1. Как правило, работа с кейс-стади строится следующим образом: преподаватель выдает текст студентам для самостоятельной проработки и затем инициирует дискуссию по вопросам, сформулированным в тексте.

На наш взгляд, эффективной является другая форма обучения, когда студенту предлагается самостоятельно подготовить кейс-стади по тематике курса. Такая форма выполнения курсовой работы была предложена студентам бакалавриата 4-го курса. При оценке работы учитывались следующие критерии:

- соответствие заявленной теме;
- логика построения материала;
- достаточность представленного в тексте материала для анализа ситуации;
- отсутствие упоминаний реальных компаний в качестве фигурантов;
- оригинальность постановки вопросов;
- уровень аналитической проработки материала;
- владение приобретенными на курсе знаниями и навыками при анализе ситуации.

Разработанные студентами кейс-стади могут быть использованы в учебном процессе, что обеспечивает постоянное пополнение банка кейс-стади. При накоплении достаточно большого числа кейс-стади мы планируем использовать этот метод при проведении экзамена как эффективный метод оценки освоения студентами пройденного материала. На рис. 2 представлены критерии оценки работы студентов с кейс-стади.

Выводы

Преимуществами подхода к преподаванию информационной безопасности, основанного на использовании метода кейс-стади, являются:

- ориентация на практические аспекты обеспечения информационной безопасности в условиях реального мира;
- высокий уровень вовлеченности студентов;
- фокусирование внимания студентов не только на технических, но и на организационных аспектах обеспечения информационной безопасности;
- демонстрация необходимости применения методов управления рисками для обеспечения защиты информации;
- возможность проведения практических занятий при минимальном уровне оснащенности аудитории;
- комплексный подход к проблеме обеспечения информационной безопасности с разных позиций — пользователя, технического специалиста, финансового директора, архитектора и топ-менеджера.

Заключение

Предлагаемый подход внедрен в учебный процесс кафедры "Управление разработкой программного обеспечения" отделения программной инженерии Государственного университета — Высшей школы экономики в рамках учебных курсов "Организация и технология защиты информации" (Магистратура; программа: Управление разработкой программного обеспечения"; 2-й курс, модуль 1, 2) и "Технологии обеспечения информационной безопасности" (Бакалавриат; специализация "Программная инженерия"; 4-й курс, модуль 3).

Апробация подхода была проведена на конференции-марафоне Training Labs'2010 в формате интерактивного тренинга "Кейс-стади: управление рисками в мире цифровых зависимостей", разработанного на основе материалов курса "Организация и технологии защиты информации".

Курс "Технологии и продукты Microsoft в обеспечении информационной безопасности", в основу которого легло использование предлагаемого подхода, на конкурсной основе получил поддержку в виде гранта "Разработка курсов по информационным технологиям", организованного компанией Microsoft (курс опубликован в библиотеке учебных курсов Центра образовательных ресурсов Microsoft [10] и в Интернет-Университете Информационных Технологий [11], где имеет высокий рейтинг популярности — 4,83 из 5 по состоянию на 19.12.2010).

Разработанная методика получила высокую оценку на Международной конференции по проблемам компьютерной безопасности "IT Security for the Next Generation — 2011" (Тур России и СНГ — 3-е место [16], Международный финал в Мюнхене — специальный приз [8]) и на семинаре "Глобальные проблемы информационной безопасности — 2011", Будапешт ("2011 Workshop on Cyber Security and Global Affairs", Budapest) [9].

На заседании Бюро Совета программы "Фонд образовательных инноваций НИУ ВШЭ" 29 июня 2011 г. по итогам весенних конкурсов образовательных инноваций работа "Методика подготовки и проведения семинарских занятий по информационной безопасности на основе изучения конкретных ситуаций" была признана победившей в номинации, посвященной разработке и внедрению в учебный процесс оригинальных методик проведения семинарских занятий, а также оригинальных методик проведения НИС в бакалавриате и магистратуре.

Список литературы

1. **Blakley B., McDermott E., Geer D.** Information security is information risk management // NSPW '01 Proceedings of the 2001 workshop on New security paradigms, ACM, 2001

2. **Brown J. S., Collins A., Duguid P.** Situated cognition and the culture of learning. Educational Researcher. 1989. Vol. 17. P. 32—42.

3. **Chipman S., Segal J., Glaser R.** Thinking and learning skills: Current research and open questions (Vol. 2). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc. 1985.

4. **Herreid C. F.** Start With a Story: The Case Method of Teaching College Science, National Science Teachers Association, 2006. 350 p.

5. **ISO/IEC** 27005:2008 Information technology — Security techniques — Information security risk management.

6. **Nickerson R. S.** On improving thinking through instruction // Review of Research in Education. 1988. Vol. 15. P. 3—57.

7. **Resnick L. B., Klopfer L. E.** Education and learning to think. Washington, DC: National Academy Press. Toward the thinking curriculum: Current cognitive research. Alexandria, VA: Association for Supervision and Curriculum Development, 1987.

8. **Savelieva A.** Special Considerations in Using the Case-study Method in Teaching Information Security // In Proceedings of 'IT Security for the Next Generation', TUM, Germany, Garching, Boltzmannstr. 3, 14—15 April 2011. URL: http://www.kaspersky.com/images/alexandra_savelieva-10-95017.pdf (дата обращения: 07.07.2011).

9. **Savelieva A. A., Avdoshin S. M.** Information Security Education and Awareness: Start with a Story // In proceedings of "2011 Workshop on Cyber Security and Global Affairs". URL: http://www.internationalcybercenter.org/workshops/cs-ga-2011/asavelieva (дата обращения: 07.07.2011).

10. **Авдошин С. М., Савельева А. А., Сердюк В. А.** Технологии и продукты Microsoft в обеспечении информационной безопасности // Библиотека учебных курсов Центра образовательных ресурсов Microsoft. 2010. URL: https://www.facultyresourcecenter.com/curriculum/pfv.aspx?ID = 8476&Login =

11. **Авдошин С. М., Савельева А. А., Сердюк В. А.** Технологии и продукты Microsoft в обеспечении информационной безопас-

ности // Интернет-Университет Информационных Технологий. 2010. URL: http://www.intuit.ru/department/security/mssec/

12. **Багиев Г. Л., Наумов В. Н.** Руководство к практическим занятиям по маркетингу с использованием кейс-метода // Энциклопедия маркетинга [Электронный ресурс]. URL: http://www.marketing.spb.ru/read/m21/

13. **Зобов А. М.** Метод изучения ситуаций (case-study) в образовании: его история и применение. Центр дистанционного образования Elitarium, 2006. URL: http://www.elitarium.ru (дата обращения: 07.07.2011).

14. **Камински Х.** Дидактико-методические основы преподавания экономики. Изучение конкретного случая (case-study) // Экономика. Вопросы школьного экономического образования. 1998. № 2.

15. **Парамонова Т. Н., Блинов А. О., Шереметьева Е. Н., Погодина Г. В.** Маркетинг: активные методы обучения. М.: КНОРУС, 2009.

16. **Савельева А. А.** Особенности использования метода case-study при преподавании информационной безопасности // Сб. тр. Международной студенческой конференции по проблемам компьютерной безопасности "IT-Security Conference for the Next Generation", 9—11 марта 2011 г., г. Москва, факультет Вычислительной математики и кибернетики Московского государственного университета [Электронный ресурс]. URL: http://www.kasperskyacademy.com/ru/view.html?id = 392 (дата обращения: 07.07.2011).

17. **Социология**: Энциклопедия / Сост. А. А. Грицанов, В. Л. Абушенко, Г. М. Евелькин, Г. Н. Соколова, О. В. Терещенко. Минск: Интерпрессервис; Книжный Дом, 2003. 1312 с. (Сер. Мир энциклопедий).

18. **Шумилова Ю. А.** Использование метода кейс-стади при преподавании маркетинговых дисциплин // Проблемы и перспективы управления экономикой и маркетингом в организации, Тюмень: Тюменский государственный университет, 2009. № 9.

IV международная конференция

"Проблемы кибернетики и информатики" (PCI'2012)

12—14 сентября 2012 г., Баку, Азербайджан

Научная тематика конференции

- информационные и коммуникационные технологии
- интеллектуальные технологии и системы
- сейсмические приборы, системы и технологии
- моделирование и идентификация
- численные методы и вычислительные технологии
- прикладной стохастический анализ
- управление и оптимизация
- принятие решений в социально-экономических системах

Доклады

Доклады представляются на английском языке объемом не более 4 страниц. Требования для оформления докладов размещены на *www.pci2012.science.az*

Финансовые вопросы

Организационные взносы участников конференции, а также расходы на издание Трудов конференции и культурные мероприятия будут оплачены организаторами конференции. Проживание в гостинице оплачивается иностранными участниками конференции самостоятельно.

Прибытие и размещение участников

Оргкомитетом для иностранных участников будут забронированы места в гостинице и организована встреча в аэропорту.

Адрес организационного комитета

Институт информационных технологий НАНА, комн. 216
Азербайджан, Az1141, Баку, ул. Б. Вагабзаде, 9
Тел.: (+994 12) 510 31 48
Факс: (+994 12) 539 61 21
pci2012az@gmail.com
www.pci2012.science.az

ЖУРНАЛ В ЖУРНАЛЕ

НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ

№ 5
МАЙ
2012

УДК 004.272.44, 004.324

С. М. Салибекян, ст. преподаватель,
П. Б. Панфилов, канд. техн. наук, проф.,
Московский институт электроники
и математики,
e-mail: salibek@yandex.ru

Реализация искусственных нейронных сетей с помощью атрибутной архитектуры вычислительной системы

Как известно, основной проблемой для воплощения нейронных сетей в "железе" является наличие огромного количества соединений между нейронами, которые невозможно реализовать с помощью современных технологий изготовления печатных плат и интегральных схем. В статье предлагается атрибутная архитектура организации вычислительной системы, которая позволяет решить данную проблему: в атрибутной системе обмен информацией между нейронами ведется всего через одну шину данных/атрибута, которая обеспечивает весь информационный обмен между любыми нейронами, входящими в сеть. Атрибутная нейронная сеть имеет и еще одно ценное качество — возможность динамической реконфигурации, причем для реконфигурации не требуется применение громоздких коммуникационных сред.

Ключевые слова: *нейронная сеть, динамическая реконфигурация, атрибутная архитектура, информационная пара*

Основной проблемой, сдерживающей широкое применение аппаратных нейронных сетей [1, 2] в современной вычислительной технике, является наличие большого количества межсоединений, объединяющих технические нейроны. Аппаратно реализовать такое громадное количество соединений по нынешний технологии производства интегральных схем и печатных плат невозможно [3]. Именно поэтому сейчас применение находят лишь программные эмуляции нейронных сетей, и именно поэтому становится невозможным реализовать теоретически высокий параллелизм вычислений, присущий данной архитектуре. Еще большие трудности аппаратной реализации вызывают реконфигурируемые сети: порой коммутационная сеть, объединяющая огромное множество нейронов, громоздка настолько, что на ее реализацию потребуется намного больше оборудования, чем для изготовления самих нейронов. А ведь именно реконфигурация нейронных сетей дала бы подобным вычислительным системам буквально неограниченные возможности.

Единственной довольно удачной попыткой аппаратной реализации нейронных сетей являются нейропроцессоры [4]. Однако это не очень удачный вариант решения проблемы, ведь нейпропроцессор — это обычный векторный процессор, который может выполнять специализированные векторные команды, эмулирующие нейронные сети.

Мы предлагаем техническое решение, которое даст возможность аппаратной реализации нейронных сетей по современным технологиям производства интегральных схем и печатных плат: изготовлять нейронную сеть на базе атрибутной архитектуры вычислительной системы, разрабатываемой в Московском институте электроники и математики [5—7].

Атрибутная система (рис. 1) состоит из функциональных устройств (ФУ) и цифровой шины данных — атрибута (ШДА), по которой ФУ осуществляют обмен информационными парами (ИП). ИП представляет собой совокупность данных (нагрузки) и атрибута (ярлыка), описывающего эти данные (атрибут — это уникальный идентификатор данных). ФУ-нейрон в данной вычислительной системе представляет собой устройство, в состав которого входят следующие элементы: регистры для хранения промежуточных данных (свой внутренний регистр соответствует каждому входному сигналу нейрона); набор регистров для весовых коэффициентов; регистр, хранящий ярлык (идентификатор) результата вычислений нейрона; мажорирующий элемент; элемент, реализующий выходную функцию нейрона. Каждой ячейке вышеперечисленных таблиц ставится в соответствие идентификатор и бит присутствия. Перед началом вычислительного процесса все биты присутствия сброшены (так как в таблицах еще нет исходных данных), и записаны значения идентификаторов полей таблиц.

ФУ-нейрон (рис. 2) работает следующим образом.

После начала вычислительного процесса по ШДА начинают передаваться ИП с данными. ШДА отслеживает передаваемые по шине ИП, и если ярлык (уникальный идентификатор) передаваемых данных совпадает с идентификатором, приписанным к определенной ячейке таблицы операндов, то данные считываются с ШДА и помещаются в соответствующую ячейку таблицы операндов и у данной ячейки выставляется бит присутствия данных.

В случае если все ячейки таблиц операндов и весовых коэффи-

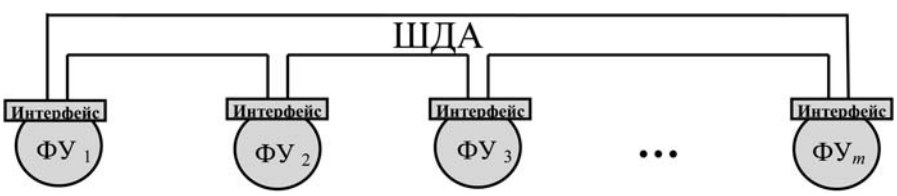


Рис. 1. Атрибутная система

циентов, регистр типа выходной функции и регистр уникального идентификатора выходного значения заполняются данными (т. е. установлены все биты присутствия), то операнды и весовые коэффициенты передаются в мажорирующий элемент для вычисления взвешенной суммы, далее полученное значение передается в блок выходной функции нейрона; к полученной выходной величине "прикрепляется" уникальный идентификатор выходного значения, и сформированная ИП через интерфейс ШДА выдается на шину, откуда данные могут быть считаны другими ФУ. Для того чтобы избежать конфликтов, когда несколько ФУ выдают ИП на ШДА одновременно, осуществляется определенный алгоритм арбитража шины.

Рис. 2. Структура ФУ-нейрона для нейронной сети без обратных связей

Приведем пример реализации нейронной сети на базе вычислительной системы (ВС) атрибутной архитектуры. Так, нейронная сеть, показанная на рис. 3, реализуется атрибутной системой, состоящей из источника входных данных нейронной сети (Ист.) и четырех ФУ, выполняющих функции технического нейрона (рис. 4).

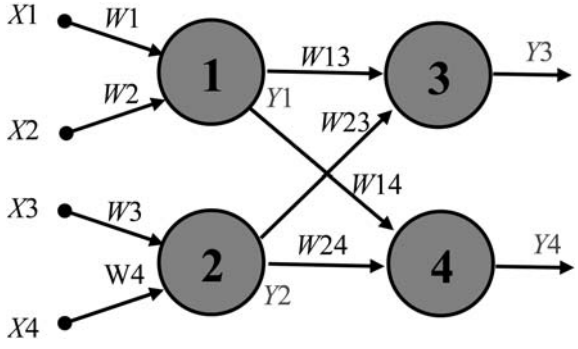


Рис. 3. Нейронная сеть без обратных связей

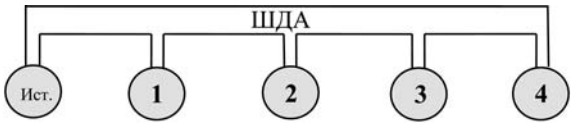


Рис. 4. Атрибутная нейронная система

Работа нейронной сети на базе атрибутной ВС делится на две фазы: инициализация и работа сети. Во время первой фазы осуществляется запись в таблицы операндов весовых коэффициентов и ярлыков (уникальных идентификаторов) передаваемых по ШДА данных (табл. 1, 2). Например, чтобы записать весовой коэффициент *W*1, необходимо выдать на ШДА следующую ИП: *W*1 = 0,4, где *W*1 — уникальный идентификатор; "=" — знак сопоставления уникального идентификатора и данных. После того как идентификатор будет выдан на ШДА, его считает ФУ1 и поместит значение в соответствующую ячейку вектора весовых коэффициентов.

Вторая фаза — выполнение. Работу атрибутной нейронной сети активизирует источник данных, который выдает последовательность ИП, содержащих данные для вычислений, на ШДА (для корректности работы системы идентификаторы мил-

Таблица 1

Таблица операндов источника заявок

Идентификатор	Значение
*X*1	...
*X*2	...
*X*3	...
*X*4	...

Таблица 2

Таблица операндов нейронов

Нейрон 1		Нейрон 2		Нейрон 3		Нейрон 4	
Входные операнды	Весовой коэффициент	Входные операнды	Весовой коэффициент	Входные операнды	Весовой коэффициент	Входные операнды	Весовой коэффициент
*X*1	*W*1	*X*3	*W*3	*Y*1	*W*13	*Y*1	*W*14
*X*2	*W*2	*X*4	*W*4	*Y*2	*W*23	*Y*2	*W*24
Идентификатор выходного значения *Y*1		Идентификатор выходного значения *Y*2		Идентификатор выходного значения *Y*3		Идентификатор выходного значения *Y*4	

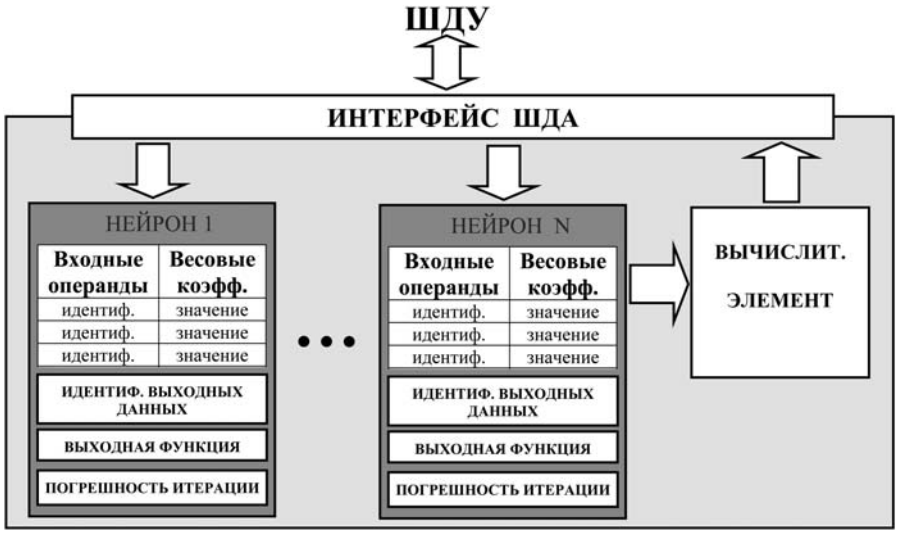


Рис. 5. Структура ФУ-нейрона для нейронной сети с обратными связями

ликоманд должны отличаться от идентификаторов данных, передаваемых по ШДА). Пусть источник данных будет последовательно выдавать на ШДА значения $X1$, $X2$, $X3$, $X4$ с соответствующими атрибутами. После выдачи на ШДА значения $X1$ ФУ1 опознает его по универсальному идентификатору и поместит в значение, считанное с ШДА, в таблицу операндов. По приходе на ШДА значения $X2$ ФУ1 помещает его в таблицу операндов нейронов и, так как все операнды, необходимые для осуществления вычисления, присутствуют в таблице операндов, ФУ передает операнды на мажорирующий элемент для вычисления выходного значения. (Атрибутная система работает по принципу управления данными (*dataflow*): вычисления на ФУ активируются не кодом команды, а приходящими по ШДА информационными парами.) Далее значения $X3$ и $X4$, выданные последовательно источником данных на ШДА, принимаются ФУ2, которое также начинает вычисление выходного значения. Как только ФУ1 закончит вычисления, оно снабдит полученный результат идентификатором $Y1$ и выдаст полученную ИП на ШДА — данную ИП считают с ШДА сразу ФУ3 и ФУ4. После того как ФУ2 выдаст полученное значение на ШДА, ФУ3 и ФУ4 будут располагать всеми необходимыми данными, осуществят вычисления и выдадут $Y3$ и $Y4$ — эти значения являются для нейронной сети выходными, и их можно будет вывести на консоль или использовать для дальнейших вычислений в других устройствах.

Атрибутная архитектура позволяет сократить объем оборудования вычислительной системы без ущерба быстродействию за счет совмещения последовательно расположенных на графе нейронов в одном ФУ. Для этого в состав ФУ вводится несколько наборов векторов идентификаторов и весовых коэффициентов (каждый набор соответствует одному нейрону): как только один из векторов операндов будет заполнен данными, операнды и весовые коэффициенты тут же поступают в мажорирующий элемент, и полученное значение, снабженное соответствующим атрибутом, будет выдано на ШДА. Так, для приведенной выше нейронной сети в одном ФУ можно объединить 1-й, 3-й нейроны и 2-й, 4-й нейроны, так как они соединены последовательно. Теперь уже атрибутная система будет состоять не из пяти, а из трех ФУ: Источник данных, Нейроны 1—3, Нейроны 2—4.

На базе атрибутной архитектуры можно реализовывать и нейронные сети с обратными связями (рис. 5). Для этой цели необходимо ввести в состав ФУ регистр, где будет храниться допустимая погрешность при проведении вычислительных итераций. ФУ-нейрон работает следующим образом: как говорилось ранее, ФУ генерирует выходное значение, когда по ШДА приходят все входные значения. В том же случае, когда ИП со входным значением приходит снова (это значит, что реализовалась обратная связь), нейрон вычисляет новое выходное значение, сравнивает его со значением старым; если новое значение отличается от старого более, чем на заданное значение погрешности, то на ШДА выдается ИП с новыми данными. В противном случае выдачи не происходит и вычислительный процесс заканчивается. Таким образом, обратные связи реализуются с помощью нескольких итераций вычислений: чем меньше допустимая погрешность, тем больше итераций совершается и тем более точным будет выходное значение.

Например, для реализации нейронной сети, показанной на рис. 6, необходимы два ФУ: одно ФУ совмещает функции нейронов 1 и 3, другое ФУ — нейронов 2 и 4. Таблицы переменных и весовых

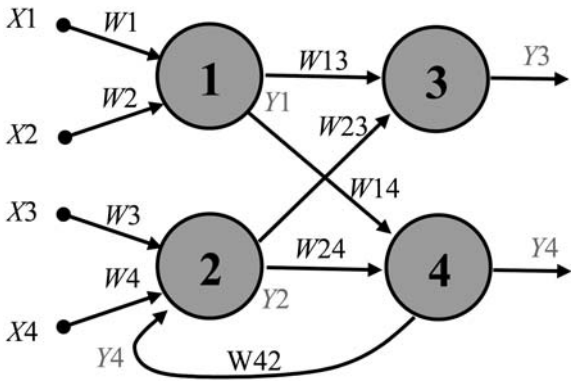


Рис. 6. Нейронная сеть с обратной связью

Таблица 3

Таблицы операндов нейронной системы с обратными связями

Нейрон 1,3		Нейрон 2,4	
Входные операнды	Весовой коэффициент	Входные операнды	Весовой коэффициент
*X*1 *X*2	*W*1 *W*2	*Y*1 *Y*2	*W*13 *W*23
Идентиф. выходных данных: *Y*1		Идентиф. выходных данных: *Y*3	
Выходная функция: сигмоида		Выходная функция: сигмоида	
Погрешность итераций: —		Погрешность итераций: —	
- ФУ нейронов 2 и 4:			
Нейрон 2		Нейрон 4	
Входные операнды	Весовой коэффициент	Входные операнды	Весовой коэффициент
*X*3 *X*4 *Y*4 (0)	*W*3 *W*4 *W*42	*Y*1 *Y*2 —	*W*14 *W*24 —
Идентиф. выходных данных: Y1		Идентиф. выходных данных: Y4	
Выходная функция: сигмоида		Выходная функция: сигмоида	
Погрешность итераций: 0,1		Погрешность итераций:	

коэффициентов каждого ФУ будут заполнены, как показано в табл. 3.

Обратите внимание: для нейрона 4 входное значение *Y*4 (оно отвечает за обратную связь) инициализируется заранее, иначе в самом начале процесса вычислений у нейрона не будет хватать одного операнда, который должен поступать уже позже по обратной связи. После того как нейроном 2 будут получены значения *X*3 и *X*4, вектор операндов заполнится и операнды будут переданы на мажорирующий элемент, далее ФУ выдаст полученное значение на ШДА, откуда его считают нейроны 3 и 4; результат же вычислений нейрона 4 по обратной связи снова поступит в нейрон 1, который вычислит новое значение. Если разница между новым и старым значениями будет больше погрешности итерации, то нейрон выдаст на свой выход новое значение, а через некоторое время по обратной цепи получит новое значение *Y*4. Так будет продолжаться до тех пор, пока разница между старым и новым значением после нескольких итераций не станет меньше погрешности, тогда нейрон 1 не станет выдавать новое значение и работа нейронной сети будет завершена.

Конечно, скорость вычислений нейронных сетей на базе атрибутной архитектуры будет меньше, чем у сетей аналоговых, однако атрибутные сети можно реализовать аппаратно с использованием современных технологий производства элементов электронной техники, и, естественно, аппаратная реализация будет работать намного быстрее, чем программная эмуляция. Атрибутная архитектура позволяет менять весовые коэффициенты нейрона уже во время выполнения вычислительного процесса: каждой ячейке вектора весовых коэффициентов приписывается свой уникальный идентификатор, с помощью которого имеется возможность в любой момент записать новое значение в данную ячейку.

Также довольно легко реконфигурируются и связи нейронных сетей, ведь виртуальная связь задается с помощью записей в таблице операндов ФУ. Для изменения виртуальной связи достаточно лишь ввести в состав ФУ вектор идентификаторов операндов, каждой ячейке которой ставится в соответствие свой уникальный идентификатор; теперь у нас появляется возможность в любой момент времени изменить идентификатор любого выходного операнда нейрона и тем самым изменить настройку виртуальной нейронной сети. Аналогично можно в любое время задать любую погрешность итераций и любую выходную функцию нейрона (каждая выходная функция имеет свой идентификатор, который можно передавать для ФУ с помощью ИП по ШДА).

Следует также отметить, что ШДА, по сути, является коммутатором между нейронами. Вернее, коммутацию осуществляют сами ФУ (они считывают с шины необходимые им данные), а ШДА является средой для передачи информации: шина работает в последовательном режиме, а ФУ работают параллельно (вычисления проходят на фоне обмена данными между ФУ). Таким образом, всего одна ШДА заменяет собой все те "неповоротливые" и аппаратоемкие коммутационные среды, что применяют в настоящее время. Именно поэтому в атрибутной архитектуре не представляет труда реализовать любой алгоритм обучения аппаратной нейронной сети и реконфигурировать связи между нейронами так, чтобы топология была наиболее подходящей для решения поставленной задачи.

Самый существенный недостаток атрибутных нейронных сетей кроется в том, что ШДА является узким местом системы: при достаточно большом числе ФУ ШДА ввиду своей малой пропускной способности может существенно снизить производительность системы. Однако этот недостаток может сгладить применение шлюзов. Шлюз — это ФУ, разделяющее ШДА на следующие отдельные сегменты: ИП, которые ФУ потребляют только в одном сегменте, шлюз в другие сегменты их не пропускает: перед началом вычислительного процесса (да, впрочем, и во время выполнения вычислительного процесса) выполняются соответствующие настройки шлюза (устанавливаются интервалы

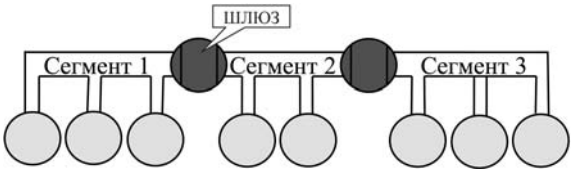


Рис. 7. Шлюзование ШДА атрибутной нейронной системы

идентификаторов данных, которые не следует "выпускать" из данного сегмента ШДА). В результате трафик, передаваемый по отдельным сегментам, будет существенно ниже общего трафика системы. Шлюз аналогичен мосту в современных локальных сетях (рис. 7).

Представляется, что реконфигурируемые атрибутные нейронные сети имеют потенциал аппаратной реализации посредством уже существующих технологий производства интегральных схем и печатных плат. Никакие другие известные технические решения подобной возможности не дают.

Список литературы

1. **Галушкин А. И.** Теория нейронных сетей. Кн. 1: учеб. пособие для вузов / Под общ. ред. А. И. Галушкина. М.: ИПРЖР, 2000.
2. **Круглов В. В., Борисов В. В.** Искусственные нейронные сети. Теория и практика. — 2-е изд., стереотип. М.: Горячая линия — Телеком, 2002.
3. **Грибачев В.** Элементная база аппаратных реализаций нейронных сетей // Компоненты и технологии. 2006. № 8. С. 100—103.
4. **Круг П. Г.** Нейронные сети и нейрокомпьютеры: учеб. пособие по курсу "Микропроцессоры". М.: Изд-во МЭИ, 2002.
5. **Салибекян С. М.** Принципы милликомандной архитектуры как основа построения высокопроизводительных адаптивных вычислительных систем // Автоматизация и современные технологии. 2002. № 5.
6. **Салибекян С. М., Панфилов П. Б.** Атрибутная архитектура как способ аппаратной реализации нейронных сетей // Тезисы IX Всероссийской научной конференции "Нейрокомпьютеры и их применение 2011". URL: http://www.it.mgppu.ru/confnc/tezisy.pdf.

УДК 004.3

Е. А. Энгель, канд. техн. наук, доц.,
Хакасский государственный университет
им. Н. Ф. Катанова, г. Абакан

Решение задач управления, принятия решений и обработки информации методом нечеткой селективной нейросети

Применение классических математических методов к решению задач управления и принятия решений затруднено, но эффективны интеллектуальные системы. Путем синтеза нейросетевых, селективных и нечетких методологий интеллектуальной поддержки при принятии управленческих решений и обработки информации обоснована разработанная автором нечеткая селективная нейросеть, устраняющая недостатки существующих методологий и более эффективная.

Ключевые слова: *нечеткая нейронная сеть, принятие решений, интеллектуальные системы*

Введение

Разработка методов анализа и управления сложными техническими объектами и системами составляет основное содержание работы специалистов в области обработки информации и предмет исследования многих направлений науки. Разработанные теории позволяют эффективно решать многие практические задачи обработки информации и управления. Однако существует значительный класс слабо формализованных задач управления и обработки информации, для которых невозможно или затруднено применение классических методов в связи с разрывом между предположениями, на которых базируются указанные методы, и свойствами информации об объектах задачи. В таком случае говорят о необходимости интеллектуальной поддержки принятия решений, т. е. о разработке набора методов, позволяющих получить достоверные решения задач управления и обработки информации. Создание и развитие новых методов решения слабо формализованных задач основано на автоматизации некоторых интеллектуальных функций анализа данных. На этом пути в настоящее время общезначимым является использование интеллектуальных информационных алгоритмов.

Реализация систем интеллектуальной поддержки при принятии управленческих решений и обработки информации на базе разработанных автором нейросетей — модифицированной и нечеткой селективной, в сравнении с другими методами повышает эффективность решения задач управления и обработки информации; снижает вычислительные затраты; позволяет автоматически корректировать функции принадлежности, т. е. создавать базы знаний, обновляемые автоматически по мере поступления новой информации. Синтез нейронных сетей и нечеткой логики сохраняет все преимущества индуктивного и дедуктивного подходов.

1. Интеллектуальная поддержка при принятии управленческих решений и обработка информации нечеткой нейросетью

Постановка задачи:

(X^k, Y^k) — выборка: обучающая для $k = \overline{1, N}$ и тестовая для $k = \overline{N+1, L}$;

X^k — k-й вектор входных переменных X_i^k, $i = \overline{1, n}$, $X^k \in R^n$;

Y^k — k-е значение выходной переменной Y_j^k, $j = \overline{1, m}$, $Y^k \in R^m$.

Требуется построить функцию f: $R^n \to Y$, $Y^k = f(X^k) \ \forall \ k = \overline{1, L}$.

В зависимости от типа выходной переменной решаются задачи:

- $Y = R^m$ — задача регрессии;
- $Y = \{y_1, ..., y_M\}$ — задача классификации, где M — число классов.

Автором разработан ***метод формирования нечеткой селективной нейросети***, включающий три этапа.

Этап 1 — составляются антецеденты нечетких продукционных правил. Осуществляется кластеризация объектов X. Если решается задача регрессии, то для определения числа кластеров M используется расширяющийся нейронный газ. В терминах нечеткой логики кластеризация определяет: нечеткие множества A'_j, где $j = \overline{1, M}$, и оценки значений функций принадлежности $\mu_j(X)$, в том числе многомерных ($n > 1$), как степень близости объекта и центроида кластера A'_j и/или, в случае задачи классификации, объектов, которые принадлежат заданному классу (функции оценки кластеризации и расчета центроидов кластера могут включать указанные характеристики). Многомерная функция принадлежности $\mu_j(X)$ является обобщением одномерных функций принадлежности $\mu_j(X_i)$, $i = \overline{1, n}$. В результате этапа 1 формируется $\mu_j(X)$ с помощью разбиения пространств входных переменных на базе кластеризации объектов и составляются антецеденты нечетких продукционных правил за два шага.

Шаг 1. Каждое из пространств входных переменных разбивается с помощью группировки одномерных функций принадлежности:

Π_j: X_1 есть A'_{1j} *И* X_2 есть A'_{2j} *И*... *И* X_n есть A'_{nj}, $j = \overline{1, M}$, A'_{ij} — нечеткие множества, определенные на X_i путем нечеткой кластеризации (Fuzzy C-Means) с функциями принадлежности $\mu_j(X_i) \in [0, 1]$, $i = \overline{1, n}$.

Шаг 2. Разбиение с помощью многомерных функций принадлежности:

Π_j: X есть A'_j, $X \in R^n$, $j = \overline{1, M}$.

Этап 2 — составляются консеквенты нечетких продукционных правил методом формирования функций принадлежности на основе модифицированной нейросети [1], который заключается в следующем: для каждого $j \in \{1, ..., M\}$ формируется модифицированная нейросеть Y'_j, реализующая классификацию $Y_j = \begin{cases} 1, \text{ если } X^i \in j \\ 0, \text{ если } X^i \notin j \end{cases}$, где $k = \overline{1, L}$. Таким образом, формируются M модифицированных нейросетей Y'_j, которые образуют M функций принадлежности $\eta_j(Y) = Y'_j$, в том числе многомерных (для $m > 1$).

Задается комплексный критерий качества классификации — H'. В данной статье рассмотрен случай, когда H' включает два показателя: допустимую погрешность классификации E' (по умолчанию $E = 0{,}4$) и функцию оценки $H(\tilde{Y})$ для нейросети $\tilde{Y}$ (по умолчанию $H(\tilde{Y}) = \max\limits_{1 \leqslant k \leqslant L} \|Y^k - (\tilde{Y})^k\|$), т. е. $H' = \{E', H(\tilde{Y})\}$. Комплексный критерий качества может быть дополнен другими показателями, например, отражающими предпочтения лица, принимающего решения, и составлен с учетом особенностей конкретной задачи.

В целях реализации функции принадлежности, в том числе многомерной, модифицированной нейросетью, алгоритм формирования последней [1] адаптирован следующим образом.

Шаг 1: $r = 0$, $I^0 = X^k$, где $k = \overline{1, N}$, r — номер ряда селекции, I^r — входные сигналы обучающей выборки на ряде селекции $r + 1$.

Шаг 2: $r = r + 1$; в функционал минимизируемой ошибки нейросети добавлена штрафная функция $\sum\limits_{j=1}^{n} \sum\limits_{l=1}^{p} \frac{(w_{l,j}^h)^2}{1 + (w_{l,j}^h)^2} + \sum\limits_{i=1}^{M} \sum\limits_{l=1}^{p} \frac{(w_{i,l}^o)^2}{1 + (w_{i,l}^o)^2}$, уменьшающая синаптические веса, где p — число нейронов скрытого слоя; $w_{l,j}^h$ — величина связи между j-м входным и l-м скрытым нейронами; $w_{i,l}^o$ — вес связи между l-м скрытым и i-м выходным нейронами; обучаем на выборке $((I^{r-1})^k, Y^k)$, где $k = \overline{1, N}$, Q двуслойных нейросетей $PY^r(I^{r-1})$, реализующих выход I^r; выбираем T нейронных сетей с наилучшей функцией оценки $H(\tilde{Y})$ на тестовой выборке $((I^{r-1})^k, Y^k)$ для $k = \overline{N+1, L}$; в результате имеем T настроенных слоистых нейросетей $\tilde{Y}_i = PY^r(PY^{r-1}(...(PY^1(X))))$, $i = \overline{1, T}$.

Шаг 3: в полученных на 2-м шаге T настроенных слоистых нейросетях $\tilde{Y}_i = PY^r(PY^{r-1}(...(PY^1(X))))$, $i = \overline{1, T}$, прореживаем незначимые нейроны, удовлетворяющие условию $\min\limits_l |w_{i,l}^o \, w_{l,j}^h| \leqslant E'$ без снижения точности классификации нейросетью.

Шаг 4: генетический алгоритм применяем к T настроенным матрицам синаптических весовых коэффициентов нейросетей $\widetilde{Y}_i$, $i = \overline{1, T}$ (поколение $s = 1$), полученных на 2-м шаге; для каждого поколения s генетического алгоритма применяем операторы кроссовера и мутации ($s = s + 1$), выбираем T нейронных сетей $\widetilde{Y}_i^s$, $i = \overline{1, T}$, с наилучшей оценкой $H(\widetilde{Y}^s)$ на выборке (X^k, Y^k) для $k = \overline{1, L}$ и вычисляем оценку поколения s по формуле: $g_s = \max\limits_{1 \leqslant i \leqslant T} H(\widetilde{Y}_i^s)$; генетический алгоритм работает до тех пор, пока $g_{s-1} > g_s$.

Шаг 5: если среди T слоистых нейросетей $\widetilde{Y}_i$, $i = \overline{1, T}$, хотя бы одна имеет функцию оценки меньше допустимой погрешности классификации, т. е. $\exists b \in \{1, ..., T\} | H(\widetilde{Y}_b) \leqslant E'$, то переход к шагу 7.

Шаг 6: вычисляем критерий несмещенности $U_r = \frac{1}{T} \sum\limits_{i=1}^{T} H(\widetilde{Y}_i^r)$. Если $U_r < U_{r-1}$, то переходим к шагу 1**.** Иначе возвращаемся на предыдущий ряд селекции $r = r - 1$ и увеличиваем поле селекции $K = 10 \cdot K$.

Шаг 7: возвращается лучшая модифицированная нейросеть $Y' = \widetilde{Y}_b$.

Схема адаптированного алгоритма формирования модифицированной нейросети изображена на рисунке.

Результатом 2-го этапа является набор из M функций принадлежности $\eta_j(Y) = Y_j'$, многомерных для $m > 1$, характеризующих лингвистические переменные y_j, где $j \in \{1, ..., M\}$.

Этап 3 — формируется нечеткая селективная нейросеть.

Шаг 1: составляются нечеткие продукционные правила

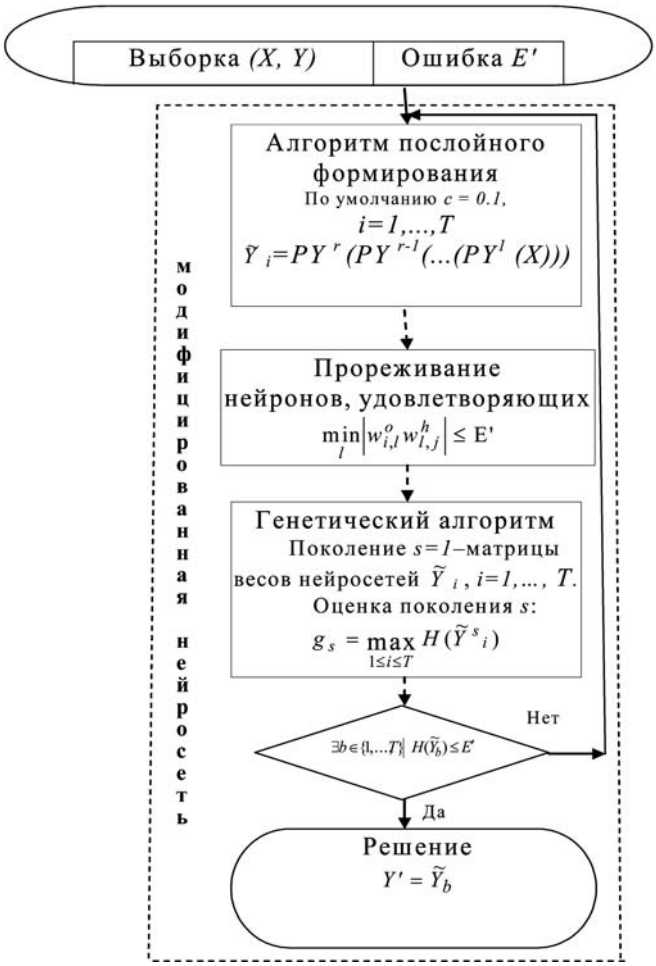
$$П_j\text{: ЕСЛИ } X \text{ есть } A_j' \text{ ТО } Y \text{ есть } y_j, \qquad (1)$$

где y_j — лингвистическая переменная, характеризуемая функцией принадлежности $\eta_j(Y)$; A_j' ($j = \overline{1, M}$) — лингвистическая переменная, характеризуемая многомерной функцией принадлежности $\mu_j(X)$.

Шаг 2: реализуется логическая схема нечеткой селективной нейросети, соответствующая задаче, путем введения в структуру:

- И-, ИЛИ-нейронов;
- нейрона, агрегирующего значения активации условия (антецедента) нечетких правил путем выбора $\max\limits_{1 \leqslant j \leqslant M} \mu_j(X)$ (1-й способ) или вычисления произведения многомерных функций принадлежности $\mu(X)$ нечетких множеств A' (2-й способ);
- нейрона, выполняющего дефаззификацию. Механизм нечеткого вывода выбирается в зависимости от особенностей задачи. Функции принадлежности $\eta_j(Y) = Y_j'$ представлены выходными функциями модифицированных нейросетей Y_j', которые непрерывны, а значит, интегрируемы. Поэтому применим наиболее распространенный способ нечеткого вывода, основанный на отыскании "центра тяжести" полученного нечеткого соответствия.

Реализация функции принадлежности как обученной модифицированной нейросети позволяет избегать априорных предположений о характере распределения данных. Поскольку модифицированная нейросеть реализует процедуру дообучения посредством сохранения основной части настроенной структуры нейросети, возможна автоматическая корректировка функции принадлежности, т. е. создание базы знаний, обновляемой автоматически по мере поступления новой информации. При данном спо-



Адаптированный метод формирования модифицированной нейросети, где *c* — параметр функции активации нейрона

собе сохраняются все преимущества индуктивного (нейроинформатика) и дедуктивного (нечеткая логика) подходов. Селективная нечеткая нейросеть имеет ряд отличий от существующих аналогов: структурно она представляет собой совокупность обученных модифицированных нейросетей и слоев специализированных нейронов, формирующих нечеткие продукционные правила и осуществляющих дефаззификацию.

2. Экспериментальные результаты

С использованием разработанного метода формирования нечеткой селективной нейросети были решены задачи Yale распознавания образов [2] и Всемирного Конгресса по Компьютерному Интеллекту (WCCI) 2010 [3].

Задача Yale распознавания образов. Задача Yale заключается в распознании 165 полутоновых изображений 15 человек, каждый человек имеет 11 изображений. Изображения одного человека различаются условиями освещения, выражением лица (нормальное, счастливое, грустное, сонное, удивленное и подмигивающее). Изображение представляется в виде 1024-мерного вектора. Данные разбиты на обучающую и тестовую выборки случайным образом в отношении *Gm*/*Pn* (*Gm* и *Pn* — число изображений одного человека, включенное в обучающую и тестовую выборки соответственно). Задача решена двумя способами.

1-й способ. Формирование модифицированной нейросети $\tilde{Y}_{Yale}$ для решения задачи осуществлялось следующим образом. Оценка константы Липшица для обучающей выборки на аналоговом представлении данных равна 0,16. На первом ряде селекции ($T = 15$ и $Q = 1000$ — параметры алгоритма формирования модифицированной нейросети) при формировании модифицированной нейросети использована двуслойная нейросеть с одним скрытым слоем (15 нейронов), оценка константы Липшица нейросети (параметр функции активации $c = 0{,}019$) равна 0,17. Оценка константы Липшица обучающей выборки второго для ряда селекции ($T = 1$, $Q = 1000$) равна 0,07. На втором ряде селекции структура модифицированной нейросети дополнена двуслойной нейросетью с одним скрытым слоем (3 нейрона), оценка константы Липшица нейросети ($c = 0{,}08$) равна 0,074. В результате сформирована четырехслойная (число нейронов на скрытых слоях соответственно 15, 15, 3) модифицированная нейросеть $\tilde{Y}_{Yale}$, эффективность распознавания образов которой отражает табл. 1.

2-й способ. Решение задачи Yale нечеткой селективной нейросетью осуществлялось следующим образом: путем нечеткой кластеризации X_i определены A'_{ij} — нечеткие множества с функциями принадлежности $\mu_j(X_i) \in [0, 1]$, $i = \overline{1, n}$, $j = \overline{1, 15}$. Методом формирования функций принадлежности, описанным в разделе 1, построены функции принадлежности $\eta_j(Y)$, $j = \overline{1, 15}$, соответствующие нечетким множествам y_j. Каждая из $\eta_j(Y)$, $j = \overline{1, 15}$, моделируется модифицированной четырехслойной нейросетью, формируемой аналогично модифицированной нейросети $\tilde{Y}_{Yale}$. В результате сформированы 15 четырехслойных модифицированных нейросетей (число нейронов на скрытых слоях соответственно 15, 15, 3), выход которых соответствует одномерным функциям принадлежности $\eta_j(X)$, $j = \overline{1, 15}$.

Нечеткая селективная нейросеть была составлена с использованием метода формирования, описанного в разделе 1, на основе следующего продукционного правила:

Если X есть A' TO Y есть y.

Таблица 1

Эффективность методов распознавания для задачи Yale

Метод	G2/P9	G3/P8	G4/P7	G5/P6
Eigenface	46,0 ± 3,4	50,0 ± 3,5	55,7 ± 3,5	57,7 ± 3,8
Fisherface	45,7 ± 4,2	62,3 ± 4,5	73,0 ± 5,4	76,9 ± 3,2
2DLDA	43,4 ± 6,2	56,3 ± 4,7	63,5 ± 5,6	66,1 ± 4,8
S-LDA	57,6 ± 4,1	72,3 ± 4,4	77,8 ± 3,0	81,7 ± 3,2
Laplacianface	54,5 ± 5,2	67,2 ± 4,1	72,7 ± 4,2	75,8 ± 4,6
MFA	45,7 ± 4,2	62,3 ± 4,5	73,0 ± 5,4	76,9 ± 3,2
S-MFA	57,2 ± 4,3	71,2 ± 4,0	76,9 ± 3,1	81,1 ± 3,1
TensorPCA	49,4 ± 3,5	54,0 ± 3,0	57,8 ± 3,3	59,8 ± 3,9
S-LPP	57,9 ± 4,5	72,0 ± 4,0	76,0 ± 3,4	81,4 ± 2,9
S-NPE	57,5 ± 4,7	71,9 ± 3,9	77,0 ± 3,4	80,9 ± 3,5
Pixel space	Нет данных	Нет данных	84,0 ± 1,5	Нет данных
Noushath et al. 2006	Нет данных	Нет данных	85,0 ± 1,5	Нет данных
Wang et al. 2007	Нет данных	Нет данных	99,0 ± 0,5	Нет данных
Модифицированная нейросеть	52,1 ± 3,2	62,8 ± 3,0	71,2 ± 3,1	71,3 ± 3,2
Нечеткая селективная нейросеть	56,3 ± 2,8	68,9 ± 2,9	76,3 ± 2,7	78,3 ± 2,8

Логическая схема нечеткой селективной нейросети, соответствующая задаче, содержит: два блока И-нейронов нечетких множеств A' и y, а также нейрон, агрегирующий значения активации условия (антецедента) нечетких правил путем нахождения произведения многомерных функций принадлежности $\mu(X)$ нечеткого множеств A' (2-й способ). Для вычисления Y использован способ нечеткого вывода, основанный на отыскании "центра тяжести" полученного нечеткого соответствия.

В табл. 1 приведена эффективность алгоритмов для задачи Yale [2]. Нечеткая селективная нейросеть решает данную задачу эффективнее модифицированной нейросети. Экспериментально, в сравнении с традиционными математическими и интеллектуальными методами, перечисленными в табл. 1, показана целесообразность использования и подтверждена высокая эффективность и точность разработанной нечеткой селективной нейросети для принятия решений и обработки информации.

Задача распознавания рукописных символов IBN_SINA (WCCI-2010). Исторические архивы трудно обрабатывать традиционными методами оптического распознавания (OCR) вследствие того, что синтаксическое и семантическое содержание древних рукописей больше не используется. Применение интеллектуальных методов обработки информации ускоряет распознавание символов древних рукописей, снижая необходимость в использовании экспертов.

Задача IBN_SINA состоит в распознании символов арабской древней рукописи IBN_SINA. Входные данные задачи IBN_SINA представлены таблицей свойств (92 переменные). Задача IBN_SINA решена модифицированной и нечеткой селективной нейросетями.

Для решения задачи IBN_SINA с использованием алгоритма послойного формирования нейросети [1] составлена и настроена четырехслойная модифицированная нейросеть (40 скрытых элементов: 30, 7 и 3 нейронов на скрытых слоях).

Для решения задачи IBN_SINA сформирована нечеткая селективная нейросеть следующим образом: на основе нечеткой кластеризации входных данных выделены нечеткие множества A'_j, соответствующие 15 кластерам, объекты одного кластера представляют изображения одного символа. Методом формирования функций принадлежности на основе модифицированной нейросети построены функции принадлежности $\eta_j(Y)$, $j = \overline{1, 15}$, соответствующие нечетким множествам y_j, каждая из которых представляет четырехслойную модифицированную нейросеть (40 нейронов на скрытых слоях соответственно: 30, 7 и 3), выход которых соответствует одномерным функциям принадлежности $\eta_j(X)$, $j = \overline{1, 15}$.

Логическая схема нечеткой селективной нейросети, соответствующая задаче IBN_SINA, включает:

- продукционные правила: Если X есть A' *ТО* Y есть y;
- два блока И-нейронов нечетких множеств A' и y;
- нейрон, агрегирующий значения активации условия (антецедента) нечетких правил путем нахождения произведения многомерных функций принадлежности $\mu(X)$ нечеткого множеств A';
- нейрон, осуществляющий нечеткий вывод, на основе отыскания "центра тяжести" полученного нечеткого соответствия.

Таблица 2

Рейтинг эффективности решения задач WCCI-2010

Рейтинг	Оценка IBN_SINA (оценка, представляющая собой площадь под кривой ROC (AUC))	Оценка ORANGE (AUC)	Оценка HIVA (AUC)
1	0,990445	0,810102	0,947128
2	0,988359	0,721316	0,884041
3	0,983816	0,813333	**0,856514**
4	0,978695	0,787346	0,812109
5	**0,97777**	**0,787243**	0,793947
6	0,977876	0,788271	**0,784561**
7	0,97781	0,78821	0,770527
8	0,977415	**0,787634**	0,769719
9	**0,977364**	0,787244	0,711866

Примечание: алгоритм вычисления оценки AUC, представляющей собой площадь под кривой ROC, приведен на сайте http://www.causality.inf.ethz.ch/activelearning.php?page=evaluation#cont

В соответствии с результатами WCCI-2010 на IBN_SINA (табл. 2) рейтинг решения задачи модифицированной и нечеткой селективной нейросетями равен соответственно 8 и 5.

Задача принятия решений— HIVA (WCCI-2010). Задача хемоинформатики HIVA — предсказать, какие соединения активны в отношении инфекции СПИД ВИЧ. Проблема сведена к бинарной классификации (активные, неактивные). Входные данные представляют 1617 бинарных переменных. Проблема относится к прогнозу количественных характеристик биологической активности (QSAR — количественное соотношение структура—активность) для скрининга новых соединений (HTS — проблема высокопроизводительного скрининга).

Задача решена модифицированной и нечеткой селективной нейросетями. Для решения задачи HIVA посредством алгоритма послойного формирования нейросети [1] реализована четырехслойная модифицированная нейросеть (57 скрытых элементов: 40, 10 и 7 нейронов на скрытых слоях соответственно).

Для решения задачи HIVA сформирована нечеткая селективная нейросеть следующим образом: на основе нечеткой кластеризации входных данных выделены нечеткие множества A'_j, соответствующие двум кластерам; 1-й кластер включал примеры активных соединений, 2-й кластер — неактивных.

Методом формирования функций принадлежности на основе модифицированной нейросети построены функции принадлежности $\eta_j(Y)$, $j = \overline{1, 2}$, соответствующие нечетким множествам y_j, каждая из которых представляет четырехслойную модифицированную нейросеть (57 скрытых элементов: 40, 10 и 7 нейронов на скрытых слоях соответственно), выход которых соответствует одномерным функциям принадлежности $\eta_j(X)$, $j = \overline{1, 2}$. Логическая схема нечеткой селективной нейросети, соответствующая задаче, основана на следующих продукционных правилах:

Если X есть A' *TO* Y есть y.

Схема содержит два блока И-нейронов нечетких множеств A' и y, а также нейрон, агрегирующий значения активации условия (антецедента) нечетких правил путем выбора $\max\limits_{1 \leqslant j \leqslant M} \mu_j(X)$. Для вычисления Y использован способ дефазификации относительно среднего максимума.

Рейтинг в соответствии с результатами WCCI-2010 решения задачи HIVA модифицированной и нечеткой селективной нейросетями равен соответственно 5 и 3 (табл. 2).

Задача принятия решений — ORANGE (WCCI-2010). Маркетинговая задача ORANGE состоит в том, чтобы предсказать склонность потребителей сменить провайдера (отток), купить новые продукты или услуги (влечение) или купить обновления или дополнения по более выгодным предложениям (сверхпродажи). Гетерогенная зашумленность данных (40 категориальных переменных), а также несбалансированное распределение класса характеризует данную задачу как слабоформализованную.

С использованием алгоритма послойного формирования нейросети [1] для решения задачи ORANGE обучена четырехслойная модифицированная нейросеть (105 скрытых элементов: 85, 15 и 5 нейронов на скрытых слоях соответственно).

С использованием разработанного метода формирования нечеткой селективной нейросети задача ORANGE решена следующим образом: на основе нечеткой кластеризации входных данных выделены нечеткие множества A'_j, соответствующие трем кластерам, каждый кластер включал примеры одинаковой склонности потребителей (отток, влечение, сверхпродажи). Методом формирования функций принадлежности на основе модифицированной нейросети построены функции принадлежности $\eta_j(Y)$, $j = \overline{1, 3}$, соответствующие нечетким множествам y_j, каждая из которых представляет четырехслойную модифицированную нейросеть (105 скрытых элементов: 85, 15 и 5 нейронов на скрытых слоях соответственно), выход которой соответствует одномерным функциям принадлежности $\eta_j(X)$, $j = \overline{1, 3}$.

Логическая схема нечеткой селективной нейросети, соответствующая задаче ORANGE, включает:

- продукционные правила: Если X есть A' *TO* Y есть y;
- два блока И-нейронов нечетких множеств A' и y;
- нейрон, агрегирующий значения активации условия (антецедента) нечетких правил путем выбора $\max\limits_{1 \leqslant j \leqslant M} \mu_j(X)$;
- нейрон, осуществляющий нечеткий вывод, на основе отыскания "центра тяжести" полученного нечеткого соответствия.

Использование нечеткой селективной нейросети позволило повысить рейтинг решения задачи с 7 до 5 по сравнению с результатом модифицированной нейросети. Разработанные автором алгоритмы и методы, представляющие оптимальные по сложности робастные модели, применены для решения задач интеллектуальной поддержки принятия решений, классификации в различных предметных областях: распознавание образов, рукописных символов, маркетинга и хемоинформатики.

Выводы

В целях реализации функции принадлежности (в том числе многомерной) модифицированной нейросетью адаптирован алгоритм формирования последней.

Для интеллектуальной поддержки при принятии решений и обработки информации разработана нечеткая селективная нейросеть, представляющая собой гибридный метод на основе модифицированной нейросети и нечеткой логики.

Результаты работы модифицированной и нечеткой селективной нейросетей при решении практических задач показали хорошую эффективность обработки информации при комплексном сравнении их с методами, оптимальность которых признана научным мировым сообществом.

Практическая значимость модифицированной и нечеткой селективной нейросетей подтверждается решением с заданной точностью задач принятия решений в различных предметных областях: маркетинг, распознавание лиц и рукописных символов, хемоинформатика.

Список литературы

1. **Энгель Е. А.** Модифицированная нейросеть для обработки информации с использованием селекции существенных связей. Автореферат диссертации на соискание ученой степени кандидата технических наук, Красноярск, 2004.

2. **Cai D., He X., Hu Y., Han J., and Huang T.** Learning a spatially smooth subspace for face recognition // Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07). 2007. P. 1—7.

3. **http:**//www.causality.inf.ethz.ch/activelearning.php?page = leaderboard#cont.

УДК 004.932.2

Нгуен Виет Хунг, аспирант,
А. В. Муравьев, аспирант,
Московский физико-технический институт,
e-mail: viethung81@yahoo.com

Применение нейросетевого алгоритма в задаче компенсации движения в видеокодировании с помощью технологии NVIDIA CUDA

Представлен алгоритм поблочного нахождения вектора движения в видеопоследовательностях и подход к его реализации на базе графической карты NVIDIA. Особенностями этого алгоритма являются новый подход к построению набора кандидатов вектора движения и высокая параллелизуемость. Приводятся результаты экспериментальных исследований, и представленный алгоритм сравнивается с другими алгоритмами компенсации движения на CPU и на CUDA.

***Ключевые слова**: компенсация движения, видеокодирование, нейронные сети, сеть Кохонена, графический процессор, CUDA*

Введение

Стандарт H.264 применяется для эффективного кодирования видео как для передачи через Интернет, так и для сохранения на носителях. Стандарт H.264 содержит ряд новых возможностей, позволяющих значительно повысить эффективность сжатия видео, по сравнению с предыдущими стандартами, а также обеспечивает высокую степень сжатия и высокое качество изображения [1]. Однако значительно больше вычислительной мощности требуется для кодирования видео, и процесс кодирования видеороликов в DVD-качестве в формате H.264 часто занимает несколько часов, даже на современном компьютере. Среди всех этапов кодирования этап оценки движения является наиболее интенсивной частью процесса кодирования H.264 с точки зрения вычислений. Принято основное предположение для всех алгоритмов компенсации движения стандарта H264: между смежными изображениями обычно меняется только малая часть сцены. В этом случае полную информацию о сцене нужно сохранять только выборочно — для опорных изображений. Для остальных изображений достаточно передавать только разностную информацию: о положении объекта, направлении и значении его смещения, о новых элементах фона (открывающихся за объектом по мере его движения).

В данной работе предлагается алгоритм поиска вектора движения с использованием нейронной сети и он сравнивается с другими алгоритмами. Далее рассматривается подход к реализации предлагаемого алгоритма и алгоритма полного перебора, так как они лучше всего подходят для распараллеливания по сравнению с другими алгоритмами.

Обзор алгоритмов компенсации движения

По общему правилу сначала текущий кадр разбивается на непересекающиеся блоки одного размера $B(x, y)$. Для каждого блока $B(x, y)$ в небольшой окрестности $-p < u, v < p$ ищется наиболее "похожий" на него блок $B_{Prev}(x + u, y + v)$ в предыдущем кадре (рис. 1). Вектор движения для блока размером $N \times M$ с координатами (x, y) выбирается из условия минимума функции стоимости, которая в общем случае является суммой абсолютных разностей между элементами текущего и опорного блоков:

$$SAD(V_x, V_y) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} |F(x + m, y + n, t_1) - F(x + V_x + m, y + V_y + n, t_2)|, \quad (1)$$

где SAD — сумма абсолютных разниц (sum of absolute differences); F — значение яркости; t_1 — временной индекс текущего кадра; t_2 — временной индекс опорного кадра; (V_x, V_y) — вектор движения.

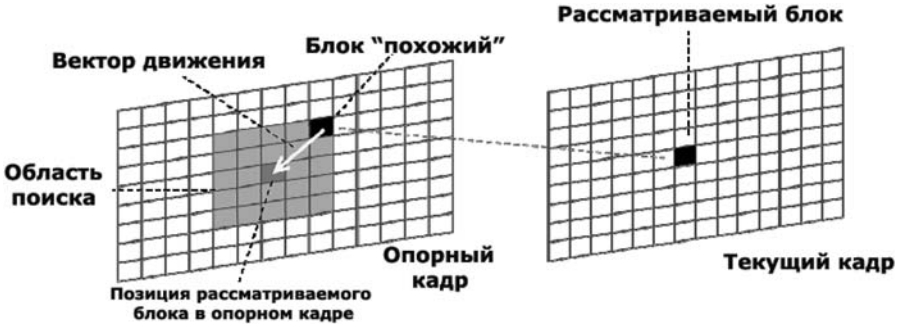


Рис. 1. Общая схема алгоритма нахождения вектора движения

Алгоритмы компенсации движения различаются методами нахождения минимума функции ошибки компенсации во всей области $-p < u, v < p$.

В алгоритме *полного перебора* проводится последовательный поиск по всему региону поиска [2], [3]. Алгоритм обеспечивает наилучшее качество приближения, однако требует выполнения большого числа операций. Этот подход считается эталонным [2] для оценки нового алгоритма компенсации движения.

Исследованы и другие алгоритмы поиска векторов движения, такие как трехшаговый поиск [4], новый трехшаговый поиск [5], четырехшаговый поиск [6], спиральный поиск [7] и т. д. Такие алгоритмы используют предположение о том, что функция *SAD* (1) строго монотонно сходится к своему минимуму в области поиска, и проверкой всего нескольких точек в этой области можно локализовать этот минимум. Поскольку функция ошибки компенсации почти никогда не бывает монотонной, эти алгоритмы, хотя демонстрируют неплохую скорость работы, часто находят локальный минимум вместо глобального.

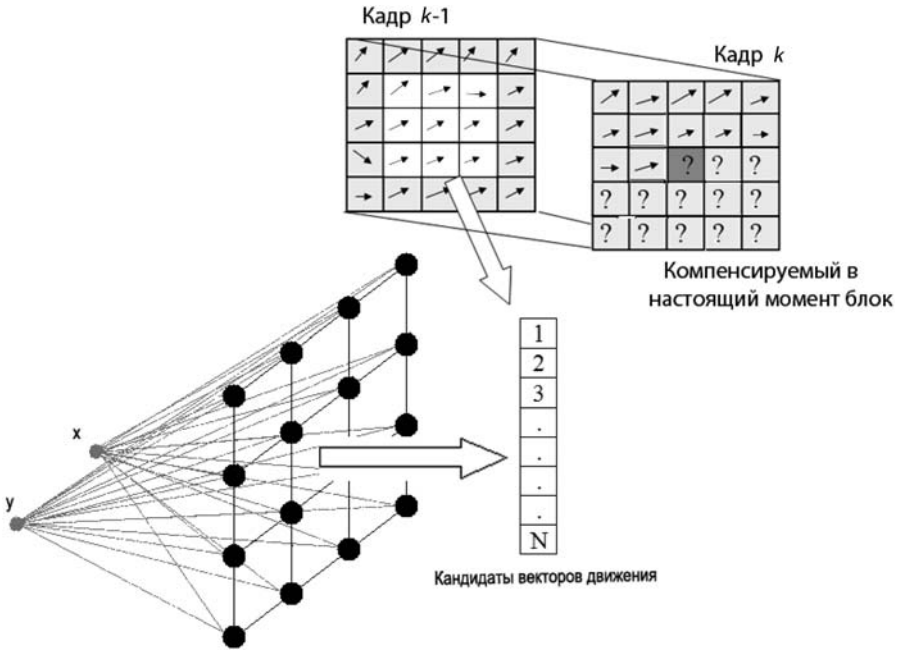


Рис. 2. Схема построения кандидатов векторов движения

Алгоритм компенсации движения с использованием нейронной сети

В данной работе предлагается новый алгоритм для задачи компенсации движения с использованием нейронных сетей Кохонена [8]. Заметим, что в видеопоследовательностях векторы движения чаще находятся в некоторых определенных направлениях. Например, в фильмах обычно горизонтальное движение превалирует над вертикальным. Поэтому с помощью сети Кохонена мы построим набор кандидатов векторов движения, которые часто встречаются в задаче компенсации движения. Входами сети Кохонена будет множество векторов движения, которые строят с помощью алгоритма полного перебора. Для построения такого множества используются видеопоследовательности по разным тематикам. Размер слоя Кохонена зависит от размера области поиска. Например, для макроблока 8 × 8 с расширением области поиска до + 7 пикселей, т. е. с размером 15 × 15 пикселей, выбирается слой Кохонена размером 8 × 8, т. е. 64 выхода соответствуют 64 кандидатам вектора движения.

Множество кандидатов векторов движения строит сеть Кохонена на основе общего характера движения, несмотря на характер самой видеопоследовательности. Чтобы улучшить эффективность алгоритма, добавим к множеству кандидатов для каждого блока отдельно векторы движения окрестности + 1 пиксель соседних блоков предыдущего кадра. Основная идея состоит в том, что поле векторов является достаточно гладким как в пространстве, так и во времени, за исключением разрывов, возникающих на границах объектов, движущихся неодинаково, и ситуации смены общего плана (рис. 2). Для сравнения эффективности алгоритмов на MATLAB были написаны трехшаговый, новый трехшаговый, четырехшаговый, спиральный поиск, полный перебор и предложенный алгоритм. Выбираем размер блока 8 × 8 пикселей и область поиска до + 7 пикселей (т. е. размер области поиска 15 × 15). В качестве меры искажения предсказуемого кадра от оригинального используется отношение сигнал/помеха (дБ):

$$PSNR = 10\lg\left(\frac{D^2 N_x N_y}{E(A, \tilde{A})}\right), \qquad (2)$$

где $N_x N_y$ — размер кадра; D — диапазон амплитуд отсчетов оригинального кадра $A(x, y)$;

$E(A, \tilde{A})$ — евклидово расстояние между оригинальным $A(x, y)$ и предсказуемым кадром $\tilde{A}(x, y)$:

$$E(A, \tilde{A}) = \sum_{x=0}^{N_x - 1} \sum_{y=0}^{N_y - 1} (A(x, y) - \tilde{A}(x, y))^2. \qquad (3)$$

В табл. 1 показаны результаты *PSNR* (2) для первых 200 кадров тестированных видеопоследовательностей: football, foreman и news. В табл. 2 по-

Таблица 1

Сравнение алгоритмов: *PSNR* средних 200 кадров разных видеопоследовательностей (в дБ)

Алгоритмы	Football	Foreman	News
Трехшаговый	23,7908	30,6627	35,4762
Четырехшаговый	23,349	30,7869	35,4656
Спиральный поиск	23,2453	30,954	35,5985
Новый трехшаговый	23,7991	31,2696	35,713
Предложенный	24,1891	31,6482	35,7502
Полный перебор	24,6354	31,8259	36,0552

Таблица 2

Время нахождения векторов движения 200 кадров разных видеопоследовательностей (в секундах)

Алгоритмы	Football	Foreman	News
Трехшаговый	77,0664	77,027	76,4872
Четырехшаговый	74,8218	66,2744	54,7127
Спиральный поиск	156,5198	117,879	83,3848
Новый трехшаговый	90,3149	72,8979	56,2794
Предложенный	379,2892	337,1631	309,9222
Полный перебор	629,1056	627,9893	618,5958

казаны соответствующее время нахождения векторов движения всех алгоритмов.

Результаты сравнения показывают, что значения *PSNR* предложенного алгоритма близки к значениям *PSNR* алгоритма полного перебора и выше значения *PSNR* других алгоритмов. Так как число вычислений функции *SAD* (1) в предложенном алгоритме меньше, чем в алгоритме полного перебора, то соответствующее время работы предложенного алгоритма тоже в 2 раза меньше.

Для задачи компенсации движения большинство из алгоритмов оценки движения требуют большого количества ветвлений, которые могут стать узким местом в реализации на CUDA. Поэтому в этой работе для реализации выбраны алгоритм полного перебора и алгоритм с использованием нейронной сети.

Реализация алгоритмов компенсации движения на CUDA

CUDA (Compute Unified Device Architecture) — это архитектура параллельных вычислений от NVIDIA, позволяющая существенно увеличить вычислительную производительность благодаря использованию GPU (графических процессоров). Типичные графические процессоры состоят из сотни высококвалифицированных процессорных ядер и способны достичь огромной параллельности вычислений. Далее рассматривается обзор модели CUDA программирования и аппаратной архитектуры для определения потенциальной производительности GPU в реализации предложенных алгоритмов.

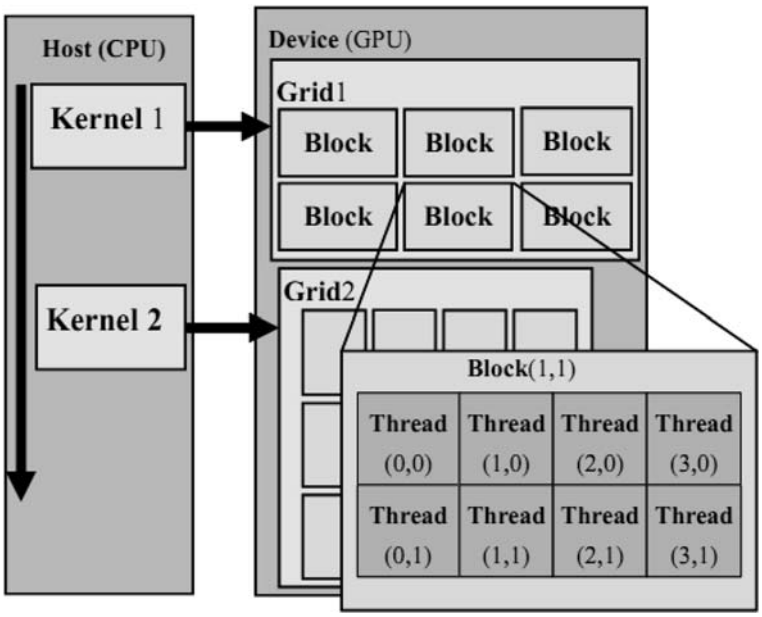


Рис. 3. Программная модель ядра GPU

CUDA основана на архитектуре SIMD (Single Instruction Multiple Data) и пригодна для эксплуатации различных уровней параллелизма данных. Данные с мелкозернистым параллелизмом могут быть захвачены потоками, и более крупнозернистый параллелизм может быть описан блоками потоков, которые представляют собой группы потоков. Вычислительная единица, которая реализует ряд потоков CUDA, называется ядром. Когда ядро выполняет работу, каждый блок потоков запланирован на каждом SM (Stream Multi-Processor). Блоки потоков запланированы и выполняются независимо друг от друга. Потоки в одном блоке выполняются пучками, и в случае перекосов поток не может перейти к следующей инструкции, если другие потоки в том же блоке еще не завершены. Программная модель ядра GPU показана на рис. 3.

В CUDA шесть видов памяти могут использоваться (регистры памяти, локальная память, разделяемая память, глобальная память и текстурная память, постоянная память) [9]. Модель памяти GPU показана на рис. 4.

В ядре для обоих алгоритмов используется общий подход: для каждого блока текущего кадра рассчитать *SAD* по всем перемещениям кандидатов и выбрать лучшее перемещение, которое сводит к минимуму функцию *SAD* (1). Мы используем два уровня параллелизма в алгоритмах поиска (рис. 5).

- *Первый уровень параллелизма* используется на уровне макроблока. Кадр разбивается на сотни или тысячи макроблоков в зависимости от разрешения изображения, и процесс поиска выполняется для каждого макроблока. Поскольку взаимозависимости между макроблоками не существует, алгоритмы могут быть применены для всех макроблоков параллельно.
- *Второй уровень параллелизма* — это параллелизм на уровне пикселей. Для вычисления *SAD* необходимо вычислить абсолютные разницы между всеми пикселями макроблока кандидата и текущего макроблока и просуммировать эти значения. При реализации абсолютные разницы одного столбца блока вычисляются параллельно в одной итерации и суммируются по алгоритму параллельной суммы префикса [10], чтобы сократить время вычисления (рис. 6).

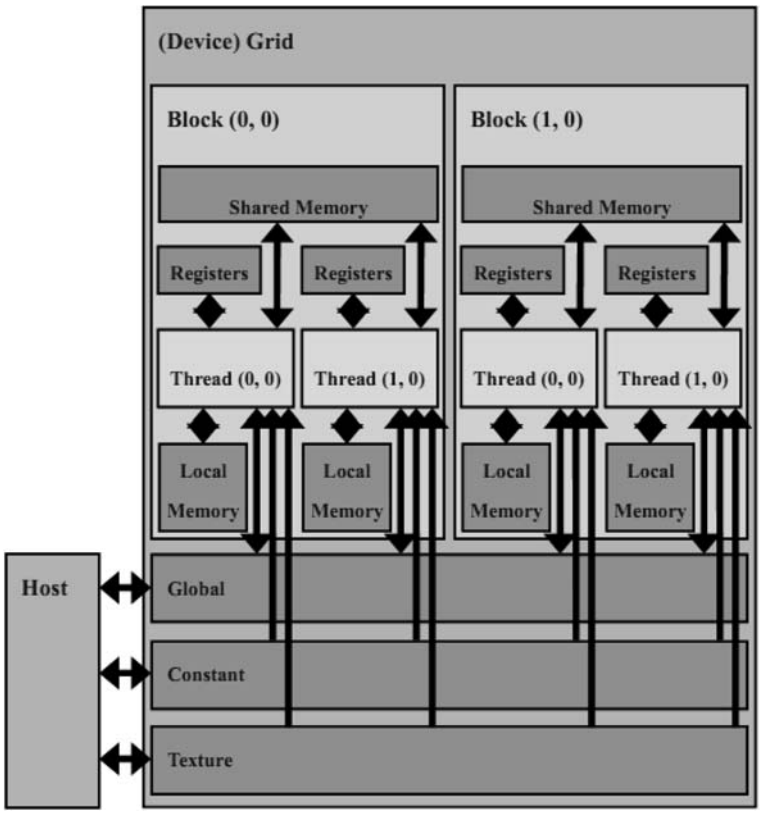


Рис. 4. Модель памяти GPU

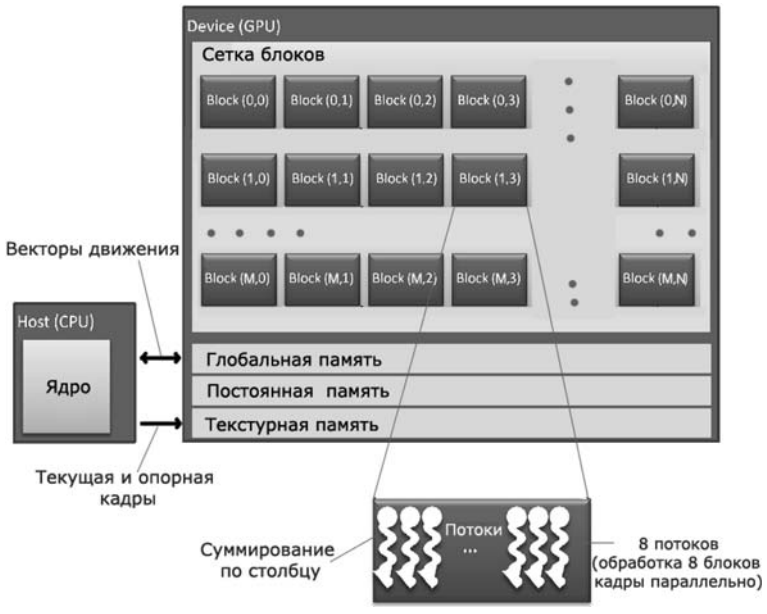


Рис. 5. Реализация алгоритмов на CUDA

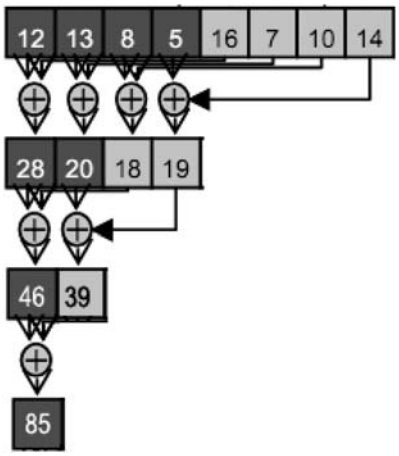


Рис. 6. Алгоритм параллельной суммы префикса

Глобальный доступ к памяти является одним из главных узких мест GPU [9]. Для минимизации потерь в реализации используются два механизма:

1) текущее, опорное изображения и кандидаты определяются нейронным способом, сохраняются в кэш-памяти 2D-текстуры;

2) все другие переменные хранятся в быстрой памяти регистров, которые связаны с процессором, расчетные значения абсолютных разниц сохраняются в разделяемой памяти, и только одна запись в глобальной памяти используется для хранения векторов движения.

Эксперименты и результаты реализации

Для эксперимента был выбран ряд видеопоследовательностей в форматах CIF (football, foreman и news) и 4CIF (crew и ice) [11]. Алгоритмы были реализованы на компьютере с процессором Intel Core i5-460M (2 ядра, 4 потока, тактовая частота 2530 МГц) и с графической картой NVIDIA GeForce 310M со следующими основными характеристиками [12]:

Процессорные ядра	16
Частота ядра, МГц	72
Частота процессора, МГц	1530
Частота памяти, МГц	800
Стандартная конфигурация	512
Интерфейс памяти	64-bit

Среднее значение *PSNR* (2) и времени нахождения векторов движения для первых 200 кадров двух алгоритмов на CUDA показаны на рис. 7 и рис. 8 (см. четвертую сторону обложки). Время вычисления такого же процесса на CPU показано на рис. 9 (см. четвертую сторону обложки).

Результаты реализации, как и в табл. 1, показывают, что значения *PSNR* предложенного алгоритма близко к значениям *PSNR* алгоритма полного перебора. Время работы предложенного алгоритма на CUDA почти в 3 раза меньше, чем алгоритма полного перебора. Результаты, приведенные на рис. 8 и рис. 9 (см. четвертую сторону обложки), тоже показывают, что время выполнения работы на CUDA гораздо меньше (в тысячи раз), чем на CPU.

Заключение

В данной работе был представлен новый алгоритм компенсации движения в видеопоследовательностях. В этом алгоритме были реализованы нейронные сети для эффективного нахождения кандидатов вектора движения. В работе рассматривается подход к реализации предложенного алгоритма и алгоритма полного перебора на базе графической карты NVIDIA. Результаты показывают, что использование GPU является хорошей альтернативой для ускорения процесса компенсации движения и, следовательно, процесса кодирования видео.

Список литературы

1. **Richardson I. E. G.** The H.264 advanced video compression standard. A John Wiley and Sons, Ltd. P. 31—40.
2. **Кубасов Д., Ватолин Д.** Обзор методов компенсации движения // Компьютерная графика и мультимедиа. 2005. Вып. № 3 (2).
3. **Yih-Chuan Lin, Shen-Chuan Tai.** Fast Full-Search Block-Matching Algorithm for Motion-Compensated Video Compression // IEEE transactions on comunications. 1997. Vol. 45, N 5.
4. **Xuan Jing, Lap-Pui Chau.** An efficient three-step search algorithm for block motion estimation // IEEE Transactions on Multimedia. 2004. Vol. 6. Is. 3.
5. **Reoxiang Li, Bing Zeng, Liou M. L.** A new three-step search algorithm for block motion estimation // IEEE Transactions on Circuits and Systems for Video Technology. Aug 1994. Vol. 4. Is. 4.
6. **Lai-Man Po, Wing-Chung Ma.** A Novel Four-Step Search Algorithm for Fast Block Motion Estimation // IEEE Trans. Circuits Syst. Video Technol. Jun. 1996. Vol. 6, N 3. P. 313—317.
7. **Shan Zhu, Kai-Kuang Ma.** A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation // IEEE Transactions on image Processing. 2000. Vol. 9, no. 2.
8. **Хайкин С.** Нейронные сети: полный курс, 2-е изд. Пер. с англ. М.: Вильямс, 2006. С. 573—622.
9. **David B. Kirk, Wen-mei, W. Hwu.** Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann Publishers, 2010. С. 78—93.
10. **Harris M.** Parallel Prefix Sum (Scan) with CUDA. URL: http://www.developer.download.nvidia.com.
11. **Richardson I. E.** The H.264 advanced video compression standard. A John Wiley and Sons, Ltd., 2010. С. 16—20.
12. **http**://www.nvidia.com/object/product_geforce_310m_us.html.

CONTENTS

The problems of long-term use of Grid-technologies for building control systems, including real-time systems are investigated. It is assumed that the supervisor system generates the flow of requests to a Host CPU of a Grid-point for the organization of distributed computing in the computer network. For each request packet of problems is solved in minimum time. A simple *compensation algorithm* for the exact optimal distribution of tasks between the packet processors uniform resource network is substantiated. In order to minimize the time of collecting the results of calculations a parallel memory structure of the baseline host processor, blocks of which are addressable objects, is proposed. Their link with the processor cluster resource allows a simultaneous communication.

Keywords: *Grid-technology, computer network, distributed calculations, Host-processor, algorithm of compensation, collecting of results*

We obtain estimates of time and hardware complexity of the computation of nonlinear polynomial function (NPF), defined over a Galois field of the form $GF(2^n)$, on multiprocessor systems with a programmable architecture, which consists of FPGA Virtex-7 series. NPF can be represented as IP-cores, that implement specific functions performed by FPGA-technology.

Keywords: FPGA, computer facilities, multiprocessing systems, programmable architecture

The paper describes a method to reconstruct the entire lifecycle of an IP packet from its creation to delivery. The method keeps track of events such as fragmentation and reassembly, multicast routing, tunneling and NAT. A software implementation for Linux-based networks is described. The method is applicable to both IPv4 and IPv6.

Keywords: network traffic analysis, IP protocol, IP packets

The majority of academic researchers present the results of their scientific activity on the Web. This trace can be used to derive useful information of their past, present activity and forecast the future intentions. Hence, social network of academic researchers can be of important value for scientific community. This information can be retrieved from various data source currently available on the Web. From each of them a separate network can be built. In this paper we present a method which can be used to combine multiple single-relational networks into a single network which will combine all relations, hence it will be multi-relational.

Keywords: multi-relational networks, academic researchers' network, multi-criteria choice

In this paper we propose a new approach to teaching practical information security in higher school based on case studies. We justify its place in information security curriculum by providing an example from our experience of using the approach for BSc and MSc students of Higher School of Economics in the courses on "Technical and Organizational Aspects of Information Security" and "Information Security Technologies". This paper fills the gap in existing practices for teaching information security which currently lack in guidelines for designing case studies and integrating them into the curriculum.

Keywords: information security, case study, Risk management, best practices

This is the major problem of neural network hardware realization is difficult to make big quantity of links between neurons with modern integration circuit or printed circuit board technologies. In this paper a new computing system architecture named attribute architecture is offered to solve this problem. In attribute architecture an exchange between neurons is made by means of information couple via single bus. Attribute neural network have ability of dynamic reconfiguration. A dynamic reconfiguration doesn't require a switch.

Keywords: neural network, dynamic reconfiguration, attribute architecture, information couple

Application of classical mathematical methods to solving decision-making problems is difficult, intelligent systems is more effective for this. Through the synthesis of neural networks, fuzzy and selective methodologies for intellectual support for management decisions, justified the author's modified fuzzy neural network, as eliminating the drawbacks of existing methodologies and more effective.

Keywords: fuzzy neural network, decision making, intelligent systems

This paper presents an algorithm for finding the motion vectors in video sequences, and its implementation approach on the basis of graphic card NVIDIA. Feature of this algorithm is a new approach of constructing a set of candidate motion vectors, and its high parallelization. The results of experimental studies were presented and compared with other algorithms for motion compensation on CPU and CUDA.

Keywords: motion compensation, video coding, neural networks, Kohonen network, GPU, CUDA

Адрес редакции:

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**

E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Технический редактор *Е. В. Конова*.

Корректор *М.Г. Джавадян*.

Сдано в набор 05.03.2012. Подписано в печать 23.04.2012. Формат 60×88 1/8. Бумага офсетная.

Усл. печ. л. 9,8. Заказ IT512. 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.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".

105120, г. Москва, ул. Нижняя Сыромятническая, д. 5/7, стр. 2, офис 2.