ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

2(138)
2008

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с ноября 1995 г.

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Аннотации статей размещены на сайте журнала по адресу http://www.**informika.ru/text/magaz/it/** или **http://novtex.ru/IT.**

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

УДК 519.6:410

О. Г. Чанышев, канд. техн. наук,
Омский филиал Института математики
им. С. Л. Соболева СО РАН

Автоматическое извлечение кандидатов в термины предметной области из представляющих ее текстов[1]

Описывается метод автоматического извлечения "доминантных" словосочетаний из текстов, представляющих предметную область. Метод основан на предварительном определении "доминант" — тематически наиболее важных слов. Представлены результаты автоматического анализа множеств текстов и эксперимента по оценке адекватности выделенных словосочетаний предметным областям.

Введение

Представляемая работа выполнена в рамках исследований возможностей автоматического построения баз знаний предметных областей путем анализа естественно-языковых текстов (ЕЯТ), ее представляющих. Вариантом может быть терминологическая база знаний (БЗ), содержащая, в простейшем случае, термины предметной области и информацию о вхождении их в предложения различных текстов.

Обзор терминоведческой литературы [например, 1—3] позволяет сделать вывод, что определения понятия "термин", позволяющего автоматически выделить его из текста, не существует. Развиваются эвристические подходы, имеющие целью извлечение из текстов *терминоподобных* [3] словосочетаний (или *кандидатов в термины* [4]). Они базируются на предположении о том, что кандидаты в термины (КТ) должны повторяться (быть устойчивыми) в одном или нескольких текстах, представляющих данную предметную область. Но к множеству устойчивых сочетаний относятся и распространенные сочетания общей лексики и семантически неполные сочетания. Смысловая завершенность подразумевает наличие существительного в составе КТ. "Наиболее часто структура словосочетаний основывается на зависимых от главного существительного прилагательных и существительных в родительном падеже" [3]. Все это вызывает необходимость фильтрации КТ. Важными представляются следующие указания: "Анализ имеющихся тезаурусов показывает, что основную массу тезаурусных единиц составляют слова-существительные, а также словосочетания из двух-трех слов", "...термины иной структуры, например, с предлогами или союзами составляют менее одного процента от общего числа терминов, включенных в тезаурусы" [3].

Авторы другой работы [4] полагают, что устойчивые цепочки должны иметь существенно большую частоту, чем цепочки, включающие их. Путем предварительной нормализации слов текста учитывается морфологическая вариативность словосочетаний.

В работе [5] рассмотрены "четыре метода автоматического извлечения двухсловных терминов из текста на основе статистики встречаемости и морфологических шаблонов". В заключении авторами указывается на сравнимую эффективность двух простых методов (*freq* и *t*-тесты). При этом *freq*-тест "использует простейшую технику — двусловия упорядочиваются по убыванию их встречаемости в тексте (т. е. частоты встречаемости отдельных слов не учитываются)."

Ассоциативная модель реального текста и доминанты

Принципиальным дополнением в рамках представляемого подхода к определению КТ является добавление условия *доминантности* [6]. Полагается, что однословные термины фиксированной предметной области следует искать среди наиболее тематически значимых слов (доминант) текстов и доминанты должны входить в состав многословных терминов.

Отбор доминант осуществляется в два этапа. Сначала отбираются *независимые лексемы связи* (НЛС) на основании ассоциативной модели реального текста (АМРТ) [7, 8]. Затем из множества НЛС выделяются *доминантные лексемы* (доминанты).

Кратко представим основные положения модели.

Формальное представление текста. За частоту слова принимается число его повторений в различных предложениях текста. Различные грамма-

[1] Работа выполнена по проекту № 1.4.2. ОМН РАН за 2007 г.

тические формы слова из его парадигмы рассматриваются как различные слова. Для исключения служебных слов языка, местоимений и некоторых других используется стоп-словарь.

Обозначим:

$L = (l_1, l_2, ..., l_i, ..., L_{Nl})$ — множество лексем (не принадлежащих стоп-словарю) в порядке появления в тексте;

$\Omega = (\omega_1, \omega_2, ..., \omega_i, ..., \omega_{Nl})$ — множество частот, соответствующих лексемам из L; $\omega_i > 1$ при любом i; $P_i = (l_{i,1}, l_{i,2}, ..., l_{i,j}, ..., l_{i,Ni})$ — предложение;

Q_i — область существования i-й лексемы, представляет собой множество из ω_i номеров предложений, в состав которых входит l_i.

Лексемы l_i, l_j ($i \neq j$) смежны, если $Q_i \cap Q_j \neq \varnothing$ ($\varnothing$ — пустое множество).

Первое предложение области существования называется порождающим предложением.

Если $Q_j = Q_i$, то l_j, l_i образуют постоянное сочетание (не обязательно контактное). Каждое постоянное сочетание можно принять за новую лексему.

Если $Q_j \subset Q_i$, то l_j есть атрибутивная лексема, а l_i — ведущая.

За установление ассоциативных связей между предложениями текста отвечает множество НЛС, обладающее тем свойством, что для каждой произвольной пары из этого множества существуют минимум два предложения, в которые они входят по отдельности.

Оставляя в силе все ранее введенные обозначения, полагаем, что множество L составляют только независимые лексемы связи (атрибутивные исключены). Введя δ_{ij} такую, что $\delta_{ij} = 1$, если j-я лексема входит в i-е предложение, и $\delta_{ij} = 0$ в противном случае, можем представить текст в виде матрицы лексических связей между предложениями. Квадратная матрица $[\lambda_{ij}]$ такая, что $\lambda_{ij} = 1$, если i-я и j-я лексемы смежны, и $\lambda_{ij} = 0$ в противном случае, представит граф $G(L, E)$, где E — множество ребер, соединяющих смежные лексемы.

На таком, чисто семиотическом уровне представления текста, единственной мерой "важности" лексемы является степень вершины в графе $G(L, E)$.

Ассоциативная мощность Ψ_i l_i-лексемы равна числу других лексем, смежных с данной, с поправкой, учитывающей только "личный" вклад лексемы в установлении связей — лексемы порождающего предложения не учитываются.

Доминантные лексемы. Из L отбираются доминантные лексемы (доминанты) [8] по критерию $\Psi > 0{,}5R$, где R — ранг текста (максимальный номер группы лексем с равными Ψ в частично упорядоченной по убыванию Ψ последовательности НЛС).

При совместном анализе множества текстов (например, в целях их кластеризации или классификации) в качестве веса доминанты принимается значение ее обратного ранга в указанной последовательности. Согласно результатам экспериментов по классификации текстов на основе анализа пересечений множеств доминант [6], можно утверждать, что доминанты представляют наиболее тематически значимое подмножество НЛС.

Таким образом, предполагается, что из каждого текста множества, представляющего собой фиксированную предметную область, будут отобраны наиболее адекватные теме словосочетания. Объединяя их, получим некоторое множество КТ предметной области. Вопрос о полноте этого множества остается открытым. Речь должна идти только о постепенном наполнении терминологической базы знаний.

Теперь уточним требования, которым должны удовлетворять КТ с нашей точки зрения.

Контактность. Слова словосочетания могут быть разделены только пробелом. Слова из стоп-списка (стоп-словаря) рассматриваются как непробельные разделители.

Устойчивость. Число повторений словосочетания в пределах анализируемого текстового файла должно быть равно или больше 2.

Доминантность. Словосочетание считается доминантным, если одно из составляющих его слов в какой-либо грамматической форме словосочетания в пределах заданного множества анализируемых текстов, является доминантой.

Смысловая завершенность и ограниченная распространенность. Словосочетания должны удовлетворять правилам, процедурно определенным в лексико-морфологическом фильтре.

Цели и задачи исследования

- Основной целью проведенных исследований явилась проверка работоспособности наших предположений о свойствах КТ. На этом пути следовало решить три основных задачи:
- оценить адекватность выделенных словосочетаний предметным областям;
- оценить потери при отклонении недоминантных сочетаний;
- разработать систему правил отклонения словосочетаний лексико-морфологического фильтра.
- Следовало оценить роль устойчивых словосочетаний в текстах деловой и художественной прозы путем определения их относительных количеств. Поэтому в состав анализируемых текстов были включены тексты жанра художественной прозы.

Основные определения

Норма (*лемма*) *слова* для каждого слова словосочетания определяется по строке морфологиче-

ской информации, выдаваемой морфопарсером *mystem*. Пример:

заводится{заводиться = V = непрош,ед,изъяв, 3-л,несов|заводить = ...

Даже в случае наличия альтернативных вариантов за норму слова принимается слово, находящееся между первой фигурной скобкой и первым знаком равенства.

В некоторых случаях мы заменяем строку морфологической информации. Например, для слова "слота" (род. падеж слова "слот") морфопарсер выдает:

слота{слота = S,жен,неод = им, ед}.

Фрагмент кода замен:

if lex = = "слота" then {norm: = "слот ";
S: = "слота{слот = S,муж, неод = род,ед}"}.

Норма словосочетания получается заменой слов сочетания на их нормы.

Парадигма словосочетания (слова) — множество грамматических вариантов словосочетания (слова) в пределах заданного множества текстов. Размер парадигмы — число вариантов.

Плотность словосочетаний есть отношение общего числа словосочетаний (до фильтрации или после) во множестве анализируемых текстовых файлов к суммарному числу предложений.

Вес доминанты в фиксированном тексте равен ее обратному рангу в убывающей по значению ассоциативной мощности последовательности доминант.

Вес нормы доминанты во множестве файлов равен сумме весов ее доминантных грамматических форм.

Вес словосочетания равен сумме весов входящих доминант.

Вес нормы сочетания равен сумме весов элементов его парадигмы.

Основные этапы выделения КТ

На вход программной системы поступает список полных имен тестовых файлов, представляющих фиксированную предметную область (или некоторый объемный текст).

- В каждом файле выделяются и классифицируются отдельные лексемы, определяются границы предложений. Полученные результаты сохраняются.
- Из каждого отдельного файла выделяются НЛС и доминанты.
- Параллельно (последовательность не важна) определяются все возможные контактные устойчивые словосочетания. Для каждой лексемы, не принадлежащей стоп-словарю, определяются контактные с ней слева и справа. Если частота контактного словосочетания больше единицы, то в справочниках соответствующих предложений вхождения эта пара лексем заменяется на их сочетание и новая "лексема" добавляется в словарь текста. Процесс повторяется для лексемы-сочетания. Доминанты (вместе с их весами) и словосочетания выводятся в отдельные файлы.
- Используя морфоанализатор *mystem*, для каждого слова сочетания получаем морфологическую информацию.
- Словосочетания отдельных файлов объединяются. Определяются веса словосочетаний, затем они группируются в парадигмы. Каждому элементу парадигмы сопоставляется список включений, содержащий наименования текстов, полные имена файлов и списки предложений вхождения для каждого файла.
- Для анализа полученных результатов определяется и выводится информация о числе словосочетаний во множестве анализируемых файлов и числе предложении. Определяются веса норм словосочетаний.
- Удаляются недоминантные словосочетания, оставшиеся словосочетания пропускаются через лексико-морфологический фильтр.
- Пересчитываются веса словосочетаний и их норм. Выводится итоговая информация.

Лексико-морфологический фильтр

Правила отклонения лексико-морфологического фильтра представлены в виде

if < условие_отклонения > then return fail

и состоят из трех последовательных групп:

- правила самого общего характера для словосочетаний произвольной длины;
- правила для двухсловных сочетаний;
- правила для словосочетаний длиной более 2.

В свою очередь, правила каждой группы по типу условия отклонения подразделяются на правила проверки принадлежности словосочетания или части словосочетания множеству стоп-словаря сочетаний и соответствия морфологическому шаблону.

В первой группе правил основным условием отклонения является наличие в составе сочетания только однобуквенных слов (например, формула или часть ее). Другое важнейшее условие отклонения — отсутствие существительного (или слова в латинице). Следует отметить, что несмотря на рискованность правила, допускающего наличие только существительных и прилагательных в последнем слове сочетания (и по частоте использования стоящего на первом месте), ничего существенного потеряно не было.

Всего процедура лексико-морфологического контроля содержит более 300 правил.

Эксперимент и результаты

Эксперимент по выделению КТ из ЕЯТ состоял из трех основных частей:

- процесс выделения словосочетаний и определения их числа в текстах жанра деловой и художественной прозы;
- оценка адекватности сочетаний предметной области;
- оценка потерь при отклонении недоминантных словосочетаний.

Выделение словосочетаний. Для первой части эксперимента и реализации информационно-поисковой системы были подобраны статьи, монографии, курсы лекций и учебники из пяти предметных областей (как правило, тексты разбивались на отдельные файлы по границам разделов, глав и т. п.): философии ("Философия", 12 текстов, 33 файла); психологии ("Психология", 19 текстов, 19 файлов); системы управления базами данных ("СУБД", 2 текста, 13 файлов); искусственного интеллекта ("Иск. Инт. ", 13 текстов, 18 файлов); политологии ("Политология", 3 текста, 32 файла). Для сравнения с результатами, представленными в работе [4], проанализирована монография Н. А. Олифер, В. Г. Олифер "Сетевые операционные системы" ("СетОпСист", 10 файлов).

"История государства Российского" Карамзина ("Карамзин", 12 файлов) включена как пример текста из слабо формализованной предметной области, но заведомо включающего большое число повторяющихся сочетаний.

Художественная литература представлена 52 рассказами Бунина ("Бунин", 52 файла), 11 рассказами Чехова ("Чехов", 11 файлов), романами Бориса Акунина ("Акунин", 5 романов, 57 файлов).

Все тексты преобразованы к линейному (*.txt) формату.

Первые 10 словосочетаний. Учитывая требования к объему статьи, приведем только первые 10 норм сочетаний множеств файлов, представляющих СУБД, искусственный интеллект "Сетевые операционные системы" и текст настоящей статьи. Словосочетания вручную преобразованы к номинативной форме. Варианты упорядоченности: а) по убыванию веса; б) по убыванию частоты (числа повторений в различных файлах).

СУБД:

а) *база данных, распределенная база данных, страница данных, сервер базы данных, объект базы данных, состояние базы данных, локальная база данных, модель данных, система баз данных, тип данных;*

б) *база данных, ограничение целостности, внешняя память, язык sql, реляционная субд, прикладная программа, оперативная память, кортеж отношения, информационная система, управление базами данных;*

СетОпСист:

а) *сетевая ос, операционная система, сервер netware, база данных, файловая система, менеджер памяти, сетевая операционная система, функции операционной системы, сервер сети, драйвер файловой системы;*

б) *операционная система, программное обеспечение, файловая система, рабочая станция, структура данных, получение доступа, передача сообщений, виртуальная память, оперативная память, реальное время;*

б.2) данные из [5] (*freq*, *t*-тест): *операционная система, файловая система, адресное пространство, ввод-вывод, оперативная память, рабочая станция, системный вызов, база данных, право доступа, программное обеспечение.*

Иск. Интел:

а) *база знаний, система искусственного интеллекта, система ии, база данных, экспертная система, решение задач, представление знаний, глобальная база данных, модель представления знаний, система продукций;*

б) *искусственный интеллект, экспертная система, представление знаний, доказательство теорем, нейронная сеть, система ии, левая часть, предметная область, база данных, база знаний.*

Настоящая статья: *норма словосочетания, недоминантное словосочетание, множество словосочетаний, число словосочетаний, плотность словосочетаний, выделение словосочетаний, множество текстов, контрольное множество, слово словосочетания, малоинформативные словосочетания.*

Числовые характеристики процесса фильтрации. Плотность словосочетаний. В табл. 1 представлены следующие показатели:

C_{all} — всего исходных словосочетаний; C_s — сумма предложений во всех текстах анализируемого множества; $P = C_{all}/C_s$ — начальное значение плотности словосочетаний; C_{nd} — число недоминантных словосочетаний; $K_{nd} = D_{nd}/C_{all}$ — коэффициент "недоминантности"; C_m — число сочетаний, не прошедших через лексико-морфологический фильтр; $C_{it} = C_{all} - (C_{nd} + C_m)$ — итоговое число словосочетаний (кандидатов в термины); $K_{flt} = (C_{nd} + C_m)/C_{all}$ — коэффициент фильтрации; P_{it} — итоговое значение плотности.

Оценка адекватности. Адекватность оценивалась путем классификации множеств норм словосочетаний, выделенных нами (контрольные множества), по предметным областям. Последние представлены наименованиями статей словарей (эталонные множества словосочетаний). Принадлежность контрольных множеств к предметной области (ПО) определялась по максимальному суммарному весу норм словосочетаний, принадлежащих пересечению с эталонными множествами. В качестве эталонных множеств словосочетаний были выбраны:

Таблица 1

Числовые характеристики процесса фильтрации

Предметная область или подборка файлов	Показатель								
	C_{all}	C_s	P	C_{nd}	K_{nd}	C_m	C_{it}	K_{flt}	P_{it}
СУБД	2009	4776	0,42	338	0,17	259	1412	0,30	0,29
СетОпСист	2187	6423	0,34	380	0,17	364	1443	0,34	0,22
Карамзин	6388	24 273	0,26	1364	0,21	942	4082	0,36	0,17
Иск. Инт.	1197	4648	0,26	257	0,21	193	747	0,37	0,16
Психология	2549	9802	0,26	689	0,27	303	1557	0,39	0,16
Философия	2615	13 580	0,19	415	0,16	316	1884	0,28	0,14
Политология	689	4155	0,12	182	0,26	64	443	0,34	0,11
Бунин	149	2264	0,07	61	0,41	23	65	0,56	0,03
Акунин	984	22 581	0,07	413	0,42	145	426	0,57	0,02
Чехов	49	1209	0,04	27	0,55	9	13	0,73	0,01

а) нормированные наименования статей "Новейшего философского словаря под редакцией Грицанова А. А. " (http://linguists.narod.ru/downloads5.html#spec, 1390 наименований, "Философия-эталон");

б) нормированные термины "Психологического словаря" (http://psi.webzone.ru/, 2172 наименования, "Психология-эталон");

в) в качестве эталонного множества сочетаний, представляющих информатику, были выбраны нормированные наименования статей "Словаря компьютерной лексики" (http://slovar.boom.ru/Head.html, 1213 наименований, "КомпЛекс-эталон").

В контрольные множества словосочетаний были включены нормы словосочетаний "СУБД", "СетОпСист", "Иск.Инт. " "Философия", "Психология". А также для контроля качества подборок мы проанализировали "Психологические теории и концепции личности. Краткий справочник" (http://www.gumer.info/bibliotek_Buks/Psihol/Psi_Teo/index.php) и нормированные двухсловные словосочетания включили в контрольную подборку ("ПсихТеор").

Результат представлен в табл. 2. Первая строка в каждой ячейке — результат до фильтрации, вторая — после фильтрации. Первое число в строке — суммарный вес словосочетаний контрольной подборки, принадлежащих множеству пересечения с эталонной подборкой, второе — размер пересечения.

Оценка потерь и малоинформативные словосочетания. Оценка потерь при удалении недоминантных словосочетаний основывалась на сравнении их множеств с множествами эталонных словосочетаний. Недоминантные сочетания из предметной области "СУБД", "Иск.Интел." и файлов "СетОпСист" были объединены в один файл ("НеДомКТ").

Результат (первое число в скобках — размер пересечения с эталоном, второе — число недоминант, третье — отношение первого ко второму):

НеДомКТ(18 910 0,020), Философия (7 415 0,017), Психология (11 648 0,017).

Процент малоинформативных словосочетаний в итоговых множествах (по визуальной субъективной оценке) также не превышает 2 %.

Обсуждение результатов

Плотность словосочетаний. Как и предполагалось, плотность словосочетаний (см. табл. 1) уменьшается в направлении от текстов деловой прозы к текстам художественной прозы. Учитывая, что значение итоговой плотности при переходе от группы текстов "Политология" к рассказам Бунина уменьшается почти в 4 раза (0,11 и 0,03), напрашивается вывод, что этот показатель может служить критерием, разделяющим два больших жанра — тексты деловой прозы и тексты художественной прозы.

Упорядоченность словосочетаний. Анализ результатов упорядочивания словосочетаний по убыванию веса, в основе которого лежит ассоциативная мощность, и по частоте повторения в раз-

Таблица 2

Результат классификации контрольных множеств словосочетаний

Эталон предметной области	КомпЛекс-эталон	Философия-эталон	Психология-эталон
СУБД	359,05 46 359,05 39		0,00 1
СетОпСист	82,05 69 82,05 56		1,79 2 1,79 1
Иск. Инт.	49,61 13 49,61 10		5,73 5 5,73 4
Философия	0,18 1 0,18 1	184,24 33 184,09 28	7,65 7 7,16 6
Психология	2,53 12 2,27 10	2,47 3 2,47 2	47,83 53 47,83 44
ПсихТеор		2,35 4 2,35 3	28,57 13 28,57 13

личных текстах позволяет сделать вывод о некоторой случайности последнего. Это хорошо заметно на примерах "СУБД" и "СетОпСист". Упорядочивание же по первому критерию хорошо укладывается в парадигму "от главного к второстепенному".

Адекватность словосочетаний предметным областям. Результат столь точной классификации итоговых множеств словосочетаний при малых размерах пересечений с эталонными словосочетаниями был ожидаем, поскольку ранее достаточно успешно классифицировались тексты по отдельным доминантам [6]. Совпадение результатов классификации итоговых и исходных множеств доказывает качество фильтрации, практически не затрагивающее наиболее тематически значимые словосочетания.

Оценка потерь. Потери являются следствием неполноты подборок текстов. Например, в подборке "Психология" недоминантными (среди прочих) оказались словосочетания "детектор лжи", "индивидуальная терапия", "корсаковский синдром". Не представляет принципиальных сложностей построение программы-агента, которая получала бы на входе недоминантные словосочетания, искала в сети тексты, их включающие, и проверяла на доминантность.

Выполнение условия доминантности во многом обеспечивает удовлетворение условию объектности — наличию существительных в словосочетаниях: процент "несуществительных" в множестве доминант не превышает 25 %. При этом наибольший процент "несуществительных" дают художественные тексты или такие, как "История государства Российского" Карамзина, том 2, глава I, — 25 %.

Малоинформативные словосочетания в основном концентрируются в конце последовательности, упорядоченной по убыванию веса, однако для введения некоторого критического значения веса пока нет фактических оснований.

Выводы

Контроль доминантности обеспечивает высокую степень адекватности выделенных словосочетаний предметной области и, совместно с лексико-морфологическим фильтром, их эффективную фильтрацию.

Избегая оценки степени "терминоподобия" итоговых множеств словосочетаний, можно считать описанный способ их выделения и "взвешивания" наиболее эффективным (по сравнению с известными) для автогенерации русскоязычных баз знаний систем терминологического типа.

Список литературы

1. **Головин Б. Н., Кобрин Р. Ю.** Лингвистические основы учения о терминах. М.: Высшая школа, 1987.
2. **Татаринов В. А.** Теория терминоведения: В 3 т. Т. 1. Теория термина: История и современное состояние. М.: Московский Лицей, 1996.
3. **Добров Б. В., Лукашевич Н. В., Сыромятников С. В.** Формирование базы терминологических словосочетаний по текстам предметной области // Тр. 5-й Всероссийской научной конференции "Электронные библиотеки: перспективные методы и технологии, электронные коллекции" — RCDL2003, Санкт-Петербург. СПб.: Изд-во СпбГУ. 2003.
4. **Гусев В. Д., Саломатина Н. В.** Алгоритм выявления устойчивых словосочетаний с учетом их вариативности (морфологической и комбинаторной) // Тр. Междунар. конф. "Диалог—2004. Компьютерная лингвистика и интеллектуальные технологии". М.: Наука, 2004. С. 530.
5. **Браславский П., Соколов Е.** Сравнение четырех методов автоматического извлечения двухсловных терминов из текста // http://www.dialog-21.ru/dialog2006/materials/html/Braslavski.htm
6. **Чанышев О. Г.** О возможности построения онтологий на основе доминантных лексем: результаты автоклассификации текстов // Вестник Омского государственного университета. Омск: ОмГУ. 2004. Вып. 3. С. 45—47.
7. **Чанышев О. Г.** Ассоциативная модель реального текста и ее применение для автогенерации баз знаний о связях. Дис. ... / Омск, ОФИМ СО РАН. 1998. 120 с.
8. **Чанышев О. Г.** Ассоциативная модель реального текста и ее применение в процессах автоиндексирования // Тр. Седьмой национ. конф. по искусственному интеллекту с международным участием КИИ'2000. М.: Физико-математическая литература, 2000. С. 430—438.

УДК 510.6

Н. П. Брусенцов, канд. техн. наук,
МГУ им. М. В. Ломоносова

Алгебраическая реконструкция силлогистики

Воссоздание категорической силлогистики Аристотеля на основе несовместимого с законом исключенного третьего диалектического принципа сосуществования противоположностей.

Логику, которая по мнению Аристотеля "прокладывает путь к началам всех учений" [1, 101b3], принятый стоиками противоестественный закон исключенного третьего превратил в парадоксальную "мертвую схоластику". Эту практически бесполезную, а в условиях нынешней формальной информатизации крайне опасную, логику по недоразумению считают аристотелевой и пытаются исправить положение, изобретая "неаристотелевы логики". Но ведь в подлинной логике Аристотеля нелепостей не было — его силлогистика непарадоксальна, а непогружаемость ее в современные

логические исчисления явно указывает на их неадекватность. Поэтому совершенствовать логику надлежит путем реанимации, осознания и развития того, что создано Аристотелем.

В силлогистике общеутвердительная посылка "Все x суть y" выражает отношение необходимого следования y из x, определенное Аристотелем в "Первой аналитике" [1, 57b1]:

"... когда два [предмета] относятся друг к другу так, что если есть один, необходимо есть и второй, тогда, если нет второго, не будет и первого; однако, если второй есть, то не необходимо, чтобы был первый. Но невозможно, чтобы одно и то же было необходимо и когда другое есть, и когда его нет".

Здесь "первый" и "второй" суть качества-термины x и y, сочетаниями присущности которых — x, y и антиприсущности — x', y' обозначаются сущности рассматриваемых вещей. Например, xy обозначает сущность вещи, которой совместно присущи качества x и y, а xy' — присущность x и антиприсущность y. То, что "второй" необходимо есть (не может не быть), когда есть "первый", составляет следование y из x, а то, что "первого" не может быть, когда нет "второго" — следование x' из y'. Ни одна x-вещь не может быть xy'-вещью и поэтому непременно будет xy-вещью, а каждая y'-вещь соответственно будет $x'y'$-вещью. Качество y, присущее x-вещам, необходимо содержится в их сущности, поэтому следование $x \Rightarrow y$ называют содержательным. Однако x'-вещам то же качество y не присуще и не антиприсуще, для них оно несущественно (может быть, а может не быть). Аристотель называет такое "привходящим". Это именно то диалектическое третье, которое в формальной логике устранено принятием закона исключенного третьего.

Лишившись своего фундаментального отношения трехзначного необходимого следования, немыслимого в условиях двухзначности, логика утратила адекватность, стала практически бесполезной, а в условиях нынешней огульной информатизации весьма небезопасной и вредной. Дело не столько в безуспешности усилий по созданию искусственного интеллекта, как в извращении и разрушении интеллекта естественного. Обученные противоестественной логике люди не полагаются на здравый смысл, а уверенно действуют по ее нелепым правилам. Поразительно, что на протяжении более двух тысяч лет буквально единицы смогли устоять против дьявольского закона (Раймонд Луллий, Дунс Скот, Уильям Оккам, Михаил Ломоносов, Льюис Кэррол), навлекая на себя всеобщую неприязнь. Пресловутый закон настойчиво приписывается Аристотелю, вопреки очевидной трехзначности его силлогистики, которая оттого и не вписывается в современные логические исчисления, что в них недостает третьего, частное не подчинено общему и из ничего следует все, что угодно, вопреки тому что из ничего следовать ничто не может.

Нельзя не признать, что традиционное (экстенсиональное) истолкование силлогистики труднопостижимо, о чем убедительно свидетельствует уже то, что в общеутвердительной посылке все еще не распознано отношение следования. Но ведь уже более ста лет существует "Символическая логика" Льюиса Кэррола [2] с содержательной интенсиональной трактовкой, но она игнорируется официальной наукой. Впрочем, игнорируют либо извращают и разъяснения самого Аристотеля, например, 43b—44b в "Первой аналитике", как и цитированное выше определение следования.

Ян Лукасевич, исследуя силлогистику "с точки зрения современной формальной логики", заключил [3, с. 94], что аристотелево утверждение "Но невозможно, чтобы одно и то же было необходимо и когда другое есть, и когда его нет" ошибочно, ибо ему не удовлетворяет двухзначный эрзац отношения следования — материальная импликация. Это впечатляющий пример заключения посредством логики, в которой нет следования: ведь несвойственность подчеркнутой Аристотелем существенной особенности отношения следования материальной импликации свидетельствует как раз о том, что эта импликация не есть следование. Действительно, ей присущи парадоксы, которых ни у полноценного аристотелева следования, ни в силлогистике в целом нет.

С тем, что парадоксы в логике недопустимы, согласны все, и бесчисленные попытки устранить их настойчиво предпринимаются на протяжении многих лет, но тщетно — в двухзначной логике они неустранимы. А поскольку в неизвращенной логике Аристотеля парадоксов нет, то надо устранять причину, по которой они возникли, — закон исключенного третьего. Адекватная логика должна быть трехзначной, допускающей наряду с "есть" и "нет" промежуточное — "может быть, может не быть", и, как справедливо отметил Аристотель [1, 43a44]: "Рассуждения и исследования имеют своим предметом главным образом, пожалуй, это промежуточное".

Как видно из аристотелева определения, первопричина следования $x \Rightarrow y$, $y' \Rightarrow x'$ в несовместимости x с y', которая равносильна несуществованию xy' при сосуществовании x-вещей и y'-вещей. Именно в силу этой несовместимости x-вещи не могут не быть y-вещами, а y'-вещи непременно будут x'-вещами.

Сосуществованием противоположностей по каждому из рассматриваемых терминов — $\mathrm{V}x\mathrm{V}x'\mathrm{V}y\mathrm{V}y'$... определена сущность положенного в основание силлогистики универсума Аристотеля УА [4]. Парадоксы логики обусловлены нарушениями принципа сосуществования противоположностей, порождающими химеры — суждения о беспредметных, ничего не означающих терминах. Для оздоровления логики достаточно неукоснительного соблюдения принципа сосуществова-

ния противоположностей, поскольку с законом исключенного третьего он несовместим.

Силлогистика в интенсиональном истолковании есть учение о сущностях качеств, об определяемых совокупностями качеств вещах и об отношениях, которыми могут быть взаимосвязаны качества и вещи, а также и сами отношения друг с другом. Первичными сущностями полагаются не единичные (индивидные) вещи, как это принято в экстенсиональной традиционной силлогистике, а простейшие несоставные качества, обозначаемые терминами x, y, z, ..., сущности которых выявляются сопоставлением вещей, обладающих данным, например x, качеством, с вещами, антиобладающими им, с x'-вещами.

Качество, которым вещь обладает, содержится в ее сущности и называется присущим этой вещи. Антиобладаемое качество называется антиприсущим либо неприсущим. Однако имеется еще третье — привходящее, т. е. такое качество, обладание которым для сущности данной вещи не имеет значения. Например, ромбу прямоугольность не присуща, хотя и не антиприсуща. Поэтому вместо присущности и антиприсущности качества x говорят о присущности *особенностей* x либо x', ни одна из которых не присуща, если качество x несущественно.

Алгебраически несущественность термина выражают умалчиванием его в характеризующем сущность вещи сочетании особенностей. Например, вещь, которой присуще x, антиприсуще z, а y не присуще и не антиприсуще, характеризуется конъюнкцией xz', умалчивающей термин y. Аналогично, в выражении необходимого следования

$$(x \Rightarrow y) \equiv \mathrm{V}'xy'\mathrm{V}xy\mathrm{V}x'y'$$

члены $\mathrm{V}xy$ и $\mathrm{V}x'y'$ означают принадлежность характеризуемой совокупности xy-вещей и $x'y'$-вещей, а член $\mathrm{V}'xy'$ — антипринадлежность xy'-вещей, тогда как несущественный член $\mathrm{V}x'y$ умалчивается.

Рассматриваемое выражение отношения следования $x \Rightarrow y$ представляет собой подмножество реализованного в УА декартова произведения

$$\{x, x'\} \times \{y, y'\} \equiv \{xy, xy', x'y, x'y'\},$$

в котором исключена принадлежность ему xy'-вещей признанием их несуществующими — $\mathrm{V}'xy'$. В таком подмножестве x-вещи и y'-вещи необходимо становятся xy- и $x'y'$-вещами, возмещающими, в частности, существование x'- и y-вещей:

$$\mathrm{V}'xy'\mathrm{V}x\mathrm{V}y \equiv \mathrm{V}'xy'\mathrm{V}xy,\ \mathrm{V}'xy'\mathrm{V}x'\mathrm{V}y' \equiv \mathrm{V}'xy'\mathrm{V}x'y',$$

$$\mathrm{V}'xy'\mathrm{V}x\mathrm{V}x'\mathrm{V}y\mathrm{V}y' \equiv \mathrm{V}'xy'\mathrm{V}xy\mathrm{V}x'y'.$$

В экстенсиональном универсуме УБ, составляющем основание двухзначной булевой логики классов, несуществование xy'-вещей равносильно дополнению класса xy', определяющему отношение материальной импликации

$$(x \to y) \equiv \neg(xy') \equiv xy \vee x'y \vee x'y' \equiv x' \vee y.$$

Это отношение соблюдено при наличии y-вещей, независимо от существования x-вещей. Более того, оно необходимо выполняется, когда x-вещей нет, причем независимо от того, существуют y-вещи или нет: "из невозможного следует все, что угодно". Но ведь следовать должно нечто определенное, и бессмысленно следование из ничего. Ясно, что материальная импликация не отношение, а типичная химера, суждение невесть о чем. Стоики подправили положение, введя *modus ponens*, требующий существования x-вещей, а в современной логике парадоксы импликации "узаконены". Поэтому безупречные модусы силлогистики математической логикой отвергаются, как недоказуемые. Более того, истинность общего суждения, вопреки здравому смыслу, не влечет истинности подчиненного ему частного. Так, из "Все x суть y" не следует "Некоторые x есть y". Причина этих нелепостей якобы в оправдываемом математическими применениями логики отклонении от аристотелева истолкования общеутвердительного суждения [5, с. 79]. На самом деле все обусловлено двухзначностью математической логики, воспроизводящей осуществленную античными стоиками подмену трехзначного содержательного следования парадоксальной материальной импликацией.

Упорядоченная на основе принципа сосуществования противоположностей силлогистика лишена возможности неадекватных истолкований. Суждение "Все x суть y" в ней неотождествимо с "Не существуют xy'", тогда как подобное допускал даже Льюис Кэррол, полагая, что общеотрицательное суждение Exy тождественно $\mathrm{V}'xy$, т. е. не требует сосуществования своих терминов. В воссозданной посредством щадящего принципа силлогистике четыре вида общих и четыре вида частных суждений:

$$\mathrm{A}xy \equiv \mathrm{A}y'x' \equiv \mathrm{V}'xy'\mathrm{V}xy\mathrm{V}x'y',\quad \mathrm{I}xy \equiv \mathrm{V}xy\mathrm{V}x'\mathrm{V}y',$$

$$\mathrm{A}xy' \equiv \mathrm{A}yx' \equiv \mathrm{V}'xy\mathrm{V}xy'\mathrm{V}x'y,\quad \mathrm{I}xy' \equiv \mathrm{V}xy'\mathrm{V}x'\mathrm{V}y,$$

$$\mathrm{A}x'y \equiv \mathrm{A}y'x \equiv \mathrm{V}'x'y'\mathrm{V}xy'\mathrm{V}x'y,\quad \mathrm{I}x'y \equiv \mathrm{V}x'y\mathrm{V}x\mathrm{V}y',$$

$$\mathrm{A}x'y' \equiv \mathrm{A}yx \equiv \mathrm{V}x'y\mathrm{V}xy\mathrm{V}x'y',\quad \mathrm{I}x'y' \equiv \mathrm{V}x'y'\mathrm{V}x\mathrm{V}y.$$

Традиционный "квадрат противоположностей" $\mathrm{A}xy$—$\mathrm{A}xy'$—$\mathrm{I}xy'$—$\mathrm{I}xy$ в действительности оказывается кубом с вершинами $\mathrm{A}xy$—$\mathrm{A}xy'$—$\mathrm{A}x'y'$—$\mathrm{A}x'y$, $\mathrm{I}xy$—$\mathrm{I}xy'$—$\mathrm{I}x'y'$—$\mathrm{I}x'y$. Каждой общей посылке подчинена пара частных. Например: $\mathrm{A}xy \models \mathrm{I}xy\mathrm{I}x'y'$, $\mathrm{A}xy' \models \mathrm{I}xy'\mathrm{I}x'y$. Каждой общей посылке две общих контрарны, а третья обратна ей. Например: посылке $\mathrm{A}xy$ контрарны $\mathrm{A}xy'$ и $\mathrm{A}x'y$, а $\mathrm{A}x'y' \equiv \mathrm{A}yx$ обратна. В случаях невыполнения обеих контрарных посылок выполняется одна из двух прочих. Например: если невозможны $\mathrm{A}xy$ и $\mathrm{A}xy'$, то имеет место $\mathrm{A}yx$ либо $\mathrm{A}x'y$. Каждая общая посылка несовместима с одной из частных и две частные подчинены ей. Например: $\mathrm{A}xy$ несовместима с $\mathrm{I}xy'$ и подчиняет $\mathrm{I}xy$ и $\mathrm{I}x'y'$, тогда как $\mathrm{A}xy'$ несовместима с $\mathrm{I}xy$ и подчиняет $\mathrm{I}x'y$ и $\mathrm{I}xy'$.

Силлогизмы, выявляющие отношение, которым взаимосвязаны друг с другом термины x и z, из отношений, связывающих x с y и y с z, сводятся к четырем видам заключений:

$$Axy Ayz \models Axz, \quad Axy Ay'z \models Ix'z,$$
$$Ixy Ayz \models Ixz, \quad Axy Iy'z \models Ix'z.$$

Инверсией одного, двух либо всех трех терминов из этих выражений получаются 32 модуса силлогизмов первой фигуры. Например, восемь модусов первой фигуры с общими заключениями:

$$Axy Ayz \mid = Axz, \quad Axy Ayz' \mid = Axz',$$
$$Axy' Ay'z \mid = Axz, \quad Ax'y Ayz' \mid = Ax'z',$$
$$Ax'y' Ay'z' \mid = Ax'z', \quad Ax'y Azy \mid = Ax'z,$$
$$Axy' Ay'z' \mid Axz', \quad Ax'y' Ay'z \mid = Ax'z.$$

Аналогично и в таком же количестве формируются модусы других фигур инвертированием терминов в соответствующих четверках заключений.

Для второй фигуры:

$$Axy Azy' \mid = Axz', \quad Axy Azy \mid = Ix'z',$$
$$Ixy Azy' \mid = Ixz', \quad Axy Iz'y' \mid = Ix'z'.$$

Для третьей фигуры:

$$Ayx Ay'z \mid = Ax'z, \quad Ayx Ayz \mid = Ixz,$$
$$Iy'x' Ay'z \mid = Ix'z, \quad Ayx Iyz \mid = Ixz.$$

Для четвертой фигуры:

$$Ayx Azy \mid = Ax'z', \quad Ay'x Azy \mid = Ixz',$$
$$Ayx Izy \mid = Ixz, \quad Iy'x Axy \mid = Ixz'.$$

Путем представленного на примере первой фигуры инвертирования терминов в четырех фигурах всего получается 128 правильных модусов категорического силлогизма вместо 19, установленных традиционной силлогистикой, и 15 из них, признаваемых математической логикой. В воссозданной таким образом на основе принципа сосуществования противоположностей силлогистике нет ни парадоксов, ни химер. Это полноценное, адекватное реальности и естественному интеллекту начало логики бытия.

Исчерпывающая алгебраизация важнейшего фрагмента силлогистики убедительно подтверждает реальность воссоздания естественной, благоразумной логики Аристотеля на основе диалектического принципа сосуществования противоположностей. Это непростая и весьма трудоемкая, но принципиально неизбежная перестройка неадекватной двоичной информатики с ее компьютерным арсеналом и вынуждаемого ею парадоксального мышления. Ключевое условие практической реализации столь основательного оздоровления логики — создание троичного компьютера, безукоризненно воплощающего диалектический принцип.

Список литературы

1. **Аристотель.** Сочинения в четырех томах. М.: Мысль. Т. 1. 1975; т. 2. 1978.
2. **Кэррол Л.** Символическая логика // Льюис Кэррол. История с узелками. М.: Мир, 1973. С. 189—361.
3. **Лукасевич Я.** Аристотелевская силлогистика с точки зрения современной формальной логики. М.: ИЛ, 1959.
4. **Брусенцов Н. П.** Блуждание в трех соснах (Приключения диалектики в информатике). М.: SvR-Аргус, 2000. http://ternarycomp.narod.ru/3PINES. DOC.
5. **Гильберт Д., Аккерман В.** Основы теоретической логики. М: ИЛ, 1947.

УДК 519.237.8

А. С. Тарасова,
Воронежский государственный университет

Методы определения оптимальной геометрической формы кластеров в задачах кластерного анализа

Многие алгоритмы кластеризации продуцируют кластеры определенной формы, что может привести к принципиально неверным результатам. Предложен метод определения оптимальной геометрической формы кластеров, основанный на подборе матриц весов признаков для каждого кластера и их последующей геометрической интерпретации. Данный метод актуален в задаче формирования базы знаний на основе "если-то" правил.

Введение

Применение многих алгоритмов кластеризации продуцирует кластеры определенной формы, т. е. навязывает данным неприсущую им структуру, что приводит не только к неоптимальным, но иногда и к принципиально неверным результатам. Поэтому появились методы кластеризации, позволяющие извлекать кластеры различной геометрической формы. Например, метод Густавсона—Кесселя [2] позволяет выделять кластеры в форме эллипсоидов, а алгоритм кластеризации К-средних и нечеткий алгоритм С-средних продуцируют кластеры сферической формы. Вопросы, касающиеся формы кластеров, описаны в [1, 2]. В [4, 8] предлагаются методы аппроксимации функций эллипсоидальной формы с помощью *если-то* правил.

В данной статье предложен алгоритм определения геометрической формы кластеров, макси-

мально приближенной к естественной. Данный алгоритм актуален в задаче формирования базы знаний при моделировании нечетких систем управления на этапе построения функций принадлежности нечетких посылок и заключений, входящих в *если-то* правила. Предложенный метод позволит повысить уровень адекватности нечетких моделей.

Итак, целью статьи является разработка метода определения геометрических форм кластеров, максимально приближенных к естественным.

Основная идея подхода состоит в следующем. Формы кластеров мы будем описывать кривыми второго порядка, каждую из которых задает матрица квадратичной формы. Находить данные матрицы мы будем для каждого кластера и каждой пары координат с помощью специально составленной задачи оптимизации, в которой целевая функция максимизирует качество разбиения.

Нами получены две модели определения оптимальной формы кластера. Одна из моделей предполагает, что матрица квадратичной формы определяет расстояние, другая же модель не накладывает такого ограничения на искомую матрицу.

Постановка задачи и необходимые теоретические сведения

Рассмотрим постановку задачи кластерного анализа. Пусть N — число рассматриваемых объектов; $j = \overline{1, N}$ — номер объекта; $X = \{x^j, j = \overline{1, N}\}$ — множество рассматриваемых объектов, где каждый объект $x^j = (x_1^j, \ldots, x_K^j)$ характеризуется K признаками. Требуется провести разбиение данных на кластеры (группы данных, которые близки между собой и далеки друг от друга).

Пусть выполнена кластеризация данных и выделено M кластеров; $i = \overline{1, M}$ — номер кластера; $C = \{c_i, i = \overline{1, M}\}$ — множество центров кластеров; S_i — i-й кластер; μ_{ij} — степень принадлежности объекта x_j кластеру S_i; r — радиус шара, описывающего кластер; $d(x_j c_i)$ — функция, позволяющая оценить близость элемента $x^j \in X$ к центру c^i.

При использовании стандартной евклидовой нормы в один кластер группируют точки, находящиеся на расстоянии не меньше r от центра кластера. В результате кластеры приобретают форму гиперсферы. Наряду с евклидовой, в кластерном анализе применяются также диагональная норма и норма Махаланобиса [2].

Известно, что форма кластеров зависит от нормы, используемой для расчета расстояния между элементами [2, 4].

В общем виде норму можно задать с помощью симметричной положительно определенной матрицы B размерностью $K \times K$, где K — число признаков [2]:

$$d_B^2(x^j, c^i) = \|x^j - c^i\|_B^2 = (x^j - c^i)B(x^j - c^i)^{\mathrm{T}}, \quad (1)$$

где т — операция транспонирования.

Для евклидовой нормы матрица B представляет собой единичную матрицу. Для диагональной нормы — матрицу, на главной диагонали которой стоят веса координат. Диагональная норма позволяет выделять кластеры в виде гиперэллипсоидов, ориентированных вдоль координатных осей. Для нормы Махаланобиса матрица $B = R^{-1}$, где R — ковариационная матрица. Норма Махаланобиса позволяет выделять кластеры в виде гиперэллипсоидов, оси которых могут быть ориентированы в произвольных направлениях.

Так как в общем случае мы можем обрабатывать данные сколь угодно большой размерности, то нет смысла находить матрицы, описывающие кластер в исходной размерности, ведь их невозможно будет изобразить. Также для обработки матриц второго порядка требуется меньшее время и уменьшается сложность вычислений по сравнению с многомерным случаем. Поэтому мы поставим задачу нахождения таких матриц для каждой пары координат, а затем будем изображать кластеры с соответствующими им кривыми на плоскости.

Квадратичной формой $f(x_1, \ldots, x_K)$ от неизвестных $x_1, \ldots x_K$ называется однородный многочлен $f(x_1, \ldots, x_K) = \sum\limits_{i=1}^{K} a_{ii}^2 x_i^2 + 2 \sum\limits_{1 \le i < j \le K} a_{ij} x_i x_j$ степени 2, где $a_{ij} = a_{ji}$ и не все коэффициенты a_{ij} нулевые [3].

Квадрикой, или *гиперповерхностью второго порядка* в аффинном пространстве называется множество решений уравнения

$$f(x_1, \ldots, x_K) + 2\alpha(x_1, \ldots, x_K) + \alpha_0 = 0,$$

где $f(x_1, \ldots, x_K)$ — квадратичная форма; $\alpha(x_1, \ldots, x_K)$ — линейная или нулевая форма; α_0 — действительное число.

Пусть $f(x_1, \ldots, x_K)$ — квадратичная форма. Согласно [5], существует такая обратимая матрица T (которую назовем матрицей преобразования или замены неизвестных), что если

$$\begin{pmatrix} x_1 \\ \vdots \\ x_K \end{pmatrix} = T \begin{pmatrix} X_1 \\ \vdots \\ X_K \end{pmatrix},$$

где $X_1, \ldots, X_K$ — новые неизвестные, то при подстановке выражений x_1, x_K через $X_1, \ldots, X_K$ в $f(x_1, \ldots, x_K)$ получим

$$f(x_1, \ldots, x_n) = f(X_1, \ldots, X_K) =$$
$$= X_1^2 + \ldots + X_s^2 - X_{s+1}^2 - \ldots - X_r^2,$$

причем для формы f такой вид единственен.

Пусть уравнение квадрики в аффинном пространстве над n-мерным векторным пространством задано в виде

$$f(x_1, ..., x_K) + 2\alpha(x_1, ..., x_K) + \alpha_0 = 0.$$

Тогда согласно [3], существует такая система координат в аффинном пространстве, в которой уравнение квадрики в новых координатах единственно и имеет вид

$$X_1^2 + ... + X_s^2 - X_{s+1}^2 - ... - X_r^2 = \begin{cases} 1 \\ -1 \\ 0 \\ 2X_{r+1} \end{cases},$$

где $s \geqslant r - s$.

Приложением аффинной классификации квадрик является случай, когда размерность пространства равна двум. На плоскости квадрика представляет собой кривую второго порядка. В таблице приведены виды уравнений, которые могут иметь кривые второго порядка (s — число положительных собственных значении, r — число собственных значений).

В расстоянии Махаланобиса используется матрица, обратная к ковариационной, которая на самом деле является матрицей квадратичной формы. Уйдем от понятия расстояния как такового, и в выражение (1) будем подставлять другие матрицы квадратичных форм. Тогда с помощью таких функций можно будет описывать классы, имеющие форму эллипса, гиперболы и прямой.

Рассмотрим, каким образом определяется тип кривой второго порядка и ее параметры. Матрица

Классификация кривых второго порядка

r	s	Уравнение	Тип кривой второго порядка
2	2	$X_1^2 + X_2^2 = 1$	Эллипс
		$X_1^2 + X_2^2 = -1$	Мнимый эллипс (∅)
		$X_1^2 + X_2^2 = 0$	Пара мнимых прямых, пересекающихся в действительной точке (точка)
2	1	$X_1^2 - X_2^2 = 1$	Гипербола
		$X_1^2 - X_2^2 = 0$	Пара пересекающихся прямых
1	1	$X_1^2 = 1$	Пара параллельных прямых
		$X_1^2 = -1$	Пара мнимых параллельных прямых (∅)
		$X_1^2 = 0$	Пара совпадающих прямых
		$X_1^2 = 2X_2$	Парабола

квадратичной формы раскладывается следующим образом:

$$B = PDP^{\mathrm{T}}, \qquad (2)$$

где $D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$ — матрица, на диагонали которой стоят собственные значения матрицы B; $P = \begin{pmatrix} e_1^1 & e_1^2 \\ e_2^1 & e_2^2 \end{pmatrix}$ — матрица единичных собственных векторов матрицы B, которая задает угол поворота системы координат Θ, т. е. $P = \begin{pmatrix} \cos\Theta & \sin\Theta \\ \sin\Theta & -\cos\Theta \end{pmatrix}$. В этой системе координат уравнение квадратичной формы является каноническим:

$$f(X_1, X_2) = \lambda_1 X_1^2 + \lambda_2 X_2^2,$$

где X_1, X_2 — переменные в новой системе координат.

Таким образом, для построения требуемых кривых нам необходимо:

- построить кривую, которая задается каноническим уравнением в плоскости XOY;
- повернуть систему координат на угол Θ;
- сдвинуть начало координат в центр рассматриваемого кластера.

Пусть $r^2 = (x^j - c^i)B(x^j - c^i)^{\mathrm{T}}$, тогда длины каждой из половинных осей эллипсоида равны соответственно $r/\sqrt{\lambda_1}$, $r/\sqrt{\lambda_2}$.

Предположим, что нами найдена матрица, задающая форму кластера. Для того чтобы определить, какой кривой соответствует данная квадратичная форма, используют следующий *алгоритм*:

1. Определение собственных векторов и собственных значений матрицы.

2. Анализ собственных значений:

a) если $\lambda_1 > 0$, $\lambda_2 > 0$, то кривая представляет собой эллипс (в нашем случае радиус положительный, поэтому случай мнимого эллипса и пары мнимых прямых можно не рассматривать);

b) если $\lambda_1\lambda_2 < 0$, то кривая представляет собой гиперболу (пару пересекающихся прямых по той же причине не рассматриваем);

c) если $\lambda_1 = 0$ или $\lambda_2 = 0$, то квадратичная форма задает пару параллельных прямых (см. выше).

Математическая модель определения оптимальной формы кластера

Модель 1. Поиск матриц квадратичной формы для кластеров. Пусть проведено разбиение данных на кластеры. Требуется описать геометрическую форму каждого из полученных кластеров. Для этого необходимо найти для каждого кластера матрицу квадратичной формы, участвующую в вычислении расстояния по формуле (1) и опреде-

ляющую геометрическую форму кластера. Для нахождения данной матрицы предлагается решить задачу безусловной оптимизации, которая с учетом введенных обозначений принимает вид

$$F(S_i,\ B_i) = \\ = \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0{,}5\}} \mu_{ij}(d^2_{B_i}(x^j,\ c^i))^2 \to \min, \quad (3)$$

где B_i обозначает искомую матрицу квадратичной формы для каждого кластера.

Заметим, что $d^2_{B_i}$ может быть отрицательным, поэтому мы не говорим о расстоянии. Задачу можно поставить и как задачу поиска расстояния, но тогда необходимо потребовать соответствующее условие неотрицательности.

Согласно формуле (2), матрица B_i может быть разложена по формуле $B_i = P_i D_i P_i^{\mathrm{T}}$, где в случае плоскости

$$D_i = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix};\ P_i = \begin{pmatrix} \cos\Theta & \sin\Theta \\ \sin\Theta & -\cos\Theta \end{pmatrix}.$$

От поиска матрицы квадратичной формы мы перейдем непосредственно к поиску кривой второго порядка, тогда переменными в нашей задаче являются Θ, λ_1, λ_2.

Данную задачу будем решать методом градиентного спуска [7]. Для этого необходимо найти частные производные целевой функции по каждой переменной. Обозначим $\Delta_i = d^2_{B_i}(x_j, \nu_i)$. С учетом введенных обозначений производные целевой функции записываются в следующем виде:

$$\frac{\partial F}{\partial\Theta} = \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0{,}5\}} 2\mu_{ij}(d^2_{B_i}(x_j,\ \nu_i))\frac{\partial\Delta_i}{\partial\Theta};$$

$$\frac{\partial F}{\partial\lambda_1} = \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0{,}5\}} 2\mu_{ij}(d^2_{B_i}(x_j,\ \nu_i))\frac{\partial\Delta_i}{\partial\lambda_1};$$

$$\frac{\partial F}{\partial\lambda_2} = \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0{,}5\}} 2\mu_{ij}(d^2_{B_i}(x_j,\ \nu_i))\frac{\partial\Delta_i}{\partial\lambda_2}.$$

Найдем частные производные функции Δ_i по каждой из переменных:

$$\frac{\partial\Delta_i}{\partial\Theta} = 2\cos\Theta\sin\Theta(x_{jp}-\nu_{ip})^2(\lambda_2-\lambda_1) + \\ 2(x_{jp}-\nu_{ip})(x_{jq}-\nu_{iq})(\lambda_1-\lambda_2)[\cos^2\Theta-\sin^2\Theta] + \\ + 2\cos\Theta\sin\Theta(x_{jq}-\nu_{iq})^2(\lambda_1-\lambda_2);$$

$$\frac{\partial\Delta_i}{\partial\lambda_1} = (x_{jp}-\nu_{ip})^2\cos^2\Theta + 2(x_{jp}-\nu_{ip}) \times \\ \times (x_{jq}-\nu_{iq})\cos\Theta\sin\Theta + (x_{jq}-\nu_{iq})^2\sin^2\Theta;$$

$$\frac{\partial\Delta_i}{\partial\lambda_2} = (x_{jp}-\nu_{ip})^2\sin^2\Theta - 2(x_{jp}-\nu_{ip}) \times \\ \times (x_{jq}-\nu_{iq})\cos\Theta\sin\Theta + (x_{jq}-\nu_{iq})^2\cos^2\Theta.$$

Подставляя значения частных производных в формулы поиска решения по методу градиентного спуска, получаем оптимальные значения переменных Θ^*, λ_1^*, λ_2^*, которые и определяют вид кривой второго порядка для данного кластера и данной пары координат.

Заметим, что запись задачи безусловной оптимизации (3) не зависит от того, рассматриваем ли мы случай плоскости или случай большей размерности. Поэтому в целом с помощью данной модели можно находить кривые большей размерности, описывающие форму кластера.

Модель 2. Поиск матриц квадратичной формы для кластеров при дополнительном ограничении. Будем интерпретировать $d^2_B(x_j,\ \nu_i)$ как расстояние. Тогда переписываем задачу следующим образом. Найти такую матрицу B, что

$$\begin{cases} F(S_i,\ B_i) = \sum\limits_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0{,}5\}} \mu_{ij}\, d_{B_i}(x_j,\ \nu_i) \to \min; \\ (d^2_{B_i}(x_j,\ \nu_i) \geqslant 0)\forall x_j : \{j:j=\overline{1,n}\ \wedge\ \mu_{ij}>0{,}5\}. \end{cases}$$

Определенная таким образом функция $F(S_i, B_i)$ не определена в точках, где $d^2_{B_i}(x_j, \nu_i) < 0$. Следовательно, данная функция не является непрерывной и тем более она не дифференцируема. Поэтому, чтобы "улучшить" эту функцию, заменим целевую функцию дифференцируемой функцией

$$F'(S_i,\ B_i) = \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0{,}5\}} \mu_{ij}\max^2\{0,\ d^2_{B_i}(x_j,\ \nu_i)\}.$$

Тогда получаем следующую задачу условной оптимизации:

$$\begin{cases} F'(S_i,\ B_i) = \sum\limits_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0{,}5\}} \mu_{ij}\max^2\{0,\ d^2_{B_i}(x_j,\ \nu_i)\} \to \min; \\ (d^2_{B_i}(x_j,\ \nu_i) \geqslant 0)\forall x_j : \{j:j=\overline{1,n}\ \wedge\ \mu_{ij}>0{,}5\}. \end{cases} \quad (4)$$

Данную задачу будем решать методом штрафных функций [7]. В соответствии с этим методом заменим задачу минимизацией следующей функции:

$$FF(S_i,\ B_i) = F'(S_i,\ B_i) + A_n\Psi(S_i,\ B_i),$$

где $\Psi(S_i,\ B_i) =$

$$= \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0{,}5\}} \max^2\{0, -d^2_{B_i}(x_j,\ \nu_i)\},\ A_n \underset{n\to\infty}{\to} \infty.$$

Раскрывая max, получаем следующий вид функции $FF(S_i, B_i)$:

$$FF(S_i, B_i) = \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0,5\wedge d^2\geq 0\}} \mu_{ij} d^4_{B_i}(x_j, \nu_i) + A_n \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0,5\wedge d^2\geq 0\}} d^4_{B_i}(x_j, \nu_i).$$

Переменными в данной задаче являются элементы матрицы $B_i = (b_{pq})$. Заметим, что для квадратичной формы характерна симметричность, т. е. переменными являются не все элементы матрицы B, а диагональные и элементы либо только верхнего треугольника, либо только нижнего треугольника.

Минимизацию функции $FF(S_i, B_i)$ будем проводить методом градиентного спуска. Для этого необходимо вычислить производные:

$$\frac{\partial FF_i(S_i, B_i)}{\partial b_{pq}} = 2\Bigg[\sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0,5\wedge d^2\geq 0\}} \mu_{ij} \times \times d^2_{B_i}(x_j, \nu_i)\frac{\partial}{\partial b_{pq}} d^2_{B_i}(x_j, \nu_i) + A_n \sum_{\{j:j=\overline{1,n}\wedge\mu_{ij}>0,5\wedge d^2\geq 0\}} d^2_{B_i}(x_j, \nu_i)\frac{\partial}{\partial b_{pq}} d^2_{B_i}(x_j, \nu_i)\Bigg],$$

где

$$\frac{\partial}{\partial b_{pq}} d^2_{B_i}(x_j, \nu_i) = \frac{\partial}{\partial b_{pq}} d^2_{B_i}(x_j - \nu_j)B_i(x_j - \nu_j)^{\mathrm{T}} =$$

$$= \frac{\partial}{\partial b_{pq}} \sum_{p_1=1}^{k}\sum_{q_1=1}^{k} (x^j_{p_1} - \nu^i_{p_1}) b_{p_1q_1}(x^j_{q_1} - \nu^i_{q_1}) =$$

$$= (x^j_p - \nu^i_p)(x^j_q - \nu^i_q).$$

Подставляя значения частных производных в формулы поиска решения по методу градиентного спуска, получаем оптимальные значения переменных Θ^*, λ_1^*, λ_2^*, которые и определяют вид кривой второго порядка для данного кластера и данной пары координат.

Данная модель также может быть с успехом применена для нахождения кривых большего порядка, описывающих форму кластера.

Алгоритм определения оптимальной геометрической формы кластеров

Рассмотрим алгоритм нахождения кривых, описывающих форму кластера. В качестве начальных данных выступает разбиение множества данных на кластеры. Задача состоит в подборе для каждого кластера и каждой пары координат кривой второго порядка, описывающей форму кластера.

Алгоритм решения задачи.

1. Установка параметров. Рассматриваем кластер S_i и пару признаков l_1, l_2.

2. Находим Θ, λ_1, λ_2, обеспечивающие минимум целевой функции (3), или решаем задачу (4).

3. Анализируем собственные значения, определяя тем самым геометрическую форму кластера.

4. Для изображения и дальнейшего распознавания новых объектов для каждого кластера также необходимо определить значение параметра r, такого что $(d^2_{B_i}(x^j, c^i) \leqslant r^2)\forall x_j : \{j:j=\overline{1,n} \wedge \mu_{ij} > 0,5\}$. Для эллипсов в качестве r в методе Густавсона—Кесселя [2] используют объем кластера $\rho_i = \sqrt{\det(B_i)}$. Для остальных кривых мы подбираем значение r так, чтобы элементы, принадлежащие данному кластеру изначально, принадлежали ему и соответственно найденным геометрическим формам кластеров.

Данную процедуру необходимо выполнить для всех кластеров и каждой пары координат.

Анализ полученных результатов

Для тестирования предложенного метода мы использовали данные о колебании курса евро—доллар за сто дней. Пусть *Open* — курс на момент открытия рынка, *Close* — на момент закрытия, *High* — самый высокий курс за день, *Low* — самый низкий курс за день. Охарактеризуем данные тремя признаками:

1. *Open—Close* — длина тела свечи.
2. *High—Low* — колебание курса за день.
3. $Ratio = \dfrac{High - Low}{Open - Close}$.

Проиллюстрируем необходимость разработки методов кластеризации и распознавания новых объектов, учитывающих естественную форму кластеров. Данные, используемые в примере, приведенном на рис. 1, содержат кластер, имеющий форму эллипса (в дальнейшем будем называть его кластер 1), и кластер с формой, близкой к прямой (в дальнейшем будем называть его кластер 2). Поэтому в данном случае целесообразно применять методы, подбирающие для каждого кластера геометрическую форму, наиболее близкую к его естественной форме.

Рассмотрим результаты работы первого метода. В процессе экспериментов мы обнаружили,

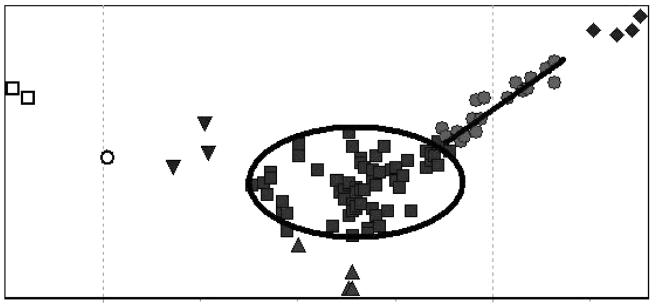


Рис 1. Визуализация данных методом многомерного шкалирования. Один из кластеров имеет форму эллипса, другой — форму, близкую к прямой

что метод является неустойчивым по отношению к начальным значениям переменных λ_1, λ_2. Данный недостаток является существенным и в дальнейшем следует модифицировать алгоритм для того, чтобы повысить его устойчивость. Отметим, однако, что многие алгоритмы кластеризации не являются устойчивыми, например, алгоритм кластеризации С-средних; и в настоящее время предложены модификации данного алгоритма [6], более устойчивые, чем исходный алгоритм.

Приведем результаты работы алгоритма. На рис. 2 (см. третью сторону обложки) изображены результаты работы алгоритма для признаков 1, 2 при различных начальных значениях параметров. На рис. 3 (см. третью сторону обложки) изображены результаты для признаков 1, 3, на рис. 4 (см. четвертую сторону обложки) — для признаков 2, 3.

Можно заметить, что для малонаселенных кластеров полученные кривые менее точно описывают геометрическую форму, чем для густонаселенных кластеров. В данном примере густонаселенными кластерами являются кластеры 1 и 2, предположение о геометрической форме которых было проиллюстрировано с помощью рис. 1.

Кривые, построенные алгоритмом для кластера 2 (в предположении, что форма — прямая), являются более устойчивыми, чем кривые для кластера 1. Они представляют собой "узкие" гиперболы, "узкие" эллипсы или прямые для пар признаков 1, 3 и 2, 3. Для признаков 1, 2 результаты представляют собой "менее узкий" эллипс и гиперболу. Кривые, полученные для кластера 1 (в предположении, что форма — эллипс), меняется от "не узкого" эллипса до "не узкой" гиперболы.

Таким образом, четкого ответа о форме кластеров получить с помощью предложенного алгоритма не удается, по всей видимости, причиной этого является неустойчивость алгоритма по отношению к начальным параметрам. Все же результаты отчетливо говорят о том, что кластер 2 стремится к прямой или "узкой" гиперболе, в то время как кластер 1 более стремится к "не узкому" эллипсу или гиперболе.

Рассмотрим результаты работы второго алгоритма. На рис. 5 (см. четвертую сторону обложки) изображены кривые, которые удалось подобрать с помощью данного алгоритма. Второй алгоритм оказался более устойчивым к начальным данным, чем первый, и поэтому мы приводим здесь результаты работы для одного значения начальных параметров для каждой пары признаков.

"Узкие" эллипсы и гиперболы можно аппроксимировать прямой. Мы ввели параметр, который отвечает за данную аппроксимацию, а именно, если отношение собственных значений меньше данного параметра (он в данном примере равен 0,0001), то эллипс или гиперболу заменяем прямой. На рис. 6 (см. четвертую сторону обложки) можно увидеть результаты работы алгоритма для признаков 1, 2 при значении данного параметра, равном 0,1. В данном случае кластер 1 предлагается описывать эллипсом, а кластер 2 — прямой, что в точности совпадает со сделанным нами предположением.

Заключение

Большинство методов построения и описания кластеризации строго завязаны на форму кластера. Формы кластеров реальных данных в большинстве случаев не одинаковы, часто они настолько причудливы, что их сложно даже отнести к какому-либо классу геометрических фигур. Поэтому разработка методов анализа данных, которые позволяют учитывать естественные формы кластеров, является важнейшей задачей.

Предложенный в данной работе метод подбирает оптимальную геометрическую форму для кластеров, что позволяет в конечном итоге увеличить точность распознавания новых объектов. В дальнейшем планируется доработка метода в сторону увеличения точности и устойчивости к начальным данным.

Список литературы

1. **Kim D.-W.** Detecting clusters of different geometrical shapes in microarray gene expression data / Dae-Won Kim, Kwang H. Leel, Doheon Lee // BIOINFORMATICS. 2005. N 21. 09. P. 1927—1934.

2. **Gustafson E.** Fuzzy clustering with a fuzzy covariance matrix / Gustafson, E. and Kessel, W. // Proc. IEEE Conf. Decision Control. 1979. P. 761—766.

3. **Кострикин А. И., Манин Ю. И.** Линейная алгебра и геометрия. М.: Наука, 1986.

4. **Babuska R.** Identification of composite linear modls via fuzzy clustering / R. Babuska, H. B. Verbruggen // Proceedings European Control Conference, Rome, Italy. 1995. P. 1207—1212.

5. **Мальцев А. И.** Основы линейной алгебры. М.: Наука, 1975.

6. **Kaymak U.** Extended fuzzy clustering algorithms / Uzay Kaymak, Magne Setnes // Erim Report Series Research in Management, November 2000, 24 c.

7. **Азарнова Т. В., Каширина Г. Д., Чернышова Г. Д.** Методы оптимизации. Элементы теории, алгоритмы и примеры. Воронеж: Изд-во ВГУ, 2000.

8. **Babuska R.** An Overview of fuzzy modeling for control / R. Babuska, H. B. Verbruggen // Control Engineering Practice. 1996. 4 (11). P. 1593—1600.

УДК 519.854.2, 519.8:658

Ю. А. Мезенцев, канд. экон. наук, доц.,
Новосибирский государственный
технический университет

Оптимизация расписаний параллельных динамических систем в календарном планировании

Предложен ряд моделей, на основе которых решены задачи синтеза оптимальных расписаний параллельных обслуживающих систем. Представленные динамические модели синтеза расписаний ориентированы на практическое применение в календарном планировании и оперативном регулировании производственных процессов с дискретным характером. Все модели принадлежат классу линейных задач оптимизации с булевыми переменными либо редуцируются в него. Оценена трудоемкость синтеза на их основе точных и приближенных к оптимальным по быстродействию расписаниям. Приведены сравнительные численные примеры использования представленных моделей для синтеза оптимальных расписаний.

Введение

Наиболее общей и актуальной задачей теории расписаний применительно к производственным системам является задача синтеза расписаний работы параллельно-последовательных обслуживающих систем (ОС). Графическое изображение примера такой системы представлено на рисунке.

Типичная параллельно-последовательная ОС состоит из совокупности блоков, или подсистем, содержащих взаимозаменяемые, в общем случае неидентичные, приборы. На производстве это группы взаимозаменяемого оборудования. На рисунке они пронумерованы от 1 до n.

Будем считать заданными технологические последовательности обслуживания всех заявок (требований, партий деталей) с точностью до группы оборудования (блока, подсистемы). Технологические маршруты фиксированы, но различны для разных заявок.

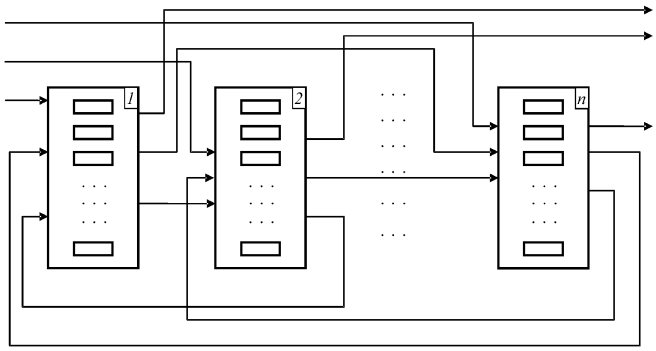
Будем считать известным время обслуживания каждой заявки каждым прибором из каждого блока.

Задача теории расписаний в данном случае заключается в том, чтобы определить расписание работы ОС (всех блоков и приборов), оптимальное по какому-либо критерию.

К настоящему времени общая задача синтеза оптимальных расписаний параллельно-последовательных ОС весьма далека от приемлемого решения при сколько-нибудь реальной размерности. Поэтому до сих пор едва ли не единственным средством автоматизации календарного планирования на практике являются имитационные модели (включая различные эвристические схемы, основанные на регулировании системы посредством варьирования приоритетов, дисциплин обслуживания заявок и очередей).

Один из возможных подходов к решению задачи синтеза оптимальных расписаний параллельно-последовательных систем заключается в декомпозиции общей задачи и выделении локальных подзадач (по числу блоков параллельных приборов) и координирующей задачи. Каждая локальная подзадача на некотором этапе определяет расписание параллельной ОС соответствующего блока.

Можно показать, что если в каждый момент времени известны расписания всех блоков параллельных ОС, то последовательными приближениями (решая по этапам координирующую задачу) можно определить оптимальное расписание всей параллельно-последовательной системы. Поэтому синтез оптимальных расписаний параллельных ОС по сути является ключевой подзадачей общей задачи теории расписаний для параллельно-последовательных систем. Соответственно поиск возможных инструментов оптимизации расписаний параллельных ОС составляет предмет настоящей статьи.



Параллельно-последовательная обслуживающая система

Существует ряд подходов к синтезу расписаний массово-параллельных систем [1—4]. Однако в данном случае ни один из них нельзя считать приемлемым, поскольку существует такое "незначительное" усложнение, как динамический вход параллельной ОС, что означает наличие некоторого расписания поступления заявок в систему. Ниже показано, что формализация данного условия приводит к одной из самых сложных задач дискретного программирования и предлагаются некоторые пути ее решения.

Формальные постановки задач синтеза расписаний параллельных статических систем

Простейшим примером модели неоднородной одностадийной параллельной ОС является классическая задача о назначениях.

Пусть имеется ряд требований (заявок), которые необходимо распределить единовременно между параллельными приборами при известной (различной) производительности таким образом, чтобы минимизировать суммарное время обслуживания всех заявок (минимизировать суммарные затраты либо максимизировать некоторый показатель качества обслуживания заявок). Формальная постановка при этом выглядит следующим образом:

$$Z_1 = \sum_{j \in J} \sum_{i \in I} t_{ij} x_{ij} \to \min; \quad (1)$$

$$\sum_{i \in I} x_{ij} = 1, \ \forall j \to J; \quad (2)$$

$$\sum_{j \in J} x_{ij} = 1, \ \forall i \to I; \quad (3)$$

$$x_{ij} = \begin{cases} 1, \text{ если заявка } j \text{ закрепляется за прибором } i; \\ 0 \text{ в противном случае,} \end{cases} \quad (4)$$

где t_{ij} — время обслуживания j-й заявки i-м прибором.

Вместо критерия (1) часто применяется

$$Z_2 = \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij} \to \max, \quad (5)$$

где c_{ij} — оценка качества обслуживания j-й заявки i-м прибором.

Совместное использование (1) и (5) приводит к бикритериальной задаче.

Как видно из контекста, число заявок и приборов в классическом варианте задачи о назначениях совпадают. Алгоритмы, используемые для решения (1)—(4), имеют полиномиальную сходимость[1].

Значительно чаще встречается ситуация, при которой число заявок больше числа параллельных приборов [1]. В этом случае естественно полагать назначение, по крайней мере на часть приборов, более одной заявки. Модель (1)—(5) модифицируется следующим образом:

$$Z_2 = \sum_{j \in J} \sum_{i \in I} t_{ij} x_{ij} \to \min; \quad (6)$$

$$\sum_{i \in I} x_{ij} = 1, \ \forall j \to J; \quad (7)$$

$$\underline{b_i} \leqslant \sum_{j \in J} x_{ij} \leqslant \overline{b_i}, \ \forall i \to I; \quad (8)$$

$$x_{ij} = \begin{cases} 1, \text{ если заявка } j \text{ закрепляется за прибором } i; \\ 0 \text{ в противном случае;} \end{cases} \quad (9)$$

$\underline{b_i}$ и $\overline{b_i}$ — минимальное и максимальное число заявок, назначаемых на i-й прибор;

$$Z_1 = \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij} \to \max. \quad (10)$$

Помимо критериев (6), (10) часто используется критерий максимального быстродействия вида

$$Z_3 = \max_i \left\{ \sum_{j \in J} t_{ij} x_{ij} \right\} \to \min. \quad (11)$$

Единый подход к решению минимаксных задач предложен Ю. Гермейером [5]. Его применение к формулам (7)—(11) приводит к следующей эквивалентной модели:

$$\sum_{i \in I} x_{ij} = 1; \ \forall j = \overline{1, J}; \quad (12)$$

$$\underline{b_i} \leqslant \sum_{j=1}^{J} x_{ij} \leqslant \overline{b_i}, \ \forall i = \overline{1, I}; \quad (13)$$

$$x_{ij} = \begin{cases} 1, \text{ если заявка } j \text{ закрепляется за прибором } i; \\ 0 \text{ в противном случае;} \end{cases} \quad (14)$$

$$\sum_{j=1}^{J} t_{ij} x_{ij} \leqslant \lambda, \ \forall i = \overline{1, I}; \quad (15)$$

$$Z_3 = \lambda \to \min. \quad (16)$$

Здесь λ имеет смысл минимального времени завершения последней операции. Если $J > \sum_{i=1}^{I} \overline{b_i}$, т. е. число заявок превышает суммарные возможности приборов, вводится дополнительный фиктивный прибор с характеристикой $\overline{b_{I+1}} = J - \sum_{i=1}^{I} \overline{b_i}$.

В отличие от (1)—(5) и (6)—(10), задача (12)—(16) является NP-трудной, поскольку ограничения (15) превращают задачу о назначениях в обобщенную задачу о ранце.

[1] Следствие свойства абсолютной унимодулярности расширенной матрицы системы ограничений.

Оценка трудоемкости алгоритмов оптимизации расписаний параллельных статических систем

При использовании современных алгоритмов решения задач линейного программирования, каковыми являются (1)—(5) и (6)—(10), не возникает затруднений с ростом числа переменных до 10^5 (в однопроцессорном варианте) [6], что превосходит фактические потребности. Универсальные комбинаторные методы, предназначенные для решения задач класса (12)—(16), теоретически позволяют находить оптимальные решения при числе переменных порядка 10^3 [7]. Фактически же, для гарантированного за приемлемое время результата, число столбцов в модели не должно превосходить 1000. Для календарного производственного планирования (параллельных ОС) в большинстве случаев этих параметров достаточно. При возникновении потребности в решении задачи, подобной (12)—(16), большей размерности целесообразно использовать декомпозиционные схемы [8, 9].

Оптимизация расписаний параллельных динамических систем

Естественным обобщением статической модели (12)—(16) является учет динамики поступления в ОС заявок для обслуживания.

Пусть известно расписание поступления заявок в параллельную обслуживающую систему. В этом случае необходимо учитывать задержки поступления заявок. Обозначим задержку поступления j-й заявки в ОС через τ_j^0 и упорядочим заявки по возрастанию τ_j^0.

Тогда динамическая модель оптимизации расписаний параллельной ОС будет иметь следующий вид:

$$\sum_{i \in I} x_{ij} = 1, \ \forall j = \overline{1, J}; \tag{17}$$

$$\underline{b_i} \leqslant \sum_{j=1}^{J} x_{ij} \leqslant \overline{b_i}, \ \forall i = \overline{1, I}; \tag{18}$$

$$x_{ij} = \begin{cases} 1, \text{ если заявка } j \text{ назначается на прибор } i; \\ 0 \text{ в противном случае;} \end{cases} \tag{19}$$

$$y_{ij} = \begin{cases} 1, \text{ если имеет место задержка заявки } j \text{ перед прибором } i; \\ 0 \text{ в противном случае;} \end{cases} \tag{20}$$

$$x_{ij} \geqslant y_{ij}, \ \forall i = \overline{1, I}, \ \forall j = \overline{1, J}; \tag{21}$$

$$\tau_{ij} = \tau_j^0 - \sum_{l=1}^{j-1} (\tau_{il} y_{il} + t_{il}) x_{il}, \quad \forall i = \overline{1, I}, \ \forall j = \overline{1, J}; \tag{22}$$

$$\tau_{ij} y_{ij} \geqslant 0, \ \forall j = \overline{1, J}; \tag{23}$$

$$\sum_{j=1}^{J} \tau_{ij} y_{ij} x_{ij} + \sum_{j=1}^{J} t_{ij} x_{ij} \leqslant \lambda, \ \forall i = \overline{1, I}; \tag{24}$$

$$Z_3 = \lambda \to \min, \tag{25}$$

где τ_{ij} имеет смысл фактической задержки начала выполнения j-й заявки i-м прибором после завершения обслуживания им предшествующей заявки. Переменные y_{ij} вводятся для того, чтобы избежать появления отрицательных задержек τ_{ij}.

Критерий (6) видоизменится следующим образом:

$$\sum_{j \in J} \sum_{i \in I} (\tau_{ij} + t_{ij}) x_{ij} \to \min.$$

Анализ модели удобно проводить по этапам.

На *первом этапе* рассмотрим подробнее (22)—(23)

$$\tau_{ij} y_{ij} = \left(\tau_j^0 - \sum_{l=1}^{j-1} \tau_{il} x_{il} y_{il} - \sum_{l=1}^{j-1} t_{il} x_{il} \right) y_{ij} \geqslant 0, \quad \forall i = \overline{1, I}, \ \forall j = \overline{1, J}. \tag{26}$$

Временно ослабим часть логических ограничений, для чего снимем ограничения (21), (23) и заменим (20) на $y_{ij} = 1$. Тогда ослабленная задача запишется следующим образом:

$$\sum_{i=1}^{I} x_{ij} = 1, \ \forall j = \overline{1, J}; \tag{27}$$

$$\underline{b_i} \leqslant \sum_{j=1}^{J} x_{ij} \leqslant \overline{b_i}, \ \forall i = \overline{1, I}; \tag{28}$$

$$x_{ij} = \begin{cases} 1, \text{ если заявка } j \text{ закрепляется за прибором } i; \\ 0 \text{ в противном случае;} \end{cases} \tag{29}$$

$$\tau_{ij} = \tau_j^0 - \sum_{l=1}^{j-1} (\tau_{il} + t_{il}) x_{il}, \ \forall i = \overline{1, I}, \ \forall j = \overline{1, J}; \tag{30}$$

$$\sum_{j=1}^{J} \tau_{ij} x_{ij} + \sum_{j=1}^{J} t_{ij} x_{ij} \leqslant \lambda, \ \forall i = \overline{1, I}; \tag{31}$$

$$Z_3 = \lambda \to \min. \tag{32}$$

На *втором этапе* подробно рассмотрим ключевые условия для учета фактических задержек исполнения заявок приборами (30)[2]. Из (30) следует:

$$\begin{gathered} \tau_{11} = \tau_1^0; \\ \tau_{12} = \tau_2^0 - \tau_{11} x_{11} - t_{11} = \\ = \tau_2^0 - \tau_1^0 x_{11} - t_{11} x_{11} = \tau_2^0 - (\tau_1^0 + t_{11}) x_{11}; \\ \tau_{13} = \tau_3^0 - [(\tau_{11} x_{11} + \tau_{12} x_{12}) + (t_{11} x_{11} + t_{12} x_{12})]. \end{gathered} \tag{33}$$

[2] Выражения (30), как и (22), определяют рекурсивные функции относительно переменных исходной задачи.

Преобразования, аналогичные (33), приводят к следующему результату:

$$\tau_{13} = \tau_3^0 - [(\tau_1^0 + t_{11})(x_{11} - x_{11}x_{12}) + (\tau_2^0 + t_{12})x_{12})]. \quad (34)$$

Аналогично (34) получаем

$$\tau_{14} = \tau_4^0 - [(\tau_{11} + t_{11})x_{11} + (\tau_{12} + t_{12})x_{12} + (\tau_{13} + t_{13})x_{13}]$$

или

$$\tau_{14} = \tau_4^0 - [(\tau_1^0 + t_{11})(x_{11} - x_{11}x_{12} - x_{11}x_{13} + x_{11}x_{12}x_{13}) + (\tau_2^0 + t_{12})(x_{12} - x_{12}x_{13}) + (\tau_3^0 + t_{13})x_{13}]. \quad (35)$$

Воспользуемся соотношением $f_1(x_{11}, x_{12}) = x_{11}(1 - x_{12}) = x_{11}\overline{x_{12}}$. Аналогично рассмотрим выражение $x_{11} - x_{11}x_{12} - x_{11}x_{13} + x_{11}x_{12}x_{13}$ в (35) как булеву функцию: $f_2(x_{11}, x_{12}, x_{13}) = x_{11}(1 - x_{12} - x_{12}x_{13}) = x_{11}(\overline{x_{12}} - x_{13}(-x_{12} + 1)) = x_{11}(\overline{x_{12}} - x_{13}(-x_{12} + 1)) = x_{11}\overline{x_{12}}\,\overline{x_{13}}$.

Таблицы истинности $f_1(x_{11}, x_{12})$ и $f_2(x_{11}, x_{12}, x_{13})$ отражены соответственно в табл. 1 и 2.

Табл. 2 построена в соответствии с выражением $f_2(x_{11}, x_{12}, x_{13}) = x_{11} - x_{11}x_{12} - x_{11}x_{13} + x_{11}x_{12}x_{13}$.

Легко убедиться в истинности $f_2(x_{11}, x_{12}, x_{13}) = x_{11}\overline{x_{12}}\,\overline{x_{13}}$, из чего следует:

$$\tau_{14} = \tau_4^0 - [(\tau_1^0 + t_{11})(x_{11}\overline{x_{12}}\,\overline{x_{13}}) + (\tau_2^0 + t_{12})(x_{12}\overline{x_{13}}) + (\tau_3^0 + t_{13})x_{13}]; \quad (36)$$

...

Индукцией по размерности получаем:

$$\tau_{ij} = \tau_j^0 - [(\tau_1^0 + t_{i1})(x_{i1}\bigcap_{l=2}^{j-1}\overline{x_{il}}) + (\tau_2^0 + t_{i2}) \times (x_{i2}\bigcap_{l=3}^{j-1}\overline{x_{il}}) + ... + (\tau_{(j-1)}^0 + t_{ij-1})x_{ij-1}],$$

$$\forall i = \overline{1, I}, \forall j = \overline{1, J},$$

или

$$\tau_{ij} = \tau_j^0 - \sum_{k=1}^{j-1}(\tau_k^0 + t_{ik})x_{ik}\bigcap_{l=k+1}^{j-1}\overline{x_{il}}, \quad \forall i = \overline{1, I}, \forall j = \overline{1, J}. \quad (37)$$

Таблица 1

x_{11}	x_{12}	f_1
0	0	0
0	1	0
1	0	1
1	1	0

Таблица 2

x_{11}	x_{12}	x_{13}	f_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Таким образом, получена задача, сложность которой в представлении (17)—(25) не адекватна возможностям современных алгоритмов оптимизации.

Вместе с тем можно предложить различные способы ее редукции.

Суть первого, заслуживающего внимание, подхода состоит в приведении булевых функций типа $x_{ik}\bigcap_{l=k+1}^{j-1}\overline{x_{il}}$ к линейному виду посредством использования вспомогательных переменных, а также системы дополнительных неравенств. Близкая по смыслу процедура представлена, например, в [10].

Обозначим $f_{ijk}(x_{ik}, x_{ik+1}, ..., x_{ij-1}) = x_{ik}\bigcap_{l=k+1}^{j-1}\overline{x_{il}}$, $\forall i = \overline{1, I}$, $\forall j = \overline{1, J}$, и введем в рассмотрение переменные u_{ijk}, значения которых истинны только тогда, когда истинны значения f_{ijk}. Примеры таких пар: u_{111} и $f_{111}(x_{11}, x_{12}) = x_{11}\overline{x_{12}}$, u_{211} и $f_{211}(x_{12}, x_{13}) = x_{12}\overline{x_{13}}$, u_{212} и $f_{212}(x_{11}, x_{12}, x_{13}) = x_{11}\overline{x_{12}}\,\overline{x_{13}}$. Соответствующие условия истинности можно задать следующей системой ограничений:

$$0 \leqslant x_{11} + \overline{x_{12}} - 2u_{111} \leqslant 1;$$

$$0 \leqslant x_{12} + \overline{x_{13}} - 2u_{211} \leqslant 1;$$

$$0 \leqslant x_{11} + \overline{x_{12}} + \overline{x_{13}} - 3u_{212} \leqslant 2;$$

...

$$0 \leqslant x_{ik} + \sum_{l=k+1}^{j-1} \overline{x_{il}} - Ku_{ijk} \leqslant K - 1,$$

где $K = j - 1 - k$.

Возвращаясь к исходным обозначениям ($\overline{x_{ij}} = 1 - x_{ij}$), получим:

$$-1 \leqslant x_{11} - x_{12} - 2u_{111} \leqslant 0;$$

$$-1 \leqslant x_{12} - x_{13} - 2u_{211} \leqslant 0;$$

$$-2 \leqslant x_{11} - x_{12} - x_{13} - 3u_{212} \leqslant 0;$$

...

$$-K + 1 \leqslant x_{ik} - \sum_{l=k+1}^{j-1} \overline{x_{il}} - Ku_{ijk} \leqslant 0. \qquad (38)$$

Условия (37) при этом запишутся следующим образом:

$$\tau_{ij} = \tau_j^0 - \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik})u_{ijk},$$

$$\forall i = \overline{1, I}, \ \forall j = \overline{1, J}. \qquad (39)$$

Возвращаясь к задаче (27)—(32) и применяя данный подход, получим:

$$\sum_{j=1}^{J} \tau_{ij}x_{ij} = \sum_{j=1}^{J} \left[\tau_j^0 - \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik})x_{ik} \bigcap_{l=k+1}^{j-1} \overline{x_{il}} \right] x_{ij} =$$

$$= \sum_{j=1}^{J} \tau_j^0 x_{ij} - \sum_{j=1}^{J} x_{ij} \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik})x_{ik} \bigcap_{l=k+1}^{j-1} \overline{x_{il}} =$$

$$= \sum_{j=1}^{J} \tau_j^0 x_{ij} - \sum_{j=1}^{J} \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik})x_{ij}x_{ik} \bigcap_{l=k+1}^{j-1} \overline{x_{il}}.$$

Расширим и дополним введенные выше булевы функции f_{ijk} и соответствующие им переменные u_{ijk}:

$$f_{ijk}(x_{ik}, x_{ik+1}, ..., x_{ij}) = x_{ij}x_{ik} \bigcap_{l=k+1}^{j-1} \overline{x_{il}},$$

$$\forall i = \overline{1, I}, \ \forall j = \overline{1, J}. \qquad (40)$$

Тогда (38) преобразуется в следующую формулу:

$$-K + 2 \leqslant x_{ij} + x_{ik} - \bigcap_{l=k+1}^{j-1} x_{il} - Ku_{ijk} \leqslant 1,$$

$$\forall i = \overline{1, I}, \ \forall j = \overline{1, J}, \ \forall k = \overline{1, J}, \qquad (41)$$

где $K = j - k$. И неравенства (31) примут вид

$$\sum_{j=1}^{J} \tau_j^0 x_{ij} - \sum_{j=1}^{J} \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik})u_{ijk} + \sum_{j=1}^{J} t_{ij}x_{ij} \leqslant \lambda. \qquad (42)$$

Окончательно динамическая модель синтеза оптимального по быстродействию расписания (27)—(32) с учетом использованных преобразований редуцируется в линейную модель с булевыми переменными:

$$\sum_{i=1}^{I} x_{ij} = 1, \ \forall j = \overline{1, J}; \qquad (43)$$

$$\underline{b_i} \leqslant \sum_{j=1}^{J} x_{ij} \leqslant \overline{b_i}, \ \forall i = \overline{1, I}; \qquad (44)$$

$$x_{ij} = \begin{cases} 1, \text{ если заявка } j \text{ закрепляется за прибором } i; \\ 0 \text{ в противном случае;} \end{cases} \qquad (45)$$

$$u_{ijk} = \begin{cases} 1, \text{ при истинности выражения (40);} \\ 0 \text{ в противном случае;} \end{cases} \qquad (46)$$

$$-K + 2 \leqslant x_{ij} + x_{ik} - \sum_{l=k+1}^{j-1} x_{il} - Ku_{ijk} \leqslant 1,$$

$$\forall i = \overline{1, I}, \ \forall j = \overline{1, J}, \ \forall k = \overline{1, J}, \ K = j - k; \qquad (47)$$

$$\sum_{j=1}^{J} (\tau_j^0 + t_{ij})x_{ij} - \sum_{j=1}^{J} \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik})u_{ijk} \leqslant \lambda,$$

$$\forall i = \overline{1, I}; \qquad (48)$$

$$Z = \lambda \to \min. \qquad (49)$$

Таким образом, задача (27)—(32) и, совершенно аналогично, (17)—(25) сводятся к задачам линейного программирования с булевыми переменными.

Несмотря на формальную разрешимость редуцированной относительно (17)—(25) задачи (43)—(49) предложенный выше подход можно считать лишь начальным этапом общего решения. Очевидно, что число булевых переменных в редуцированной задаче, как и число ограничений, увеличивается в $J!$ раз по сравнению с (17)—(25). Тем самым формальная сложность (17)—(25) редуцируется в вычислительную сложность (43)—(49). Ее непосредственное численное решение для динамических параллельных ОС сколько-нибудь реальной размерности существующими комбинаторными алгоритмами невозможно. Имеет место надежда на построение декомпозиционного алгоритма, подобного предложенному автором для задачи оптимальной комплектации [9], однако на сегодняшний день этот вопрос открыт.

Второй возможный подход заключается в максимальном упрощении условий задачи. Его суть можно пояснить, модифицировав исходную задачу следующим образом:

$$\sum_{i=1}^{I} x_{ij} = 1, \ \forall j = \overline{1, J}; \qquad (50)$$

$$\underline{b_i} \leqslant \sum_{j=1}^{J} x_{ij} \leqslant \overline{b_i}, \ \forall i = \overline{1, I}; \quad (51)$$

$$x_{ij} = \begin{cases} 1, \text{ если заявка } j \text{ назначается на прибор } i; \\ 0 \text{ в противном случае;} \end{cases} \quad (52)$$

$$\sum_{j=1}^{J} \tau_j^0 x_{ij} \leqslant \beta, \ \forall i = \overline{1, I}; \quad (53)$$

$$\sum_{j=1}^{J} t_{ij} x_{ij} \leqslant \lambda, \ \forall i = \overline{1, I}; \quad (54)$$

$$Z_3 = \lambda \to \min, \ Z_4 = \beta \to \min. \quad (55)$$

Из контекста понятно, что второй подход к распределению заявок между приборами предполагает компромиссное решение по "чистому" быстродействию ОС без учета задержек и по равномерности распределения задержек между приборами.

Линеаризовать векторный критерий $Z = \{Z_3, Z_4\}$ можно посредством линейной свертки:

$$Z_5 = \alpha\lambda + (1 - \alpha)\beta, \quad (56)$$

где $0 \leqslant \alpha \leqslant 1$ — параметр модели.

Данную идею можно критиковать, однако модель (50)—(56) обладает неоспоримым преимуществом в размерности по сравнению с рассмотренной выше точной редукцией модели (17)—(25), а следовательно, по крайней мере на текущий момент времени, имеет большую практическую ценность. Численные эксперименты с обеими моделями подтверждают эффективность второго подхода, так как при использовании (50)—(56) наблюдается падение быстродействия синтезируемых расписаний ОС не более чем на 5 % от оптимального уровня, что для рассматриваемых объектов вполне приемлемо.

Числовой пример синтеза расписаний параллельной динамической системы

Проиллюстрируем использование рассмотренных выше моделей для синтеза расписаний параллельных ОС на небольшом примере.

Рассмотрим параллельную обслуживающую систему, состоящую из двух неидентичных приборов, в которую на обслуживание не одновременно поступают семь заявок.

Данные о нормах времени обслуживания заявок и задержках (расписании) поступления заявок в систему приведены в табл. 3.

Сначала сравним оптимальные расписания, синтезируемые посредством применения статических моделей (6)—(9) и (12)—(16) (без учета задержек поступления заявок в систему τ_j^0). Затем сопоставим расписания, синтезируемые с помощью динамических моделей (43)—(49) и (50)—(55).

Таблица 3

Номер заявки j	Время обслуживания t_{ij}		Задержка поступления заявки τ_j^0
	Номер прибора i		
	1	2	
1	2	4	0
2	3	2	0
3	5	4	2
4	2	4	3
5	4	2	4
6	3	3	5
7	4	3	6

Решение задачи (6)—(9) обозначим x^{1*} значение критериальных показателей Z_2 (суммарное время обслуживания заявок) и Z_3 (раннее время завершения обслуживания последней заявки) при x^{1*} соответственно через Z_2^{1*} и Z_3^{1*}. Аналогичным образом решение задачи (12)—(16) обозначим x^{2*}, Z_2 и Z_3 на x^{2*} — соответственно через Z_2^{2*} и Z_3^{2*}:

$$x^{1*} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}; \ Z_2^{1*} = 18; \ Z_3^{1*} = 14; \ x^{2*} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix};$$

$$Z_2^{2*} = 19; \ Z_3^{2*} = 10.$$

Для наглядности сформируем матрицы времени завершения операций T_{po}^{1*}, T_{po}^{2*} для каждой задачи. С учетом задержек поступления заявок τ_j^0 будем иметь:

$$T_{po}^{1*} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 0 & 6 \\ 5 & 0 \\ 0 & 8 \\ 0 & 11 \\ 0 & 14 \end{pmatrix}; \ T_{po}^{2*} = \begin{pmatrix} 2 & 0 \\ 5 & 0 \\ 0 & 6 \\ 7 & 0 \\ 0 & 8 \\ 10 & 0 \\ 0 & 11 \end{pmatrix}.$$

Видно, что значение $Z_3^{1*} = 14$ не изменилось, значение же $Z_3^{2*} = 11$ увеличилось на единицу. В данном случае расписание поступления заявок в ОС серьезного влияния на расписание обслуживания этих заявок не оказывает. Однако более существенные задержки могут иметь решающее влияние на расписание обслуживания заявок. Так, если задержки в рассматриваемом примере увеличить вдвое, то T_{po}^{1*}, T_{po}^{2*} для моделей (6)—(9) и (12)—(16) будут выглядеть следующим образом:

$$T_{po}^{1*} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 0 & 8 \\ 8 & 0 \\ 0 & 10 \\ 0 & 13 \\ 0 & 16 \end{pmatrix}; \; T_{po}^{2*} = \begin{pmatrix} 2 & 0 \\ 5 & 0 \\ 0 & 8 \\ 8 & 0 \\ 0 & 10 \\ 13 & 0 \\ 0 & 15 \end{pmatrix}.$$

При этом значение $Z_3^{2*} = 15$ является минимально возможным, так как превышает значение $\tau_7^0 = 12$ на значение минимального времени обслуживания последней заявки.

Модель (43)—(49) даже для рассматриваемого примера выглядит достаточно громоздко:

$x_{11} + x_{21} = 1;$

$x_{12} + x_{22} = 1;$

$x_{13} + x_{23} = 1;$

...

$x_{17} + x_{27} = 1;$

$1 \leqslant x_{11} + x_{12} + x_{13} + ... + x_{17} \leqslant 7;$

$1 \leqslant x_{21} + x_{22} + x_{23} + ... + x_{27} \leqslant 7;$

$x_{ij} = \begin{cases} 1, \text{ если заявка } j \text{ закрепляется за прибором } i, \\ i = \overline{1,2}, j = \overline{1,7}; \\ 0 \text{ в противном случае;} \end{cases}$

$u_{ijk} = \begin{cases} 1, \text{ при истинности выражения} \\ x_{ij} x_{ik} \bigcap\limits_{l=k+1}^{j-1} \overline{x_{il}}, \; i = \overline{1,2}, j = \overline{1,7}, k = \overline{1,7}; \\ 0 \text{ в противном случае;} \end{cases}$

$-4 \leqslant x_{17} + x_{11} - x_{12} - x_{13} - x_{14} - x_{15} - x_{16} - 6u_{171} \leqslant 1;$

$-3 \leqslant x_{17} + x_{12} - x_{13} - x_{14} - x_{15} - x_{16} - 5u_{172} \leqslant 1;$

$-2 \leqslant x_{17} + x_{13} - x_{14} - x_{15} - x_{16} - 4u_{173} \leqslant 1;$

$-1 \leqslant x_{17} + x_{14} - x_{15} - x_{16} - 3u_{174} \leqslant 1;$

$0 \leqslant x_{17} + x_{15} - x_{16} - 2u_{175} \leqslant 1;$

$0 \leqslant x_{17} + x_{16} - 2u_{176} \leqslant 1;$

$-3 \leqslant x_{16} + x_{11} - x_{12} - x_{13} - x_{14} - x_{15} - 5u_{161} \leqslant 1;$

$-2 \leqslant x_{16} + x_{12} - x_{13} - x_{14} - x_{15} - 4u_{162} \leqslant 1;$

$-1 \leqslant x_{16} + x_{13} - x_{14} - x_{15} - 3u_{163} \leqslant 1;$

$0 \leqslant x_{16} + x_{14} - x_{15} - 2u_{164} \leqslant 1;$

$0 \leqslant x_{16} + x_{15} - 2u_{165} \leqslant 1;$

$-2 \leqslant x_{15} + x_{11} - x_{12} - x_{13} - x_{14} - 4u_{151} \leqslant 1;$

$-1 \leqslant x_{15} + x_{12} - x_{13} - x_{14} - 3u_{152} \leqslant 1;$

$0 \leqslant x_{15} + x_{13} - x_{14} - 2u_{153} \leqslant 1;$

$0 \leqslant x_{15} + x_{14} - 2u_{154} \leqslant 1;$

$-1 \leqslant x_{14} + x_{11} - x_{12} - x_{13} - 3u_{141} \leqslant 1;$

$0 \leqslant x_{14} + x_{12} - x_{13} - 2u_{142} \leqslant 1;$

$0 \leqslant x_{14} + x_{13} - 2u_{143} \leqslant 1;$

$0 \leqslant x_{13} + x_{11} - x_{12} - 2u_{131} \leqslant 1;$

$0 \leqslant x_{13} + x_{12} - 2u_{132} \leqslant 1;$

$0 \leqslant x_{12} + x_{11} - 2u_{121} \leqslant 1;$

$-4 \leqslant x_{27} + x_{21} - x_{22} - x_{23} - x_{24} - x_{25} - x_{26} - 6u_{271} \leqslant 1;$

...

$-3 \leqslant x_{26} + x_{21} - x_{22} - x_{23} - x_{24} - x_{25} - 5u_{261} \leqslant 1;$

...

$-2 \leqslant x_{25} + x_{21} - x_{22} - x_{23} - x_{24} - 4u_{251} \leqslant 1;$

...

$-1 \leqslant x_{24} + x_{21} - x_{22} - x_{23} - 3u_{241} \leqslant 1;$

$0 \leqslant x_{24} + x_{22} - x_{23} - 2u_{242} \leqslant 1;$

...

$0 \leqslant x_{22} + x_{21} - 2u_{221} \leqslant 1;$

$2x_{11} + 3x_{12} + 7x_{13} + 5x_{14} + 8x_{15} + 8x_{16} + 10x_{17} - 2u_{171} - 3u_{172} - 7u_{173} - ... - 8u_{176} - 2u_{161} - 3u_{162} - ... - 8u_{165} - 2u_{151} - 3u_{152} - ... - 5u_{154} - 2u_{141} - 3u_{142} - 7u_{143} - 2u_{131} - 3u_{132} - 2u_{121} \leqslant \lambda;$

$4x_{21} + 2x_{22} + 6x_{23} + 7x_{24} + 6x_{25} + 8x_{26} + 9x_{27} - 4u_{271} - 2u_{272} - 6u_{273} - ... - 8u_{276} - 4u_{261} - 2u_{262} - ... - 6u_{265} - 4u_{251} - 2u_{252} - ... - 7u_{254} - 4u_{241} - 2u_{242} - 6u_{243} - 4u_{231} - 2u_{232} - 4u_{221} \leqslant \lambda;$

$Z_3^3 = \lambda \to \min.$

Результатом решения задачи (43)—(49) с данными, приведенными в табл. 3, является вектор x^{3*}, на котором определены значения критериев Z_2^{3*} и Z_3^{3*}. Сформируем также матрицу времени завершения операций T_{po}^{3*}. С учетом задержек поступления заявок τ_j^0 будем иметь:

$$x^{3*} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix};\ Z_2^{3*} = 18;\ Z_3^{3*} = 11;\ T_{po}^{3*} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 0 & 6 \\ 5 & 0 \\ 0 & 8 \\ 8 & 0 \\ 0 & 11 \end{pmatrix}.$$

Отметим, что данный результат является наилучшим по всем рассматриваемым критериям.

Наконец, применение модели (50)—(56) при разных значениях параметра α приводит к следующим результатам:

$$0 \leqslant \alpha \leqslant 0{,}66$$

$$x^{4*} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix};\ Z_2^{4*} = 21;\ Z_3^{4*} = 11;\ T_{po}^{4*} = \begin{pmatrix} 0 & 4 \\ 0 & 6 \\ 7 & 0 \\ 9 & 0 \\ 0 & 8 \\ 12 & 0 \\ 0 & 11 \end{pmatrix};$$

$$0{,}66 < \alpha \leqslant 1$$

$$x^{4*} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix};\ Z_2^{4*} = 19;\ Z_3^{4*} = 10;\ T_{po}^{4*} = \begin{pmatrix} 2 & 0 \\ 5 & 0 \\ 0 & 6 \\ 7 & 0 \\ 0 & 8 \\ 10 & 0 \\ 0 & 11 \end{pmatrix}.$$

Анализ результатов применения предложенных моделей для синтеза расписаний параллельных ОС подтверждает вывод о существенном преимуществе в размерности модели (50)—(56) перед точной редукцией модели (17)—(25) и о незначительном проигрыше в эффективности синтезируемых расписаний при тех же обстоятельствах.

Заключение

Кратко перечислим полученные результаты.

Разработана модель оптимального календарного планирования динамических параллельных систем, являющаяся задачей булева программирования, которая, в свою очередь, трансформируется в линейную булеву задачу большой размерности специальной структуры. В перспективе возможно ее успешное решение для реальных объектов с использованием методов линеаризации и декомпозиции.

Предложена бикритериальная модель небольшой размерности, позволяющая определять приближенные к оптимальным расписания динамических параллельных ОС.

В конечном итоге обе модели нацелены на решение общей задачи синтеза оптимальных расписаний параллельно-последовательных ОС, каковыми является большинство промышленных объектов.

Учитывая высокую достоверность и точность информации на данном уровне планирования, а также рост эффективности использования ресурсов при оптимальном календарном планировании, можно предсказать высокую востребованность подобного рода разработок промышленностью[3].

Список литературы

1. **Левин В. И.** Структурно-логические методы исследования сложных систем с применением ЭВМ. М.: Наука, 1987. 304 с.
2. **Иванов Л. Н., Корчагин И. Я.** Проблемно-ориентированная концепция построения структур массово-параллельных вычислительных систем // Автометрия, Новосибирск, 2002. Т. 38. № 5. С. 68—75.
3. **Коффман Э. Г.** и др. Теория расписаний и вычислительные машины. М.: Наука, 1984. 336 с.
4. **Корчагин И. Я.** Использование метода аналогий в задачах распределения ресурсов и маршрутизации сообщений в массово-параллельных вычислительных системах // Вестник Хакасского государственного университета. 2003. Вып. 5. Сер. 1. С. 53—58.
5. **Гермейер Ю. Б.** Введение в теорию исследования операций. М.: Наука, 1971. 368 с.
6. **Мезенцев Ю. А.** Практические аспекты эффективности алгоритма следования центральному пути метода внутренних точек // Научный вестник НГТУ. Новосибирск: Изд-во НГТУ. 2006. № 4 (25). С. 67—104.
7. **Мезенцев Ю. А.** Алгоритмы синтеза расписаний многостадийных обслуживающих систем в календарном планировании // Омский научный вестник. Омск: Изд-во ОГТУ. 2006. № 3 (36). С. 97—102.
8. **Сергиенко И. В.** Математические модели и методы решения задач дискретной оптимизации. Киев: Наукова думка, 1985. 382 с.
9. **Мезенцев Ю. А.** Декомпозиционный метод решения одного класса задач оптимального проектирования // Научный вестник НГТУ. Новосибирск: Изд-во НГТУ. 2006. № 3 (24). С. 67—100.
10. **Муртаф Б.** Современное линейное программирование. М.: Мир, 1984. 224 с.

[3] На сегодняшний день ни одна корпоративная информационная система класса ERP из распространяемых в России подобного рода инструментов не содержит.

УДК 004.3

В. М. Картак, канд. физ.-мат. наук,
Уфимский государственный авиационный технический университет

Матричный алгоритм поиска оптимального решения для задачи упаковки прямоугольников в полубесконечную полосу

Рассматривается задача упаковки прямоугольников в полубесконечную полосу (two-dimensional strip packing problem, 2DSPP), которая является NP-трудной. Для ее решения известны немногие точные алгоритмы типа ветвей и границ (method branch and boundary, MMB). Они позволяют находить оптимальное решение для небольшого числа различных прямоугольников. Предлагается новый переборный метод на базе матричного представления прямоугольной упаковки. Алгоритм трансформируется на задачу упаковки n-мерных параллелепипедов.

Введение

Задача упаковки прямоугольников в полубесконечную полосу (также двумерная задача раскроя-упаковки, *two-dimensional strip packing problem*, **2DSPP**) состоит в следующем: требуется разместить заданные прямоугольники в полубесконечной полосе так, чтобы длина занятой части полосы достигла минимума. Согласно Dyckhoff's типологии раскроя и упаковки [1], двумерная задача раскроя-упаковки имеет тип 2/N/O. Задача **2DSPP** возникла в середине XX века и с тех пор не теряет актуальности, поскольку к ее решению сводятся прикладные задачи распределения двумерного ресурса, такие как: задачи планирования занятости и составления расписаний; задачи распределения памяти ЭВМ; собственно проблемы упаковки и раскроя. Вместе с тем, большинство работ в указанной области посвящены приближенным методам решения данной задачи. Это обусловлено тем, что **2DSPP** является NP-трудной задачей. На данный момент в мире существует лишь несколько подходов, позволяющих решать двумерную задачу раскроя-упаковки точно [2].

Большинство точных методов решения **2DSPP** сводится к перебору всего множества допустимых решений и нахождения среди них оптимума. Такой перебор разного вида улучшениями может быть доведен до эффективных алгоритмов [3]. Значительная группа методов улучшенного перебора известна под названием "метода ветвей и границ" (MMB). Martello S. and Vigo D. применили MBB для решения **2DSPP** [4].

Точный метод решения **2DSPP**, ориентированный на небольшое число прямоугольников, был также разработан Липовецким А. И. (УрО АН СССР). В работе [5] он ввел понятие "зоны" таким образом, что решение двумерной задачи раскроя-упаковки стало сводиться к перебору допустимых зон, а также показал, что задача упаковки прямоугольников в полубесконечную полосу имеет сложность $O(m!^2)$, где m — число прямоугольников. Использование ряда правил отсечения бесперспективных ветвей позволило сократить перебор. Метод исследован и реализован Бухваловой В. В. [6].

S. P. Fekete, J. Schepers [7] предложили представлять упаковки в виде набора интервальных графов. Разработанный ими алгоритм осуществляет перебор указанных графов. В работе показано, что одному интервальному графу соответствует некоторое семейство упаковок. Они использовали этот подход для решения рюкзачной задачи.

Другой подход к решению **2DSPP** был предложен Ю. Г. Стояном [8] и М. В. Новожиловой [9], которые предлагают использовать методы поиска глобального минимума функции цели для задачи **2DSPP**. Условия размещения прямоугольников описываются системой из $2m(m-1)$ линейных неравенств относительно координат полюсов прямоугольников и таким же числом целочисленных переменных. Полученная система неравенств может быть решена с использованием методов линейного программирования. Недостатком данного подхода является то обстоятельство, что число неравенств и целочисленных переменных быстро растет с увеличением числа прямоугольников.

В данной статье описан новый подход к нахождению оптимальной упаковки, базирующийся на ее матричном представлении. Первоначально упаковка разбивается на вертикальные и горизонтальные полосы. Такое вертикальное разбиение впервые было предложено Э. А. Мухачевой и позже применено для эвристического поиска А. С. Мухачевой [10]. Далее каждому разбиению сопоставляется бинарная матрица с элементами $a_{ij} = 1$, если i-й прямоугольник пересекается с j-м полосой, в противном случае $a_{ij} = 0$.

Алгоритм сводится к построению представленных бинарных матриц и выбора из них тех, которые отвечают оптимальному результату.

1. Постановка задачи

Имеется полубесконечная полоса S фиксированной ширины W и множество прямоугольников $R = \{R_i,\ i = \overline{1, m}\}$ с размерами (l_i, w_i), где l_i — длина; w_i — ширина i-го прямоугольника; m — число прямоугольников. Обозначим исходные данные

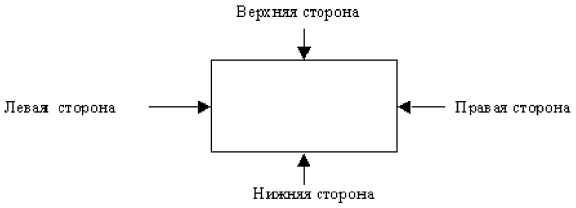


Рис. 1. Наименование сторон прямоугольника

как (W, m, R). Требуется разместить прямоугольники в полосе так, чтобы длина занятой части полосы достигла минимума. Для определенности назовем каждую из сторон прямоугольника в соответствии с рис. 1.

Допустимой упаковкой прямоугольников R в полубесконечную полосу S называется размещение прямоугольников внутри полосы S, удовлетворяющее следующим условиям:

- стороны прямоугольников из R параллельны сторонам полосы S;
- прямоугольники из R между собой не перекрываются (не имеют общих внутренних точек).

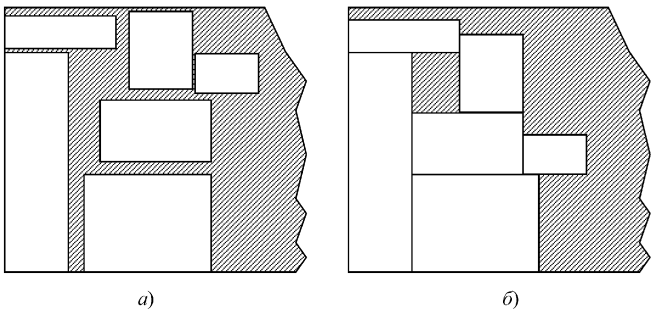
Введем прямоугольную систему координат, в которой оси Ox, Oy совпадают с нижней и левой сторонами полосы, и обозначим допустимую упаковку $P = \{P_1, P_2, ..., P_m\}$, где $P_i = (x_i, y_i)$ — координаты левого нижнего угла i-го прямоугольника. Тогда координаты прямоугольников в допустимой упаковке должны удовлетворять следующим условиям:

$$x_i \geqslant 0,\ y_i \geqslant 0,\ y_i + w_i \leqslant W,\ i = \overline{1, m}; \qquad (1)$$

$$(x_i + l_i \leqslant x_j) \vee (x_j + l_j \leqslant x_i) \vee (y_i + w_i \leqslant y_j) \vee$$
$$\vee (y_j + w_j \leqslant y_i),\ i, j = \overline{1, m},\ i \neq j. \qquad (2)$$

Условие (1) означает, что все прямоугольники располагаются внутри полосы S, а условие (2) — что они не перекрываются между собой, т. е. любые два прямоугольника из R "раздвинуты" по горизонтальному или (и) вертикальному направлению.

Допустимая упаковка P_o, для которой длина занятой части полосы $L = \max\limits_{i = \overline{1, m}} (x_i + l_i)$ достигает минимума, называется *оптимальной*.



а) *б*)

Рис. 2. Пример упаковок: *а* — неплотная упаковка P; *б* — плотная упаковка P

Упаковка P называется *плотной*, если левая (нижняя) сторона каждого прямоугольника касается либо боковой (нижней) границы полосы, либо правой (верхней) стороны другого прямоугольника (рис. 2)

В работе [5] показано, что для нахождения оптимальной упаковки P_o достаточно рассматривать только плотные упаковки P.

2. Матричное представление упаковки

Как уже говорилось во введении, **2DSPP** принадлежит к классу NP-трудных задач. На данный момент известен один путь, позволяющий гарантировать получение оптимального решения данного рода задач — это метод полного перебора допустимых вариантов. Представленный в статье алгоритм принадлежит к классу таких методов. Его отличие от ранее известных методов заключается в том, что каждой упаковке сопоставляются бинарные матрицы специального вида и процесс перебора допустимых вариантов сводится к операциям над этими бинарными матрицами.

2.1. Вертикальное и горизонтальное разбиение упаковки

Пусть для некоторой задачи (W, m, R) известна упаковка P. Выполним мысленно вертикальные "резы", проходящие через правые стороны прямоугольников R. Таким образом, вся упаковка делится на вертикальные полосы. Обозначим k_x — число вертикальных полос.

Пример 1. Для задачи **2DSPP** с исходными данными (W, m, R), где $W = 55$, $m = 6$, $R_1 = (20,28)$, $R_2 = (30,27)$, $R_3 = (63,10)$, $R_4 = (22,45)$, $R_5 = (26,45)$, $R_6 = (15,15)$, известна упаковка $P = \{P_1 = (0,27), P_2 = (0,0), P_3 = (20,45), P_4 = (30,0), P_5 = (52,0), P_6 = (78,0)\}$, изображенная на рис. 3.

В нашем случае упаковка P оказывается разбитой на $k_x = 6$ вертикальных полос (рис. 3, *а*). Аналогично выполним мысленно горизонтальные резы, проходящие через верхние стороны прямоугольников, k_y — число горизонтальных полос. При этом упаковка P оказывается разбитой на $k_y = 4$ горизонтальных полос (рис. 3, *б*).

2.2. Матричное представление упаковки

Вертикальному разбиению упаковки на полосы сопоставим следующую пару:

1) матрицу $\mathbf{A^x}$ размерности $m \times k_x$ по правилу:

$$a_{ij}^x = \begin{cases} 1, \text{ если } i\text{-й прямоугольник пересекается} \\ j\text{-й вертикальной полосой;} \\ 0, \text{ в противном случае;} \end{cases}$$

2) вектор $\boldsymbol{z^x} = (z_1^x,\ z_1^x,\ ...,\ z_{k_x}^x)$, элементами которого являются значения длин вертикальных полос.

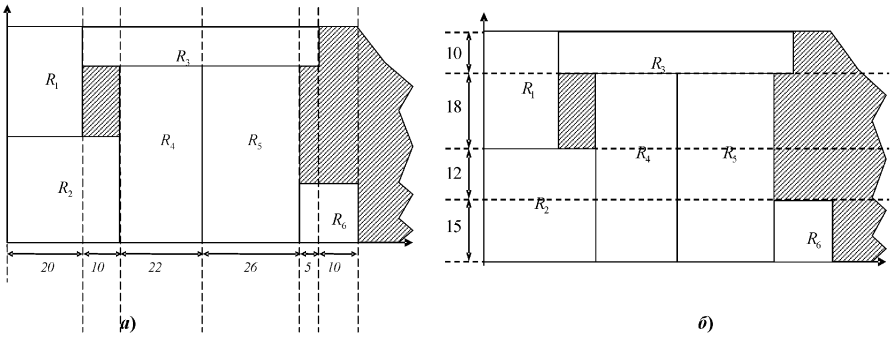


Рис. 3. Блок структуры упаковки: *а* — вертикальная; *б* — горизонтальная

Аналогично горизонтальному разбиению сопоставим матрицу $\mathbf{A^y}$ размерности $m \times k_y$ и вектор $\boldsymbol{z}^{\mathbf{y}} = (z_1^y, z_2^y, ..., z_{k_y}^y)$. Для упаковки P, представленной в примере 1, имеем:

$$\mathbf{A^x} = \begin{pmatrix} 1&0&0&0&0&0 \\ 1&1&0&0&0&0 \\ 0&1&1&1&1&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&1 \end{pmatrix}; \quad \mathbf{A^y} = \begin{pmatrix} 0&0&1&1 \\ 1&1&0&0 \\ 0&0&0&1 \\ 1&1&1&0 \\ 1&0&0&0 \end{pmatrix};$$

$\boldsymbol{z}^{\mathbf{x}} = (20, 10, 22, 26, 5, 10)$; $\boldsymbol{z}^{\mathbf{y}} = (5, 12, 18, 10)$.

Полученные указанным способом матрицы $\mathbf{A^x}$ и $\mathbf{A^y}$ будем называть *матрицами упаковки*, а векторы $\boldsymbol{z}^{\mathbf{x}}$ и $\boldsymbol{z}^{\mathbf{y}}$ — *векторами упаковки*.

2.3. Свойства матриц и векторов упаковки

В силу специфики своего построения матрицы $\mathbf{A^x}$, $\mathbf{A^y}$ и векторы $\boldsymbol{z}^{\boldsymbol{x}}$ и $\boldsymbol{z}^{\mathbf{y}}$ обладают следующими свойствами.

Свойство 1. (Ограниченность). Число столбцов в матрицах $\mathbf{A^x}$ и $\mathbf{A^y}$ не превосходит m число прямоугольников в R:

$$k_x \leqslant m,\ k_y \leqslant m. \tag{3}$$

Действительно, максимальное число вертикальных и горизонтальных полос не может превосходить числа заданных прямоугольников из R, так как они проходят по правым (верхним) границам прямоугольников.

Свойство 2. (Продолженность единиц). Для каждой строки i $(1 \leqslant i \leqslant m)$ матрицы $\mathbf{A^x}$ найдутся такие столбцы j^i_{begin} и j^i_{end}, что

$$a^x_{ij} = \begin{cases} 1, \text{ если } j^i_{begin} \leqslant j \leqslant j^i_{end}; \\ 0, \text{ в противном случае.} \end{cases} \tag{4}$$

Это свойство следует из неразрывности прямоугольников, а именно: каждый прямоугольник может занимать одну или несколько соседних вертикальных (горизонтальных) полос. Данное свойство также верно и для $\mathbf{A^y}$.

Свойство 3. (Положительность). Все координаты векторов упаковки $\boldsymbol{z}^{\mathbf{x}}$ и $\boldsymbol{z}^{\mathbf{y}}$ неотрицательные.

Это свойство является очевидным, так как элементами данных векторов являются длины (ширины) полос.

Свойство 4. Связь между компонентами векторов $\boldsymbol{z}^{\mathbf{x}}$, $\boldsymbol{z}^{\mathbf{y}}$ и элементами матриц $\mathbf{A^x}$, $\mathbf{A^y}$ с размерами прямоугольников R устанавливается следующим соотношением:

$$\sum_{i=1}^{k_x} a^x_{ij} z^x_i = l_j; \quad \sum_{i=1}^{k_y} a^y_{ij} z^y_i = w_j, \quad j = \overline{1, m}. \tag{5}$$

Формулы (5) показывают равенство длины (ширины) прямоугольника сумме длин (ширин) полос, пересекающих этот прямоугольник.

Свойство 5. Пусть W — ширина полосы S, тогда связь между компонентами векторов $\boldsymbol{z}^{\mathbf{x}}$, $\boldsymbol{z}^{\mathbf{y}}$ и границами S выражается формулой

$$\sum_{i=1}^{k_y} z^y_i \leqslant W. \tag{6}$$

Эти неравенства означают, что суммарная ширина полос не может превосходить границ полосы S.

Таким образом, любая упаковка P может быть представлена в своей матричной интерпретации — в виде набора из пары бинарных матриц $\mathbf{A^x}$, $\mathbf{A^y}$ и пары векторов $\boldsymbol{z}^{\mathbf{x}}$ и $\boldsymbol{z}^{\mathbf{y}}$, удовлетворяющих свойствам 1—5. Покажем, что между упаковкой и ее матричной интерпретацией существует и обратная связь.

Пусть нам известны $(\mathbf{A^x}, \boldsymbol{z}^{\mathbf{x}})$ и $(\mathbf{A^y}, \boldsymbol{z}^{\mathbf{y}})$. Тогда связь между матрицами и векторами упаковки и координатами прямоугольников R в упаковке P устанавливается следующим утверждением.

Утверждение 1. Координаты (x_i, y_j) прямоугольников вычисляются по формулам

$$x_i = \sum_{k=1}^{k_i - 1} z^x_k, \text{ где } k_i = \min(j | a^x_{ij} = 1),\ i = \overline{1, m}; \tag{7}$$

$$y_i = \sum_{k=1}^{k_i - 1} z^y_k, \text{ где } k_i = \min(j | a^y_{ij} = 1),\ i = \overline{1, m}. \tag{8}$$

Однако оказывается, что не для любого произвольно заданного набора $(\mathbf{A^x}, \boldsymbol{z}^{\mathbf{x}})$ и $(\mathbf{A^y}, \boldsymbol{z}^{\mathbf{y}})$ существует соответствующая допустимая упаковка P.

Свойства 1—5 не являются достаточными для получения допустимых упаковок, а именно, в них не отражено условие не перекрытия прямоугольников.

Свойство 6. (Не перекрытие прямоугольников). Если в некотором столбце k_0 матрицы $\mathbf{A^x}$ имеются строки i_1 и i_2 такие, что $a^x_{i_1 k_0} = 1$ и $a^x_{i_2 k_0} = 1$, то для любого столбца $1 \leqslant k \leqslant k_y$ матрицы $\mathbf{A^y}$ верно: $a^y_{i_q k} \cdot a^y_{i_2 k} = 0$ (т. е. $a^y_{i_1 k}$ и $a^y_{i_2 k}$ одновременно не могут быть равны 1).

Действительно, если в столбце k_0 матрицы $\mathbf{A^x}$ элементы в i_1-й и i_2-й строках равны 1, то это означает, что вертикальный блок k_0 пересекает i_1-й и i_2-й прямоугольник, т. е. они не "раздвинуты" по горизонтальному направлению. Тогда, следуя определению допустимой упаковки, они должны быть раздвинутыми по "вертикальному" направлению. Следовательно, для любого столбца k матрицы $\mathbf{A^y}$ выполняется равенство $a^y_{i_q k} \cdot a^y_{i_2 k} = 0$.

Утверждение 2. Всякой допустимой упаковке P можно сопоставить единственным образом матрицы $\mathbf{A^x}$, $\mathbf{A^y}$ и векторы $\boldsymbol{z}^{\mathbf{x}}$ и $\boldsymbol{z}^{\mathbf{y}}$, удовлетворяющие свойствам 1—6.

Истинность утверждения следует из способа построения матриц.

Теорема. Упаковка P, полученная по формулам (7) и (8) из матриц $\mathbf{A^x}$, $\mathbf{A^y}$ и векторов $\boldsymbol{z}^{\mathbf{x}}$ и $\boldsymbol{z}^{\mathbf{y}}$, удовлетворяющих свойствам 1—6, является допустимой.

Доказательство. Пусть у нас есть две матрицы $\mathbf{A^x}$, $\mathbf{A^y}$ и два вектора $\boldsymbol{z}^{\mathbf{x}}$, $\boldsymbol{z}^{\mathbf{y}}$ такие, что выполнены свойства 1—6. Построим соответствующую (единственную) упаковку P, используя формулы (7) и (8), т. е. присвоим прямоугольникам R координаты относительно начала полосы S. Полученная упаковка P удовлетворяет условию (1): прямоугольники не выходят за границу полосы S (свойства 4 и 5).

Предположим, что для упаковки P условие (2) неверно. Это значит, что существуют такие прямоугольники i_1 и i_2, для которых верно $(x_{i_1} + l_{i_1} > x_{i_2}) \wedge (x_{i_2} + l_{i_2} > x_{i_1}) \wedge (y_{i_1} + w_{i_1} > y_{i_2}) \wedge \wedge (y_{i_2} + w_{i_2} > y_{i_1})$. Пусть $x_{i_1} \geqslant x_{i_2}$, тогда $x_{i_2} + l_{i_2} > x_{i_1} \geqslant x_{i_2}$. Из свойств 2 и 4 следует, что в матрице $\mathbf{A^x}$ в строке i_1 единицы начинаются раньше, чем они заканчиваются в строке i_2. Следовательно, существует такой столбец k^x_0, что $a^x_{i_1 k^x_0} = 1$ и $a^x_{i_2 k^x_0} = 1$. Предположение $x_{i_1} \leqslant x_{i_2}$ приводит к такому же результату. Повторим аналогичные рассуждения для y_{i_1} и y_{i_2}, получим $a^y_{i_1 k^y_0} = 1$ и $a^y_{i_2 k^y_0} = 1$. Таким образом упаковка P не удовлетворяет свойству 6, что противоречит предположению теоремы.

Пары $(\mathbf{A^x}, \boldsymbol{z}^{\mathbf{x}})$ и $(\mathbf{A^y}, \boldsymbol{z}^{\mathbf{y}})$, удовлетворяющие свойствам 1—6, назовем *допустимыми*.

Утверждение 3. Для нахождения оптимальной упаковки достаточно найти допустимые пары $(\mathbf{A^x}, \boldsymbol{z}^{\mathbf{x}})$ и $(\mathbf{A^y}, \boldsymbol{z}^{\mathbf{y}})$, такие что $\sum\limits_{i=1}^{k_x} z^x_i \to \min$.

3. Нижняя граница для 2DSPP

Утверждение 3 можно интерпретировать следующим образом: в задаче **2DSPP** в качестве нижней границы можно использовать оценку величины $\min \sum\limits_{i=1}^{k_x} z^x_i$, на значение которой влияет только пара $(\mathbf{A^x}, \boldsymbol{z}^{\mathbf{x}})$. (Пара $(\mathbf{A^y}, \boldsymbol{z}^{\mathbf{y}})$ необходима для проверки допустимости построенной упаковки.) Подсчитаем нижнюю границу, используя аналогию между задачей построения пары $(\mathbf{A^x}, \boldsymbol{z}^{\mathbf{x}})$ и задачей линейного раскроя.

Задача линейного раскроя (*C*utting Stock Problem, **1DCSP**). Имеется материал, поступающий в виде стержней длины H. Путем его раскроя требуется получить набор из m различных предметов заданных длин λ_i, $i = \overline{1, m}$, и в необходимом количестве b_i каждого вида, $i = \overline{1, m}$. Требуется раскроить материал на линейные предметы (заготовки) с минимальными затратами материала [11].

Добавим в задачу **1DCSP** дополнительное условие: потребуем, чтобы раскрой стержня содержал только одну заготовку одного типа. Обозначим новую задачу (Binary Cutting Stock Problem, **1DBCS**) как $E = (H,\ m,\ \boldsymbol{\lambda},\ \boldsymbol{b})$; $\boldsymbol{\lambda} = (\lambda_1,\ ...,\ \lambda_m)$; $\boldsymbol{b} = (b_1,\ ...,\ b_m)$.

Пусть вектор $\boldsymbol{\alpha}^j = (a_{1j},\ a_{21},\ ...,\ a_{mj})^{\mathrm{T}} \in Z^m_+$, где $a_{ij} \in \{0,\ 1\}$, описывает j-ю карту раскроя (способ раскроя стержня): его компонента $a_{ij} = 1$, если заготовка типа i получается из данного стержня, и $a_{ij} = 0$ в противном случае. Матрица $\mathbf{A_p} = (a_{ij})$, $i = \overline{1, m}$, $j = \overline{1, N}$, называется *раскройной матрицей*, здесь N — число всевозможных карт раскроя. Обозначим x_j, $j = \overline{1, N}$, *интенсивность* применения карты раскроя j, т. е. число прутков, которые должны быть разделены в соответствии с картой раскроя $\boldsymbol{\alpha}^j$. Тогда задача **1DBCSP** сводится к решению следующей задачи линейного целочисленного программирования:

$$Z^*(E) = \sum_{j=1}^{N} x_j \to \min,\ \mathbf{A_p}\boldsymbol{x} = \boldsymbol{b},\ x \in Z^N_+. \qquad (9)$$

Для получения нижней границы оптимального значения задачи (9) рассмотрим аналогичную задачу линейного программирования, в которой от-

сутствует условие целочисленности компонент вектора $\boldsymbol{x}$:

$$Z^*(E) = \sum_{j=1}^{N} x_j \to \min,\ \mathbf{A_p}\boldsymbol{x} = \boldsymbol{b},\ x \in R_+^N. \quad (10)$$

Поскольку число N для реальных задач достаточно велико, то задача (10) решается симплекс-методом с неявно заданной матрицей ограничений. При этом максимальное число ненулевых элементов x_j не превосходит заданного числа m (размера базисного множества).

Вернемся к задаче **2DSPP**. Пусть заданы исходные данные (W, m, R) для прямоугольной упаковки. Если положить $H = W$; $\lambda_i = w_i$; $b_i = l_i$; $i = \overline{1, m}$, то задаче построения пары ($\mathbf{A^x}$, $\boldsymbol{z}^{\mathbf{x}}$) можно сопоставить **1DBCSP** с параметрами ($H = W$; m; $\boldsymbol{\lambda} = w$; $\boldsymbol{b} = l$). Очевидно, что любое допустимое заполнение матриц ($\mathbf{A^x}$, $\boldsymbol{z}^{\mathbf{x}}$) даст нам допустимое решение соответствующей задачи **1DBCSP**. Обратное не верно, так как решение **1DBCSP** не удовлетворяет свойству 2 (продолженности единиц). Тем не менее любая нижняя граница для задачи **1DBCSP** будет также являться нижней границей для $\min \sum_{i=1}^{k_x} z_i^x$, а следовательно, и для задачи **2DSPP**. Таким образом, в качестве нижней границы можно использовать решение (10).

4. Матричный алгоритм поиска оптимального решения 2DSPP

Теперь мы готовы описать алгоритм поиска оптимального решения задачи **2DSPP**, базирующийся на матричном представлении упаковки. На каждом шаге алгоритма допустимая упаковка P представляется в виде пары ($\mathbf{A^x}$, $\boldsymbol{z}^{\mathbf{x}}$) и ($\mathbf{A^y}$, $\boldsymbol{z}^{\mathbf{y}}$). На следующем шаге ищется упаковка с меньшей длиной занятой полосы, нежели было получено ранее. Процесс поиска повторяется до тех пор, пока не будет достигнута нижняя граница (в этом случае лучшего решения просто не может быть) или улучшения не удалось найти.

В силу конечности числа прямоугольников, а также с учетом того, что рассматриваются только плотные упаковки, процесс поиска оптимальной упаковки будет конечным.

Алгоритм последовательного улучшения допустимых матриц и векторов состоит из выполнения следующих базовых процедур.

Шаг 1. Решается задача линейного программирования и ее результат берется в качестве нижней границы L_{down}. В качестве верхней границы работы алгоритма берется $L_{up} = \infty$.

Шаг 2. Строится пара ($\mathbf{A^x}$, $\boldsymbol{z}^{\mathbf{x}}$), удовлетворяющая свойствам 1—5 и условию $\sum_{i=1}^{k_x} z_i^x < L_{up}$. Если пара существует, то переход к шагу 3, иначе — к шагу 4.

Шаг 3. Строится допустимая пара ($\mathbf{A^y}$, $\boldsymbol{z}^{\mathbf{y}}$), удовлетворяющая свойствам 1—6 и $\sum_{i=1}^{k_y} z_i^y \leqslant W$. Если пара существует, то переход к шагу 4, иначе — к шагу 2.

Шаг 4. $L_{up} = \sum_{i=1}^{k_x} z_i^x$. Если $L_{up} = L_{down}$, то переход к шагу 5, иначе переход к шагу 2.

Шаг 5. L_{up} — оптимум.

5. Численный эксперимент

Численный эксперимент состоит из двух этапов: первый — расчет случайно сгенерированных тестовых примеров; второй — сравнение результатов работы предложенного алгоритма и "метода зон" (реализация Бухваловой В. В.).

Для проведения эксперимента использовались случайно сгенерированные тесты по методике, описанной в работе [7]. Ширина полосы постоянна для всех тестов и равна $W = 100$. Прямоугольники, которые должны быть упакованы в полосу, разделены на четыре типа (табл. 1), в каждом из которых размеры прямоугольников генерируются случайно из соответственно указанных диапазонов. Задачи разбиты на три класса (табл. 2), состоящие из прямоугольников разных типов в указанном процентном соотношении. Для каждого класса задач и m = (12, 14, 16, 18) было сгенерировано по 10 примеров. Численный эксперимент проводился на компьютере P-IV 2000 GHz, 1 Gb. Время работы программы было ограничено 90 мин. Результаты экспериментов сведены в табл. 3.

Как видно из табл. 3, время работы алгоритма в сильной степени зависит от того, достигает решение нижней границы или нет. Так, например, для $m = 12$ оптимум всегда достигался за время, меньшее 7 с, в то время как при $m = 18$ только в

Таблица 1

Типы прямоугольников

Тип прямоугольника	Диапазон длинн заготовок	Диапазон ширин заготовок
1	[1, 50]	[75, 100]
2	[75, 100]	[1, 50]
3	[50, 100]	[50, 100]
4	[1, 50]	[1, 50]

Таблица 2

Классы задач

Класс задач	Типы прямоугольников			
	1	2	3	4
I	20 %	20 %	20 %	40 %
II	15 %	15 %	15 %	55 %
III	10 %	10 %	10 %	70 %

Таблица 3

Результаты экспериментов (время 90 мин)

Классы задач	m	Число задач, где был получен доказуемый оптимум*	Число задач, где была достигнута нижняя граница**	Минимальное время, с	Максимальное время, с***
I	12	10	6	<1	3
	14	10	1	<1	358
	16	10	6	<1	2169
	18	6	4	<1	4441
II	12	10	3	<1	1
	14	10	4	<1	78
	16	8	2	15	1569
	18	5	4	<1	2566
III	12	10	6	<1	7
	14	10	1	<1	454
	16	5	0	57	3005
	18	3	1	314	4301

*Указано число примеров из 10 сгенерированных, для которых удалось получить оптимальные решения. Оптимальность была доказана либо достижением нижней границы, либо перебором всех допустимых вариантов.

** Представлено число примеров, в которых оптимальность решения была доказана путем достижения нижней границы.

***Максимальное время зафиксировано только для тех задач, у которых было получено оптимальное решение.

редких примерах удалось получить оптимум, значение которого не совпадает с нижней границей.

Сравнение с "методом зон" (табл. 4) проводилось на тех же тестовых задачах, время счета было ограничено 10 мин. В качестве показателя эффективности алгоритма служит коэффициент раскроя (Cutting Coefficient, **CC**), равный отношению суммарной площади прямоугольников к площади занятой части полосы $CC = \frac{\sum_{i=1}^{m} l_i w_i}{WL_{up}}$. Чем ближе коэффициент раскроя к единице, тем меньше остается "обрезков" полосы, идущих в отходы. Таким образом, из двух аналогичных по прочим характеристикам упаковок предпочтительнее та упаковка, у которой коэффициент раскроя выше. При этом учитываются коэффициенты раскроя для лучших найденных решений.

Как видно из табл. 4, алгоритм, основанный на "методе зон", может гарантированно получать оптимальные решения за 10 мин только для задач размерности $m = 12$.

При $m = 14$ алгоритмом, основанным на "методе зон", были получены упаковки, длина занятой полосы у которых совпадает с оптимальным значением, но в большинстве случаев их оптимальность доказать не удалось. Матричный алгоритм получил оптимальные решения для всех примеров во всех группах данной размерности.

При $m = 16$ и $m = 18$ матричный алгоритм за существующее временное ограничение находил лучшие упаковки (даже в случаях, когда оптимальность ее не удалось доказать), нежели "метод зон", что иллюстрирует табл. 4.

Заключение

Приведенный в статье матричный подход к решению **2BDPP** имеет существенные отличия от известных ранее методов решения этой задачи.

Во-первых, алгоритм "работает" с бинарными матрицами определенного типа, а не с конкретными геометрическими объектами (прямоугольниками, "зонами", блок-структурами и т. д.), что является немаловажным фактом при реализации алгоритма, а также свидетельствует о его универсальности в смысле возможности применения модификации алгоритма для решения задач упаковки более сложных геометрических объектов.

Во-вторых, алгоритм сводится к решению двух задач линейного раскроя, трудоемкость решения которых существенно меньше трудоемкости решения **2BDPP**. Базируясь на приведенном матричном построении упаковки, используя методы математического программирования, можно строить более точные нижние границы, а также

Таблица 4

Сравнение с "методом зон" (время 10 мин)

Классы задач	m	Матричный алгоритм		"Метод зон"	
		Число задач, где был получен доказуемый оптимум	Средний коэффициент раскроя	Число задач, где был получен доказуемый оптимум	Средний коэффициент раскроя
I	12	10	0,85	10	0,85
	14	10	0,92	1	0,92
	16	8	0,88	1	0,76
	18	4	0,89	0	0,73
II	12	10	0,88	10	0,88
	14	10	0,86	0	0,86
	16	4	0,9	0	0,81
	18	4	0,88	0	0,7
III	12	10	0,85	10	0,85
	14	10	0,91	0	0,91
	16	2	0,91	0	0,83
	18	1	0,88	0	0,75

использовать новые виды отсечений недопустимых вариантов. Этот факт был подтвержден численным экспериментом.

В-третьих, данный подход обобщается на случаи, когда мерность рассматриваемых объектов больше двух. Число матриц и векторов упаковки будет равна мерности рассматриваемых прямоугольников. Например, в трехмерном случае их будет три ($\mathbf{A}^{\mathbf{x}}$, z^x), ($\mathbf{A}^{\mathbf{y}}$, z^y), ($\mathbf{A}^{\mathbf{z}}$, z^z); в четырехмерном — четыре и т. д. При этом свойства 1—5 остаются неизменными, а свойство 6 модифицируется в зависимости от рассматриваемой мерности.

Автор благодарен профессору, д-ру техн. наук Э. А. Мухачевой за ценные замечания и за интерес, проявленный к работе.

Список литературы

1. **Dyckoff H.** A typology of cutting and packing problems // European Journal of Operational Research. 1990. Vol. 44. P. 145—159.

2. **Wascherr G., Haussner H. and Schumann H.** An improved typology of cutting and packing problems // Working paper, n. 24/2005, at Faculty of Economics and Management, Otto von Guericke University Magdeburg. 2005.

3. **Beasley J. E.** An exact two-dimensional non-guillotine cutting tree search procedure // Operational Research. 1985. Vol. 33. P. 49—64.

4. **Martello S., Vigo D.** Exact solution of two-dimensional finite bin packing problem // Managment Science. 1997. Vol. 35. P. 64—68.

5. **Липовецкий А. И.** К оптимизации свободного размещения прямоугольников // Автоматическое проектирование в машиностроении. Минск: ИТК АН БССР, 1985. С. 80—87.

6. **Бухвалова В. В.** Задача прямоугольного раскроя: метод зон и другие алгоритмы. СПб.: СПбГУ, 2001. 96 с.

7. **Fekete S., Schepers J.** A combinatorial characterization of higher dimensional orthogonal packing // Mathematics of Operations Research 2004 29. P. 353—368.

8. **Стоян Ю. Г., Яковлев С. В.** Математические модели и оптимизационные методы геометрического проектирования // Киев: Наук. Думка. 1986.

9. **Новожилова М. В.** Решение задачи поиска глобального экстремума линейной функции цели на структуре линейных неравенств // Препринт института проблем машиностроения. Харьков; АН УССР, 1988.

10. **Мухачева Э. А., Мухачева А. С., Чиглинцев А. В.** Генетический алгоритм блочной структуры в задачах двухмерной упаковки // Информационные технологии. 1999. № 11. С. 13—18.

11. **Мухачева Э. А., Картак В. М.** Модифицированный метод ветвей и границ: алгоритм и численный эксперимент для задачи одномерного раскроя // Информационные технологии. 2000. № 9. С. 15—22.

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И СЕТИ

УДК 004.4’416

В. А. Богатырев, д-р техн. наук., проф.,
Санкт-Петербургский государственный
университет информационных технологий
механики и оптики

Оптимизация отказоустойчивых кластеров с неполнодоступностью узлов и неоднородностью потока запросов

Сформулированы задачи оптимизации кластеров с неоднородностью потока запросов и функциональной неполнодоступностью узлов. В результате оптимизации находится распределение весовых коэффициентов, задающих доли потока запросов, направляемых при балансировке нагрузки в различные узлы. При оптимизации кластерной структуры определяются кратности резервирования узлов, обеспечивающие максимальную производительность и надежность кластера при ограничении средств на его разработку.

1. Распределение потока запросов в кластерах

Требования повышения надежности, отказоустойчивости и производительности информационных систем приводят к необходимости кластеризации серверных систем с обеспечением при распределении запросов балансировки нагрузки и адаптации к отказам узлов [1—3].

Под кластером понимается группа независимых компьютеров (серверов), совместно выполняющих общий набор приложений (функциональных задач), воспринимаемая внешними клиентами как единый ресурс. Все поступающие в кластер запросы по мере возможности равномерно распределяются между компьютерами (узлами) кластера, снимая риск их перегрузки.

Организация распределения нагрузки в кластере во многом усложняется при неоднородности потоков запросов и не равной доступности узлов (серверов) кластера, зависящей от типа функционального запроса и от параметрических и функциональных возможностей узлов по их выполнению. Несовпадение параметрических и функциональных возможностей узлов кластера может закладываться при его проектировании или формироваться в результате отказов и модернизации системы.

Поток, для которого выделяется несколько типов запросов, различающихся параметрами их обслуживания или доступностью узлов кластера, будем считать неоднородным.

Кластеры, для которых существуют узлы, не способные обслужить все типы запросов, отнесем к кластерам с неполнодоступностью узлов.

Кластер имеет функциональную неполнодоступность узлов, если при неоднородном потоке функциональных запросов $F = \{f_1, f_2, f_3, ..., f_n\}$ любой запрос $f_i \in F$ не может быть выполнен любым узлом. Заметим, что изучение такого класса объектов особенно важно для отказоустойчивых кластеров, сохраняющих свое функционирование по мере накопления отказов при деградации, приводящей к функциональной неполнодоступности (и соответственно к функциональной неоднородности) узлов. Функциональная неоднородность узлов кластера может закладываться при проектировании, исходя, например, из экономии средств на реализацию системы.

Рис. 1. Классификация кластеров по возможности распределения потока запросов

Кластер, для которого узлы различаются качеством (параметрами) обслуживания (временем выполнения, вероятностью сбоя, точностью и др.) однотипных запросов, будем считать неоднородным по параметрам узлов. Параметрическая неоднородность узлов может формироваться, например, в результате модернизации (масштабирования) кластера, при которой к ранее работающим узлам добавляются новые, и соответственно, более производительные.

Классификация кластеров по возможности распределения потока запросов, доступности и однородности узлов представлена на рис. 1.

Функциональные возможности узлов кластерной системы (их доступности по выполнению функциональных запросов $f_i \in F$) отображается матрицей $\|\varphi_{ij}\|_{n \times m}$, элемент которой $\varphi_{ij} = 1$, если j-й узел способен выполнить функцию $f_i \in F$ (доступен для выполнения запроса f_i), в противном случае $\varphi_{ij} = 0$ [4].

Информация о среднем времени выполнения запросов $f_i \in F$ различными узлами задается в общем случае матрицей $\|\nu_{ij}\|_{n \times m}$, в которой ν_{ij} — среднее время выполнения запроса $f_i \in F$ j-м узлом кластера. Если параметры обслуживания запросов $f_i \in F$ для всех узлов одинаковы, то указанная матрица преобразуется к вектору-столбцу $<\nu_i>$.

Для однородного потока запросов (когда запросы не различаются по типам выполняемых функций) при параметрической неоднородности узлов среднее время выполнения запросов в различных узлах кластера характеризуется вектором-строкой $(\nu_1, \nu_2, ..., \nu_m)$, а при их параметрической однородности это время задается числом ν.

Для систем с неоднородным потоком функциональных запросов при полной доступности узлов в матрице $\|\varphi_{ij}\|_{n \times m}$ выполняется условие $(\forall i)(\forall j)(\varphi_{ij} = 1)$, а при их неполнодоступности условие — $(\exists i)(\exists j)(\varphi_{ij} = 0)$.

Обозначим множество узлов, способных обслужить запросы $f_i \in F$ и $f_k \in F$, через M_i и M_k, тогда в общем случае

$$(\exists i)(\exists k)(M_i \cap M_k \neq \varnothing). \qquad (1)$$

Выделим частные случаи:

$$(\forall i)(\forall k)(M_i \cap M_k) = \varnothing), \qquad (2)$$

$$(\forall i)(\forall k)((M_i \cap M_k = \varnothing) \vee (M_i = M_k)), \qquad (3)$$

когда узлы кластера разделены на группы, каждая из которых выделена на выполнение одного (2) или нескольких (3) типов запросов (приложений). Например, в серверной системе выделяются группы Web-серверов, почтовых серверов, FTP-серверов, серверов-печати и др.

Матрицы $\|\varphi_{ij}\|_{n \times m}$ для случаев (1), (2), (3) имеют, например, соответственно вид:

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

При проектировании и эксплуатации кластерных систем с неоднородностью потока запросов и функциональной неполнодоступностью узлов, с целью повышения эффективности балансировки нагрузки, возникает задача оптимизации весовых коэффициентов, определяющих доли распределяемых в различные узлы запросов $f_i \in F$.

В результате оптимизации каждому узлу присваивается вес (весовой коэффициент) в зависимости от типа функционального запроса $f_i \in F$ с учетом числа выполняющих его узлов и их нагрузки, создаваемой потоком на выполнение других функций $F = \{f_1, f_2, f_3, ..., f_n\}$. При параметрической неоднородности узлов искомые веса должны назначаться с учетом (пропорционально) производительности узлов. Узлы кластера с большим весом получают на выполнение большую долю запросов $f_i \in F$.

Оптимизация, осуществляемая при проектировании системы, помимо определения весовых коэффициентов, может включать поиск кратности резервирования различных узлов, обеспечивающей максимальную эффективность кластера при ограничении средств на его разработку.

2. Оптимизация распределения неоднородного потока запросов

Для кластерных систем с неоднородностью потока запросов и функциональной неполнодоступностью узлов задача оптимизации весовых коэффициентов, определяющих доли направляемых в различные узлы запросов $f_i \in F$, с целью балансировки нагрузки узлов кластера может быть сформулирована следующим образом.

Задано:

- число узлов в кластере m;
- функциональные возможности узлов по обработке запросов $f_i \in F$, представленные матрицей $\|\varphi_{ij}\|_{n \times m}$;
- параметрические возможности узлов по выполнению запросов $F = \{f_1, f_2, f_3, ..., f_n\}$ (среднее время выполнения запроса $f_i \in F$), представленные вектором-столбцом $\langle v_i \rangle$ или, если используются узлы разной производительности, — матрицей $\|v_{ij}\|_{n \times m}$;
- интенсивность λ суммарного потока запросов $F = \{f_1, f_2, f_3, ..., f_n\}$;
- вероятности (доли) поступления запросов различных типов $\{f_1, f_2, f_3, ..., f_n\}$, представленные вектором-столбцом $\langle b_i \rangle$, причем $\sum_{i=1}^{n} b_i = 1$.

Требуется найти распределение весов $\|a_{ij}\|$, используемых при балансировке нагрузки, обеспечивающее минимум среднего времени T_0 пребывания запросов в системе:

$$T_0 \to \min, \text{ при этом } \sum_{j=1}^{m} a_{ij} = 1,$$

где $a_{ij} = \begin{cases} 0 \leqslant a_{ij} \geqslant 1 & if\ \varphi_{ij} = 1, \\ a_{ij} = 0 & if\ \varphi_{ij} = 0. \end{cases}$

Кроме того, должно соблюдаться условие неперегруженности всех узлов, т. е. $(\forall j)\left[\left(\lambda \sum_{i=1}^{n} a_{ij} b_i v_i\right) < 1\right]$.

Заметим, что поставленная задача оптимизации весовых коэффициентов применима к системам, соответствующим общему виду (1), в частных случаях (2) и (3) при параметрической однородности узлов балансировку нагрузки обеспечивает равномерное циклическое распределение запросов (алгоритм балансировки нагрузки — *simple round—robin* [1—3]).

Модель обслуживания запросов в кластере. Будем считать все поступающие в кластер потоки запросов простейшими, а распределение времен их выполнения экспоненциальным. Вычислительный процесс в каждом сервере проинтерпретируем системой массового обслуживания типа M/M/1 с неограниченной очередью. Модель обслуживания в кластерной системе с распределением запросов представлена на рис 2.

Для стационарного режима среднее время пребывания запросов в системе (с параметрической однородностью узлов) определим по формуле

$$T_0 = \sum_{i=1}^{n} b_i \sum_{j=1}^{m} \frac{a_{ij} v_i}{1 - \lambda \sum_{k=1}^{n} b_k a_{kj} v_k}. \qquad (4)$$

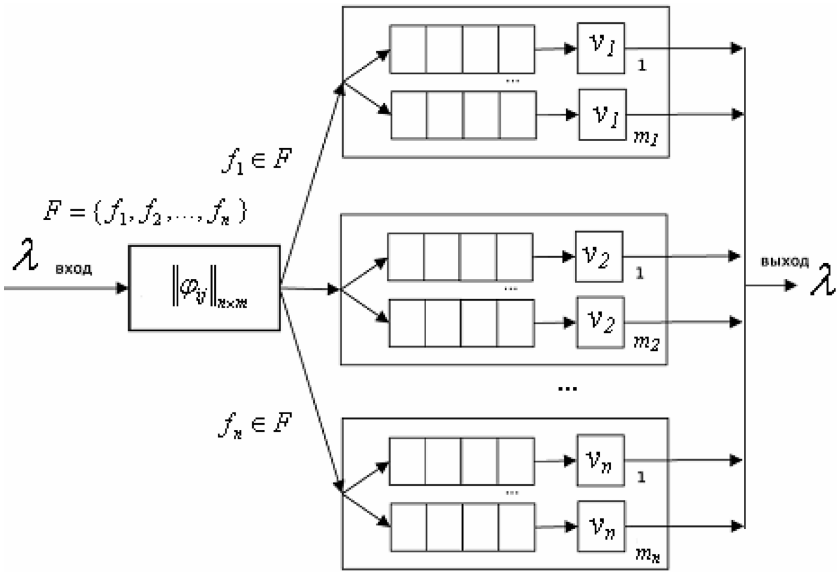


Рис. 2. Модель обслуживания запросов в кластерной системе

Заметим, что b_i — вероятность того, что поступивший запрос относится к i-му типу, а вес a_{ij} (искомый при оптимизации) соответствует вероятности распределения запроса $f_i \in F$ на обслуживание в j-й узел; $\lambda b_k a_{kj} \nu_k$ и $\lambda \sum_{k=1}^{n} b_k a_{kj} \nu_k$ — загрузка j-го узла запросами на выполнение k-го типа функций $f_k \in F$ и на выполнение всех типов функций, $k = 1, 2, 3, ..., n$.

Формула (4) позволяет на основе процедуры оптимизации определить распределение весов $\|a_{ij}\|$, обеспечивающее минимум среднего времени T_0 пребывания запросов в системе.

При параметрической неоднородности узлов среднее время пребывания запросов в кластере

$$T_0 = \sum_{i=1}^{n} b_i \sum_{j=1}^{m} \frac{a_{ij}\nu_{ij}}{1 - \lambda \sum_{k=1}^{n} b_k a_{kj} \nu_{kj}}.$$

Если проводится резервирование первого, второго, ..., m-го узлов кластера соответственно с кратностью резервирования $(m_1, m_2, m_3, ..., m_m)$, то функциональная доступность узлов отображается матрицей:

$$\|\varphi_{ij}\|_{n \times \sum_{j=1}^{m} m_j}.$$

В этом случае в кластере выделяется m групп из $(m_1, m_2, m_3, ..., m_m)$ одноготипных узлов.

При равномерном распределении запросов внутри группы идентичных резервированных узлов среднее время пребывания запросов в кластере будет

$$T_0 = \sum_{i=1}^{n} b_i \sum_{j=1}^{m} \frac{a_{ij}\nu_i}{1 - \lambda \sum_{k=1}^{n} b_k \left(\frac{a_{kj}}{m_j}\right) \nu_k}.$$

При этом весовой коэффициент a_{ij} соответствует вероятности распределения запроса $f_i \in F$ на обслуживание в j-ю группу из m_j узлов (кратность резервирования j-го узла). Загрузка узла j-й резервированной группы запросами на выполнение $f_k \in F$ и на выполнение всех функций $F = \{f_1, f_2, ..., f_n\}$ соответственно равна

$$\lambda b_k \left(\frac{a_{kj}}{m_j}\right) \nu_k \text{ и } \lambda \sum_{k=1}^{n} b_k \left(\frac{a_{kj}}{m_j}\right) \nu_k.$$

Пример нахождения оптимального распределения весов. Проиллюстрируем применение оптимизации кластера на конкретном примере. Пусть задана интенсивность запросов $\lambda = 1{,}5\ \text{с}^{-1}$, а матрица $\|\varphi_{ij}\|_{n \times m}$ и векторы $<b_i>$, $<v_i>$ соответственно имеют вид:

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}, \begin{pmatrix} 0{,}5 \\ 0{,}3 \\ 0{,}2 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \\ 4 \end{pmatrix}.$$

Ниже приведен пример оптимизации рассматриваемого кластера с использованием пакета математических расчетов MathCAD13:

$$T(a) := \sum_{i=0}^{2} \left[b_i \sum_{j=0}^{4} \frac{a_{i,j} v_i}{1 - \left[\sum_{k=0}^{2} (\lambda b_k a_{k,j} v_k) \right]} \right]$$

$$a := \begin{pmatrix} 0 & 0.333 & 0.333 & 0 & 0.334 \\ 0.25 & 0.25 & 0 & 0.25 & 0.25 \\ 0.25 & 0 & 0.25 & 0.25 & 0.25 \end{pmatrix}$$

Given

$a_{0,0} = 0 \quad 0 \leqslant a_{0,1} \leqslant 1 \quad 0 \leqslant a_{0,2} \leqslant 1 \quad 0 \leqslant a_{0,3} \leqslant 1$
$0 \leqslant a_{0,4} \leqslant 1$

$a_{1,2} = 0 \quad 0 \leqslant a_{1,1} \leqslant 1 \quad 0 \leqslant a_{1,0} \leqslant 1 \quad 0 \leqslant a_{1,3} \leqslant 1$
$0 \leqslant a_{1,4} \leqslant 1$

$0 \leqslant a_{2,0} \leqslant 1 \quad 0 \leqslant a_{2,2} \leqslant 1 \quad 0 \leqslant a_{2,3} \leqslant 1$
$0 \leqslant a_{2,4} \leqslant 1 \quad a_{2,1} = 0$

$a_{0,0} + a_{0,1} + a_{0,2} + a_{0,3} + a_{0,4} = 1$

$a_{1,0} + a_{1,1} + a_{1,2} + a_{1,3} + a_{1,4} = 1$

$a_{2,0} + a_{2,1} + a_{2,2} + a_{2,3} + a_{2,4} = 1$

$(a_{0,0} \cdot b_0 \cdot v_0 + a_{1,0} \cdot b_1 \cdot v_1 + a_{2,0} \cdot b_2 \cdot v_2) \cdot \lambda < 0{,}99999$

$(a_{0,1} \cdot b_0 \cdot v_0 + a_{1,1} \cdot b_1 \cdot v_1 + a_{2,1} \cdot b_2 \cdot v_2) \cdot \lambda < 0{,}99999$

$(a_{0,4} \cdot b_0 \cdot v_0 + a_{1,4} \cdot b_1 \cdot v_1 + a_{2,4} \cdot b_2 \cdot v_2) \cdot \lambda < 0{,}999999$

$(a_{0,2} \cdot b_0 \cdot v_0 + a_{1,2} \cdot b_1 \cdot v_1 + a_{2,2} \cdot b_2 \cdot v_2) \cdot \lambda < 0{,}9999999$

$(a_{0,3} \cdot b_0 \cdot v_0 + a_{1,3} \cdot b_1 \cdot v_1 + a_{2,3} \cdot b_2 \cdot v_2) \cdot \lambda < 0{,}99999$

$w := \text{Minimize}(T, a)$

$$w = \begin{pmatrix} 0 & 0.541 & 0.343 & 0 & 0.116 \\ 0.36 & 0.188 & 0 & 0.143 & 0.309 \\ 0.145 & 0 & 0.335 & 0.389 & 0.13 \end{pmatrix}$$

$T(w) = 6{,}470589453081987$

$T(a) = 9{,}39305014934025$

В результате расчетов установлено, что минимум среднего времени пребывания запросов в кластере T_0, равный 9,4 с, достигается при оптимальном распределении весов, заданном матрицей w

$$\begin{bmatrix} 0 & 0.541 & 0.343 & 0 & 0.116 \\ 0.36 & 0.188 & 0 & 0.143 & 0.309 \\ 0.145 & 0 & 0.335 & 0.389 & 0.13 \end{bmatrix}.$$

В приведенных расчетах для оценки эффективности оптимизации весовых коэффициентов вычислено среднее время пребывания запросов в кластере ($T_r = 9{,}4$ с), поддерживающем алгоритм равномерного циклического распределения запросов, для которого $a_{ij} = 1/\sum_{j=1}^{m} \varphi_{ij}$.

Таким образом, для рассмотренного примера в результате оптимизации весов среднее время пребывания запросов сокращается с 9,4 до 6,5 с, т. е. ускорение $T_r - T_0$ составляет примерно 30 %, что свидетельствует о значительной эффективности оптимизации.

Оптимальное распределение весов, используемых при балансировке нагрузки, зависит от интенсивности потока запросов. Результаты поиска оптимального распределения весов и расчета соответствующего ему среднего времени пребывания запросов в системе для заданных выше исходных данных при изменении интенсивности потока запросов λ сведены в таблицу. В таблице также представлены результаты оценки среднего времени пребывания запросов при балансировке нагрузки на основе алгоритма равномерного циклического распределения без оптимизации весов.

Из таблицы видно, что по мере возрастания интенсивности запросов (и соответственно нагрузки узлов) эффективность оптимизации возрастает, причем при интенсивности запросов $\lambda \geqslant 2$ с$^{-1}$ в результате оптимизации удается сохранить стационарный режим работы системы (отсутствие перегруженности системы). Таким образом, рассмотренная оптимизация позволяет повысить

Эффект от оптимизации весов, используемых при балансировке нагрузки кластера

Интенсивность запросов λ, с$^{-1}$	Оптимальное распределение весов $\|a_{ij}\|$	Среднее время пребывания запросов		Эффект от оптимизации весов ($T_r - T_0$), с
		при оптимизации весов T_0, с	без оптимизации весов T_r, с	
0,5	$\begin{bmatrix} 0 & 0{,}397 & 0{,}421 & 0 & 0{,}182 \\ 0{,}27 & 0{,}268 & 0 & 0{,}259 & 0{,}203 \\ 0{,}247 & 0 & 0{,}287 & 0{,}258 & 0{,}208 \end{bmatrix}$	2,82	2,85	0,035
1	$\begin{bmatrix} 0 & 0{,}397 & 0{,}421 & 0 & 0{,}182 \\ 0{,}27 & 0{,}268 & 0 & 0{,}259 & 0{,}203 \\ 0{,}247 & 0 & 0{,}287 & 0{,}258 & 0{,}208 \end{bmatrix}$	3,92	4,15	0,22
1,25	$\begin{bmatrix} 0 & 0{,}541 & 0{,}343 & 0 & 0{,}116 \\ 0{,}36 & 0{,}188 & 0 & 0{,}143 & 0{,}309 \\ 0{,}145 & 0 & 0{,}335 & 0{,}389 & 0{,}13 \end{bmatrix}$	4,89	5,53	0,644
1,5	$\begin{bmatrix} 0 & 0{,}541 & 0{,}343 & 0 & 0{,}116 \\ 0{,}36 & 0{,}188 & 0 & 0{,}143 & 0{,}309 \\ 0{,}145 & 0 & 0{,}335 & 0{,}389 & 0{,}13 \end{bmatrix}$	6,47	9,39	2,92
1,6	$\begin{bmatrix} 0 & 0{,}397 & 0{,}421 & 0 & 0{,}182 \\ 0{,}27 & 0{,}268 & 0 & 0{,}259 & 0{,}203 \\ 0{,}247 & 0 & 0{,}287 & 0{,}258 & 0{,}208 \end{bmatrix}$	7,43	15,8	8,37
1,65	$\begin{bmatrix} 0 & 0{,}541 & 0{,}343 & 0 & 0{,}116 \\ 0{,}36 & 0{,}188 & 0 & 0{,}143 & 0{,}309 \\ 0{,}145 & 0 & 0{,}335 & 0{,}389 & 0{,}13 \end{bmatrix}$	8,03	30,39	22,37
2	$\begin{bmatrix} 0 & 0{,}471 & 0{,}367 & 0 & 0{,}162 \\ 0{,}362 & 0{,}227 & 0 & 0{,}131 & 0{,}28 \\ 0{,}143 & 0 & 0{,}321 & 0{,}402 & 0{,}134 \end{bmatrix}$	18,33	Нарушены условия стационарности (перегруженность системы)	Сохранение работоспособности
2,2	$\begin{bmatrix} 0 & 0{,}376 & 0{,}424 & 0 & 0{,}2 \\ 0{,}25 & 0{,}28 & 0 & 0{,}273 & 0{,}197 \\ 0{,}269 & 0 & 0{,}285 & 0{,}243 & 0{,}204 \end{bmatrix}$	68,75	Перегруженность системы	Сохранение работоспособности

производительность системы и ее устойчивость к увеличению интенсивности потока запросов λ (для приведенного примера, по крайней мере, от $\lambda = 1{,}7\ с^{-1}$ до $\lambda = 2{,}2\ с^{-1}$).

3. Структурная оптимизация неоднородных кластеров с балансировкой нагрузки

Для кластерных систем с неоднородностью потока запросов и функциональной неполнодоступностью узлов задача структурной оптимизации (проводимой на стадии проектирования системы) может быть сформулирована следующим образом.

Задано:

- интенсивность λ суммарного потока запросов $F = \{f_1, f_2, f_3, ..., f_n\}$;
- вероятности запросов всех типов $f_i \in F$, представленные вектором-столбцом $< b_i >$ причем $\sum_{i=1}^{n} b_i = 1$;
- минимальное число узлов в кластере m и их возможности по обработке запросов $f_i \in F$, отображаемые матрицей $\| \varphi_{ij} \|_{n \times m}$;
- параметры обслуживания узлами различных запросов (средние времена выполнения запросов $f_i \in F$), представленные вектором-столбцом $< v_i >$ (при параметрической однородности узлов);
- суммарные средства c_0, выделяемые на построение серверной системы;
- стоимости (c_1, c_2, c_3, ..., c_m) для всех m типов узлов.

Требуется найти такое распределение кратности резервирования узлов каждого типа (m_1, m_2, m_3, ..., m_m) и распределение весов $\| a_{ij} \|$, используемых при направлении запросов в каждую резервированную группу узлов, которое обеспечивает минимум среднего времени пребывания запросов T_0 в системе:

$$T_0 = \sum_{i=1}^{n} b_i \sum_{j=1}^{m} \frac{a_{ij} \nu_i}{1 - \lambda \sum_{k=1}^{n} b_k \left(\frac{a_{kj}}{m_j} \right) \nu_k} \to \min.$$

При оптимизации должны соблюдаться следующие условия:

$$\sum_{j=1}^{m} a_{ij} = 1 \text{ и } a_{ij} = \begin{cases} 0 \leqslant a_{ij} \geqslant 1 \ if \ \varphi_{ij} = 1 \\ a_{ij} = 0 \ if \ \varphi_{ij} = 0 \end{cases};$$

$$\lambda \sum_{k=1}^{n} b_k \left(\frac{a_{kj}}{m_j} \right) \nu_k < 1;$$

$$\sum_{j=1}^{m} c_j m_j \leqslant c_0, \ m_i \geqslant 1.$$

Заметим, что представленная задача может быть модифицирована так, что задается не минимальное число узлов, требуемое для функционирования системы, а множество типов узлов по их функциональной комплектации, которые могут использоваться при построении кластера. При этом требуется найти число узлов каждого типа, которое при полном выполнении набора возлагаемых на систему функций обеспечивало бы максимум эффективности системы при ограничении затрат на ее реализацию. При такой постановке некоторые узлы из заданного набора в результате оптимизации могут не включаться в структуру кластера (условие $m_i \geqslant 1$ заменяется на $m_i \geqslant 0$).

В качестве дополнительного условия при структурной оптимизации кластера может быть задан нижний допустимый уровень надежности кластера.

При выборе структуры систем ответственного назначения, особенно работающих в реальном времени, помимо минимизации среднего времени пребывания запросов требуется проводить и максимизацию надежности кластера.

При оптимизации ищется распределение кратности резервирования узлов каждого типа (m_1, m_2, m_3, ..., m_m) и распределение весов $\| a_{ij} \|$, используемых при направлении запросов в каждую резервированную группу узлов, с учетом достижения компромисса между минимумом среднего времени пребывания запросов T_0 в системе и максимумом ее надежности при ограничении средств c_0 на ее реализацию.

В этом случае задача оптимизации становится векторной (многокритериальной).

В качестве целевой функции при векторной оптимизации можно рассматривать мультипликативный критерий, задаваемый в виде $K = P/T$.

Расчет надежности системы проведем по следующей формуле [5]:

$$P = \sum_{k_1=0}^{m_1} \sum_{k_2=0}^{m_2} \dots \sum_{k_m=0}^{m_m} C_{m_1}^{k_1} C_{m_2}^{k_2} \dots$$

$$\dots C_{m_m}^{k_m} p_\Omega^{\sum_{j=1}^{m} k_j} (1 - p_\Omega)^{\sum_{j=1}^{m} m_j - \sum_{j=1}^{m} k_j} \times$$

$$\times \prod_{i=1}^{n} (1 - (1 - p_i)^{k_1 a_{i1} + k_2 a_{i2} + \dots + k_m a_{im}}),$$

где p_Ω и p_i показатели надежности базового оборудования Ω узла, отказ которого приводит к выходу из строя всего узла, и его оборудования $\Phi_i \backslash \Omega$, задействованного при выполнении только i-го типа функции. Отказ оборудования $\Phi_i \backslash \Omega$ узла приводит к потере им выполнения только одной функции $f_i \in F$.

Таким образом, сформулированы задачи оптимизации кластеров с неоднородностью потока запросов и функциональной неполнодоступностью узлов. В результате предлагаемой оптимизации находится распределение весовых коэффициентов, определяющих доли направляемых в различные узлы запросов $f_i \in F$, обеспечивающие балансировку нагрузки кластера. Оптимизация, помимо определения весовых коэффициентов, используемых при балансировке нагрузки, может включать определение кратности резервирования узлов, обеспечивающее максимальную производительность и надежность кластера при ограничении средств на его разработку.

Показано, что эффективность предлагаемой оптимизации возрастает по мере увеличения интенсивности потока запросов. В результате оптимизации удается увеличить верхнюю границу интенсивности запросов входного потока, при которой обеспечивается стабильная работа кластера (его работа в стационарном режиме).

Список литературы

1. **Балансировка** нагрузки сети. http://www.microsoft.com/Rus.

2. **Бруно Ч., Килмартин Г.** Способы выравнивания нагрузки средства повышения производительности Web-узлов // Сети. 2000. № 1.

3. **Немков М.** Распределение нагрузки с помощью коммутатора // КомпьютерПресс. 2000. № 10.

4. **Богатырев В. А.** К повышению надежности вычислительных систем на основе динамического распределения функций // Изв. вузов СССР. Приборостроение. 1981. № 8. С. 62—65.

5. **Богатырев В. А.** Оценка надежности и оптимизация комплектации вычислительных систем при резервировании функционально неоднородных компьютерных узлов // Информационные технологии. 2007. № 5. С. 41—47.

УДК 004.7

И. К. Коханенко, д-р техн. наук, проф.,
Ростовский военный институт ракетных войск

Правила распределения моделей по узлам вычислительной сети

Обосновываются правила распределения моделей по узлам однородных и гетерогенных вычислительных сетей. Правила в качестве критериев включают сложность моделей и разнообразие, описываемое удельной энтропией и связанное с объемом и временем обмена. Показано, что зависимость частоты включения модели в определенное число вычислительных узлов от ранга сложности является фракталом. Приведены численные примеры распределения моделей.

Введение

Повышение производительности и снижение стоимости высокопроизводительных вычислительных систем, а также рост требований к ресурсам со стороны приложений приводят к тому, что технологии распределенных вычислений начинают широко использоваться как в научных лабораториях, так и в бизнесе.

Широкий спектр моделей параллельных компьютеров (SIMD, MIMD), сред параллельного программирования, включающих такие продукты, как стандарты MPI, HPF, модель параллелизма по управлению, отечественная система DVM, технологий типа CORBA, RMI, DCOM, отвечающих на ряд вопросов проблемы распределения вычислений, может создать впечатление завершенности задачи распределения. Практика и публикации свидетельствуют об ошибочности такого видения. В целом ряде приложений проблемным вопросом остается процедура распределения моделей по вычислительным узлам.

Часто задача распределения сводится к эвристикам равномерного распределения нагрузки среди составляющих вычислительного комплекса (балансировке загрузки), использования узлов комплекса с максимальной загрузкой. Такие традиционные эвристики-правила приводили бы к универсальным алгоритмам распределения. Но задача распределения во многих приложениях содержит элементы неопределенности вследствие динамичности конфигурации сети (число узлов, их архитектура и производительность), обусловленной непостоянством загрузки [1]. В связи с этим распределение, как правило, происходит на основе трудоемких алгоритмов прямого адаптивного поиска, априорной эвристической оценки времен вычисления и обмена. Отсюда необходимость синтезировать систему правил распределения [2, 3]. В данной работе на базе иерархического представления моделей обосновываются правила распределения моделей по вычислительным узлам в форме количественных критериев. Понятие "модель" здесь используется в классической интерпретации [4, 5]. Полагается, что моделируемые объекты представляются иерархической и масштабируемой структурой алгоритмически описанных элементов. Мощность множества таких объектов

достаточно велика. Это могут быть сложная техническая система, система передачи сигналов с управляемой кодовой структурой, функциональные модули рассредоточенной системы управления, программы пользователей и т. п. В частности, это могут быть подробно представленные в работе [5] системы из физически неоднородных объектов, описываемые топологически и физически с помощью направленных сигнальных графов (бонд-графов).

Правила формулируются для однородных и гетерогенных систем и позволяют определять как целесообразность, так и структуру распределения при заданной топологии вычислительной системы.

Правила распределений

Известно, что математическая модель физического объекта может быть представлена многоуровневой иерархической структурой. Она включает, например, на нижнем уровне такие элементы, как машинные инструкции, на среднем уровне — операции, и на высшем уровне — элементарные структурные модели (например, дифференциальные уравнения). На каждом уровне имеется определенное число элементов. Уровень определяется степенью сложности решений. Очевидно, что нижние уровни являются наиболее доступными для большинства компонентов вычислительной системы и задача распределения должна инициализироваться на верхних уровнях, на которых сложность существенно увеличивается. Полагается, что на i_k-м уровне сложность com_k пропорциональна уровню: $com_k = c(i_k - i_0)$. Здесь всем элементам уровня i присвоен одинаковый номер k, причем существует некоторый нижний уровень i_0, на котором сложность минимальна, незначительна. Между этими номерами существует некоторая функциональная зависимость $i_k(k)$, т. е. сложность $com_k = c[i_k(k) - i_0]$ есть к тому же и функция номера k. Параметр c может быть определен, например, как сложность при $i_0 = 0$ и $i_k(k) = 1$. Если сложность измеряется временем, то c — это время решения элемента первого уровня. Так получилось ранжирование по сложности всех элементов, нуждающихся в реализации на вычислителях.

Теперь следует определить частоты включения элементов в вычислительные узлы. Пусть мощность множества элементов всех уровней равна N. Тогда частоту φ_j включения j-го элемента можно определить отношением числа n_j вычислительных узлов, решающих этот элемент, к суммарному числу N_s элементов, решаемых всеми n вычислительными узлами: $\varphi_j = (n_j/N_s)$. В частном случае $N_s = N$, и этот параметр можно интерпретировать как максимально необходимое число вычислительных узлов в системе. Присваивая каждому элементу с номером k ранг r по значению частоты включения, несложно получить среднюю сложность решения одного элемента вычислителями системы как сумму от 1 до N по индексу следующего вида:

$$Com = \Sigma\varphi_r com_r = \Sigma\{c\varphi_r[i_r(r) - i_0]\}. \qquad (1)$$

Меньший ранг будет у элемента, имеющего большую частоту включения. Очевидно, $\Sigma\varphi_r = 1$. Сейчас время напомнить, что предпочтительна меньшая средняя сложность. Поэтому следует воспользоваться тем, что полученного вида суммы произведений обладают одним известным свойством: сумма принимает наименьшее значение при упорядоченных уменьшении первого сомножителя и увеличении второго. Следовательно, для получения минимальной сложности решений включения элементов вычислительными узлами должны быть такими, чтобы в выражении (1) номера i_r были упорядочены так же, как упорядочены ранги соответствующих им элементов. Это можно назвать первым правилом распределения. Сравнительно сложная его формулировка содержательно поясняется следующим образом: чем сложнее модель, тем меньшее число вычислительных узлов должно ее включать, т. е. среди меньшего числа вычислителей она должна быть распределена.

Но распределенные вычисления характеризуются не только сложностью решений. Одной из особенностей многопроцессорной вычислительной системы является сеть обмена, с помощью которой процессоры соединяются друг с другом или с памятью. В связи с важностью модели обмена для распределенных вычислений релевантными показателями для них являются объем и скорость обменных процедур. Отсюда целесообразно обратиться к выражению для удельной энтропии — энтропии, приходящейся на один модельный элемент. Удельная энтропия физически определяет среднюю степень разнообразия элементов, решаемых типичным вычислительным узлом системы. Очевидно, удельная энтропия равна $H = -\sum_{r=1}^{N} \varphi_r \log_2 \varphi_r$.

Естественно считать, что степень разнообразия и время обмена находятся в обратной зависимости (чем больше степень разнообразия, тем меньше объем обменных операций и время обмена). Поэтому удельная энтропия также является критерием качества распределения. На основе этого критерия второе правило распределения формулируется так: следует стремиться к большей степени разнообразия моделей, решаемых вычислительным узлом, т. е. к максимуму удельной энтропии при заданном N.

Стремление уменьшить сложность и увеличить удельную энтропию (уменьшить время обмена) приводит к обобщенному критерию оценки качества распределения в виде следующего отношения:

$$K(\varphi_r, r) = Com(\varphi_r, r)/H(\varphi_r). \qquad (2)$$

Отсюда третье правило распределения, таким образом, состоит в том, что распределение моделей по вычислительным узлам должно соответствовать минимуму $K(\varphi_r, r)$ — доли средней сложности вычислений, приходящейся на единицу удельной энтропии (2). Величина, обратная K, может определять эффективность распределения.

3. Фрактальность распределения

Фракталы — понятие, вошедшее в научную картину мира сравнительно недавно. Исследования, связанные с фракталами, меняют многие привычные представления о поведении природных и искусственных объектов во времени. Разрабатываемые на основе фракталов теории открывают новые возможности в различных областях знаний, в том числе в информационных и коммуникационных технологиях. В частности, фрактальность системы означает, что ее поведение отличает случайность в локальном и детерминированность в глобальном, т. е. оно характеризуется взаимодействием порядка со случайностью. Такое взаимодействие иногда трактуют как синтез упорядоченной структуры (порядка) комбинацией случаев. При этом порядок может проявляться в наличии некоторых тенденций поведения, таких, например, когда поведение системы, имея тренд на некотором промежутке времени, вероятнее всего сохранит его и в последующем, демонстрируя эффект долговременной памяти. Сохранение тенденции, тренда, связано с другой особенностью — с нестрого периодической цикличностью поведения (своеобразная спираль развития) систем, которая объясняется динамикой памяти системы. Известны и другие не менее интересные свойства фрактальных систем [7]. В связи с этим ниже рассматриваются отдельные варианты зависимостей частоты включения от ранга, приводящих к фрактальности.

Частоту включения φ_r можно найти при решении вариационной задачи Лагранжа [7]: $\min K(\varphi_r, r)$ при условии $\Sigma\varphi_r = 1$.

В первом, частном, но интересном для приложений, варианте, когда на последующем уровне число элементов удваивается по сравнению с предыдущим, сложность имеет вид:

$$Com = \sum_{r=1}^{n} \varphi_r \log_2\left(\frac{1+2r}{1+2r_0}\right).$$

Указанный характер изменения числа элементов от уровня к уровню подобен связи числа бифуркаций и номера сечения в логистических отображениях [8], которые используются при феноменологическом описании ряда процессов в технике, экономике, гидродинамике, электронике. Для логистического отображения характерно наличие взаимодействия двух противодействующих процессов: концентрации (пропорционального роста, экспоненциального развития) и рассеивания (эффекта тесноты). Концентрация и рассеивание часто характерны и для изменения числа элементов от уровня к уровню. Процесс концентрации при повышении уровня проявляется в росте, пропорциональном числу элементов, числа элементов. А эффект тесноты, обусловленный ограниченностью различных ресурсов или допустимых затрат на разработку моделей, приводит к уменьшению числа элементов. Указанное уменьшение в случае логистического отображения пропорционально квадрату числа элементов. По аналогии с биологами, экономистами и др. подобное можно объяснить повышением возможностей ресурсного конфликта (сопротивления среды) при увеличении вероятности встречи двух элементов на одном узле при росте числа элементов. Учитывая естественную грубость, структурную устойчивость [8] процесса распределения моделей, описанная гипотеза относительно характера двух рассмотренных эффектов, приводящих к логистической модели, может претендовать на релевантность. Более того, ее можно отнести к фундаментальным свойствам систем моделирования.

Задача Лагранжа в этом варианте сводится к уравнениям

$$\begin{cases} H\log_2\left(\dfrac{1+2r}{1+2r_0}\right) + Com(\log_2\varphi_r + 1) + \lambda H^2 = 0, \\ \sum\limits_{r=1}^{n} \varphi_r - 1 = 0, \end{cases}$$

из которых плотность распределения — частота распределения по вычислительным узлам — получается в виде фрактального закона Ципфа:

$$\varphi_r = \begin{cases} G(1+2r)^{-1/\alpha}, & r > r_0 \\ 1, & r \leqslant r_0 \end{cases},$$

где $G = (1+2r_0)^{1/\alpha}$, $\alpha = K = Com/H$.

Отсюда следует, что частота включения элементов в вычислительные узлы может быть фракталом. Это соответствует свойственным для сетевых процессов самоподобию, масштабной инвариантности статистических характеристик [9]. Здесь уместно указать на пример, когда система структурного моделирования представляется со-

вокупностью кластеров. Кластер иллюстрирует самоподобие тем, что, являясь отдельным элементом системы, обладает функциями всей системы в целом.

Несмотря на отмеченную выше значимость логистического отображения, его ограничения достаточно жесткие. Поэтому во втором варианте в развитие первого рассматривается случай, когда на последующем уровне по сравнению с предыдущим число элементов больше в m раз (геометрическая прогрессия). Решение задачи Лагранжа при этом приводит к фракталу вида:

$$\varphi_r = \begin{cases} G_1(1 + br)^{-\nu/\alpha}, & r > r_0 \\ 1, & r \leqslant r_0 \end{cases},$$

$$G_1 = B(1 + br_0)^{\nu/\alpha},\ \nu = \log_m 2,\ \alpha = K = Com/H.$$

Постоянные в выражении для частоты φ_r можно найти из следующих простых соотношений. Если число элементарных моделей равно N, число вычислительных узлов — n, то наибольшее возможное значение частоты будет $\varphi_r = (n - 1)/N$, ему соответствует наименьшее значение ранга $r = 1$ (наименьшая сложность). Значение ранга $r = 0$ соответствует, как было отмечено ранее, элементам, которые решаются на всех n узлах. Наименьшее значение частоты, очевидно, равно $\varphi_r = 1/N$, ему соответствует наибольшее значение ранга $r = N - 1$. Отсюда при заданном K величины G_1 и b несложно найти из нелинейной алгебраической системы вида:

$$(n - 1)/N = G_1(1 + b)^{-\nu/\alpha};$$

$$1/N = G_1[1 + b(N - 1)]^{-\nu/\alpha}.$$

Наконец, третий, наиболее простой, а потому, с учетом отмеченной грубости системы распределения, прагматичный частный вариант. Для него характерна линейная зависимость сложности решения одного элемента вычислителями системы и ранга этого элемента:

$$com_r = cr$$

или для средней сложности решения одного элемента $Com = \Sigma\varphi_r$

$$com_r = \Sigma c\varphi_r r.$$

Решение задачи Лагранжа в этом случае также приводит к фрактальной зависимости:

$$\varphi_r = B_2 r^{-1/\alpha}. \qquad (3)$$

Появление фрактала в этом варианте является следствием известного факта: вариант при малом числе уровней есть линейное приближение варианта с логистическим отображением.

Используя уже приведенную аргументацию относительно граничных значений рангов и частот, несложно получить выражения для констант полученной зависимости в виде:

$$B_2 = (n - 1)/N,\ 1/\alpha = H/Com = \ln(n - 1)/\ln(N).$$

Важно обратить внимание на то, что фрактальная размерность $1/\alpha$ равна усредненному обратному значению обобщенного критерия (2). Известно [6], что она является системообразующей характеристикой. В данном случае распределение моделей по вычислительным узлам определяется в основном значением отношения средней сложности к удельной энтропии.

Третий вариант позволяет просто провести количественные оценки. Все полученные соотношения для частот целесообразно нормировать. Нормирование заключается в обеспечении выполнения равенства $k \sum_{r=1}^{N} \varphi_r = 1$ с нормирующим коэффициентом k.

Приведенные три фрактальных выражения для частоты включения являются конструктивным инструментом решения задачи распределения по вычислительным узлам многопроцессорного комплекса элементов математической модели физического объекта. Алгоритм распределения включает назначение требуемых значений критерия K, т. е. средней сложности Com и энтропии H, и расчет частот включения φ_r, которые при известном общем числе элементов N определяют, сколько вычислительных узлов решают элемент заданного ранга r.

На рис. 1 приведен график зависимости частоты $\varphi_r(n, N)$ от масштаба системы — числа вычислительных узлов n и числа элементарных моделей N при $d = 1/\alpha = H/Com = 0{,}281$, что соответствует числу узлов 4 и числу элементарных моделей 50.

На рис. 2 показано распределение частоты $\varphi_r(r) = \Phi i$ ($i = 4, 5, 6$) по рангам при заданном числе элементарных моделей $N = 200$ и различном числе вычислительных узлов $n_1 = 10$, $n_2 = 30$, $n_3 = 100$ соответственно.

На рис. 3 показано распределение частоты $\varphi_r(r) = \Phi i$ ($i = 1, 2, 3$) по рангам при заданном

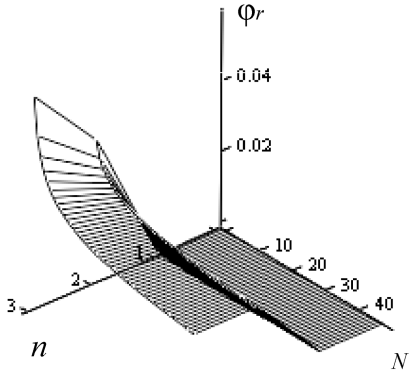


Рис. 1. Зависимость частоты от числа процессоров и числа элементарных моделей

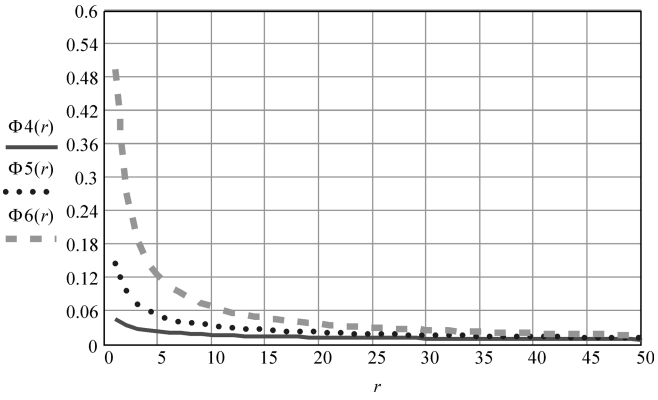


Рис. 2. Зависимость частоты от ранга при разном числе вычислительных узлов

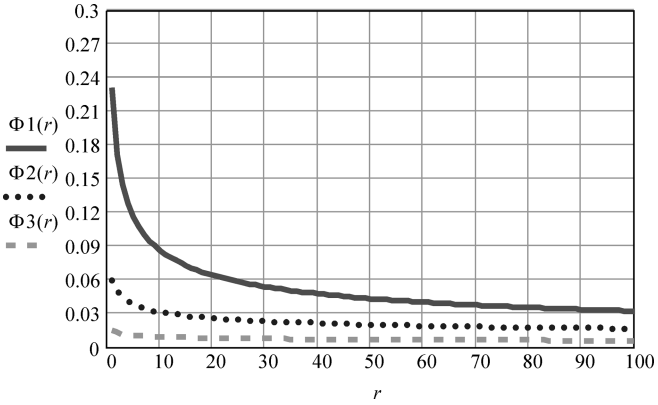


Рис. 3. Зависимость частоты от числа моделей при $n = 4$

числе вычислительных узлов $n = 4$ и различном числе элементарных моделей $N_1 = 13$, $N_2 = 50$, $N_3 = 200$ соответственно.

Решения (рис. 1—3) получены для результата (3) третьего варианта.

Графики свидетельствуют о следующих закономерностях. Существенно бóльшая частота включения — для элементарных моделей первого ранга, т. е. менее сложных моделей. Указанная существенность возрастает с увеличением фрактальной размерности $d = 1/\alpha = H/Com$. При $d < 0{,}2$ распределение носит равномерный характер и соответствует традиционному в практике распределенных вычислений правилу равномерной загрузки (балансировки загрузки). При фиксированном числе вычислительных узлов n увеличение числа элементарных моделей N приводит к уменьшению фрактальной размерности d и незначительному изменению частоты, что эквивалентно равномерной загрузке вычислительной системы. Поскольку параметр d определяется фрактальной размерностью, то последняя может считаться структурообразующей для многомашинного комплекса в смысле включения элементарных моделей в его компоненты.

Вариант гетерогенности вычислительного комплекса

Приведенные выше результаты в большей степени относятся к случаю однородных вычислительных узлов. Когда информационный комплекс гетерогенен (по производительности, памяти, скорости обмена), сохранение описанного выше подхода приводит к следующему алгоритму распределения. Пусть комплекс включает m групп вычислительных узлов. Каждая группа однородна по составу узлов. Компоненты группы имеют примерно одинаковые вычислительные возможности. Полагается, что большие производительность, память и скорость обмена обеспечивают всем узлам группы возможность решать элементарные модели повышенной сложности в сравнении с нулевым рангом. Например, если масштаб группы равен n_j и группа занимает в последовательности роста вычислительных возможностей j-е место (чем больше j, тем больше возможности, $j = 1, 2, ..., m$), то все узлы этой группы решают элементарные модели ранга j. Следовательно, после определения из физических соображений наибольшего значения частоты $\varphi_{\max}$ в этой группе, соответствующего наименьшего ранга $r_{\min}$, наименьшего значения частоты и наибольшего значения ранга, так же как и в однородном случае получаются два алгебраических уравнения, из которых находятся выражения для постоянных G_{1j}, b_j, d_j, B_{2j}. Например, для третьего случая

$$d = [\ln(\varphi_{\min}/\varphi_{\max})]/[\ln(\varphi_{\min}/\varphi_{\max})];$$

$$B = (\varphi_{\max}\varphi_{\min} r_{\min}^{d} r_{\max}^{d})^{0,5}.$$

В этих выражениях индекс j для упрощения записи предполагается по умолчанию.

Теперь алгоритм распределения элементарных моделей описывается следующим образом. Распределение начинается с группы узлов наибольших возможностей, т. е. с группы $j = m$. В группе распределяется N_m элементарных моделей, соответствующих рангам, при которых значения частот близки: $(\varphi_i - \varphi_{i-1}) \leqslant \Delta$, где Δ — допуск. Оставшиеся $(N - N_m)$ передаются в группу $j = (m - 1)$ с меньшими возможностями, где процедура распределения повторяется и т. д. Значение допуска определяется из требования целого числа занятых узлов в группе.

Для иллюстрации работоспособности предлагаемого подхода и характера получаемых решений на графике (рис. 4) не приведена зависимость частоты включения (на графике $\Phi(r)$) от ранга (сложности) для варианта четырех гетерогенных групп информационного комплекса — $m = 4$. Каждая группа состоит из однородных вычислительных

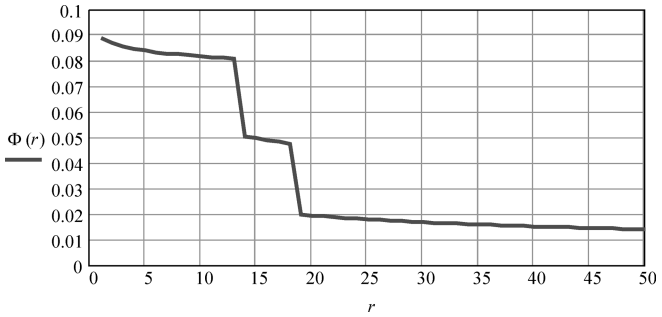


Рис. 4. Зависимость частоты включения от ранга при $m = 4$

узлов. Причем общее число элементарных моделей $N = 50$. Распределение этих моделей получается следующим. Модели ранга 19...50 распределены в двух группах с наиболее производительными узлами, модели ранга 18...14 — в следующей по производительности группе, и модели ранга 1...13 — в последней группе узлов.

Заключение

Правила распределения моделей по вычислительным узлам сводятся к следующему.

1. Чем сложнее модель, тем меньшее число вычислительных узлов должно ее включать.

2. Следует стремиться к большей степени разнообразия моделей, решаемых вычислительным узлом.

3. Распределение моделей по вычислительным узлам должно соответствовать минимуму доли средней сложности вычислений, приходящейся на единицу удельной энтропии.

Первые два из сформулированных правил содержательно не связаны между собой. Но формально через частоту $\varphi_r(r)$ зависимы. Можно проектировать систему распределения по первому правилу, и тогда наиболее сложные задачи будут решаться на небольшом числе вычислительных узлов. Если создавать систему по второму правилу, то распределение будет характеризоваться решением каждым вычислительным узлом как можно большего числа разных моделей. Третье правило по отношению к первым двум является интегральным. В системе распределения, построенной на основе этого правила, достигается минимальная средняя вычислительная сложность, приходящаяся на единицу удельной энтропии. Какое из правил использовать — это отдельная самостоятельная задача. В ней можно изучать стоимость работ, производительность, коллективные решения и т. п.

В целом синтезированные правила могут использоваться при развитии архитектуры распределенного моделирования HLA, предназначенной для обеспечения взаимодействия территориально распределенных участников моделирования, объединяя разноцелевые системы, технологии различных поколений, программные продукты и платформы различных фирм в единое окружение.

Кроме того, практически являясь составной частью диспетчеризации вычислительного процесса, правила представляют интерес для целей наращивания производительности системы распределенных вычислений, разработки и развития принципов организации функционирования систем моделирования сложных объектов.

Список литературы

1. **Крюков В. А.** Разработка параллельных программ для вычислительных кластеров и сетей. // Информационные технологии и вычислительные системы. 2003. № 1—2. С. 42—61.

2. **Каляев И. А.** Использование принципов коллективного принятия решений при распределении потока задач в компьютерных сетях. // Информационные технологии. 2002. № 9. С. 31—37.

3. **Коробкин В. В., Чернов Е. И.** Алгоритмы распределения задач по компьютерам вычислительной сети // Высокопроизводительные вычислительные системы (ВПВС—2005). Материалы научной международной школы, 2005 г. Таганрог: Изд-во ТРТУ, 2005. С. 133—137.

4. **Самарский А. А., Михайлов А. П.** Математическое моделирование: Идеи. Методы. Примеры. М.: Наука, 1997.

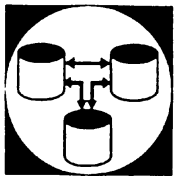
5. **Норенков И. П.** Основы автоматизированного проектирования. М.: Изд-во МГТУ им. Н. Э. Баумана, 2000.

6. **Коханенко И. К.** Фрактальная размерность как критерий системной устойчивости // Известия РАН. Теория и системы управления. 2003. № 2.

7. **Гельфанд И. М., Фомин С. В.** Вариационное исчисление. М.: Физматгиз, 1961.

8. **Малинецкий Г. Г.** Введение в нелинейную динамику. М.: Эдиториал УРСС, 2000.

9. **Заборовский В. С., Городецкий А. Я.** Фракталы и степенные законы: модели процессов в компьютерных сетях // Архив семинара "Современные сетевые технологии" в МГУ, 2003.



УДК 519.6

И. А. Кантор,
Московский технический университет
связи и информатики

Многокритериальные ограниченные сортирующие выборки в реляционных СУБД. Метод деревьев битовых карт

Рассматриваются индексные структуры в реляционных СУБД, позволяющие эффективно осуществлять выборки, где кроме сильно варьирующегося набора условий на записи требуется сортировка по одному или нескольким атрибутам. Такие выборки применяются как в системах принятия решений, так и в задачах оптимизации, хранилищах данных.

Перечислен ряд известных подходов и предложен новый подход, основанный на специализированной индексной структуре — "дереве битовых карт".

Даны оценки числа операций и объема новой индексной структуры, применимые в стоимостных оптимизаторах.

Введение

Реляционные базы данных широко используются для больших хранилищ данных и быстрых операций поиска по ним. В статье рассматривается индексная структура для оптимизации важного частного случая запросов к хранилищам, который можно определить как "ограниченные сортированные многокритериальные выборки".

Такие выборки проводятся из одной большой таблицы и обладают следующими свойствами:

- каждое условие запроса ограничивает значение одного поля;
- условия комбинируются логическими операциями И/ИЛИ/НЕ и т. п.;
- набор условий часто меняется;
- записи требуется возвращать в отсортированном порядке;
- выборка возвращает не все записи, удовлетворяющие условиям, а с N-й по M-ю в порядке, задаваемом сортировкой.

Эти запросы естественным образом появляются при использовании поисковых интерфейсов к хранилищам, когда человек выбирает условия на записи, сортировку по дате, цене, удаленности и т. п. и желает видеть не все записи (которых могут быть миллионы), а некоторое небольшое окно.

Число полей в таблице варьируется исходя из предметной области, например, при поиске товаров может быть лишь несколько полей, а в задачах астрономии — 100 полей и более. Например, для поиска товаров это может быть 100 самых дешевых предложений, в железнодорожных перевозках — 50 ближайших маршрутов.

В системах принятия решений используются аналогичные выборки. Число атрибутов, на которые налагаются условия, также широко варьируется. Например, в задачах астрономии их может быть 100 и более.

Сформулируем задачу поиска формально и рассмотрим существующие пути ее выполнения.

Задача ограниченной сортированной многокритериальной выборки

Будем рассматривать выборки из отношения (или таблицы) T с заголовком $<A_1, ">, <A_2,">, ..., <A_{fieldsCount}, " >$.

Элементы множества кортежей (записи) обозначим T_k. Каждый триплет кортежа имеет вид $<A_i, ", v_i>$.

Таким образом, если не оговорено особо, то в таблице находятся $|T|$ записей, каждая запись состоит из $|A| = fieldsCount$ натуральных чисел — значений атрибутов A_1, ..., $fieldsCount$. Упрощенным обозначением кортежа является запись в виде последовательности значений атрибутов: $\{v_1, ..., v_{fieldsCount}\}$.

Выбор $\mathbb{N}$ в качестве домена сделан для простоты рассуждений, которые также применимы ко всем предметным моделям, в которых домен преобразуется к $\mathbb{N}$. Это, например, дата, булевы значения и 32-битные числа с плавающей точкой в формате IEEE-754.

Будем обозначать $V_i = \cup v_i$ — множество всех значений атрибута A_i, $V_i \in \mathbb{N}$, $\forall i$.

Определение 1. *Задачей (или условием задачи) ограниченной сортированной многокритериальной выборки* называется упорядоченный набор

$$Q = (Cond, f, c, s, Skip, Count, Top),$$

где

- $Cond = \{C_i(x), i = 1, ..., n\}$ — упорядоченный набор условий (предикатов) равенства $EqCond \subset Cond$ и неравенства или нахождения в промежутке $RangeCond \subset Cond$: $Cond = EqCond \cup RangeCond$;

- $f : \{Cond_1, ..., Cond_n\} \to \{0, 1\}$ — логическая функция над условиями;
- c — перестановка, которая задает отображение номера *Cond* на номер атрибута A и, таким образом, привязывает условие к атрибуту. Одному атрибуту может соответствовать несколько условий;
- $s \in 1, ..., |A|$ — номер атрибута сортировки;
- *Skip* $\in \mathbb{N}$ — число пропускаемых записей;
- *Count* $\in \mathbb{N}$ — размер окна поиска;
- *Top* $\in \mathbb{N}$ — число максимального предпросмотра.

Определение 2. Запись $\{v_1, ..., v_{fieldsCount}\}$ называется *удовлетворяющей критериям задачи*, если $f(Cond_1(v_{c(1)}), ..., Cond_n(v_{c(n)})) = 1$.

Определение 3. *Полным множеством результатов* для задачи Q является множество *ResultsAll* упорядоченных по атрибуту A_s записей, удовлетворяющих критериям задачи.

Определение 4. *Искомым множеством*, или *окном поиска* для задачи Q называется упорядоченное множество $ResultsWindow = (ResultsAll_{Skip + 1}, ..., ResultsAll_{Skip + Count})$. Будем говорить, что окно поиска полное, если $|ResultsWindow| = Count$.

Определение 5. *Числом предпросмотра* для задачи Q называется число $CountMore = \min(|ResultsAll| - (Skip + Count), Top)$.

Определение 6. *Решением* задачи Q называется упорядоченная пара (*ResultsWindow*, *CountMore*), где *ResultsWindow* — искомое множество, а *CountMore* — число предпросмотра.

Наличие чисел *Skip*, *Count*, *Top* в условии задачи дает возможность организовать постраничный вывод результатов.

Можно упростить задачу выборки, объединив эти числа в одно

$$TotalCount = Skip + Count + Top$$

и определив искомое множество как

$$(Records\ All_1, ..., RecordsAll_{\min(|RecordsAll|,\ TotalCount)}).$$

С одной стороны, из искомого множества такой упрощенной задачи можно получить решение задачи Q, с другой — отдельное выделение *Skip* и *Top* позволяет рассмотреть специальные алгоритмы оптимизации, так как доступ к данным можно реализовать более эффективно, если важно лишь общее число, а не атрибуты записей.

OLAP-системы и пользовательские интерфейсы требуют решать задачи ограниченной сортированной многокритериальной выборки в реальном времени, что обусловливает большие требования к производительности.

Способы решения задачи Q

Для описанной задачи, как и для многих других задач выборки данных, невозможно предложить универсальный метод получения решения. Существуют разные методы с разными характеристиками. Кратко перечислим различные способы решения, некоторые из которых уже реализованы в известных реляционных СУБД (РСУБД).

Пересечение битовых карт. Применение битовых индексов в большинстве известных РСУБД (MSSQL, DB2, Oracle) позволяет лишь выбрать записи, на которых верна функция f, но для нахождения искомого множества *Records* требуется дополнительная операция сортировки.

Для поиска с помощью битовых карт создается по одной битовой карте длиной $|T|$ для каждой пары (атрибут, значение). При поиске на первом шаге с помощью логических операций выбирается полное множество результатов, а на втором — происходят сортировка и выборка *ResultsWindow*.

Традиционно для этого используются кодированные битовые индексы[1] [1], битовые индексы для объединения[2] [2], индексы на основе промежутков[3] [3] и многие другие их виды [4].

Сканирование Б-дерева. Создается набор индексов на основе Б-дерева с одинаковым префиксом A_S и различными комбинациями остальных атрибутов. Например, $I_{2, 1} = (A_S, A_2, A_1)$, $I_{3, 1, 2} = (A_S, A_3, A_1, A_2)$, ... Просмотр любого такого Б-дерева в прямом порядке [5] будет выдавать записи, заведомо отсортированные по A_S.

На основе статистических данных и условия задачи выбирается наиболее подходящий индекс и просматривается в направлении прямого обхода, пока не будет набрано $\min(|ResultsAll|, Skip + Count + Top)$ записей.

Индекс выбирается таким образом, чтобы делать отсечения ветвей на возможно более высоком уровне. При обходе и хранении Б-дерева применяются многочисленные оптимизации, сжатие по префиксу и т. п.

Метод сортированных битовых карт. Построение индекса в этом методе состоит из двух шагов:

1) каждой записи присваивается номер, соответствующий ее порядку по атрибуту A_s;

2) битовые карты по атрибутам поиска строятся в порядке п. 1.

При поиске вычисление функции f проводится применением операций AND/OR/NOT к битовым картам, от начала вектора и до конца либо пока не наберется $Skip + Count + Top$ записей.

Этот метод удобен в системах однократной загрузки данных или там, где добавление записи в поиск может подождать полной перестройки битовых карт.

Применяются также комбинации упомянутых методов.

Дерево битовых карт

Для решения задачи сортированной ограниченной многокритериальной выборки предлагается новая индексная структура, в которой Б-де-

[1] В англоязычной литературе "encoded bitmap index".
[2] В англоязычной литературе "bitmap join index".
[3] В англоязычной литературе "interval bitmap index".

ревья применяются для поддержки обновлений в сортированных битовых картах.

В ней используются Б-деревья с данными, т. е. такие, где каждая запись (или элемент) является парой (k, d), где k — ключ, возможно, составной, по которому происходит упорядочение элементов дерева, а d — некоторые данные, соответствующие ключу k.

Для Б-дерева *tree* данные по ключу k обозначаются как $d = tree(k)$.

Пусть *map* — битовая карта. Тогда $Len(map)$ обозначает длину битовой карты, $BitCount(map, 1)$ — число единиц, т. е. векторную норму, $BitCount(map, 0)$ — число нулей.

Структура индекса. Индексная структура состоит из нескольких частей. Первая из них — *основное Б-дерево SearchTree*.

Каждому ключу K_i основного Б-дерева соответствует значение $K_{i,\ sort} \in V_s$, причем отображение ключей в V_s инъективно, т. е. не должно быть более одного ключа в дереве с данным значением атрибута.

В элементе, который задает ключ K_i, хранятся данные о записях таблицы со значениями v_s, такими что

$$K_{i,\ sort} \leqslant v_s < K_{i+1,\ sort}. \qquad (1)$$

Для последнего ключа правая часть неравенства отсутствует. У первого узла значение ключа нулевое.

Одна запись таблицы может находиться лишь в данных одного ключа K_i, поэтому будем говорить, что запись *принадлежит* этому ключу. Промежуток $[K_i, sort, K_{i+1,\ sort})$ или, для последнего ключа, $[K_i, sort, \infty)$ называется *промежутком значений ключа*.

Данные элемента основного дерева состоят из нескольких компонентов.

1. Набор пар (A_m, D_m), $m = 1, ..., fieldsCount$, где A_m — имя атрибута, а D_m — *Б-дерево значений атрибута*. Ключами D_m являются значения атрибута $v \in V_m$, а данными — битовые карты. Каждому ключу v соответствует ровно одна битовая карта, которая содержит столько единиц, сколько записей таблицы имеют значение $v_m = v$.

В отличие от традиционных битовых карт, по номеру позиции нельзя восстановить саму запись без дополнительной информации.

2. *Б-дерево данных RowidTree* имеет своими элементами пары (p, d), $d = (rowid, s)$, где p — позиция в битовых картах, *rowid* — идентификатор записи в таблице, а s — значение v_s этой записи. Каждая позиция появляется в дереве не более одного раза.

Отображение позиции в идентификатор, задаваемое деревом *RowidTree*, поддерживается всеми битовыми картами, т. е. во всех битовых картах-компонентах одна запись находится на единственной, одинаковой позиции. Поэтому имеет место следующая лемма.

Если запись T_{rowid} с идентификатором *rowid* из $d = RowidTree(p)$ состоит из упорядоченного набора значений $\{v_1, ..., v_{fieldsCount}\}$ и принадлежит данному узлу *SearchTree*, то соответствующая битовая карта имеет 1 на позиции p:

$$BitRecords_{k,\ v}(p) = 1 \Leftrightarrow v_k = v.$$

Таким образом, логическим операциям между битовыми картами соответствуют логические операции между условиями поиска, и обратно.

Например,

$$(BitRecords_{k,\ a}(p) \vee BitRecords_{k,\ b})(p) = 1 \Leftrightarrow (v_k = a) \vee (v_k = b);$$

$$(BitRecords_{a,\ b}(p) \wedge BitRecords_{c,\ d})(p) = 1 \Leftrightarrow (v_a = b) \wedge (v_c = d).$$

Значение s используется в алгоритме поиска, который будет приведен ниже.

3. *Б-дерево позиций OrderTree* состоит из элементов вида $(rowid, p)$ и хранит отображение идентификатора записи в позицию в битовых картах. Оно необходимо для эффективных операций обновления и удаления.

4. *Битовая карта свободных мест* $Extent_k$ содержит 1 на тех позициях, которые есть в *RowidTree*, и 0 — на остальных.

Карту свободных мест можно использовать, например, для того, чтобы быстро найти те позиции, которым еще не сопоставлена ни одна запись, так как поиск нулей в сжатой битовой карте можно реализовать гораздо эффективнее, чем поиск промежутков среди ключей Б-дерева.

Вставка записи в основное дерево поиска. Для вставки новой записи узел ищется в порядке убывания приоритетов:

а) не увеличивать число элементов *SearchTree*;

б) не увеличивать число записей в узле, единая верхняя граница — *BitmapSize*;

в) целостность индексной структуры.

То есть алгоритм вставки в общих чертах такой.

1. Найти место в подходящем элементе *SearchTree*.

2. Если там *BitmapSize* или более записей, то добавить новый элемент, нарушив п. *а*).

3. Если это не удалось, то вставить запись, нарушив п. *б*).

Как следствие, в элементе основного дерева может быть больше *BitmapSize* записей только тогда, когда его нельзя разделить, т. е. все записи в нем имеют одинаковое значение A_s.

Поиск по дереву битовых карт. Внутрь каждого узла дерева записи добавляются хаотически. В отличие от сортированных битовых карт, нельзя сказать, у какой записи v_s меньше, руководствуясь их позициями.

Вместе с тем, если записи принадлежат разным элементам основного дерева, то отношение порядка между ними такое же, как между ключами этих элементов.

Поэтому поиск можно осуществлять, сканируя данные *SearchTree* в порядке возрастания ключей K_i.

Из каждого элемента дерева узла извлекаются нужные $BitRecords_{k, v}$ и, с помощью логических операций над ними, получается битовая карта $resMap_i$ записей, удовлетворяющих критериям задачи.

Сканирование следует завершить, как только

$$\sum_i BitCount(resMap_{i,\ 1}) \geqslant Skip + Count + Top$$

Полученные карты $resMap_i$ обладают двумя важными свойствами:

- $resMap_i$ имеет 1 на позициях p таких, что в соответствующем дереве позиций $RowidTree_i$: $rowid = RowidTree_i(p)$ — идентификатор записи, удовлетворяющей критериям задачи;
- наборы записей, соответствующих картам $resMap_k$, $resMap_l$, упорядочены между собой так же, как k и l.

Таким образом, можно отбросить те $resMap_k$, записи в которых заведомо не попадут в окно поиска:

$$\sum_{i=1}^{k} BitCount(resMap_i, 1) \leqslant Skip.$$

Затем собрать записи из карт $resMap_l$, которые могут попасть в окно поиска, т. е. таких, где $l = 1, ..., k + 1$, число k выбирается из условия:

$$Skip < \sum_{i=1}^{k} BitCount(resMap_i, 1) \leqslant Skip + Count.$$

Среди этих записей могут быть лишние, так как выбор идет с точностью до номера битовой карты. Поэтому они сортируются и дают искомое множество. Для сортировки используется поле s из *RowidTree*, а для получения данных из таблицы — *rowid* из *RowidTree*.

Оценки числа сортируемых записей. Так как записи внутри битовых карт расположены хаотично, то наилучший случай — когда сортируются все результаты поиска и только они. Этот случай достигается, когда все окно поиска находится в одной или нескольких битовых картах, причем каждая запись из битовой карты входит в окно.

Нижняя оценка сортируемых записей: 0, причем она достигается только тогда, когда для некоторых чисел k, $l \in \mathbb{N}$ выполняются условия

$$Skip = \sum_{i=1}^{k} BitCount(resMap_i, 1);$$

$$Count = \sum_{i=k+1}^{l} BitCount(resMap_i, 1);$$

$$l - k = Count.$$

Эти условия необходимы и достаточны для того, чтобы записи были распределены по одной на битовую карту $resMap_i$, т. е. $BitCount(resMap_i, 1) = 1$, $\forall i = k + 1, ..., l$. А записи в различных битовых картах отсортированы друг относительно друга по построению.

Верхняя оценка числа сортируемых записей по размеру окна поиска:

$$2 * (BitmapSize - 1) + |ResultWindow|.$$

Эта оценка естественным образом появляется при анализе числа "лишних" записей, т. е. тех, которые участвуют в процессе сортировки, но не попадают в результаты поиска.

Число сортируемых записей максимально тогда, когда максимально число лишних записей в первой и последней битовых картах, дающих окно поиска.

Лишних записей не может быть $\geqslant BitmapSize$, так как иначе соответствующая битовая карта была бы отброшена при поиске. Но если битовые карты полностью заполнены 1 и содержат по одному результату, а остальные записи — лишние, то $2 * (BitmapSize - 1)$ записей будут лишними, откуда следует оценка.

Из этой верхней оценки получается верхняя оценка числа сортируемых записей по параметрам задачи: $2 * (BitmapSize - 1) + Count$.

Дальнейшие оценки даны в байтах и рассчитаны исходя из общепринятой физической структуры Б-дерева на машинах семейства x86.

Размер индексной структуры. Формула для размера:

$$32\,|T| + \frac{|T|}{4}\left(1 + \sum_{i=1}^{fieldsCount} |V_i| compress\left(\frac{1}{V_i}\right)\right).$$

Здесь $compress(z)$ — средний коэффициент сжатия битовой карты в зависимости от ее плотности z.

Число операций при поиске. Число операций при поиске складывается из следующих компонентов:

- $Elem * Map_f$ чтений битовых карт длиной *BitmapSize*;
- $Elem * Cost_f$ логических операций над битовыми картами длиной *BitmapSize*;
- *ResultsPerElement* поисков на каждое из *Elem* Б-деревьев высотой $RowidTree_h$;
- *Elem* операций сортировки *ResultsPerElement* чисел.

Здесь

$$Elem = \left\lceil \min(Skip + Count, |ResultsAll|) \frac{|SearchTree|}{|ResultsAll|} \right\rceil - \left\lceil \min(Skip, |ResultsAll|) \frac{|SearchTree|}{|ResultsAll|} \right\rceil,$$

Map_f, $Cost_f$, $|ResultsAll|$ вычисляются исходя из статистики по данным и условий задачи Q;

$$ResultsPerElement = \frac{|ResultsAll|}{|SearchTree|};$$

$$RowidTree_h = \left\lceil \log_{\frac{BlockSize}{18}} BitmapSize \right\rceil.$$

Заключение

Предлагаемая индексная структура может быть с успехом использована наравне с существующими алгоритмами индексирования данных.

Ее минусами являются классические недостатки битовых индексов и тот факт, что дерево битовых карт позволяет проводить выборки лишь по одному, жестко заданному атрибуту сортировки.

Стоимостные оценки для традиционных подходов широко известны, и их сравнение с рассмотренными стоимостными оценками для дерева битовых карт позволяет оценить эффективность применения дерева битовых карт по статистике и условиям выборки и, таким образом, использовать его для тех запросов, когда оно даст выигрыш в производительности.

Список литературы

1. **Buchmann A., Wu M. C.** Encoded bitmap indexing for data warehouseman // In 14th ICDE. 1998.
2. **Graefe G. O'Neil P.** Multi-table joins through bitmapped join indices // In SIGMOD Record, 1995. 24 (3).
3. **Yu P. S., Wu K. L.** Range-based bitmap indexing for high cardinality attributes with skew // Technical report, IBM Watson Research Center. 1996.
4. **Sarawagi S.** Indexing olap data // In Bulletin of the Technical Committee on Data Eng. 1997. Vol. 20. N 1.
5. **Яблонский С. В.** Введение в дискретную математику. М.: Высш. шк., 2002.

УДК 519.687.1

Р. С. Самарев,
НИС НУК ИУ МГТУ им. Н. Э. Баумана

Организация внутризапросного параллелизма в унаследованных СУБД

Рассматривается метод модернизации не параллельных СУБД посредством организации внутризапросного параллелизма на уровне внутренних операций, контролируемое выполнение которых обеспечивает этот метод. В частности, метод применен для снижения времени отклика выполнения запросов в объектной СУБД ODB — Jupiter в режимах неполной загрузки. Разработанный метод может применяться в широком классе программных систем.

Введение

Проблемы параллельного программирования традиционно рассматриваются в контексте вычислительных систем (ВС) с массовым параллелизмом. В то время как в последние годы становятся все более распространенными недорогие многопроцессорные высокопроизводительные ВС с архитектурой x86. Разработанные несколько лет назад (унаследованные) программные продукты (ПП) для данного класса ВС часто неэффективно используют предоставляемые ресурсы ВС, что могло бы снизить время их выполнения и таким образом увеличить пропускную способность системы в ненасыщенных режимах работы. В полной мере это относится и к большинству СУБД, разработанных для ВС с архитектурой x86, которые не рассчитаны на параллельное выполнение одиночных запросов (Microsoft Access, MySQL до 5.1, SQLite, InterBase/Firebird).

Теоретические вопросы параллельного программирования в настоящее время хорошо проработаны для ВС специального назначения, а также для случая монопольного выполнения ПП на ВС [4]. Однако эти методы могут быть применены для ВС общего назначения в условиях совместного использования ресурсов ВС несколькими ПП далеко не во всех случаях и не всегда без изменений. Часто в работах, посвященных параллельным вычислениям, рассматриваются идеальные случаи, когда выполняемая программа работает в монопольном режиме на некотором вычислительном комплексе, что позволяет планировать ее выполнение, считая ресурсы ВС постоянными [3].

Выполнение поступающих запросов к СУБД предполагает выполнение ряда внутренних операций СУБД. При планировании параллельного выполнения внутренних операции требуется учитывать их разнородность по отношению к используемым ресурсам ВС. В зависимости от реализации СУБД операции можно разделить на следующие пересекающиеся группы: с интенсивным обращением к дисковым накопителям по чтению/записи (одна БД может размещаться на нескольких дисках); с интенсивным использованием ОЗУ (для обработки промежуточных результатов); с интенсивным использованием процессоров. В распределенной системе также есть ограничения на одновременный доступ по каналам связи. Проблема непостоянства доступных ресурсов ВС на момент планирования выполнения операций внутри запроса обусловлена тем, что в большинстве малых и средних коммерческих организаций сервер ВС используют одновременно в качестве сервера СУБД, WEB, файл-сервера и сервера печати, что

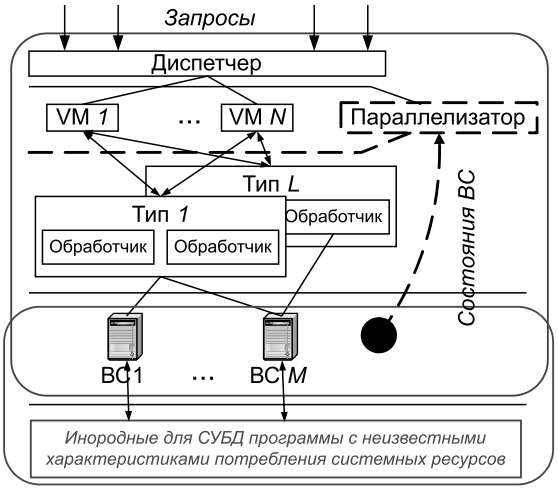


Рис. 1. Обобщенная схема обработки запросов в параллельной СУБД

требует динамически отслеживать реально доступные в настоящий момент ресурсы.

Данная работа рассматривает особенности организации разработанного автором программного модуля — параллелизатора в широко распространенном классе ВС с общей памятью типа "SE" [10] семейства x86 (Intel Xeon, AMD Opteron) с учетом многопараметрического контроля параллелизма и динамического контроля состояния ВС в условиях не монопольного выполнения программы на ВС. Разработанный параллелизатор был использован для модификации ОСУБД ODB-Jupiter (НПЦ "ИНТЕЛТЕК ПЛЮС") [1, 2] с целью обеспечения не конвейерной внутризапросной межоперационной схемы параллельного выполнения.

Обобщенно можно представить схему обработки запросов в СУБД с программным модулем — параллелизатором следующим образом (рис. 1).

Запросы пользователей поступают на вход диспетчеру БД, после чего поступают в модули VM, обеспечивающие трансляцию, преобразование, контроль порядка выполнения запроса и подготовку результата для возврата пользователю. При этом реальное выполнение операций обеспечивается типизированными обработчиками запросов. В общем случае обработчики могут размещаться на различных ВС, однако такой вариант в этой статье не рассматривается. На всех этапах выполнения запроса процесс передачи управления контролируется программным модулем — параллелизатором. Анализ состояния ВС проводится средствами, позволяющими оценить состояние доступных ресурсов ВС в целом, с учетом их использования сторонними процессами.

1. Основные формы параллелизма

При рассмотрении вопроса реорганизации последовательных СУБД, а также для уточнения используемой здесь терминологии нельзя не отметить формы параллелизма в СУБД, рассмотренные в работах [6, 8, 9]. Применяется как межзапросный параллелизм (параллельное выполнение нескольких запросов при их последовательном внутреннем выполнении), так и различные виды внутризапросного параллелизма, обеспечивающие как параллельное выполнение одной и той же операции сервера СУБД над разными данными (внутриоперационный параллелизм), так и параллельное выполнение разных микроопераций с разными данными (межоперационный параллелизм). Кроме того, различают конвейерные и не конвейерные методы обработки.

В большинстве коммерческих параллельных СУБД для ВС с массовым параллелизмом для обработки больших объемов данных исторически применяется конвейерный принцип. С позиции способа параллельного выполнения различают два основных принципа — скобочный (*bracket model*) и операторный (*operator model*) [7, 8]. В работе [7] к тому же рассматривается прототип программно-аппаратного комплекса с иерархической аппаратной архитектурой СУБД "ОМЕГА" и многопроцессорного вычислительного комплекса МВС 100/1000, обеспечивающей внутризапросную конвейерную параллельную обработку каждого запроса. Однако массовое использование таких комплексов невозможно по причине высокой стоимости оборудования и ограниченной области применения СУБД такого уровня производительности.

Для модификации унаследованных последовательных СУБД пригодны неконвейерные методы обработки, поскольку изменение программ, реализующих последовательные алгоритмы в параллельные конвейерные, является задачей по сложности соизмеримой, а часто и превосходящей разработку конвейерных программ с нуля. В то же время неконвейерный параллелизм часто является достаточным для улучшения характеристик таких СУБД, поскольку преимущества конвейера можно получить лишь в случае, когда существует возможность обработки части данных и последовательной передачи результатов между этапами обработки всего запроса, а также в условиях ограничения ресурсов, например, когда объем требуемого ОЗУ для выполнения операции над всем объемом данных сразу превышает объем доступного ОЗУ.

Итак, модификация ранее разработанной непараллельной СУБД для реализации внутризапросной межоперационной схемы выполнения включает: выбор метода создания параллельного физического плана выполнения одиночного запроса; анализ взаимонезависимых операций сервера БД и их составляющих в контексте обращения к общим ресурсам ВС; выбор и реализацию метода организации межоперационного параллелизма.

2. План выполнения запросов

Трансляция запроса пользователя в СУБД в общем случае происходит следующим образом. Вход-

ной запрос, например, на языке высокого уровня SQL преобразуется в логический план выполнения, являющийся внутренней формой его представления. Логический план транслируется в физический план выполнения, операции которого в точности соответствуют тем внутренним операциям, которые выполняет сервер СУБД. Будем называть их микрооперациями. Под физическим планом понимается строгая последовательность микроопераций, которая должна быть выполнена для выполнения всего запроса.

Важно отметить, что высокоуровневый язык запросов — не единственный интерфейс взаимодействия с СУБД. Часто предусматривают прямой доступ к данным посредством ряда специальных операций, что, в первую очередь, относится к объектным СУБД, где по спецификации ODMG 3.0 [11] модификация данных осуществляется только так. Независимо от того, какой интерфейс доступа используется, доступ к данным осуществляется посредством набора внутренних микроопераций, специфичных для каждой конкретной СУБД. Микрооперации обеспечивают элементарную выборку данных, пересечение списков ключей или данных, выполнение каких-либо более сложных операций. Под данными подразумеваются объекты — результаты выборок, поиска, сортировки.

Рассматривая физический план как линейный процесс, основной задачей распараллеливания на момент его создания становится выявление взаимонезависимых операций и микроопераций. Формальные методы анализа взаимозависимостей в программах приводятся в работе [4]. Кроме того, при динамическом планировании порядка параллельного выполнения операций можно воспользоваться методом польской инверсной записи [3].

На рис. 2 представлены схема параллельного физического плана и предлагаемая обобщенная схема его обработки с помощью модуля-параллелизатора. Для того чтобы обеспечить параллельное межоперационное выполнение, разделим выполнение операции на два этапа. Первый этап — асинхронный — постановка заявки на выполнение конкретной операции. Второй этап — синхронный — запрос результатов выполнения этой заявки. Следовательно, линейный управляющий процесс выполнения физического плана контролирует лишь порядок выполнения, обеспечивая максимально раннюю постановку заявок на выполнение операций и максимально позднее получение результатов их выполнения, а задача параллельного выполнения операций решается специализированным модулем-параллелизатором.

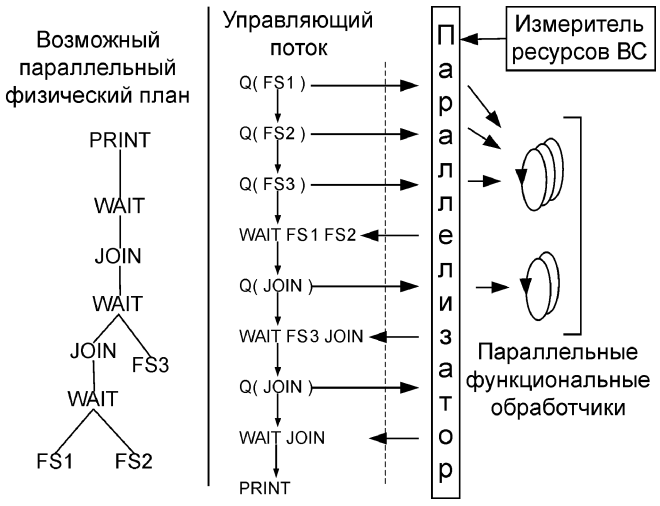


Рис. 2. Параллельное выполнение запроса

3. Параллелизатор

Рассмотрим реализацию унифицированного программного модуля — параллелизатора для ВС с общей памятью типа "SE" [10], который обеспечивает выполнение операций с максимально возможной степенью параллельности, контролируя при этом порядок выполнения и контроль состояния ВС. В состав параллелизатора входят следующие программные модули: модуль управления обработчиками; планировщик, который принимает решение о постановке заявки на выполнение; модуль корректора, обеспечивающий контроль текущего состояния вычислительной системы.

Модуль управления обработчиками. Основной единицей данных для параллельного выполнения операций является сообщение, в рамках которого формируются заявка на выполнение и результат. Сообщение содержит все необходимые параметры для выполнения операции и идентификаторы для возврата результатов. Логическим следствием являются очереди сообщений, соответственно входные и выходные. На рис. 3 представлена функциональная схема параллелизатора операций.

Диспетчер очередей позволяет обеспечить сортировку заявок-сообщений перед выполнением в порядке их приоритета, задаваемом, например, на основании прав пользователя, сформировавшего запрос. Заявка на выполнение операции, таким образом, поступает диспетчеру очередей, который помещает ее во входную очередь обработчика соответствующего типа в соответствии с принятой политикой размещения заявок в очереди. Принятый подход заключается в том, чтобы помещать заявки с большим приоритетом в порядке возрастания ближе к началу очереди (рис. 4), однако такой способ в принципе допускает состояние блокировки выполнения запросов клиентов с низким приоритетом.

Обработчик — это некоторый программный функциональный блок, который обеспечивает обработку заявок. Обработчики могут быть разделены по типу выполняемых заявок либо использоваться по схеме универсального обработчика, который может исполнять все заявки. Возможны комбинации этих двух вариантов. Число входных и выходных параметров варьируется в зависимости от типа операторов. Служебный поток обра-

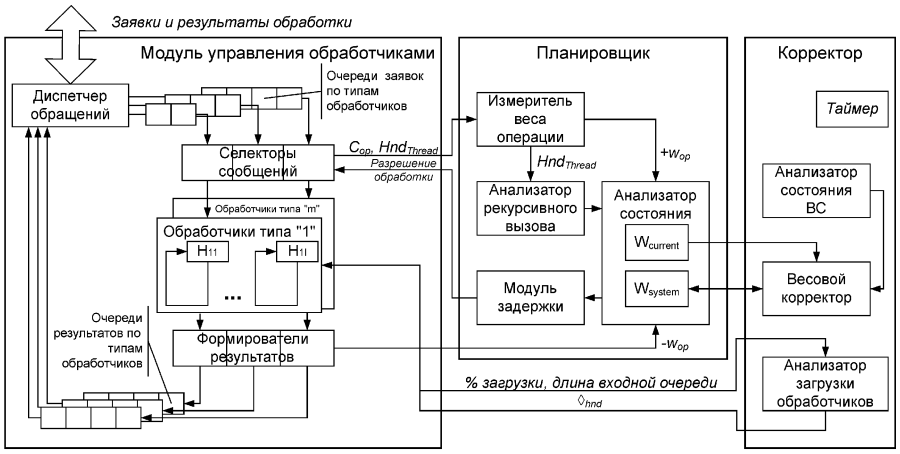


Рис. 3. Функциональная схема асинхронного адаптивного параллелизатора

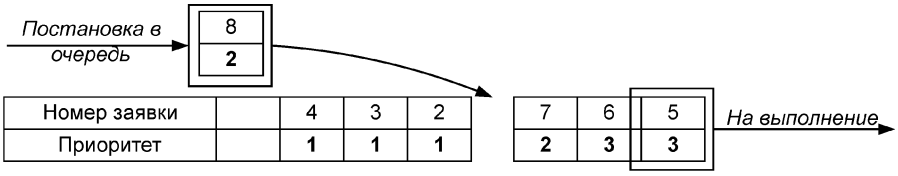


Рис. 4. Схема постановки заявок во входную очередь

ботчиков определенного типа осуществляет отслеживание заявок во входной очереди, соответствующей типу потока, и при наличии возможности выполнения (наличие заявки во входной очереди, наличие свободных потоков) запрашивает разрешение на выполнение у планировщика. Если выполнение возможно — заявка из входной очереди помечается как обрабатываемая и передается свободному потоку обработчика. После окончания обработки формируется (если это предусмотрено типом операции) результат, помещаемый в выходную очередь, а заявка из входной очереди удаляется. Подразумевается, что свободные потоки обработчиков находятся в состоянии ожидания заявки, а после окончания обработки переходят в исходное ждущее состояние, т. е. не завершаются (см. рис. 3, иллюстрируется циклами). Число параллельных потоков обработки может меняться в зависимости от текущего состояния ВС.

Внешний по отношению к параллелизатору поток выполнения, контролирующий последовательность выполнения, формирует заявки на выполнение операции, добавляя к ним уникальный порядковый номер, значения специфических параметров данного типа. Возврат управления происходит в момент постановки заявки во входную очередь, что позволяет провести постановку нескольких заявок одновременно. Точки синхронизации реализуются следующим образом: управляющий процесс ждет, в зависимости от типа операции, либо момента отсутствия заявки с соответствующим номером во входной очереди, либо появления заявки с этим номером в выходной очереди.

Планировщик и корректор. Планировщик обеспечивает контролируемую постановку заявок на выполнение. Заявки поступают в соответствующие их типу входные очереди, откуда изымаются для постановки на выполнение свободным обработчикам. Момент постановки заявки на выполнение может быть задержан до момента появления требуемых ресурсов. Поскольку возможность выполнения заявки определяется внешними факторами, необходимо контролировать и вносить коррекцию в текущие значения планировщика, что обеспечивает программный модуль-корректор. Если определение состояния вычислительной системы проводится в момент постановки сообщения на выполнение, метод управления будем называть синхронным. Если же оно не зависит от момента постановки сообщения, будем называть управление асинхронным, что является более простым в реализации, так как менее требовательно ко времени анализа системы.

Неотделимыми задачами планировщика являются оценка текущего состояния и прогноз загрузки ВС на момент выполнения операции в целях предотвращения превышения некоторого допустимого уровня, выход за который может вызвать деградацию производительности ВС в целом и фактический отказ обслуживания поступающих запросов. Простой способ организации планировщика заключается в том, чтобы рассматривать доступные ресурсы как некоторый вектор, размерности которого ассоциированы с физическими ресурсами, типа процента использования процессора, интенсивности использования канала диска и пр. Задача определения необходимых для выполнения заявки ресурсов в общем случае является сложной, поскольку длительность выполнения операции и объем необходимых ресурсов в общем случае зависят от значений параметров. Однако на основании приблизительной оценки можно принять, что для данной операции, например, требуется 1 процессор и 1 диск для чтения.

ВС в текущей реализации планировщика представляется двумя векторами — $\boldsymbol{W}_{CUR}$ и вектор $\boldsymbol{W}_{SYS}$. Под ресурсом здесь понимается некоторая численная величина, которая, в общем случае, может и не иметь реальной физической интерпретации, такой как скорость обмена данными, процент использования и пр. Любая заявка для выполнения требует определенных ресурсов, описываемых вектором $\boldsymbol{w}_{op}$, иначе:

$$\boldsymbol{W}_{CUR} = \begin{bmatrix} rs_1 \\ rs_2 \\ \dots \\ rs_N \end{bmatrix}, \ \boldsymbol{w}_{op,\,i} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_N \end{bmatrix}, \ \boldsymbol{W}_{SYS} = \begin{bmatrix} RS_1 \\ RS_2 \\ \dots \\ RS_N \end{bmatrix},$$

где rs_j — значение потребления j-го ресурса; N — число типов ресурсов; w_j — значение потребления j-го ресурса, полученное на основе априорных данных или в результате накопленных статистических данных во время работы системы; i — тип операции; RS_j — допустимое значение потребления j-го ресурса, полученное на основе априорных данных или выполненного тестирования. Критерием постановки заявки на выполнение является неравенство

$$\boldsymbol{W}^j_{SYS} \geqslant \boldsymbol{W}_{CUR} + \boldsymbol{w}^j_{op,\,i}, \ \forall j$$

где j — номер элемента вектора ресурсов.

Следует отметить возможность рекурсивной постановки заявок из обработчиков, требующей ожидания окончания выполнения. Для исключения состояния взаимной блокировки детектор рекурсивного вызова планировщика в момент ожидания окончания обработки установленной заявки временно высвобождает ресурсы, выполняя операцию $\boldsymbol{W}_{CUR} = \boldsymbol{W}_{CUR} - \boldsymbol{w}^r_{op}$, где $\boldsymbol{w}^r_{op}$ — вектор использования ресурсов временно приостановленного потока выполнения. После получения результата обработки планировщик возобновляет работу потока выполнения и выполняет операцию $\boldsymbol{W}_{CUR} = \boldsymbol{W}_{CUR} + \boldsymbol{w}^r_{op}$.

Модуль-корректор в данной реализации активируется по таймеру, проводя две независимые проверки. Первая проверка — контроль достаточности числа параллельных обработчиков каждого типа. Вторая проверка — оценка текущего состояния ВС по некоторым внешним для данной программы характеристикам и сопоставление их с представленными в планировщике векторами состояний.

Число параллельных каналов обработки каждого типа может быть фиксировано или быть переменным числом, изменяемым корректором в зависимости от текущей возможности ВС.

Обратная связь управления по состоянию ВС. Анализ текущего состояния ВС при реализации модуля-корректора можно проводить на основании системных счетчиков производительности (*Performance Counters* в терминологии *Microsoft* в ОС *MS Windows*) и проверки соответствия их значений векторам использования ресурсов. В общем случае нет прямого соответствия между элементами векторов состояний и счетчиками производительности. Характеристики производительности являются функциями многих параметров, установить точные аналитические зависимости от которых в целом не представляется возможным. Предоставляемый в ОС *MS Windows* набор параметров, доступных для съема, избыточен, поскольку содержит множество параметров, являющихся линейной комбинацией других, а также параметры, косвенно зависимые от других.

Параметры *MS Windows* можно разнести по следующим группам: параметры процессоров, параметры подсистемы памяти, параметры дисковой подсистемы, общесистемные параметры, а также параметры специального назначения, которые предоставляют информацию, специфичную для конкретных приложений, работающих на данной ВС.

Учет всех параметров в реальном времени в рамках одной ВС невозможен в связи с их большим числом, например *MS Windows* 2003 *Server* предоставляет в среднем 10—15 тысяч параметров в зависимости от системной конфигурации ВС, а также установленного набора приложений.

Методы снижения размерности и анализа адекватности значений счетчиков производительности выходят за рамки данной работы, однако следует отметить, что в общем виде для исключения избыточных счетчиков производительности необходимо использование методов статистического анализа данных, полученных от них на основе некоторых эталонных тестов. Кроме того, возможно применение методов нейросетевого прогнозирования [5]. Тем не менее возможно и непосредственное использование ряда счетчиков, предоставляющих понятные относительные значения, типа "% Processor Time".

4. Экспериментальные данные

Для проведения экспериментов автору статьи была доступна в исходном коде объектная СУБД ODB-Jupiter, разработанная в НПЦ "ИНТЕЛТЕК ПЛЮС", а также созданные на ее основе ИС (в т. ч. ИПС "Обзор СМИ") [1, 2]. Параллелизатор был встроен в сервер ОСУБД, что обеспечило контроль ресурсов на этапе диспетчеризации запросов, а также параллельное выполнение ряда внутренних операций, в частности, параллельный поиск объектов по типам, размещенным на различных хранилищах данных. На рис. 5 представлен порядок выполнения внутренних операций ОСУБД при выполнении операции поиска. Параллельной секцией является "обработка группы подзапросов по хранилищу" при наличии независимых хранилищ.

Целью проведенного эксперимента являлся анализ характеристик производительности параллельного обслуживания запросов полнотекстового поиска документов ИПС "Обзор СМИ", поэтому для исключения избыточного влияния факторов, снижающих степень параллельной обработки, были установлены следующие условия проведения эксперимента.

- Тестовая БД состоит из восьми хранилищ данных, что обеспечило параллельный поиск дан-

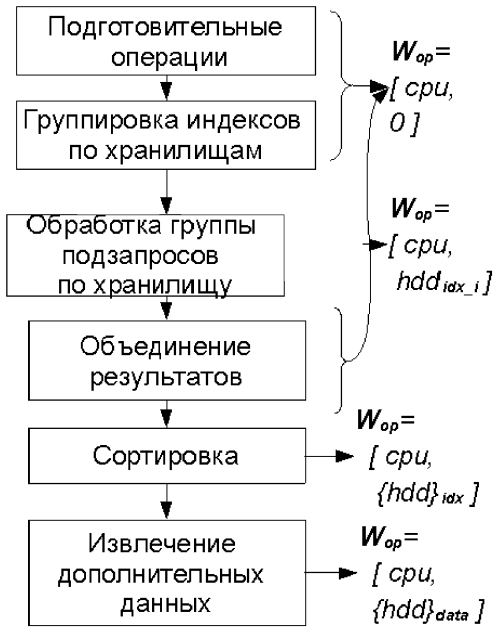


Рис. 5. Порядок выполнения внутренних операций при поиске данных

ных до восьми каналов одновременно. Каждое хранилище содержит около 30 тыс. документов со средним объемом текстовых данных 2 Кбайт.

- Поиск проводится на малом наборе текстовых запросов, требующих выполнения большого объема операций позиционирования по индексам данных, Размер кэшей индексных данных подобран таким образом, чтобы все необходимые данные полностью умещались в отведенном пространстве памяти, а обращения к дисковым накопителям были минимизированы — обеспечивается анализ ускорения параллельной обработки запросов в режиме ЦП—ОЗУ.
- Запросы формировались модифицированными программами-клиентами ИПС "Обзор СМИ" таким образом, чтобы каждый клиент циклически обеспечивал непрерывный поток запросов на сервер с синхронизацией по окончании обработки каждого запроса. Число одновременно обслуживаемых клиентов в результатах измерения соответствует числу одновременно работающих клиентов комплекса, так как время работы самого клиента является величиной, многократно меньшей времени обработки запроса.
- Значения средней производительности получены непосредственно от сервера СУБД по результатам обработки реальных запросов. Время работы для всех ВС для исключения влияния переходных процессов было одинаково и принято равным 600 с.

В качестве количественного показателя эффективности параллельной работы примем коэффициент ускорения $k^n_{\text{уск}} = t^n_{\text{посл}} / t^n_{\text{парал}}$, где $t^n_{\text{посл}}$ и $t^n_{\text{парал}}$ — среднее время последовательного и параллельного выполнения запросов соответственно; n — число клиентов, циклически формирующих запросы.

Полученные результаты сведены в таблицу.

Графики производительности для тех же ВС представлены на рис. 6. На графиках представлена производительность ИПС (число выполненных запросов текстового поиска в секунду) в зависимости от числа одновременно обслуживаемых клиентов на тех же ВС в случаях ОСУБД до (последоват.) и после (парал.) модификации. Графики иллюстрируют наибольший рост производительности в случае использования ВС в режимах до половины предельной загрузки (8 × Opteron — от 1 до 4; 4 × Intel Xeon — от 1 до 2, и AMD Athlon 64 — один поток обработки). Дальнейший рост плотности потока запросов приводит к сближению характеристик обработки запросов ОСУБД с последовательным и параллельным режимами выполнения одиночных запросов.

Графики, демонстрирующие суммарный процент загрузки процессоров для тех же ВС, представлены на рис. 7. Процент загрузки, представленный здесь, получен на основании значения счетчика производительности "% Processor Time" ОС *MS Windows.*

Полученные данные показывают снижение времени отклика за счет использования доступных процессоров в ненасыщенных режимах. Режим насыщения по производительности наступает при числе потоков обработки, равном числу процессоров, кроме ВС с процессорами Intel Xeon (три процессора из четырех), что обусловлено общей шиной ОЗУ для всех процессоров. Неполная загрузка доступных процессоров обусловлена лишь частичным распараллеливанием рассмот-

Коэффициент ускорения параллельной системы относительно последовательной

Конфигурация ВС	Коэффициент $k^n_{\text{уск}}$ от числа клиентов							
	1	2	3	4	5	6	7	8
Процессор AMD Athlon 64 X2 4400+, ОЗУ 2 ГБ DDR 400	1,6	0,98						
Процессор 4 x Intel Xeon 2,6 ГГц, ОЗУ 1024 МБ DDR 333	1,75	1,33	0,93	0,9				
Процессор 4 x Opteron 880 2,4 ГГц, ОЗУ 16 ГБ DDR 400	2,9	2	1,7	1,4	1,2	1,06	1	1
Примечание: процессоры AMD Athlon 64 X2 и AMD Opteron 880 являются двухъядерными.								

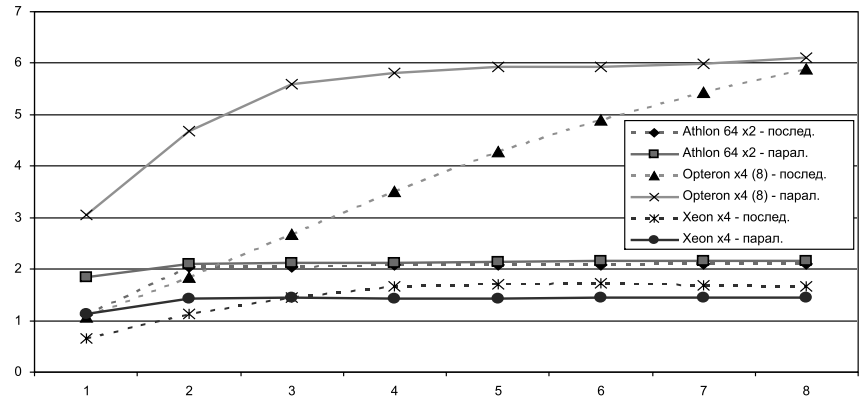


Рис. 6. Производительность сервера ИПС на полнотекстовых запросах

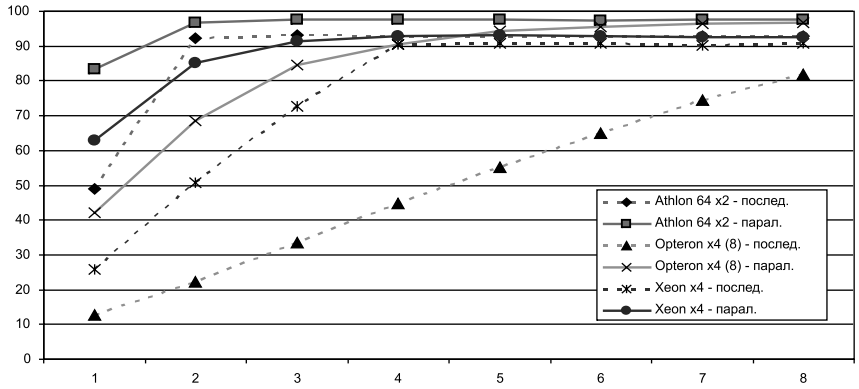


Рис. 7. Процент загрузки процессоров ВС сервера ИПС на полнотекстовых запросах

ренной операции. Во всех экспериментах параллельной системы асинхронный корректор обеспечивал регулирование предельного процента загрузки процессоров на уровне 95 % с интервалом контроля 1 с.

Заключение

В статье предложен реализованный метод параллельного выполнения внутренних операций СУБД, который предполагает регулирование загрузки обработчиков и контроль состояния ВС в реальном времени, что позволяет предотвратить состояние деградации производительности ВС в целом. Результаты экспериментальных данных показывают снижение времени отклика в ИПС с параллельным обслуживанием запросов в режимах неполной загрузки серверов в 1,6—2,9 раза в зависимости от ВС и соответственно увеличением производительности на этих режимах. Улучшено использование ресурсов процессоров. Ухудшение характеристик на режимах насыщения незначительно для ВС на основе архитектур AMD. Снижение производительности для платформы Intel Xeon не является недостатком схемы управления параллельным обслуживанием, а обусловлено ее ориентацией на выполнение задач, не требующих обмена большими объемами данных в ОЗУ, что затруднительно в классе объектных СУБД. Таким образом, можно заключить о положительном влиянии параллелизатора на процесс обслуживания запросов СУБД.

Предложенный метод организации параллельного выполнения программ с модульной структурой параллелизатора и обратной связью по состоянию ВС может найти применение в широком классе задач. Управление параллельным выполнением операций с учетом текущего состояния системы позволяет избежать монополизации ресурсов и, как следствие, системной деградации производительности. В то же время для увеличения точности регулирования постановки операций на параллельное выполнение необходимы эффективные средства мониторинга ресурсов, существующие не во всех ОС, и средства предсказания состояния ВС на ближайшее время, что в настоящее время не реализовано и является целью будущей работы.

Список литературы

1. **Андреев А. М., Березкин Д. В., Самарев Р. С.** Внутренний мир объектно-ориентированных СУБД // Открытые системы. 2001. № 3. С. 44—54.

2. **Андреев А. М., Березкин Д. В., Самарев Р. С.** Особенности создания корпоративных информационных систем на основе ОСУБД ODB-Jupiter // Материалы конференции "Корпоративные базы данных". М.: Цит-Форум, 2003.

3. **Барский А. Б.** Параллельные информационные технологии: Учебное пособие. М.: ИНТУИТ; БИНОМ. Лаборатория знаний, 2007. 503 с.

4. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

5. **Оссовский С.** Нейронные сети для обработки информации / Пер. с польск. И. Д. Рудинского. М.: Финансы и статистика, 2002. 344 с.

6. **Соколинский Л. Б.** Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой // Программирование. 2001. № 6. С. 13—29.

7. **Соколинский Л. Б.** Параллельные машины баз данных // Природа. 2001. № 8. С. 10—17.

8. **Graefe G.** Encapsulation of Parallelism in the Volcano Query Processing System // ACM 089791 365 5/90/O005/0102 — 1990.

9. **Graefe G.** Query Evaluation Techniques for Large Databases // ACM Computing Surveys. June_1993. Vol. 25. N 2.

10. **Stonebraker M.** The Case for Shared Nothing. // Database Engineering Bulletin. March 1986. Vol. 9. N 1. P. 4—9.

11. **The Object** Data Standard: ODMG 3.0 / Ed. by R. G. G. Cattel, Douglas K. Barry. — Morgan Kauffmann Publishers, 2000.

УДК 004.434

П. П. Олейник,
ООО "Торговый Дом Волшебный Рай", г. Шахты

Технология представления логических выражений с помощью XML-документов

Рассмотрена унифицированная реализация представления логических выражений с помощью XML-документов. Уделено внимание практическому применению созданных документов, описаны механизмы их обработки. Разработан алгоритм преобразования XML-документа в логическое выражение.

Одной из ключевых проблем, возникающих в процессе разработки программного обеспечения, является проблема реализации унифицированного подхода к фильтрации данных и формирования логического выражения (условия), которому должен соответствовать результирующий набор данных. К оптимальной реализации механизма фильтрации данных предъявляются следующие требования:

- *унифицированность.* Формат, используемый для представления логических выражений, должен быть максимально гибким и позволять описывать любые операторы, используемые в приложении вне зависимости от возможностей источника данных;
- *расширяемость.* Добавление нового оператора (логической функции) должно потребовать от разработчика минимального числа изменений в имеющейся системе;
- *обработка.* Как на клиентском приложении, так и на источнике данных должны присутствовать механизмы создания и обработки логических выражений, а также их трансформации в более подходящий вид.

Описанные критерии требуют пояснений. Будем предполагать, что информационная система строится по архитектуре клиент—сервер. В качестве сервера СУБД используется реляционная база данных (БД), а в качестве клиента — приложение, написанное на объектно-ориентированном языке программирования. Задача выборки данных заключается в формировании запроса клиентом с помощью графического пользовательского интерфейса (GUI) и передача его на сервер СУБД. Современные СУБД используют встроенный язык запросов, такой как SQL для реляционных БД и OQL — для объектно-ориентированных [1—3]. Существуют различные диалекты, которые не совместимы друг с другом [4]. Это одна из ключевых проблем разработки приложений, использующих в качестве хранилища информации БД от различных производителей. Кроме того, не существует унифицированного механизма генерации SQL-запросов на клиентском приложении с помощью высокоуровневого программного интерфейса (API).

Распространенным решением генерации SQL-запросов на клиентском приложении является конкатенация строк, содержащих ключевые слова, и переменных с определенными значениями [5—6]. Применяется также модифицированный подход, заключающийся в использовании текстовых строк, содержащих маркеры для подстановки параметров. Во время выполнения запроса все маркеры заменяются фактическими значениями. Недостатками данных подходов являются:

- невозможность модификации запросов. Так как строки запросов сохраняются в переменных приложения, то для их изменения придется модифицировать исходный текст программы;
- сильная привязка к синтаксису конкретной СУБД. Перенос на другую СУБД невозможен ввиду отличия в диалектах языка запросов. Проблема решается созданием нескольких строк единого запроса по одному для каждой СУБД. При этом модификация исходного кода приложения сильно усложняется.

Существуют и альтернативные решения данной задачи. В частности, современные объектно-ориентированные преобразователи (ORM, Object-Relation Mapper) включают в себя высокоуровневый язык запросов. Например, Hibernate имеет в своем составе язык HQL [7], и проблема конвертации запросов с HQL в SQL решается с помощью выбора соответствующего провайдера данных, имеющего развитый программный интерфейс (API), который позволяет создавать запрос с помощью встроенных классов. Несмотря на огромные преимущества использования этого способа по сравнению с другими решениями у него существует серьезный недостаток — преобразование запроса в SQL-операторы происходит на стороне клиента, из-за чего затрудняется оптимизация планов выполнения под конкретную СУБД.

В настоящее время получили широкое распространение программные реализации инфраструктуры MDA (Model Drive Architecture) [8—9]. Данные инструменты предполагают создание единой объектной модели на основе какого-либо хранилища данных, например, на основе реляционной базы данных. При этом имеется встроенный язык запросов (OCL, Object Constraint Language), который оперирует объектами предметной области. Ключевым недостатком данного решения проблемы является невозможность однозначного преобразования из встроенного языка запросов (OCL) в диалект конкретной СУБД. В связи с этим возникает про-

блема снижения производительности данных инструментов при реализации больших приложений.

В последние годы широкую популярность получил расширяемый язык разметки XML. Это связано с наличием единой древовидной структуры в документе и множеством технологий, которые используют XML [10]. Современные объектно-ориентированные языки программирования имеют встроенные библиотеки классов для формирования XML-документов. Выделяют две основные модели обработки документов — это SAX и DOM [11]. Несмотря на различия в подходах эти библиотеки позволяют программисту абстрагироваться от текстового содержимого XML-документа и предоставляют набор классов и методов для получения атрибутов и узлов.

Многие современные СУБД имеют в своем составе механизмы формирования и извлечения данных из XML с помощью языка запросов XQuery. Введен новый тип данных — xml, который включен в стандарт языка SQL [12—15]. Таким образом, использование XML-документов для составления запросов к БД является оптимальным решением, позволяющим унифицировать процесс фильтрации данных.

Структура XML-документа может быть представлена в виде дерева, поэтому исходная задача сводится к задаче преобразования условного выражения в дерево. Подобная задача решается при реализации языков программирования, когда для определенной строки, соответствующей введенной грамматике, строится бинарное дерево разбора [17]. Рассмотрим процесс преобразования выражения

(a = 5 and b = 4) or c <> 17
or (d like 'abc%' and e in (1,4,6))

в бинарное дерево разбора. На рис. 1 представлено данное дерево.

Для упрощения структуры дерева будем считать, что логические выражения, не содержащие булевых операторов (and и or) являются терминальными символами. После совершения симметричного (флангового, in-ordertraversal) обхода дерева (вначале посещается левое поддерево, потом сама вершина, а затем ее правое поддерево) и выполнения некоторых элементарных операций получим исходное выражение.

Теперь представим это же выражение в XML-формате, т. е. представим дерево разбора в виде

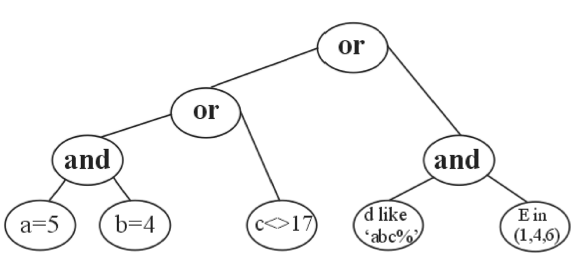


Рис. 1. Дерево разбора тестового выражения

```
<Criterias>
<Or>
  <Or>
    <And>
      <Criteria Name="a" Operator="equal">
        <Item Value="5"/>
      </Criteria>
      <Criteria Name="b" Operator="equal">
        <Item Value="4"/>
      </Criteria>
    </And>
    <Criteria Name="c" Operator="notequal">
      <Item Value="17"/>
    </Criteria>
  </Or>
  <And>
    <Criteria Name="d" Operator="like">
      <Item Value="abc%"/>
     </Criteria>
     <Criteria Name="e" Operator="in">
       <Item Value="1"/>
       <Item Value="4"/>
       <Item Value="6"/>
     </Criteria>
  </And>
</Or>
</Criterias>
```

Рис. 2. Представление логического выражения с помощью XML

XML-узлов (рис. 2). Основной идеей является размещение каждого элемента, представляющего критерий фильтрации данных (например, a = 5) внутри элемента, определяющего ту логическую операцию, с помощью которой он соединяется с другими критериями.

На рис. 2 видно, что полученный документ имеет вложенность однотипных тегов. В частности, имеются два вложенных тега < Or >. С точки зрения конечного пользователя, строить такие выражения сложно. Проведем некоторую оптимизацию полученного дерева в целях исключения вложенности однотипных тегов. В результате получили документ, представленный на рис. 3.

Полученный XML-документ имеет меньшую вложенность элементов (по сравнению с документом, представленным на рис. 2), поэтому он более прост для понимания, формирования и конечной обработки, как на стороне клиента, так и на сервере.

В примере рассматривается только представление логических выражений, которые записываются в операторе where языка SQL. Другие синтаксические конструкции, например, отвечающие за состав и порядок следования столбцов (select), или оператор сортировки (order by) не представлены. Но в случае необходимости эти элементы могут быть добавлены в качестве соседнего узла к Criterias.

Формировать подобное логическое выражение на стороне клиента несложно. Для этого необхо-

```
<Criterias>
<Or>
  <And>
    <Criteria Name="a" Operator="equal">
      <Item Value="5"/>
    </Criteria>
    <Criteria Name="b" Operator="equal">
      <Item Value="4"/>
    </Criteria>
  </And>
  <Criteria Name="c" Operator="notequal">
    <Item Value="17"/>
  </Criteria>
  <And>
    <Criteria Name="d" Operator="like">
      <Item Value="abc%"/>
     </Criteria>
     <Criteria Name="e" Operator="in">
       <Item Value="1"/>
       <Item Value="4"/>
       <Item Value="6"/>
     </Criteria>
  </And>
</Or>
</Criterias>
```

Рис. 3. Оптимизированное представление логического выражения

дим редактор, представляющий всю структуру в виде дерева. Заметим, что у элементов управления DevExpress имеется компонент FilterControl, который выглядит именно таким образом [16]. На рис. 4 изображено логическое выражение, эквивалентное тестовому.

Компонент формирует на выходе не XML (нижняя часть на рис. 4), а простую строку. При этом фильтрация выполняется на клиентском приложении. Следовательно, ему присущи недостатки, описанные ранее.

Рассмотрим процесс разбора полученного логического выражения на стороне СУБД. Представим XML-документ (см. рис. 3) в виде дерева, где корневыми узлами являются логические операции (and и or), а в листах расположены элементы Criteria (рис. 5). Будем называть такое дерево деревом логического выражения.

Дерево логического выражения является 3-арным (в общем случае *n*-арным), поэтому к нему нельзя применить классические алгоритмы обхода, применимые к бинарным деревьям [8].

Для обхода дерева логического выражения разработан следующий алгоритм.

Шаг 1. Обход дерева начинается с корня.

Шаг 2. Для каждой логической операции (and, or) все выражения, размещенные в поддеревьях, соединяются этой операцией.

Шаг 3. Обход поддеревьев узла выполняется слева направо, т. е. сначала обходится самое левое поддерево корня, содержащего логическую операцию.

Шаг 4. Посещение начинается с самого левого листового узла для каждой операции.

Выполняя обход, на каждом шаге необходимо анализировать тип обрабатываемого узла (логический оператор, либо выражение) и выполнять преобразования, соответствующие синтаксису целевой СУБД. При переносе приложения с одной СУБД на другую достаточно переписать процедуру преобразования XML-документа в SQL-выражения.

Отметим, что можно упростить процедуру преобразования, если использовать XSLT-шаблон и воспользоваться соответствующей функциональностью выбранного парсера [19].

Ранее мы предположили, что разрабатывается стандартное клиент-серверное приложение. В случае использования многозвенных приложений XML-структура логического выражения не изменится. На сервере приложений (при разработке трехзвенного приложения) можно выполнить идентификацию и аутентификацию пользователя и, в случае необходимости, дописать дополнительные критерии в запрос. Таким образом, реализуется разграничение прав доступа к информации для каждого пользователя.

Структура разработанного XML-документа не привязана непосредственно к модели хранения данных. Описанный механизм представления логических выражений может быть использован как

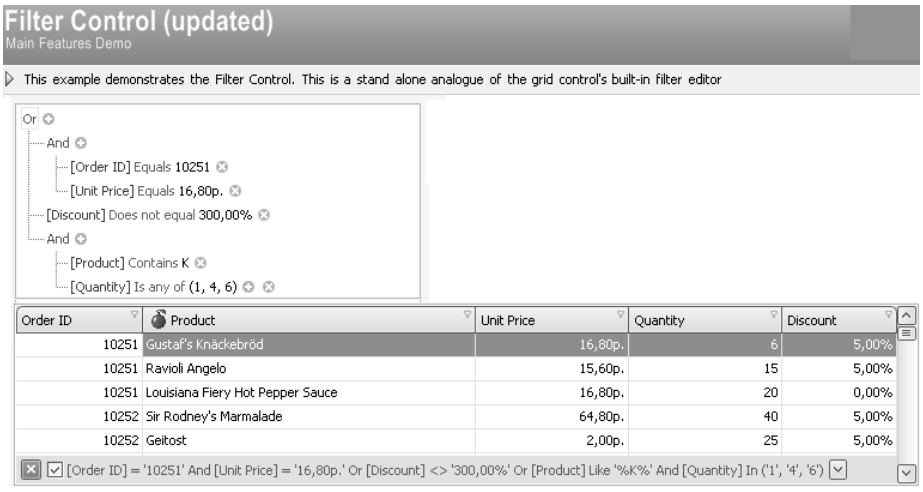


Рис. 4. Компонент FilterControl

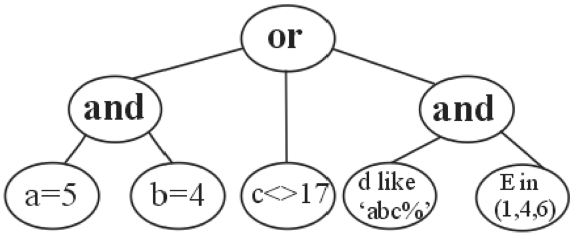


Рис. 5. Дерево логического выражения

в среде реляционной, так и в объектно-ориентированной СУБД. Кроме этого, возможно применение данных механизмов в различных объектных настройках, таких как рассмотренные в [20—21]. При этом могут быть использованы возможности выбора допустимых значений, так как доступна метаинформация о классах и объектах системы.

Список литературы

1. **Дейт К. Дж.** Введение в системы баз данных. 7-е издание.: Пер. с англ. М.: Издательский дом "Вильямс", 2001. 1072 с.
2. **Коннолли Т., Бегг К., Страчан А.** Базы данных: проектирование, реализация и сопровождение. Теория и практика. 2-е изд. М.: Издательский дом "Вильямс", 2000. 1120 с.
3. **Джордан Д.** Обработка объектных баз данных в C++. Программирование по стандарту ODMG / Пер. с англ.: Учеб. пос. М.: Издательский дом "Вильямс", 2001. 384 с.
4. **Грофф Дж., Вайнберг П.** SQL: Полное руководство: Пер. с англ. 2-е изд., перераб. и доп. К.: Издательская группа BHV, 2001. 816 с.
5. **Фаронов В. В., Шумаков П. В.** Delphi 5. Руководство разработчика баз данных. М.: Нолидж, 2001. 640 с.
6. **Флауер М.** Архитектура корпоративных программных приложений: Пер. с англ. М.: Издательский дом "Вильямс", 2004. 544 с.
7. **Bauer C., King G.** Hibernate in Action. Manning Publications. 2005. 431 p.
8. **Mukerji J., Miller J.** MDA Guide V1.0.1, http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf
9. **Грибачев К. Г.** Delphi и Model Driven Architecture. Разработка приложений баз данных, СПб.: Питер, 2004. 348 с.
10. **Дейтел Х. М., Дейтел П. Дж.** и др. Как программировать на XML. М.: Бином, 2001. 944 с.
11. **Дейтел Х. М., Дейтел П. Дж.** и др. Как программировать для Internet и WWW. М.: Бином, 2003. 1184 с.
12. **Кузнецов С.** Наиболее интересные новшества в стандарте SQL: 2003, http://citforum.ru/database/sql/sql2003/
13. **Klein S.** Professional SQL Server 2005 XML. Wiley Publishing. 2006. 549 p.
14. **Чанг Б., Скардина М.** и др. Oracle9i XML. Разработка приложений электронной коммерции с использованием технологии XML. М.: Лори, 2003. 492 с.
15. **IBM Corp.** Книга по DB2 и XML, http://www.ibm.com/developerworks/ru/library/db2xml-intro/contents.html?S_TACT = 105AGX99&S CMP = GR01
16. **DevExpress** Corp, FilterControl, http://www.devexpress.com/Products/NET/WinForms/XtraEditors/editors/FilterControl.xml
17. **Ахо А., Сети Р., Ульман Дж.** Компиляторы: принципы, технологии и инструменты.: Пер. с англ. М.: Издательский дом "Вильямс", 2003. 768 с.
18. **Топп У., Форд У.** Структуры данных в C++. М.: Бином, 2001. 816 с.
19. **Кэй М. XSLT.** Справочник программиста. 2-е изд. М.: Символ-Плюс, 2002. 1016 с.
20. **Олейник П. П.** Представление метамодели объектной системы в реляционной базе данных // Известия высших учебных заведений. Северо-Кавказский регион. Спецвыпуск "Математическое моделирование и компьютерные технологии". 2005. С. 3—8.
21. **Олейник П. П.** Организация метамодели объектной системы на основе реляционной СУБД // Научное творчество молодежи: Материалы X Всероссийской научно-практической конференции (21—22 апреля 2006 г.). Ч. 1. Томск: Изд-во Том. ун-та, 2006. С. 87—88.

ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫЕ СИСТЕМЫ

УДК 004.4:621.398.001.4

А. К. Плесовских,
Московский государственный университет леса

Решение задачи распределения опросов датчиков по потоку телеметрической информации с определением их адресов в системе "Орбита-IVMO"

Представлено решение задачи автоматизированного распределения опросов датчиков по потоку телеметрической информации с определением их адресов в системе "Орбита-IVMO" с учетом ограничений бортовой телеметрической аппаратуры.

В современном мире выяснение причин аварий при испытаниях новых образцов ракетной техники невозможно без использования радиотелеметрических средств. Одной из важнейших задач является получение измерительной информации, характеризующей состояние и поведение изделия в процессе испытаний. На выходе современной радиотелеметрической системы формируется последовательный поток информации, в котором циклически в заранее определенном порядке следуют значения всех измерительных каналов приборов. Практически каждому измерительному каналу соответствует какой-либо датчик на испытываемом объекте.

Распределение опросов датчиков по потоку телеметрической информации (ТМИ) является сложной и многосторонней задачей. Мы имеем дело с большим числом датчиков, имеющих частоты опросов, различающиеся во много раз, а также накладываются конструктивные ограничения по работе интерфейсов и запоминающих устройств самой радиотелеметрической системы.

В статье рассмотрено решение данной задачи для системы "Орбита-IVMO".

Система "Орбита-IVMO"

В качестве примера современной информационно-телеметрической системы (ИТС) рассмотрим бортовую аппаратуру "Орбита-IVMO". Это многоканальная высокоинформативная система сбора, преобразования и передачи сигналов от датчиковой аппаратуры (датчиков), устанавливаемой на испытуемом изделии. Число и типы датчиков определяются требованиями программы измерений изделия и характеристиками измеряемых параметров.

Исходными данными для проектирования бортовой телеметрической системы на основе программы измерений заказчика являются: полный перечень датчиков, установленных на испытываемом изделии, их типы и места расположения в изделии, желательные частоты опросов, режимы передачи информации на хранение и воспроизведение, структура информации, поступающей по интерфейсу RS-232 (длина и частота следования пачек), количество и содержание программ опроса.

Аппаратура "Орбита-IVMO" осуществляет периодический опрос выходных сигналов датчиков в дискретные моменты времени. Разнесение моментов опросов датчиков по времени дает возможность коммутации сигналов в единый поток измерительной информации.

Опрос каждого датчика циклический и равномерный по времени, а частота опроса датчиков может быть различной и определяется заказчиком в зависимости от ожидаемой скорости изменения измеряемого сигнала.

Аппаратура "Орбита-IVMO" обеспечивает передачу информации без дополнительного преобразования от следующих типов датчиков: аналоговых, сигнальных (дискретных), температурных (термосопротивлений и термопар), вибрационных, тензодатчиков, цифровых и аппаратурных (цифровой сигнал от аппаратуры НАП по интерфейсу RS232).

Для запоминания всего потока информации или его части и последующего воспроизведения используется запоминающее устройство (ЗУ). Возможны следующие режимы его работы: запись, хранение, воспроизведение информации, совмещение в реальном времени режимов записи и ускоренного воспроизведения ранее записанной информации с увеличением скорости воспроизведения в 2, 4, 8, 16 и 32 раза, а также задержки текущего информационного потока на определенное время. При распределении адресов записи и воспроизведения в потоке ТМИ необходимо, чтобы адрес записи информации всегда предшествовал адресу воспроизведения.

Таблица 1

Индекс информативности М	М01	М02	М04	М08	М16
Информативность, слов/с (Гц)	16 384	32 768	65 536	131 072	262 144

Сигналы с датчиков, с учетом их группирования в контролируемой аппаратуре и требований по частоте их опроса, поступают на приборы ИТС, являющиеся коммутаторами электрических сигналов датчиков. Приборы имеют унифицированную конструкцию, которая предусматривает возможность их комплектования в моноблоки, называемые модулями. Модуль, в свою очередь, является законченным изделием и позволяет осуществлять сбор информации от датчиков и внешних источников данных.

Сумма частот опроса всех датчиков, подключаемых к аппаратуре "Орбита-IVMO", и частота опроса воспроизведения, ранее записанной информации с датчиков, определяет общую информативность системы.

Информативность системы, определяющая скорость передачи данных (число слов в секунду), может принимать одно из значений, указанных в табл. 1.

Единицей распределения информативности с ИТС "Орбита-IVMO" является группа конкретного прибора. В пределах этой группы все входы датчиков опрашиваются равномерно. Большинство приборов (коммутаторов) разбиты на четыре группы. Опрос каждой группы каналов в этих коммутаторах происходит независимо от других групп и задается своим временным адресом в потоке ТМИ.

Адрес канала характеризует положение этого канала в телеметрическом кадре и его частоту опроса. Адрес канала обеспечивает однозначное его выделение из общего потока телеметрических сигналов. Система адресации каналов позволяет однозначно связать временную расстановку опроса каналов с программой распределения опросов в формирователях и коммутаторах системы.

Особенности адресации в системе "Орбита-IVMO"

В ИТС "Орбита-IVMO" принято иерархическое октавное распределение опросов, где каждая ступень распределения (всего их пять) представляет собой виртуальный регистр-делитель на 8, на выходе которого вырабатывается 8 последовательностей выходных сигналов, частота которых

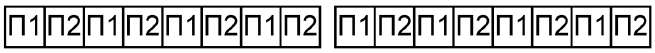


Рис. 1. Телеметрический кадр при информативности М16

в 8 раз меньше частоты сигнала, поданного на этот регистр. Для увеличения частоты опросов используется запараллеливание выходных потоков.

Адрес каждого канала задается буквенными и цифровыми символами, составляющими набор координат и ряд вспомогательных обозначений. Адрес канала задается: символом информативности системы М и шестью дополнительными координатами.

Символ информативности системы М может принимать одно из значений: М01, М02, М04, М08 или М16 (см. табл. 1).

При информативности 262 144 слов/с (М16) все опросы делятся на два потока — П1 и П2 по 131 072 слова/с каждый. Первый поток содержит все нечетные слова, второй — все четные (рис. 1).

При информативности 131 072 слов/с (М08) и менее все они укладываются в один поток.

Шесть координат — *A*, *B*, *C*, *D*, *E*, *X* — определяют виртуальную ступень деления общей информативности и фазу выходного сигнала с каждого виртуального делителя, которая записывается двумя цифрами после обозначения координаты, где первая из цифр — номер координаты или фаза выходного сигнала данного виртуального делителя, вторая цифра — коэффициент запараллеливания или число объединений на этом выходе делителя.

Решение задачи в общем виде

На основе программы измерения заказчика мы имеем: полный перечень датчиков, их типы и места расположения в изделии, желательные частоты опросов, структуру информации, поступающей по линии RS-232 (длина и частота следования пачки), а также режимы передачи информации на хранение и воспроизведение.

Распределение опросов датчиков по потоку ТМИ начнем с информации, поступающей по линии RS-232 (НАП). Частота опроса $F_{\text{НАП}}$, требуемая для передачи информации, поступающей с НАП, вычисляется по формуле

$$F_{\text{НАП}} = LT, \qquad (1)$$

где L — длина пачки информации (байт); T — частота следования пачки (пачек в секунду (Гц)).

В силу того, что частота информации, поступающей с НАП, является асинхронной по отношению к частоте потока телеметрической системы, поток НАП распределяют особым образом. Как правило, НАП делят на два потока: НАП1 и НАП2 с разными частотами, что позволяет приблизить суммарную частоту опроса НАП к частоте потока ТМИ.

Частоты опросов НАП1 и НАП2 равны

$$\text{НАП1} = \max f < F_{\text{НАП}}, \qquad (2)$$

$$\text{НАП2} = \min f > (F_{\text{НАП}} - \text{НАП1}), \qquad (3)$$

где f — частота передаваемой информации, кратная степени двойки.

Исходя из требуемой опросности датчиков изделия определяется ориентировочная информативность системы. Информативность системы (*It*) рассматривается как сумма частот опросов всех групп приборов (групп коммутаторов сигналов датчиков) ИТС ($F_{\text{г}}$), плюс частоты опросов воспроизведения, ранее записанной информации ($F_{\text{гв}}$), плюс частоты опросов и воспроизведения информации, поступающей с цифровых источников данных (НАП) ($F_{\text{НАП}}$ и $F_{\text{НАПв}}$):

$$It = F_{\text{г}} + F_{\text{гв}} + F_{\text{НАП}} + F_{\text{НАПв}}. \qquad (4)$$

Следует иметь в виду, что вся информативность, выделенная на опрос группы прибора, распределяется равномерно на опрос каждого канала в этой группе, и если какие-либо каналы данной группы не задействованы, то эта информативность пропадает.

Полученное значение информативности округляется в большую сторону до одной из стандартных информативностей ИТС "Орбита-IVMO" (см. табл. 1):

$$I = \min M > It. \qquad (5)$$

Сначала происходит распределение приборов/НАП по потоку ТМИ, а потом, с использованием определенных формул, происходит вычисление октавных адресов в формате ИТС "Орбита-IVMO".

Поток ТМИ (телеметрический кадр) разбит на четыре цикла. Каждый элемент потока назовем ячейкой. Частота опроса ячейки всегда равна 4 Гц. Размер каждого цикла ($R_{\text{ц}}$) определяется как:

$$R_{\text{ц}} = I/4 \text{ (ячеек)}. \qquad (6)$$

Для **распределения опросов групп приборов/НАП** необходимо знать начальное смещение группы/НАП S и период P следования этих опросов в цикле.

Период следования опросов групп приборов/НАП определяется как

$$P = (R_{\text{ц}} \cdot 4)/F_{\text{г}}(F_{\text{НАП}}), \qquad (7)$$

где $F_{\text{г}}$ — частота опроса группы; $F_{\text{НАП}}$ — частота опроса информации, поступающей с НАП.

Поиск начальных смещений S для записи и воспроизведения информации с групп приборов/НАП и их распределение по потоку ТМИ происходит по следующему алгоритму, представленному на рис. 2:

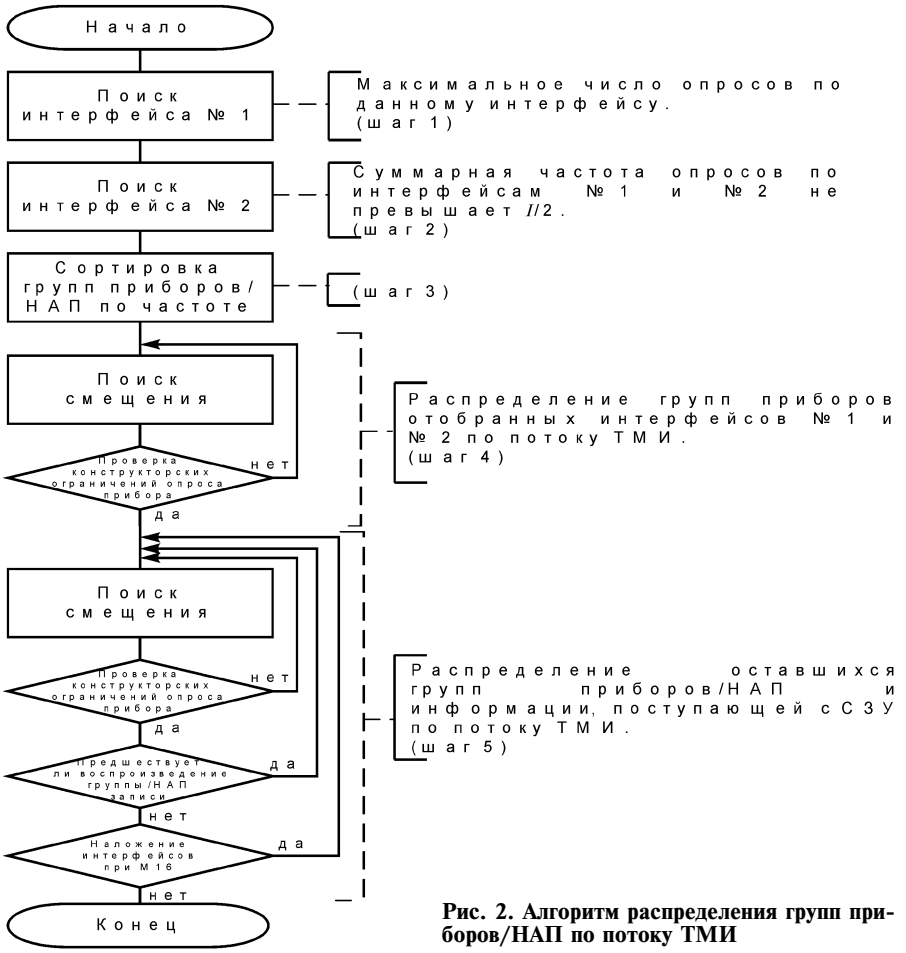


Рис. 2. Алгоритм распределения групп приборов/НАП по потоку ТМИ

Шаг 1 — просматриваются все опрашиваемые группы приборов, и определяется, по какому интерфейсу происходит максимальное число опросов.

Шаг 2 — для оставшихся групп приборов определяется интерфейс, при котором суммарная частота опросов по обоим интерфейсам не превышает $I/2$ Гц.

Шаг 3 — сортировка всех опрашиваемых групп приборов/НАП и воспроизведения их с ЗУ по частоте.

Шаг 4 — происходит распределение групп приборов отобранных интерфейсов по потоку ТМИ. Начальное смещение опроса определяется как средняя ячейка в максимально длинной неопрашиваемой группе ячеек. Поиск необходимых ячеек происходит с двойным периодом. Таким образом, на данном этапе исчезает необходимость при информативности М16 проверки наличия двух смежных опросов по одному интерфейсу. При распределении еще учитывается, чтобы частота между опросами групп одного прибора не превышала максимальной частоты опроса данного прибора.

Шаг 5 — распределение опросов оставшихся групп приборов/НАП и информации, поступающей с ЗУ. При этом необходимо, чтобы адрес записи информации всегда предшествовал адресу воспроизведения, чтобы частота между опросами групп одного прибора не превышала максимальной частоты опроса данного прибора и при информативности М16 не было двух смежных опросов по одному интерфейсу, поскольку частота передачи командных слов контроллером не может превышать 131 072 Гц.

Определение октавных адресов групп приборов/НАП

Адрес группы прибора/НАП задается символом информативности системы М и пятью координатами — *A*, *B*, *C*, *D*, и *E*.

После каждой координаты следуют две цифры, где первая — номер координаты или фаза выходного сигнала данного виртуального делителя, вторая — число объединений на этом выходе делителя.

Зная начальное смещение S и период следования опросов группы или НАП P, можно определить значения цифр, следующих после каждой координаты.

Для координаты $A(a_1a_2)$:

$$a_1 = [(S - R_A[S/R_A] + 1)/K_{\text{п}}]; \quad (8)$$

$$a_2 = \begin{cases} \log_2(R_A/P), \text{ если } P \leqslant R_A, \\ 0, \text{ если } P \geqslant R_A. \end{cases} \quad (9)$$

Здесь [ ] — вычисление целого значения отношения, заключенного в скобки, с отбрасыванием дробной части; R_A — размер координаты A (число ячеек), R_A принимает значения 16, 8, 4, 2, 1 для информативностей М16, М08, М04, М02, М01 соответственно; $K_{\text{п}}$ — число потоков в системе; при информативности М16 — два потока; при остальных значениях М — один. При информативности М16 номер потока $N_{\text{п}}$ будет

$$N_{\text{п}} = \begin{cases} 1, \text{ если } S \text{ четное}, \\ 2, \text{ если } S \text{ нечетное}. \end{cases} \quad (10)$$

Для координат $B(b_1b_2)$, $C(c_1c_2)$, $D(d_1d_2)$, $E(e_1e_2)$ значения, следующие после координат (k_1k_2), вычисляются следующим образом:

$$k_1 = [S/(R_A \cdot 8^{n-2})] - 8[S/(R_A \cdot 8^{n-1})] + 1; \quad (11)$$

$$k_2 = \begin{cases} \log_2((R_A \cdot 8^{n-1})/P), \text{ если } P \leqslant 8^{n-1}R_A, \\ 0, \text{ если P} \geqslant 8^{n-1}R_A, \end{cases} \quad (12)$$

где n — номер координаты в адресе; A — первая координата; B — вторая координата и т. д. В адресе группы прибора/НАП могут присутствовать не все координаты.

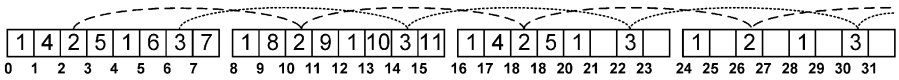


Рис. 3. Поток ТМИ при информативности М08

Если

$$P > R_A \cdot 8^{n-2}, \qquad (13)$$

то n-я координата в адресе присутствует.

Практический пример

Рассмотрим пример. По потоку телеметрической информации (ТМИ) требуется распределить прибор МКБ2, имеющий 4 группы по 8 каналов с частотами опроса 32 768 ($F_{г1}$), 16 384 ($F_{г2}$), 8192 ($F_{г3}$) и 4096 ($F_{г4}$) Гц, и еще поток информации, поступающей с приемника RS-232 (НАП). Длина пачки L равна 512 байт, частота следования пачки T равна 10 пачек в секунду, МКБ2 — коммутатор быстроменяющихся сигналов, поступающих с датчиков. Вторую, третью и четвертую группу прибора, а также всю информацию с НАП необходимо записать и впоследствии воспроизвести с той же скоростью, например в режиме линии задержки.

Частота опроса, требуемая для передачи информации, поступающей с НАП [формула (1)]: $F_{НАП} = 5120$ Гц. НАП делится на два потока: НАП1 и НАП2, с частотами [формулы (2), (3)], равными 4096 Гц и 2048 Гц соответственно.

Общая информативность системы [формула (4)]:

$$It = F_{г1} + F_{г2} + F_{г3} + F_{г4} + F_{г2в} + F_{г3в} + F_{г4в} + F_{НАП} + F_{НАПв} = 101\ 376.$$

Информативность с учетом выбора одной из стандартных информативностей ИТС "Орбита-IVMO": $I = 131\ 072$ ГЦ (М08).

Затем по приведенному выше алгоритму (см. рис. 2) происходит распределение групп приборов/НАП и информации, поступающей с СЗУ по потоку ТМИ.

Результаты распределения опросов групп приборов и НАП представлены на рис. 3 и в табл. 2.

В табл. 2 позиция в потоке ТМИ соответствует цифре данной группы/НАП на рис. 3; название группы прибора МКБ2(1)-2 означает, что это вторая группа первого прибора МКБ2 в системе; r — передача информации в реальном времени; z — задержка поступающей информации на определенное время; aN — воспроизведение ранее записанной информации, где N — коэффициент трансформации скорости воспроизведения (2, 4, 8, 16, 32). Если скорость воспроизведения равна скорости записи, то коэффициент N не указывается.

Как видно из рис. 3 и табл. 2, вторая группа прибора МКБ2 (позиция 2) имеет начальное смещение $S = 2$ и период опроса восьми ячеек $P = 8$. Размер R_A координаты A и число потоков $K_П$ при информативности М08 будут равны 8 и 1 соответственно.

Так как неравенство $P > R_A 8^{n-2}$ [формула (13)] верно только при первой координате, то адрес записи группы будет состоять только из координаты $A(a_1a_2)$, где a_1 и a_2 [формулы (8), (9)] равны 3 и 0 соответственно. Адрес записи второй группы прибора МКБ2 — $A30$.

Воспроизведение данной группы (позиция 3) имеет начальное смещение 6 и тот же период опроса. Адрес воспроизведения второй группы прибора МКБ2 — $A70$.

Адрес записи группы предшествует адресу воспроизведения, что не противоречит работе запоминающего устройства.

Просматривая полученные данные, можно сделать вывод, что рассматриваемая система не

Таблица 2

Позиция в потоке ТМИ	Название группы прибора/НАП	Частота опроса, Гц	Номер модуля	Номер интерфейса	Режим работы запоминающего устройства	Адрес в потоке ТМИ*
1	МКБ2(1)-1	32 768	1	1	*r*	A11
2	МКБ2(1)-2	16 384	1	1	*z*	A30
3	МКБ2(1)-2В**	16 384	1	0	*a*	A70
4	МКБ2(1)-3	8192	1	1	*z*	A20B22**
5	МКБ2(1)-3В	8192	1	0	*a*	A40B22
6	МКБ2(1)-4	4096	1	1	*z*	A60B11
7	МКБ2(1)-4В	4096	1	0	*a*	A80B11
8	НАП1	4096	1	0	*z*	A20B21
9	НАП1В	4096	1	0	*a*	A40B21
10	НАП2	2048	1	0	*z*	A60B20
11	НАП2В	2048	1	0	*a*	A80B20

Примечания. * — номер потока в адресах не указывается, так как информативность системы М08; ** — символ "В" означает, что данная группа/НАП воспроизводится с запоминающего устройства.

использует заметную часть выбранной информативности, и есть возможность опрашивать группы приборов с большей частотой.

Заключение

Решение задачи, рассмотренной в данной статье, с помощью ЭВМ позволяет значительно сократить временные затраты, а также исключить грубые ошибки при распределении опросов.

В настоящее время создана программа, реализующая данный алгоритм, и идет ее опытная отработка в ОКБ МЭИ. По результатам этой отработки планируется добавить пользовательский интерфейс и передать программу для использования всем заинтересованным потребителям системы "Орбита-IVMO".

Список литературы

1. **Горбатюк Л. Е., Недошивин С. Н., Победоносцев К. А.** и др. Бортовая аппаратура "Орбита-IVMO", опыт ее производства и применения для натурной отработки ракет различных классов. // Радиотехнические тетради. 2006. № 33.

2. **Фомин А. Ф., Новоселов О. Н., Победоносцев К. А., Чернышов Ю. Н.** Цифровые информационно-измерительные системы: Теория и практика / Под ред. Фомина А. Ф., Новоселова О. Н. М.: Энергоатомиздат, 1996.

УДК 004.032

М. Д. Керимов, канд. техн. наук,
Азербайджанский политехнический университет

Вопросы оптимизации транзитивно-нечетких информационных систем дистанционного зондирования

Рассматривается вопрос об оптимизации выделенного подкласса транзитивно-нечетких систем. Показано, что раздельная и последовательная оптимизация по двум выбранным сходным критериям позволяет выявить инвариантность отношения основных показателей этих систем.

Транзитивно-нечеткие отношения в информационных системах исторически первыми рассматривались применительно к мультиагентным системам [1, 2], где были исследованы транзитивно-нечеткие — энтропийные закономерности самоорганизации мультиагентов. В указанных работах мультиагентные системы рассматриваются в качестве динамической системы, в которой агенты обеспечиваются информацией, аргументированно связывая этот обмен со своими знаниями о поставленной цели. Оптимальный объем существующих знаний соответствует оптимальному уровню информационной организации, и нечеткая энтропия в этом случае рассматривается в качестве меры степени порядка, установившегося между агентами. Наличие транзитивно-нечетких отношений в таких системах приводит к тому, что процедура минимизации нечеткой энтропии может быть рассмотрена в качестве механизма превращения мультиагентной системы в голоническую структуру [1], т. е. в структуру, наиболее приспособленную для выполнения поставленной задачи.

Однако применительно к информационно-измерительным системам такую постановку и решение задачи структурно-функциональной оптимальной эволюции систем вряд ли можно считать удачными, так как такой подход основан на нечеткой энтропии, в то время как исследователей информационно-измерительных систем интересовало бы совместное рассмотрение как нечеткой энтропии, так и информационной энтропии и их связи с функционально-динамическими возможностями исследуемых систем, характеризующихся такими фундаментальными понятиями, как транзитивность и нечеткость.

Далее, говоря о транзитивно-нечетких отношениях в информационных системах, и конкретно о транзитивно-нечетких информационных системах, будем полагать фактическое разделение этих понятий, т. е. будем считать, что рассматриваемая система характеризуется двумя независимыми показателями, один из которых характеризует транзитивные отношения в системе, а другой — нечеткие отношения.

Далее вопрос об оптимальных отношениях в транзитивно-нечетких системах будем рассматривать в контексте сканирующей системы дистанционного зондирования, осуществляющего построчное формирование информационного кадра.

Рассмотрим вопрос полета носителя сканера по наклонной траектории (рис. 1), где носитель сканера, двигаясь по наклонной траектории при перемещении из пространственной точки A в точку A_1, осуществляет сканирование изучаемого участка. Заметим, что на рис. 1 показан оптимальный вариант выбора режимных показателей системы в соответствии с результатами, полученными в работе [3], согласно которым на более высоких траекториях значение T_i должно уменьшаться. Однако заметим, что такой порядок проведения оптимальных съемок справедлив только в случае транзитивности информативного параметра ис-

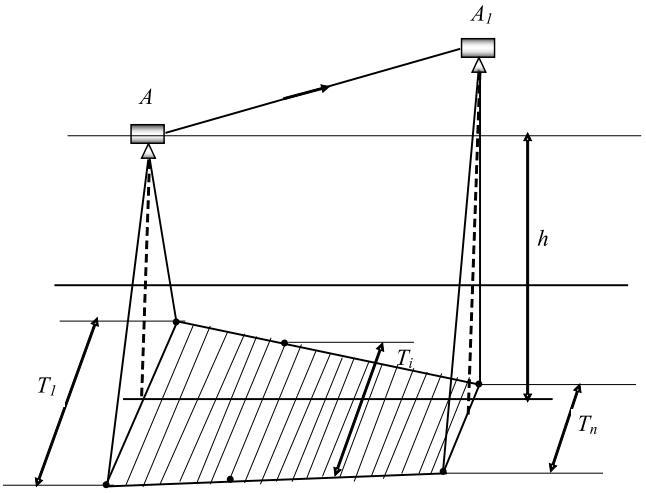


Рис. 1. Графическое представление оптимального режима

следуемых строчных импульсов. Требование транзитивности соответствует классическому определению транзитивности элементов множества по выбранному типу отношения. Например, если оценку информативности строки T_i обозначить как $m(T_i)$, то транзитивность элементов множества $\{m_i(T_i)\}$ с учетом отношения, сформированного в виде "не связана в информационном смысле" и обозначаемого как R, может быть выражена как

$$If \lfloor M(T_{i-1}) \rfloor R \lfloor M(T_i) \rfloor \wedge [M(T_i)] R \lfloor M(T_{i+1}) \rfloor \Rightarrow \Rightarrow \lfloor M(T_{i-1}) \rfloor R \lfloor M(T_{i+1}) \rfloor. \quad (1)$$

Физически транзитивность информативности смежных строк кадра означает автономность исследуемых участков (строк) по тематическим информативным признакам. При этом допускаем однородность участков, следовательно, строки с меньшей длительностью не могут быть более информативными, чем строки с большей длительностью. Следует отметить, транзитивность исследуемых объектов является общей вероятной характерной чертой многих видов систем дистанционного зондирования, таких как радары, тепловизоры, сонары, сканеры и т. д.

С учетом сказанного можно выделить особый класс транзитивно-нечетких систем, которые должны иметь следующие особенности.

1. Эти системы характеризуются транзитивным параметром $M(T_i)$, $i = \overline{1, n}$.

2. Эти системы характеризуются нечетким параметром S — сигнал/шум, который проявляет дуальный характер, заключающийся в том, что:

2.1. S является детерминированной функцией детерминированного аргумента T_i, в то же время оставаясь нечеткой величиной из-за наличия большого числа внешних воздействующих факторов.

2.2. Так как S является нечеткой величиной, то эта величина характеризуется функцией принадлежности, которая меняется динамически в зависимости от T_i.

3. Транзитивно-нечеткие системы характеризуются двумя предельными ресурсными параметрами.

3.1. Интегральный показатель нечеткости системы, определяемый как

$$F = \int_0^{S_{\max}} \mu(S) dS, \quad (2)$$

где $\mu(S)$ — значение функции принадлежности S; $F = \text{const}$. При этом F может быть назван ресурсом нечеткости системы, так как этот показатель характеризует суммарную характеристику системы по всем возможным значениям S.

3.2. Интегральный детерминированный показатель системы, определяемый как

$$D = \int_0^{T_{\max}} k_{01} S(T) dT, \quad (3)$$

где k_{01} — коэффициент пропорциональности.

При этом параметр D может быть назван достоверностным ресурсом системы, так как характеризует интегрированное значение отношения сигнал/шум в системе.

Перейдем непосредственно к вопросу об оптимизации систем выделенного класса транзитивно-нечетких систем. Общая задача оптимизации таких систем может быть представлена в виде последовательно выполняемых двух подзадач оптимизации.

Подзадача 1. Нахождение оптимального вида функции $S = S(T)$ при выбранном функционале оптимизации, с учетом ранее принятого ограничительного условия (3).

Для решения данной подзадачи используем энергоинформационный критерий оптимизации [3], имеющий следующий вид:

$$z_g = \frac{\int_0^{T_{\max}} T \ln[S(T)] dT}{\int_0^{T_{\max}} S(T) dT}. \quad (4)$$

Подзадача 2. Нахождение оптимального вида функции $\mu = \mu(S)$ при выбранном функционале оптимизации, с учетом ранее принятого ограничительного условия (2).

Для решения второй подзадачи также используем критерий оптимизации, аналогичный (4):

$$z_\Phi = \frac{\int_0^{S_{\max}} S \ln[\mu(S)] dS}{\int_0^{S_{\max}} \mu(S) dS}. \quad (5)$$

Использование для решения двух указанных выше подзадач энергоинформационного и нечет-

кого критериев, а также трехступенчатого принципа оптимизации, предложенного в работе [3], имеет следующие очевидные преимущества:

- ограничения типа (3) и (2) автоматически учитываются при решении оптимизации функционалов (4) и (5);
- применяемый трехступенчатый принцип оптимизации позволяет не только находить оптимальный вид функции $S(T)$ и $\mu(S)$, но и вычислить оптимальные значения заданных ограничений (2) и (3).

Решение вышеуказанных двух подзадач с помощью трехступенчатого принципа оптимизации дало следующие выражения для искомых функций $\mu = \mu(S)$ и $S = S(T)$:

$$\mu(S)_{\text{опт}} = \frac{2SF_{\text{опт}}}{S^2_{\max}}; \qquad (6)$$

$$S(T)_{\text{опт}} = \frac{2TD_{\text{опт}}}{k_{01}T^2_{\max}}. \qquad (7)$$

С учетом выражений (6) и (7) вычислим инвариант, характеризующий выделенный класс транзитивно-нечетких систем (ТНС).

Так как решение двух подзадач осуществляется раздельно, результат (7) используем в следующем выражении:

$$\mu(S)_{\text{опт}} = \frac{4TF_{\text{опт}}D_{\text{опт}}}{k_{01}T^2_{\max}S^2_{\max}}. \qquad (8)$$

С учетом (7) и (8) имеем

$$\text{Inv[ТНС]} = \frac{\mu[S(T)_{\text{опт}}]}{S(T)_{\text{опт}}} = \frac{2D_{\text{опт}}}{S^2_{\max}}. \qquad (9)$$

Таким образом, выявлено важное качество систем выделенного класса — транзитивно-нечетких систем, заключающееся в существовании инварианта, определяемого в виде отношения найденных оптимальных функциональных зависимостей в трехступенчатой процедуре оптимизации этих систем.

Отметим, что транзитивное качество информативности в рассматриваемых системах может проявляться не только как свойство исследуемых удаленных объектов (сцен), но и как результат воздействия других изменяющихся параметров систем. Например, функция $M(T_c)$, отображающая информативность выделенных полос исследуемого кадра, может потерять корреляционные связи между $M(T_i)$ и $M(T_j)$; $i, j = 1, n$; $i \neq j$, вследствие динамичного вхождения носителя в зону с меньшей прозрачностью атмосферы, возникшей, например, вследствие различных течений переноса атмосферного аэрозоля. С учетом указанного обстоятельства можно предложить алгоритм правильного выбора транзитивного аргумента при идентификации принадлежности конкретной системы к выделенному классу транзитивно-нечетких систем. Блок-схема алгоритма показана на рис. 2.

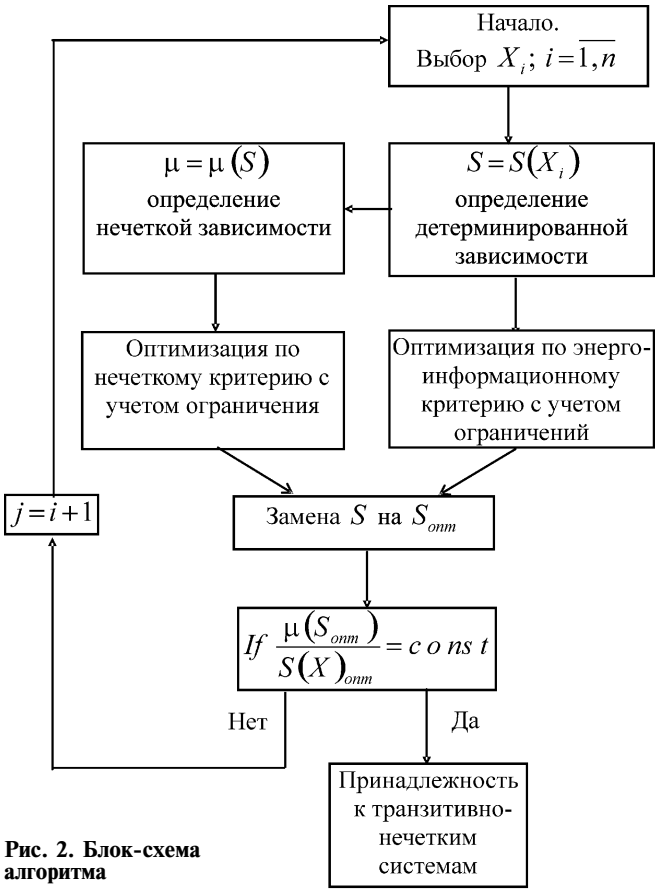


Рис. 2. Блок-схема алгоритма

***

В заключение отметим, что выявленное инвариантное качество оптимального режима систем выделенного подкласса транзитивно-нечетких систем имеет ясный физический смысл, заключающийся в том, что в оптимальном режиме увеличение отношения сигнал/шум в таких системах непременно должно сопровождаться увеличением соответствующего максимального значения функции принадлежности, что означает движение системы в состоянии большей определенности.

Список литературы

1. **Ulieri M.** A fuzzy mathematic approach to modeling emergent holonic structures. www.cs.unb.ca/~ulieri/Publicant/Pub.Web/Romaniz-paper-Last.doc.

2. **Ulieri M.** Holonic. Enterprise as an Information Ecosystem // Proc. Workshop on "Holous for Industry", Autonomous Agents International Conference, Montreal, Canada, May 2—6. 2001. P. 3—20.

3. **Асадов Х. Г., Керимов М. Д.** Новый принцип оптимизации информационных систем дистанционного зондирования в нестационарных режимах работы // Информационные технологии. 2006. № 5. С. 73—76.

УДК 330.322

А. Н. Важдаев,
Юргинский технологический институт (филиал)
Томского политехнического университета

Информационные системы, применяемые при работе, оценке и анализе инвестиционных проектов

Рассмотрены наиболее популярные программы, применяемые при работе, оценке и анализе инвестиционных проектов во многих государственных организациях и частных компаниях. Подробно описаны критерии анализа при выборе подходящего программного обеспечения для решения вопросов инвестиционного характера. Описан новый программный продукт, используемый для инвестиционного анализа в работе предприятий малых форм собственности и в ходе учебного процесса вузов.

Введение

Современные условия развития малого бизнеса в нашей стране делают особенно важными знания и умения спланировать развитие своего дела, оценить достигнутые результаты в сравнении с первоначальным планом, проанализировать и сделать соответствующие выводы для принятия правильных управленческих решений на всех этапах осуществления инвестиционных проектов.

Экономическая оценка инвестиционных проектов или предприятий в целом представляет собой довольно сложную и трудоемкую систему расчетных операций, для проведения которых требуется весьма значительная по объему информация. К тому же должны иметься достаточные материальные и временные ресурсы. Поэтому очень часто у представителей малого бизнеса возникает потребность в использовании более простых и доступных инструментов оценки и анализа.

При существующих объемах информации, числе сравниваемых проектов и малых сроках ожидания получения готовых результатов расчет эффективности инвестиционных проектов и параметров инвестиционной привлекательности организации лучше всего осуществлять с помощью специального программного обеспечения. Использование таких программ обеспечивает более высокое качество оценки и позволяет увеличить вероятность адекватности полученных результатов существующей реальности. С помощью компьютерного моделирования можно довольно быстро проанализировать большое число проектов, выбрать из них лучший по соответствующим критериям или разработать мероприятия по повышению эффективности уже реализуемого проекта [1, 2, 3].

1. Обзор существующих информационных систем по оценке и анализу инвестиционных проектов

По мнению автора, современная Россия только находится на пути становления рыночной экономики. Поэтому при оценке и анализе эффективности инвестиционных проектов в нашей стране следует учитывать специфику российской экономики [4]. Именно вследствие этого будет правильным использовать именно отечественные программные продукты. Внедрение же в российские программы международной методики по оценке инвестиционных проектов UNIDO и методики финансового анализа по стандартам IAS является, по мнению автора, пока лишь только попыткой применить мировые общепризнанные стандарты к российской действительности.

В настоящее время в нашей стране наибольшее распространение получили два вида программных продуктов для оценки инвестиционных проектов.

1. Программные продукты, предназначенные для оценки финансовых результатов деятельности, отраженных в отчетности за истекший (на момент анализа) период, а также будущего потенциала предприятия, т. е. для экономической диагностики хозяйственной деятельности и инвестиционной привлекательности предприятия в целях выработки рекомендаций по совершенствованию хозяйственной деятельности и повышению инвестиционной привлекательности.

2. Программные продукты для планирования, расчета и сравнительного анализа инвестиционных проектов в целях выбора наиболее эффективного и оптимального.

1.1. Программы для экономической диагностики хозяйственной деятельности предприятия

1. Программный продукт *Альт-Финансы* предназначен для выполнения комплексной оценки деятельности предприятия, выявления основных

тенденций его развития, расчета базовых нормативов для планирования и прогнозирования, оценки кредитоспособности предприятия. Проведение детального финансового анализа и управленческая интерпретация полученных результатов позволят найти оптимальный путь развития, разработать программу финансового оздоровления предприятия, находящегося на грани банкротства, обосновать инвестиционное решение [1].

2. Аналитическая система *Audit Expert* предназначена для диагностики, оценки и мониторинга финансового состояния предприятия. *Audit Expert* позволяет: сформировать сопоставимые финансовые данные для решения аналитических задач; быстро оценить финансовое состояние предприятия по его финансовой отчетности; провести углубленное исследование финансово-хозяйственной деятельности предприятия на основе собственных методик; осуществить бенчмаркинг; оценить реальную стоимость имущества компании. Программа позволяет провести переоценку различных статей активов и пассивов баланса для проведения финансового анализа по реальным данным [2].

3. Система *Мастер финансов* — это профессиональная система оценки финансового состояния предприятия, позволяющая в десятки раз снизить затраты времени при проведении анализа финансово-хозяйственной деятельности и составлении различных отчетов и заключений по результатам анализа. Исходными данными для программы являются обычные формы бухгалтерской и статистической отчетности, такие как баланс и отчет о прибылях и убытках [1].

4. Программно-методический комплекс *КИС*: *Бюджетирование* предназначен как для внедрения всеобъемлющей системы бюджетного управления, охватывающей все подразделения предприятия и требующей кардинальных изменений в процедурах и организационной структуре планирования, так и для составления бюджета предприятия, интегрирующего и гармонизирующего существующие планы предприятия силами двух—трех специалистов [1].

5. Программа *ИНЭК-АФСП* предназначена для финансового анализа (на основе данных внешней бухгалтерской отчетности), мониторинга и сравнения предприятий и организаций, занимающихся различными видами деятельности [3].

6. Программа *ИНЭК-АДП* предназначена для финансового анализа (на основе данных внешней бухгалтерской отчетности), мониторинга и сравнения предприятий и организаций, занимающихся различными видами деятельности. Кроме того, программа позволяет выполнять расчет финансовых показателей в соответствии с действующим законодательством, оценивать эффективность использования материально-сырьевых, топливно-энергетических и трудовых ресурсов, проводить факторный анализ прибыли, оценивать эффективность производства и реализации каждого вида продукции, рассчитывать уровень безубыточности и запас финансовой прочности [3].

1.2. Программные продукты анализа инвестиционных проектов

1. Аналитическая система *Project Expert* — система поддержки принятия решений для разработки и выбора оптимального плана развития бизнеса, создания финансовых планов и инвестиционных проектов. Эта система позволяет моделировать деятельность предприятий различных масштабов — от небольшого частного предприятия до холдинговых структур [2].

Создаваемый в системе бизнес-план соответствует международным требованиям: в основу *Project Expert* положены методика UNIDO по оценке инвестиционных проектов и методика финансового анализа, определенная международными стандартами IAS. В то же время в *Project Expert* учитывается специфика российской экономики. Система рекомендована к использованию госструктурами федерального и регионального уровня как стандартный инструмент для разработки планов развития предприятия.

2. Программный продукт *Альт-Инвест* предназначен для подготовки, анализа и оптимизации инвестиционных проектов различных отраслей, масштабов и направленности. С помощью *Альт-Инвест* можно эффективно и корректно решить такие задачи, как подготовка финансовых разделов технико-экономического обоснования и бизнес-планов, моделирование и оптимизация схемы осуществления проекта, проведение экспертизы инвестиционных проектов, ранжирование инвестиционных проектов [1].

3. *Альт-Инвест Сумм* — компьютерная модель, предназначенная для оценки инвестиционных проектов различных отраслей, масштабов и направленности. Данный продукт является расширенной версией программы *Альт-Инвест* [1].

4. Программный продукт *Альт-Инвест-Прим* дает пользователю возможность выполнить предварительную оценку коммерческой состоятельности проекта, используя минимум исходной информации. Результатом работы *Альт-Инвест-Прим* являются: получение базовых форм финансовой оценки проекта; расчет коэффициентов, характеризующих финансовую состоятельность проекта; определение чистой текущей стоимости проекта, внутренней нормы прибыли, срока окупаемости и других показателей эффективности инвестиций; анализ чувствительности проекта к изменению

основных параметров; моделирование поведения проекта при использовании любой схемы финансирования [1].

5. Программный комплекс *ИНЭК-Аналитик* предназначен для решения задач финансово-экономического анализа, мониторинга, инвестиций, сравнения, планирования и план-факторного контроля предприятий, занимающихся производством и(или) оказанием услуг [3].

6. Программный комплекс *ИНЭК-Холдинг* предназначен для финансово-экономического анализа, мониторинга, сравнения, планирования и план-факторного контроля предприятий и организаций любых видов деятельности (производство, предоставление услуг, торговля и пр.) [3].

7. Программный продукт *Информационная система оценки и анализа инвестиционной привлекательности предприятий, эффективности бизнеса и инвестиционных проектов* (ИС оценки и анализа инвестиционной привлекательности предприятий) построен на платформе программы 1С: Предприятие, является независимой информационной базой (конфигурацией) [5, 6].

Система поддерживает ввод, расчет и хранение основных показателей будущих инвестиционных проектов. Рабочие инструменты системы позволяют:

- проводить экономическую экспертизу инвестиционных проектов;
- сравнивать и выбирать наиболее эффективный и оптимальный из множества инвестиционных проектов, построенных на разных методологических схемах;
- рассчитывать и прогнозировать значения наиболее важных экономических показателей будущего инвестиционного проекта;
- строить разнообразные графики, диаграммы и таблицы по результатам проведенного анализа.

Благодаря особенности платформы 1С: Предприятие созданный программный продукт сочетает в себе положительные стороны "открытых" (опытный пользователь может изменять алгоритм "под себя") и "закрытых" систем (для внесения изменений в конфигурацию необходимо иметь практический опыт работы в системе 1С: Предприятие).

1.3. Стоимость программных продуктов

Стоимость описанных выше программных продуктов приведена ниже в таблице. Следует отметить, что внедрение, сопровождение и настройка программ под конкретную организацию осуществляются специалистами фирмы-поставщика за отдельную плату.

Описанные выше программы являются универсальными, наиболее популярными и используются во многих государственных организациях и частных компаниях. Часть этих программных продуктов в качестве основной программной среды использует пакет MS Office, поэтому, покупая программу оценки инвестиционных проектов, пользователю придется отдельно приобретать пакет MS Office.

Безусловно, проведенные исследования затронули лишь небольшую часть существующих программных продуктов, применяемых для оценки инвестиционных проектов. В этой области также существует множество малоизвестных или известных малому кругу лиц программных продуктов, созданных в компаниях, учебных заведениях, государственных и частных организациях. В большинстве своем они не являются универсальными и созданы специально для решения задач инвестирования в конкретной структурной модели.

По вопросу сравнения отечественного программного обеспечения с западным можно сказать следующее. Со временем, когда российская экономическая действительность приблизится по осуществлению основных экономических процедур к мировым стандартам, использование западных программных продуктов для оценки и анализа эффективности инвестиционных проектов станет более обоснованным и востребованным. Тогда будет возможным и необходимым описать зарубежные программы в сравнении с отечественными.

2. Выбор программного продукта

Для выбора оптимального программного обеспечения организации или непосредственно пользователю необходимо провести предварительный анализ программного обеспечения, исходя из ряда критериев. Все программные продукты практически сопоставимы по таким критериям, как функциональные возможности, качество программной реализации, удобство пользовательского интерфейса, степень "закрытости" пакета.

Функциональные возможности включают следующее:

- использование современной методики расчета, основанной на имитации потока реальных денег (*cash-flow*);
- минимизация ограничений на горизонт и шаг расчета;
- минимизация ограничений на количество продуктов;
- широта набора финансовых экономических показателей, используемых для оценки проекта;
- разнообразие сценариев реализации проекта;
- возможность и способы учета инфляции, расчеты в неизменных и текущих ценах;

Стоимость программных продуктов по работе с инвестиционными проектами (по данным источников фирм-разработчиков по состоянию на начало 2007 г.)

№ п/п	Программный продукт	Стоимость, руб.					
		Несетевая версия	Сетевая версия (число рабочих мест)				
			1	3	5	10	20
1	Альт-Финансы	18 500					
2	Мастер финансов	9300					
3	ПМК КИС: Бюджетирование	30 000					
4	ИНЭК-АФСП	35 400	—	67 500	79 500	109 500	169 500
5	ИНЭК-АДП	48 000	—	88 800	102 000	135 000	201 000
6	Audit Expert 3 Professional	33 300	34 800	53 400	66 000	89 400	125 100
7	Audit Expert 4 Standard	51 300	55 800	74 700	90 000	122 700	180 900
8	PIC Holding 7	99 000	100 500	150 000	181 500	236 700	312 600
9	Project Expert 7 Professional	75 000	76 500	114 600	139 200	183 000	243 900
10	Project Expert 7 Standard	38 100	39 600	60 600	74 400	100 200	138 600
11	Альт-Инвест	38 600					
12	Альт-Инвест Сумм	59 300					
13	Альт-Инвест-Прим	26 200					
14	ИНЭК-Аналитик	75 000	—	122 400	136 800	172 800	244 800
15	ИНЭК-Холдинг	105 000	—	163 800	178 800	216 300	291 300
16	ИС оценки и анализа инвестиционной привлекательности предприятий (* — с учетом платформы)	2000 *4000	2000 *16 400	5000 *19 400	7000 *21 400	12 000 *26 400	22 000 *36 400
		Стоимость платформы 1С 7.7: несетевая версия — 3000, сетевая версия — 14 400					

- возможность и способы учета неопределенности и рисков;
- возможность сохранения в памяти ЭВМ приемлемых вариантов расчета для последующего сравнения и окончательного отбора.

Качество программной реализации определяется:

- возможностью реализации пакета на распространенных типах ЭВМ с использованием достаточно доступной операционной среды;
- надежностью в работе;
- быстродействием, позволяющим в приемлемые сроки проводить расчеты необходимого числа вариантов, их сравнение, учет неопределенности;
- защитой пакетов от несанкционированного использования и копирования.

Удобство пользовательского интерфейса предполагает:

- упрощение и минимизацию трудоемкости и ввода информации;
- возможность защиты от ошибок при вводе;
- наглядность результатов;
- достаточный объем графической информации.

Под "закрытостью" пакета понимают невозможность изменения пользователем формул и алгоритмов, по которым происходят вычисления в программном пакете. К "закрытым" относят пакеты, написанные на каком-либо языке программирования и функционирующие вне специфической среды, к "открытым" — написанные на базе электронных таблиц в соответствующей среде, где пользователь имеет возможность модифицировать формулы. "Закрытость" пакета может быть как достоинством, так и недостатком в зависимости от целей, которые стоят перед инвестором. Достаточно квалифицированные пользователи могут имитировать довольно сложные и разнообразные сценарии реализации проекта, используя гибкие возможности "открытой" системы. В то же время такие возможности пакета таят в себе угрозу ошибок. Заметим, что при использовании "закрытых" пакетов возможность ошибок возникает вследствие отсутствия алгоритма, позволяющего следить за расчетом, а также ввиду невозможности модификации расчетных формул с учетом особенностей проекта.

Критерии, по которым проводится выбор программного обеспечения, можно разделить на три группы:

1) операционные критерии, относящиеся к функциональным возможностям программного обеспечения;

2) критерии, по которым оценивается возможность функционирования программного обеспечения в рамках любой информационно-управляющей системы; они соотносятся с требованиями программного обеспечения к аппаратным средствам и оборудованию, возможностью интеграции с другими приложениями;

3) критерии, связанные с затратами на программное обеспечение, — покупка, инсталляция,

оплата технической поддержки, обслуживание на протяжении всего времени функционирования.

Высокая стоимость, необходимость затрат временных ресурсов и системные требования отдельных программных продуктов не позволили автору проверить возможность использования собственных научных и практических наработок в их среде. Однако тщательное изучение инструкций, рекомендаций, описаний и отзывов о работе этих программ дали возможность автору статьи сделать вывод, что в настоящее время большинство предприятий малого бизнеса не могут позволить себе тратить такие финансовые и временные ресурсы на покупку и освоение методик работы отдельных программных продуктов, описанных выше. Это приведет и уже приводит к созданию большого числа малых программных продуктов, удовлетворяющих требованиям именно малых предприятий и организаций.

Список литературы

1. http://softmagazin.ru
2. http://www.expert-systems.com
3. http ://www.inec.ru
4. **Важдаев А. Н.** Модель благоприятного инвестиционного климата компании // Материалы докладов Всероссийской научно-технической конференции студентов, аспирантов и молодых специалистов, Томск, 4—7 мая 2006 г. Ч. 4. Томск: В-Спектр, 2006. С. 228—230.
5. **Важдаев А. Н.** Информационная система оценки эффективности бизнеса и инвестиций. Конференция // Сборник материалов Конференции лауреатов и стипендиатов Международного научного фонда экономических исследований академика Н. П. Федоренко. Москва, 1 декабря 2005 года. М.: ЦЭМИ РАН, 2006. С. 16—23.
6. **Важдаев А. Н.** Использование программного продукта 1С: Предприятие для создания Информационной системы оценки и анализа инвестиционных проектов: Новые информационные технологии в образовании // Доклады и выступления участников Седьмой международной научно-практической конференции "Использование программных продуктов фирмы "1С" в инновационной деятельности учебных заведений", 30—31 января 2007 г. Москва, 2007. С. 297—300.

УДК 519.86

А. Н. Соломатин, канд. физ.-мат. наук,
Вычислительный центр
им. А. А. Дородницына РАН

Формирование оптимального состава бизнес-единиц компании с учетом стратегий их развития

Рассматривается задача формирования портфеля диверсифицированной компании, обеспечивающего максимизацию накопленной прибыли компании при наличии у бизнес-единиц различных возможных стратегий развития. Для решения этой неполиномиальной задачи предложен комбинированный алгоритм, основанный на совместном использовании метода ветвей и границ и метода последовательных расчетов.

Введение

Последнее время в России и за рубежом интенсивно развивается наука стратегического управления, что обусловлено постоянным ростом сложности и нестабильности современной бизнес-среды [1]. Однако эффективность стратегического управления снижается из-за недостаточного применения современных информационных технологий, что во многом связано с преимущественно качественным характером используемых моделей и методов. В работе [2] были предложены некоторые новые формализованные модели стратегического управления и концепция разработки соответствующей системы автоматизации.

Важным этапом процесса стратегического управления является портфельный анализ, задача которого состоит в определении состава и связей стратегических бизнес-единиц (СБЕ) диверсифицированной компании, позволяющих оптимизировать денежные потоки компании [3]. Отдельные СБЕ могут соответствовать различным подразделениям компании, товарам (услугам), предлагаемым на рынке, или различным направлениям деятельности компании, т. е. любым компонентам, которые могут иметь самостоятельные стратегии развития. Портфель должен быть сбалансирован, т. е. должно быть обеспечено правильное сочетание СБЕ, испытывающих потребность в капитале для обеспечения роста, и СБЕ, располагающих избытком капитала.

В настоящей статье исследуется и решается одна из задач, поставленных в [2], — задача формирования сбалансированного портфеля компании, обеспечивающего максимизацию накопленной прибыли; при этом дополнительно предполагается, что различные бизнес-единицы компании могут выбирать одну из нескольких возможных стратегий развития. Другими словами, одновременно решаются как задача оптимизации портфельной стратегии компании, так и задача оптимизации ее конкурентных стратегий. Для решения этой неполиномиальной задачи предложен комбинированный алгоритм, основанный на совместном ис-

пользовании метода ветвей и границ [4, 5] и метода последовательных расчетов [6, 7].

1. Модель и постановка задачи оптимизации

Пусть задано множество стратегических бизнес-единиц I мощностью $n = |I|$, каждая из которых является кандидатом на включение в состав диверсифицированной компании, т. е. портфель компании может быть образован на основе любого подмножества СБЕ $\omega \subseteq I$.

Делаются следующие основные предположения, характеризующие модель диверсифицированной компании:

- функционирование и развитие компании рассматриваются в динамике на заданном периоде времени длиной T, причем учитывается дисконтирование с заданной ставкой δ;
- для каждой СБЕ из ω может быть выбрана одна из нескольких возможных стратегий развития из заданного множества J, $m = |J|$;
- для каждой j-й стратегии каждой СБЕ с номером $i \in I$ в рамках заданного периода планирования задана прогнозная динамика средств, которые продуцируются данной СБЕ — $d_i^j(t)$ и которые ею потребляются — $e_i^j(t)$;
- тем самым для каждой СБЕ с номером i можно определить функцию прибыли вида $f_i^j(t) = d_i^j(t) - e_i^j(t)$;
- задана функция прибыли $F_0(\omega)$, отражающая баланс доходов и затрат, которые относятся ко всей компании (а не к отдельным СБЕ), рассматриваемой как единое целое (см. п. 5).

Стратегия развития каждой СБЕ из ω определяется с помощью матрицы $X(\omega)$ размерностью $n \times m$, зависящей от ω, такой, что $X_{ij}(\omega) = 1$, если i-я СБЕ, $i \in \omega$, использует j-ю стратегию, $j \in J$; $X_{ij}(\omega) = 0$ в противном случае либо если $i \in I \backslash \omega$. Поскольку для каждой СБЕ может быть выбрана единственная стратегия, в каждой строке матрицы $X(\omega)$ может находиться не более одной единицы, т. е.

$$\sum_{j=1}^{m} X_{ij}(\omega) = 1 \text{ для любого } i \in \omega;$$

$$\sum_{j=1}^{m} X_{ij}(\omega) = 0 \text{ для любого } i \in I \backslash \omega.$$

Можно сформулировать следующую оптимизационную задачу: определить такую совокупность СБЕ компании $\omega^* \subseteq I$ и такую стратегию развития каждой СБЕ из ω^* (задается матрицей $X^*(\omega)$), чтобы максимизировать накопленную прибыль всей компании:

$$\sum_{i \in \omega} \sum_{j=1}^{m} \sum_{t=1}^{T} X_{ij}(\omega) f_i^j(t) \exp(-\delta t) + F_0(\omega) \to \max_{\omega \subseteq I} \max_{X(\omega)}. \quad (1)$$

Ограничения для задачи (1) определим ниже на основе анализа свойств соответствующих функций.

2. Свойства функций прибыли бизнес-единиц

На вид функций $f_i^j(t)$, $t \in \{1, 2, ..., T\}$, не накладывается особых ограничений: это могут быть достаточно произвольные кусочно-гладкие функции, принимающие как положительные, так и отрицательные значения.

Накопленная функция прибыли СБЕ $F_i^j(t)$ отражает суммарный объем прибыли СБЕ с номером i, полученный к моменту времени t при использовании j-й стратегии, и определяется как

$$F_i^j(t) = \sum_{t'=1}^{t} f_i^j(t') \exp(-\delta t'); \quad (2)$$

$$F_i^j(T) = F_i^j.$$

Поскольку СБЕ функционируют в составе единой компании и могут использовать денежные средства друг друга (в оптимизации такого взаимодействия как раз и состоит смысл решаемой задачи), то, вообще говоря, любая СБЕ может быть убыточна как в некоторые моменты времени ($F_i^j(t) < 0$), так и в целом ($F_i^j < 0$). На самом деле возможен случай, когда такая СБЕ будет служить источником прибыли для других СБЕ в течение некоторого периода времени, а потом ее деятельность будет приносить убытки. Однако если СБЕ устойчиво убыточна после некоторого момента времени t', т. е. $F_i^j(t) < 0$ для $t' < t \leqslant T$, то такая СБЕ должна быть ликвидирована, т. е. $F_i^j(t) = 0$ для $t' < t \leqslant T$.

Накопленная функция прибыли компании $F^*(\omega, X, t)$ отражает суммарный объем прибыли всех СБЕ компании, полученный к моменту времени t, если каждая СБЕ с номером $i \in \omega$ использует стратегию развития с номером j, зависящим от i:

$$F^*(\omega, X, t) = \sum_{i \in \omega} \sum_{j=1}^{m} \sum_{t'=1}^{t} X_{ij}(\omega) f_i^j(t') \exp(-\delta t') = \sum_{i \in \omega} \sum_{j=1}^{m} X_{ij}(\omega) F_i^j(t). \quad (3)$$

Очевидно, что ни в один из годов периода планирования деятельность всей компании не может быть убыточной, т. е.

$$F^*(\omega, X, t) \geqslant 0 \text{ для любого } t \in \{1, 2, ..., T\}. \quad (4)$$

Относительно этого важного условия справедливо следующее:

- условие отражает взаимные связи СБЕ компании, т. е. служит решению важнейшей задачи портфельного анализа — обеспечить сбалансированность денежных потоков диверсифицированной компании;
- условие должно проверяться для каждого рассматриваемого набора стратегий;
- при отсутствии условия (4) поиск оптимального набора стратегий для каждой СБЕ из ω не имеет смысла: в силу аддитивности функционала (3) для каждой СБЕ достаточно просто взять наиболее прибыльную стратегию;
- наличие условия (3) делает невозможным применение стандартной схемы метода ветвей и границ для поиска оптимального решения: такое решение может не удовлетворять (4), а другие решения, близкие к оптимальному и удовлетворяющие (4), будут отброшены.

Обозначим накопленную функцию прибыли для всего периода времени как

$$F(\omega, X) = F^*(\omega, X, T).$$

Отметим, что $F(\omega, X)$ является аддитивной неотрицательной функцией, что следует из (3) и условия (4).

Кроме того, на область значений функции $F(\omega, X)$ накладывается следующее ограничение:

$$F(\omega, X) \geqslant L, \quad (5)$$

т. е. компания должна в итоге обеспечить заданный уровень прибыльности, отражающий целевые установки менеджмента и собственников компании.

С учетом всего вышесказанного задача (1) может быть сформулирована следующим образом:

$$\sum_{i \in \omega} \sum_{j=1}^{m} \sum_{t=1}^{T} X_{ij}(\omega) f_i^j(t) \exp(-\delta t) + F_0(\omega) \rightarrow \rightarrow \max_{\omega \subseteq I} \max_{X(\omega)} \quad (6)$$

при условиях $F^*(\omega, X, t) \geqslant 0$, $t \in \{1, 2, ..., T\}$;

$\sum_{j=1}^{m} X_{ij}(\omega) = 1$ для любого $i \in \omega$;

$\sum_{j=1}^{m} X_{ij}(\omega) = 0$ для любого $i \in I \backslash \omega$) и ограничении $F(\omega, X) \geqslant L$.

3. Сложность задачи и подход к решению задачи

Оценим вычислительную сложность задачи (1). Поиск оптимального набора стратегий для каждого множества СБЕ ω должен проводиться на множестве всех кортежей декартова произведения J^k, $k = |\omega|$, что потребует $O(m^k)$ операций. Далее, проверка условия (4) для каждого решения требует $O(kT)$ операций. Тогда с учетом перебора различных подмножеств СБЕ из множества I временная сложность задачи составит

$$O\left(\sum_{\omega \subset I} |\omega| \, T m^{|\omega|}\right). \quad (7)$$

Преобразуем сумму (7):

$$\sum_{\omega \subseteq I} |\omega| \, T m^{|\omega|} \leqslant \sum_{\omega \subseteq I} |I| \, T m^{|I|} = \sum_{\omega \subseteq I} n \, T m^n = 2^n n T m^n = n T (2m)^n. \quad (8)$$

Таким образом, данная неполиномиальная задача имеет сложность $O(nT(2m)^n)$ и является чрезвычайно трудоемкой. Например, даже при таких небольших значениях параметров, как $m = n = T = 10$, оценка числа операций составит $O(10 \times 10 \times 20^{10}) = O(10^2 \times 2^{10} \times 10^{12}) = O(10^{16})$.

Кроме того, сложность задачи (6) определяется следующими причинами:

- данная задача принадлежит к классу нелинейных многоэкстремальных комбинаторных задач дискретной оптимизации;
- необходимо проводить оптимизацию одновременно по двум направлениям: и по составу СБЕ, входящих в состав компании, и по составу стратегий, используемых различными СБЕ;
- необходимо проверять условие (4) для каждого возможного решения, что исключает стандартные способы отбраковки неперспективных решений.

Предлагается следующий подход к решению задачи (6), основанный на одновременном применении таких известных методов дискретной оптимизации, как метод ветвей и границ (МВГ) [4, 5] и метод последовательных расчетов (МПР) [6, 7]; использование МВГ рассматривается в п. 4, использование МПР — в п. 5.

Для каждого фиксированного множества СБЕ $\omega \subseteq I$ может быть найден оптимальный набор стратегий всех СБЕ из ω, т. е. матрица $X^*(\omega)$. Для решения этой задачи используется модифицированный метод ветвей и границ, позволяющий искать не только оптимальное, но и близкие к нему решения. В результате становится возможным найти значения функционала

$$P(\omega) = F(\omega, X^*) + F_0(\omega).$$

А для оптимизации этого функционала, т. е. для решения исходной задачи (6), используется

другой метод дискретной оптимизации — метод последовательных расчетов.

4. Применение метода ветвей и границ для нахождения оптимальных стратегий развития бизнес-единиц

В данном разделе рассматривается решение следующей оптимизационной задачи. Для фиксированного множества СБЕ $\omega \subseteq I$ найти для каждой СБЕ с номером $i \in \omega$ такую стратегию развития $j \in J$, зависящую от i, чтобы максимизировать накопленную дисконтированную прибыль всей компании, т. е. найти матрицу $X^*(\omega)$ такую, что

$$F(\omega, X^*) = \max_{X(\omega)} \sum_{i \in \omega} \sum_{j=1}^{m} \sum_{t=1}^{T} X_{ij}(\omega) f_i^j(t) \exp(-\delta t) \quad (9)$$

при условиях $F^*(\omega, X, t) \geqslant 0$, $t \in \{1, 2, ..., T\}$;

$$\sum_{j=1}^{m} X_{ij}(\omega) = 1 \text{ для любого } i \in \omega,$$

$$\sum_{j=1}^{m} X_{ij}(\omega) = 0 \text{ для любого } i \in I \setminus \omega)$$

и ограничении $F(\omega, X) \geqslant L$.

На первый взгляд может показаться, что решение данной задачи тривиально. Поскольку для каждой j-й стратегии развития, $j \in J$, каждой СБЕ с номером $i \in \omega$ известно значение дисконтированной накопленной прибыли F_i^j, то можно построить оптимальное решение $\overline{X}(\omega)$ так, что для каждой СБЕ будет выбрана наиболее прибыльная для нее стратегия, т. е.

$$F(\omega, \overline{X}) = \sum_{i \in \omega} \overline{F}_i, \ \overline{F}_i = \max_{j \in J} F_i^j. \quad (10)$$

Однако для этого решения может не выполняться условие (4), что потребует нахождения решения, которое является ближайшим к оптимальному (по значению функционала) и для которого выполняется (4). Предлагается использовать модифицированный метод ветвей и границ [5], который наряду с оптимальным решением находит также все решения, близкие к оптимальному, т. е. отличающиеся от него на заданную величину R. Тогда решение задачи проводится следующим образом.

1. В качестве величины R берется $F(\omega, \overline{X}) - L$, где $\overline{X}(\omega)$ — оптимальное решение задачи (9) без учета проверки условия (4), найденное в соответствии с (10), а L — нижнее ограничение на суммарную накопленную прибыль компании из условия (5).

2. В результате работы МВГ находятся решения задачи (9), отличающиеся от оптимального решения не более чем на R и упорядоченные по убыванию значения функционала $F(\omega, X)$.

3. Полученные близкие решения просматриваются прямым перебором, и для каждого из них проверяется выполнение условия (4).

4. Первое решение, для которого выполняется условие (4), считается оптимальным решением задачи (9), и работа оканчивается с положительным результатом.

5. Если ни одно из R-близких решений не удовлетворяет условию (4), то работа оканчивается с отрицательным результатом: не нашлось ни одного решения задачи (9), удовлетворяющего и условию (4), и ограничению на прибыль (5).

Рассмотрим теперь особенности применения метода ветвей и границ для решения задачи (9).

1. При работе алгоритма используется матрица S размерностью $m \times k$, $k = |\omega|$, в которой хранятся значения накопленной прибыли для каждой j-й стратегии каждой l-й СБЕ с номером i ($l \in \{1, 2, ..., k\}$), т. е. $s_{jl} = F_i^j$, $i \in \omega$, $j \in J$. После построения матрицы S она специальным образом упорядочивается:

- для каждого l-го столбца матрицы элементы столбца упорядочиваются по убыванию, в результате чего в первой строке матрицы S для каждой l-й СБЕ будет находиться стратегия, обеспечивающая максимальную накопленную прибыль;
- столбцы матрицы упорядочиваются по убыванию среднего арифметического значений элементов s_{ji} для каждого столбца.

2. Способ разбиения множеств решений отражает специфику рассматриваемой задачи. Ветвление множеств может проводиться не более чем на k уровнях, причем на каждом уровне с номером l выбранное для ветвления множество G_l^j разбивается на m непересекающихся подмножеств следующего $(l + 1)$-го уровня. Каждое множество G_l^j содержит все возможные решения задачи (9) такие, что стратегии для СБЕ с номерами, не превосходящими l, совпадают.

3. Верхняя оценка $g(G_l^j)$ для каждого множества G_l^j находится по следующей формуле:

$$g(G_l^j) = \sum_{p=1}^{l} s_{j(p), p} + \sum_{p=l+1}^{k} s_{1, p}. \quad (11)$$

Индекс $j(p)$ означает, что для каждой p-й СБЕ выбирается своя стратегия, зависящая от p. Использование элементов $s_{1, p}$ во второй сумме означает, что при расчете оценки для СБЕ с номерами, большими номера уровня разбиения l, берутся наиболее прибыльные стратегии, стоящие в первой строке матрицы S.

Например, для матрицы S на рисунке ($m = 3$, $k = 4$) множество G_2^3 содержит все решения, в которых для первой СБЕ выбрана вторая стратегия, а для второй СБЕ — третья; при этом оценка $g(G_2^3) = 12 + 9 + 10 + 8 = 39$.

15	11	**10**	**8**	
12	10	6	5	
7	**9**	3	2	**Верхняя оценка $g(G_2^3)$**

4. Поскольку при расчете каждой оценки $g(G_l^j)$ для СБЕ с номерами, большими l, берутся наиболее прибыльные стратегии, то значение такой оценки будет не меньше, чем значение функционала для любого решения из G_l^j. Другими словами, оценка (11) на самом деле является верхней оценкой.

5. Необходимым условием применения МВГ является неубывание нижних оценок для задач минимизации (соответственно невозрастание верхних оценок для задач максимизации). Очевидно, что верхняя оценка (11) не возрастает по мере ветвления. На самом деле по мере ветвления число слагаемых первой суммы формулы (11) увеличивается от 1 до k, а второй — уменьшается от $k-1$ до 0. При этом если во второй сумме складываются значения F_i^j для наиболее прибыльных стратегий каждой СБЕ, то в первой — значения для произвольных стратегий.

6. Для дальнейшего ветвления выбирается множество G_l^j с максимальной оценкой $g(G_l^j)$, причем выбор проводится только среди множеств данного l-го уровня.

7. В качестве рекорда $\bar{X}(\omega)$ берется оптимальное решение задачи (9) без учета проверки условия (4), т. е.

$$F(\omega, \bar{X}) = \sum_{p=1}^{k} s_{1,p}. \qquad (12)$$

Очевидно, что в процессе работы МВГ дальнейшего улучшения рекорда не проводится.

8. Отсев некоторого множества G_l^j осуществляется в том случае, если выполняется условие

$$g(G_l^j) < F(\omega, \bar{X}) - R, \qquad (13)$$

при этом использование R в условии отсева позволяет не отсеивать решения, близкие к оптимальному.

9. После получения очередного близкого решения это решение сохраняется в массиве близких решений в порядке уменьшения $F(\omega, X)$.

10. После получения массива R-близких решений для каждого из этих решений проверяется условие (4). Поскольку решения упорядочены по убыванию значений функционала $F(\omega, X)$, то самое первое решение, для которого выполняется (4), будет оптимальным решением задачи (9). Если ни для одного из R-близких решений условие (4) не выполняется, то задача (9) не имеет решения. На самом деле если для некоторого $R^* > R$ и существует решение, для которого выполняется условие (4), то в силу $R = F(\omega, \bar{X}) - L$ это решение не будет удовлетворять ограничению (5).

11. Следует отметить, что при решении задачи (9) МВГ работает достаточно быстро, поскольку рекордом является оптимальное решение задачи, а способ построения матрицы S позволяет находить решения, близкие к оптимальному, в первую очередь.

5. Применение метода последовательных расчетов для нахождения оптимального состава бизнес-единиц

Для решения оптимизационной задачи (6) было предложено использовать метод последовательных расчетов, предложенный В. П. Черениным и развитый В. Р. Хачатуровым [6, 7]. В данном методе рассматриваются функции $P(\omega)$, определенные на множестве всех подмножеств ω множества $I = \{1, 2, ..., n\}$; очевидно, что число элементов ω составляет 2^n.

Достаточным условием применимости МПР для определения экстремума $P(\omega_0) = \min P(\omega)$ по всем $\omega \subseteq I$ является выполнение следующего неравенства для любых двух подмножеств δ и γ множества I:

$$S(\delta, \gamma) = P(\delta) + P(\gamma) - P(\delta \cup \gamma) - P(\delta \cap \gamma) \leqslant 0. \qquad (14)$$

Функции $P(\omega)$, удовлетворяющие условию $S(\delta, \gamma) \leqslant 0$, называются супермодулярными. Функции $P(\omega)$, удовлетворяющие условию $S(\delta, \gamma) \geqslant 0$, называются субмодулярными. Наконец, функции $P(\omega)$, для которых $S(\delta, \gamma) = 0$, называются модулярными.

Приведем в соответствии с [7] ряд определений.

Определение 1. Функция $P(\omega)$ на подмножестве $\alpha \subseteq I$ имеет локальный минимум (а само подмножество α называется локальным минимумом), если выполняются условия:

$$P(\alpha) \leqslant P(\alpha \cup \{i\}) \text{ для всех } i \in I \backslash \alpha;$$

$$P(\alpha) \leqslant P(\alpha \backslash \{i\}) \text{ для всех } i \in \alpha.$$

Определение 2. Функция $P(\omega)$ на подмножестве $\alpha \subseteq I$ имеет глобальный минимум, если для любого $\omega \subseteq I$ справедливо $P(\alpha) \leqslant P(\omega)$, при этом подмножество α называется глобальным минимумом или оптимальным подмножеством.

Определение 3. Рядом подмножеств, соединяющим подмножества ω^r и ω^t ($\omega^r \subset \omega^t$), называется последовательность подмножеств множества I вида $\omega^r, \omega^{r+1}, ..., \omega^p, ..., \omega^{t-1}, \omega^t$, где $|\omega^p| = p$, $\omega^{p-1} \subset \omega^p$, $r \geqslant 0$, $t \leqslant n$, а p последовательно пробегает значения от r до t. Ряд подмножеств называется главным, если $r = 0$, $t = n$, т. е. главный ряд имеет следующий вид:

$$\{0\} = \omega^0, \omega^1, ..., \omega^p, ..., \omega^n = I.$$

Определение 4. Функция $P(\omega)$ называется монотонно убывающей по ряду подмножеств, если для любых $\omega^1 \subset \omega^2$ из этого ряда выполняется $P(\omega^1) \geqslant P(\omega^2)$; аналогично определяется монотонно возрастающая функция.

Теорема (основная теорема МПР). На любом главном ряду подмножеств, содержащем локальный минимум α, функция $P(\omega)$, удовлетворяющая условию супермодулярности (14), монотонно убывает вплоть до α и монотонно возрастает после α.

Из основной теоремы и ее следствий вытекают правила отбраковки подмножеств, позволяющие исключать из рассмотрения большие группы подмножеств, не теряя при этом глобальный минимум [6].

Были сформулированы и доказаны также три обобщенных правила отбраковки, применение которых позволяет найти не только оптимальное подмножество α, но и все близкие к оптимальному подмножества ω, т. е. $\omega \subseteq I$, для которых $P(\alpha) \leqslant P(\omega) \leqslant P(\alpha) + R$ для некоторого фиксированного $R \geqslant 0$. Приведем в качестве примера первое правило.

Первое обобщенное правило отбраковки. Если для каких-либо подмножеств $\omega^1 \subset \omega^2 \subseteq I$ известны значения $P(\omega^1)$ и $P(\omega^2)$ и если $P(\omega^1) + R < P(\omega^2)$, то можно исключить из рассмотрения все $n - |\omega^2|$ подмножеств $\omega^2 \subset \omega$, так как для них заведомо $P(\omega) > P(\alpha) + R$.

Обобщенные правила отбраковки используются в модифицированном алгоритме последовательных расчетов для определения оптимального и всех близких к оптимальному подмножеств $\omega \subseteq I$. Данный алгоритм состоит из двух этапов. На этапе предварительной отбраковки последовательно применяются первое и второе правила отбраковки, что позволяет исключить из рассмотрения подмножества $\omega \subseteq I$, заведомо не являющиеся оптимальными. На втором этапе работает основной алгоритм МПР, который, используя первое и третье правила отбраковки, позволяет определить оптимальное и все близкие к нему подмножества $\omega \subseteq I$.

Как показала практика, использование правил отбраковки в алгоритме последовательных расчетов позволяет определять глобальный минимум функции $P(\omega)$, просматривая порядка kn^3 вариантов из общего числа 2^n вариантов, где $1/n^2 < k < n^2$.

Рассмотрим использование МПР для решения поставленной задачи оптимизации (6). Поскольку алгоритм последовательных расчетов может быть применен для минимизации супермодулярной функции $P(\omega)$, то он может быть применен и для максимизации субмодулярной функции $-P(\omega)$. Поэтому для применения МПР при максимизации функции $P(\omega) = F(\omega, X^*) + F_0(\omega)$ следует доказать ее субмодулярность.

Докажем вначале модулярность функции $F(\omega, X^*)$. Очевидно, что для этой аддитивной функции (сумма значений накопленной прибыли для всех СБЕ из ω) справедливо соотношение $F(\delta \cup \gamma) = F(\delta) + F(\gamma)$ для любых подмножеств δ, $\gamma \subseteq I$ таких, что $\delta \cap \gamma = \varnothing$.

Представим подмножества δ и γ как $\delta = \delta^* \cup \sigma$, $\gamma = \gamma^* \cup \sigma$, где $\sigma = \delta \cap \gamma$. Тогда $F(\delta) + F(\gamma) = F(\delta^*) + F(\sigma) + F(\gamma^*) + F(\sigma) = F(\delta^* \cup \sigma \cup \gamma^*) + F(\sigma) = F(\delta \cup \gamma) + F(\delta \cap \gamma)$, что и требовалось доказать.

Чтобы функция $P(\omega) = F(\omega, X^*) + F_0(\omega)$ была субмодулярной при модулярности $F(\omega, X^*)$, субмодулярной должна быть функция $F_0(\omega)$.

Представим $F_0(\omega)$ в виде

$$F_0(\omega) = I(\omega) - E(\omega), \qquad (15)$$

где $I(\omega) = \alpha \lg|\omega|$ отражает дополнительную прибыль всей компании, возникающую от взаимодействия СБЕ в рамках единой системы, включая воздействие внутрифирменных цен; $E(\omega) = \beta|\omega|^2$ отражает дополнительные затраты всей компании, состоящей из многих СБЕ, возникающие вследствие усложнения управления; коэффициенты α и β определяются спецификой конкретной компании.

Докажем субмодулярность функции $I(\omega)$ для любых подмножеств δ, $\gamma \subseteq$ I. Пусть $\delta = \delta^* \cup \sigma$, $\gamma = \gamma^* \cup \sigma$, $\sigma = \delta \cap \gamma$ и $a = |\delta^*|$, $b = |\gamma^*|$, $x = |\sigma|$.

Тогда

$$I(\delta) + I(\gamma) = \alpha \lg(a + x) + \alpha \lg(b + x) = \alpha \lg((a + x)(b + x)) = \alpha \lg(ab + ax + bx + x^2);$$

$$I(\delta \cup \gamma) + I(\delta \cap \gamma) = \alpha \lg(a + b + x) + \alpha \lg(x) = \alpha \lg(ax + bx + x^2) = \alpha \lg((ab + ax + bx + x^2) - ab) \leqslant \alpha \lg(ab + ax + bx + x^2) = I(\delta) + I(\gamma).$$

Аналогично доказывается супермодулярность функции $E(\omega)$, т. е. субмодулярность функции $-E(\omega)$. Поэтому функция $F_0(\omega)$ является субмодулярной, так же как и функция $P(\omega) = F(\omega, X^*) + + F_0(\omega)$, что делает возможным применение базового алгоритма последовательных расчетов для нахождения глобального максимума этой функции, т. е. решения задачи (6).

Список литературы

1. **Зуб А. Т.** Системный стратегический менеджмент: теория и практика. М.: Генезис, 2001. 751 с.
2. **Соломатин А. Н.** Модели и средства автоматизации стратегического управления газодобывающими предприятиями. М.: ВЦ РАН, 2005. 40 с.
3. **Ансофф И.** Новая корпоративная стратегия / Пер. с англ. СПб.: Питер Ком, 1999. 414 с.
4. **Корбут А. А., Финкельштейн Ю. Ю.** Дискретное программирование. М.: Наука, 1969. 368 с.
5. **Сигал И. Х., Иванова А. П.** Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. М.: Физматлит, 2002. 240 с.
6. **Хачатуров В. Р.** Математические методы регионального программирования. М.: Наука, 1989. 304 с.
7. **Хачатуров В. Р., Веселовский В. Е., Злотов А. В.** и др. Комбинаторные методы и алгоритмы решения задач дискретной оптимизации большой размерности. М.: Наука, 2000. 354 с.

УДК 004.891.2

Л. Е. Муханов,
ЗАО МЦСТ
lmukhanov@fraudprevention.ru

Адаптация модели сетей Байеса для обнаружения мошенничества с платежными картами

Число случаев мошенничества с использованием пластиковых карт постоянно увеличивается. Поэтому в настоящее время есть потребность в системе предотвращения мошенничества, большинство функций принятия решений которой было бы автоматизировано. Приводится описание базовых моделей разработанной системы обнаружения и предотвращения мошенничества — "простого" классификатора Байеса и сетей Байеса. Также рассматриваются их недостатки для решения данной задачи и механизмы адаптации. В заключение приводятся результаты оценочного тестирования адаптированных моделей и делаются выводы об эффективности их использования.

Введение

Число людей, использующих платежные карты в качестве средства безналичной оплаты, постоянно растет. Как и в любой сфере деятельности, касающейся финансов или их движения, банковские организации при работе с платежными картами также сталкиваются с мошенничеством. За последние несколько лет сложилась явная тенденция к увеличению числа мошенничеств в этой области несмотря на постоянное усовершенствование систем защиты. Отмечается ежегодный рост потерь международных платежных сетей и банков из-за подобного вида преступлений. Поэтому в настоящий момент существует потребность в системе, которая позволяла бы среди всего потока транзакций, обрабатываемых процессинговой системой банка, достаточно быстро выявлять "подозрительные" и предотвращать мошенничество.

По данным аналитической компании "*Frost & Sullivan*", совокупные потери финансовых организаций от карточного мошенничества во всем мире в 2005 г. составили 7,9 млрд долл. США, а к 2009 г. этот показатель должен вырасти до 15,5 млрд долл. США [1]. Наибольший ущерб банки несут от следующих преступлений:

- предъявление чужих или поддельных карт в магазинах;
- незаконное использование реквизитов карты для оплаты товаров через Интернет.

Первый вид мошенничества широко распространен (в России) из-за того, что продавцы, как правило, не требуют предъявления документов у человека, оплачивающего товар с использованием платежной карты. Также в большинстве POS-терминалов, обслуживающих магазин, не требуется вводить PIN-код. Очевидно, что данные факты только способствуют росту числа мошенничеств с использованием платежных карт [2, 3].

Второй вид мошенничества также распространен из-за простоты его осуществления. В большинстве существующих электронных магазинов для осуществления покупки достаточно ввести только номер карты и код CVV. Подобную информацию злоумышленник может выяснить довольно легко.

В данной статье приводится общее описание разработанной системы обнаружения мошенничества, описание "простого" классификатора Байеса, адаптированной модели сети Байеса и оценка эффективности использования данных моделей для выявления мошенничества.

Модель выявления мошеннических транзакций, базирующаяся на использовании сетей Байеса

"Простой "классификатор Байеса. Данная модель относится к вероятностным моделям классификации данных и является частным случаем модели сетей Байеса. В качестве входных параметров в данной модели используются значения полей (параметров, атрибутов) транзакций, но в некоторых случаях входные параметры могут и не совпадать с этими значениями, а только формироваться на их основе [4, 5].

"Простой" классификатор Байеса базируется на вычислении вероятности того, что данная транзакция принадлежит определенному классу. Эта вероятность определяется следующим образом [6]:

$$P(C = c|\boldsymbol{X} = \boldsymbol{x}) = P(C = c) \times \\ \times P(\boldsymbol{X} = \boldsymbol{x} \mid C = c)/P(\boldsymbol{X} = \boldsymbol{x}); \\ P(C = c \mid \boldsymbol{X} = \boldsymbol{x}) = P(C = c) \times \\ \times \Pi_{i = 1, \ldots, n} P(X_i = x_i \mid C = c)/\Pi_{i = 1, \ldots, n} P(X_i = x_i). \quad (1)$$

Сети Байеса и принцип MDL. В данной модели вероятность того, что транзакция с параметрами x относится к классу c, также вычисляется по формуле (1).

Отличие этой модели от предыдущей заключается в том, что здесь допускается наличие зависимостей между параметрами транзакции [8, 9]. Таким образом, в данной модели для вычисления вероятности (1) необходимо сначала определить вид сети Байеса, т. е. необходимо ввести некоторую количественную характеристику, которая бы позволила охарактеризовать сеть Байеса и найти

наиболее подходящую из всех возможных. В качестве такой характеристики в данной модели будет использоваться MDL-характеристика (*Minimal Description Length*) [8]:

$$\text{MDL} = \log_2 N \cdot |B|/2\text{-LL};$$

$$\text{LL} = N \times \Sigma_{i=1,\ldots,N} \Sigma_{\Pi_{x_i}} I(X_i, \Pi_{x_i});$$

$$I(X, Y) = P(x, y)\log_2(P(x, y)/(P(x) \times P(y))),$$

где N — число параметров транзакции (число узлов в сети Байеса); $|B|$ — число дуг в соответствующей сети Байеса; Π_{x_i} — совокупность всех параметров, от которых зависит параметр X_i, например, если X_1 зависит от X_2 и X_3, то $\Pi_{X_1} = \{X_2, X_3\}$; $\log_2 N \cdot |B|/2$ — параметр, характеризующий число битов, необходимое для хранения соответствующей модели сети Байеса; $I(X; Y)$ — характеризует степень зависимости между X и Y, при вычислении этого параметра проводится суммирование по всем возможным значениям X и Y; LL — параметр, характеризующий суммарную степень зависимости между всеми параметрами в соответствующей сети Байеса.

Таким образом, чем больше зависимостей будет обнаружено (т. е. чем больше LL), тем меньше будет ошибка при осуществлении классификации [9]. Вместе с тем, если в сети Байеса будет учитываться избыточное число зависимостей (т. е. значение $\log_2 N \cdot |B|/2$ увеличится), то это приведет к излишним затратам на хранение самой модели. Из этого следует, что оптимальной сетью Байеса является сеть с минимальным значением MDL.

В данной модели для простоты построения оптимальной сети Байеса (исходя из принципа MDL) делается предположение о том, что X_i ($i = 0, \ldots, n$) может зависеть только от X_j ($j = 0, \ldots, i-1$). Таким образом, найденная сеть может и не являться оптимальной в действительности, но считается таковой исходя из условий, наложенных на зависимость между параметрами транзакции.

Определение оптимальной сети Байеса (исходя из условий, наложенных на зависимости между параметрами сети) будет происходить с использованием следующего алгоритма:

```
Procedure SERCH_MIN_NET
INPUT: B_in; OUTPUT: B_out, minMDL_out;
begin
  1. Compute MDL;
  1. if B_in=NULL then i<-0 else i<-the last
  element in B_in
  2. for i+1<=q<= j-1
     begin
       B<-B_in attached q at the end;
       SEARCH_MIN_NET (B, B_temp, MDL_temp);
       if MDL>minMDL_temp
       then
          begin
             B_out = B_temp
             minMDL_out = MDL_temp
          end
        else
          begin
             B_out = B;
             minMDL_out = MDL;
          end
     end
end
```

Данный алгоритм вычисления позволяет найти оптимальную сеть зависимых параметров только для одного из атрибутов (параметров) транзакции. Таким образом, для нахождения оптимальной сети Байеса необходимо запустить алгоритм SEARCH_MIN_NET для каждого из параметров транзакции [9]. То есть сначала следует запустить алгоритм для параметра, обозначающего класс транзакции C (этой "вершине" будет соответствовать значение i, равное 0). Затем данный алгоритм вычисления необходимо запустить для второго параметра транзакции (X_1), т. е. значение i, равное 0, в данном алгоритме уже будет соответствовать X_1. После этого алгоритм повторяется для всех остальных параметров (X_2, X_3, ..., X_n).

Вероятность того, что транзакция с параметрами x относится к классу c, вычисляется также исходя из найденной оптимальной сети Байеса.

Например, для сети, изображенной на рис. 1, вероятность будет вычисляться по следующей формуле:

$$P(C = c|\boldsymbol{X} = \boldsymbol{x}) = P(C = c) \times P(X_1 = x_1|C = c) \times$$
$$\times P(X_2 = x_2|\, C = c, X_1 = x_1) \times P(X_3 = x_3|\, C = c) \times$$
$$\times P(X_4 = x_4|\, C = c, X_1 = x_1)/(P(X_1 = x_1) \times$$
$$\times P(X_2 = x_2|X_1 = x_1) \times$$
$$\times P(X_3 = x_3) \times P(X_4 = x_4|X_1 = x_1)).$$

Модели формирования статистической информации и вычисления вероятности. В "простом" классификаторе Байеса для формирования статистической информации и вычисления вероятности того, что данная транзакция принадлежит соответствующему классу (в данном случае — классу мошеннических или легальных транзакций), возможно использовать следующие три модели [6].

В *первой модели* предполагается, что параметры транзакций могут принимать только дис-

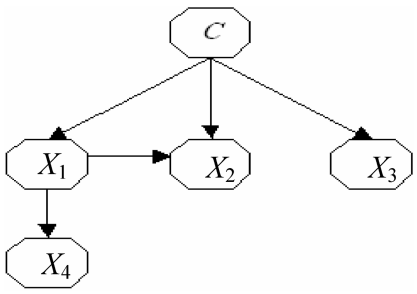


Рис. 1. Сеть Байеса

кретные значения. В случае существования параметров транзакций, значения которых могут принимать непрерывные значения, необходимо использовать механизм, позволяющий осуществлять преобразование от непрерывных значений к дискретным. Вероятность того, что атрибут принимает конкретное значение для данного класса транзакции, вычисляется следующим образом:

$$P(X_i = x \mid C = c) = (K_c + 1)/(N + 1),$$

где K_c — это число значений x_i параметра i транзакции класса c, которое наблюдалось при наборе статистической информации (т. е. при обучении модели); N — общее число транзакций (значений параметра) в выборке, предназначенной для обучения модели.

Для использования подобной модели вычисления вероятности в качестве статистической информации необходимо хранить значения параметров всех транзакций, предназначенных для обучения модели.

Недостаток данного метода заключается в том, что при наличии атрибутов транзакций, которые принимают непрерывные значения, необходимо использовать какой-то механизм преобразования непрерывных величин в дискретные. Также подобный метод вычисления вероятностей требует значительных ресурсов при осуществлении хранения статистической информации [6].

Во *второй модели* на основе статистических данных для каждого параметра и соответствующего класса высчитываются среднее значение и дисперсия:

$$\mu_c^i = 1/n \times \Sigma_{j = 1,\ldots,n} x_{cj}^i;$$

$$D_c^i = 1/n \times \Sigma_{j = 1,\ldots,n} (x_{cj}^i - \mu_c^i)^2.$$

Здесь μ_c^i — среднее значений атрибута i для транзакций класса c; D_c^i — дисперсия значений атрибута i для транзакций класса c; x_{cj}^i — j-е значение атрибута i для транзакции класса c.

Вероятность высчитывается следующим образом:

$$P(X_i = x | C = c) = g(x, \mu_c^i, D_c^i);$$

$$g(x, \mu_c^i, D_c^i) = 1/((2 \times \pi)^{0,5} \times D_c^i)) \times$$
$$\times \exp(-(x - \mu_c^i)^2/2(D_c^i)^2).$$

Таким образом, исключается недостаток первого метода. Но здесь также есть существенный недостаток, который заключается в том, что в данной модели мы сделали предположение, что распределение всех параметров соответствуют нормальному распределению, хотя в реальности может оказаться, что это не так.

Для использования подобной модели вычисления вероятности в качестве статистической информации необходимо хранить средние значения и соответствующие дисперсии параметров транзакций, предназначенных для обучения модели, для каждой из используемых платежных карт [6].

Чтобы исключить недостаток описанной выше модели, необходимо $P(X = x | C = c)$ вычислять следующим образом (*третья модель*):

$$P(X = x | C = c) = 1/n \times \Sigma_{i = 1,\ldots,n} g(x, x_i, D)$$

(оценка плотности среднего); $D = 1/(n)^{0,5}$.

В данном случае суммируется $g(x, x_i, D)$ для всех x_i, где x_i ($i = 1, \ldots, n$) — все наблюдаемые в статистических данных значения параметра X при условии, что случайная величина C принимает значение c. Соответственно n в данном случае — число подобных наблюдений [6].

Для использования данной модели вычисления вероятности в качестве статистической информации также необходимо хранить значения параметров всех транзакций, предназначенных для обучения модели.

Модель представления входных параметров. В качестве входных параметров в данных моделях могут использоваться значения полей транзакций либо значения, сформированные на их основе. Необходимо отметить, что значения большинства из параметров транзакций несут определенную идентификационную нагрузку, и их распределение может не подчиняться какому-то определенному закону распределения. Например, значения параметра типа транзакции могут варьироваться либо в районе числа 800, либо в районе числа 1000.

В процессе исследования описанных выше моделей выявления мошеннических транзакций, был выявлен тот факт, что значительный вклад в окончательный результат классификации вносит именно представление значений входных параметров этих моделей. Эффективность классификации транзакций при использовании сетей Байеса растет с уменьшением диапазона значений, которые принимают входные параметры, причем это происходит независимо от того, какой метод вычисления вероятности используется. В данном случае можно отказаться от использования значений параметров транзакций в качестве входных параметров сети Байеса и предложить следующее решение:

- значение входного параметра равно 1, если значение соответствующего поля (параметра или атрибута) транзакции "характерно" для транзакций, осуществленных владельцем платежной карты (т. е. значение соответствующего поля транзакции содержится в выборке, предоставленной для обучения модели);
- значение входного параметра равно 0, если значение соответствующего поля транзакции

"нехарактерно" для транзакций, осуществленных владельцем платежной карты (т. е. значение соответствующего поля транзакции не содержится в выборке, предоставленной для обучения модели).

Для использования подобного механизма представления значений входных параметров можно хранить все значения соответствующих полей транзакции, которые были получены в процессе сбора статистической информации. Но такое решение является недопустимым, поскольку это требует использования значительных ресурсов для осуществления хранения и является крайне избыточным решением в случае представления данных одного из полей транзакции, которое может принимать непрерывные значения. Поэтому предлагается для каждого из параметров транзакции разбивать все значения, полученные из обучающей выборки, на интервалы, для которых находить среднее (центр кластеризации) от значений, расположенных в этом интервале, и соответствующее отклонение. Таким образом, все значения каждого из параметров транзакции будут представлены в виде областей кластеризации, т. е. средних значений и соответствующих отклонений. Если значение параметра классифицируемой транзакций "входит" в одну из областей кластеризации, то оно признается "характерным" для данного параметра.

Необходимо отметить, что подобный способ представления входных параметров пригоден только для "простого" классификатора Байеса (частный случай сетей Байеса). В общем случае в качестве входных параметров можно использовать порядковый номер центра кластеризации, в который "входит" значение соответствующего поля транзакции, что позволяет отразить зависимость между параметрами. Но существенным недостатком подобного метода является то, что необходимо хранить все цепочки номеров центров кластеризации для соответствующих участков зависимости между параметрами, а также значительные требования к ресурсам при вычислении вероятности (сложность и число запросов к базе данных). Поэтому можно объединить несколько параметров транзакции, среди которых была обнаружена зависимость, в один общий параметр. Например, для сети, изображенной на рис. 1, можно объединить параметры X_1, X_2, X_4 (векторизация). Следует уточнить, что один и тот же параметр не может входить в несколько объединений. Таким образом, получится совокупность независимых векторизованных параметров. Вычисление вероятности при классификации транзакции в данном случае сводится к определению того, "входят" ли значения векторизованного параметра в соответствующие области кластеризации (в данном случае векторизованный параметр принимает значение 1, если все значения его атрибутов "входят" в соответствующие области кластеризации, иначе — 0) и вычислению вероятности по частоте появлений значений векторизованного параметра. Поиск структуры векторизованного параметра, т. е. цепочки параметров транзакции, которые в него входят, базируется на использовании механизма MDL. В результате работы данного механизма будут получены цепочки зависимых параметров транзакции, причем в случае обнаружения общих параметров в двух разных цепочках происходит объединение данных цепочек.

Результаты оценочного тестирования

Тестирование осуществлялось с помощью разных наборов транзакций. Каждый набор состоит из выборки транзакций, предназначенных для обучения модели, и выборки, предназначенной для проведения оценочного тестирования. Первый набор транзакций предназначен для оценки эффективности работы модели на основе одного из параметров. В выборке, предназначенной для обучения, содержится ряд транзакций, в которых значение соответствующих параметров транзакции одинаковы, за исключением параметра времени осуществления транзакции. Тестирование "простого" классификатора Байеса показало, что наилучший результат классификации транзакций достигается при использовании модели представления входных параметров, базирующейся на использовании областей кластеризации значений параметров (рис. 2, см. вторую сторону обложки).

На данном рисунке *Plegal* — это вероятность того, что классифицируемая транзакция является легальной, причем:

- *Naive Bayes* 1 отражает результаты работы "простого" классификатора Байеса, базирующегося на использовании областей кластеризации значений параметров;
- *Naive Bayes* 2 отражает результаты работы "простого" классификатора Байеса, базирующегося на использовании нормального распределения для представления значений параметров;
- *Naive Bayes* 3 отражает результаты работы "простого" классификатора Байеса, базирующегося на использовании метода оценки отклонения среднего;
- *Naive Bayes* 4 отражает результаты работы "простого" классификатора Байеса, базирующегося на использовании частоты появления значений параметров транзакции.

Транзакции с номерами 1, 2, 3, 4, 5, 6, 7, 8 и 9 соответствуют экземплярам легальных транзакций, причем транзакции 7 и 8 имеют один "нехарактерный" параметр. Транзакции с номерами 9 и 10 со-

ответствуют экземплярам мошеннических транзакций.

Очевидно, что в данном случае результаты работы "простого" классификатора Байеса, базирующегося на использовании областей кластеризации значений параметров, значительно лучше, чем при использовании остальных методов представления входных параметров, так как все легальные транзакции были признаны таковыми с большой вероятностью. Данный метод эффективен и при вычислении вероятности *Pfraud* (рис. 3, см. вторую сторону обложки), т. е. вероятности того, что классифицируемая транзакция является мошеннической.

Результаты работы "простого" классификатора Байеса, базирующегося на использовании областей кластеризации значений параметров, больше всех соответствуют действительности. Видно, что вероятность *Plegal* для транзакций 7 и 8, которые имеют один "нехарактерный" параметр, меньше, чем у остальных легальных транзакций, а соответствующая вероятность *Pfraud* — больше.

Также было проведено сравнительное тестирование "простого" классификатора Байеса, базирующегося на использовании областей кластеризации значений параметров, и сети Байеса, базирующейся на использовании векторизованных параметров. Поиск структуры векторизованных параметров проводился на основе реальных транзакций, осуществленных в одном из банков. Сам процесс тестирования моделей проводился на основе 56 транзакций, предназначенных для обучения системы, и 11 транзакций, предназначенных для классификации. Все транзакции были осуществлены с одной и той же карты. Среди транзакций, предназначенных для обучения, 52 экземпляра соответствуют классу легальных транзакций, а 4 — классу мошеннических транзакций. Легальные транзакции были осуществлены с использованием 6 различных банкоматов данного банка, 12 внешних финансовых институтов (через платежную систему). Транзакции проводились в разное время и в разных странах. Мошеннические транзакции были осуществлены из определенного финансового института, страны и в определенное время. Среди транзакций, предназначенных для классификации, девять транзакций содержат параметры, "характерные" для легальных транзакций, а две соответствуют классу мошеннических транзакций. Результаты тестирования показывают, что данная адаптация модели сети Байеса (*Bayesian Network*) эффективнее модели "простого" классификатора Байеса (*Naive Bayes*), базирующегося на использовании областей кластеризации, для классификации транзакций (рис. 4 и 5, см. вторую сторону обложки). Причем транзакции под номерами 7 и 8 имеют один "нехарактерный" параметр (время осуществления транзакции) при сопоставлении с классом легальных транзакций. Таким образом, можно сделать вывод о том, что данная адаптация модели сети Байеса эффективнее "простого" классификатора Байеса, базирующегося на использовании областей кластеризации. В основном это связано с тем, что в модели сети Байеса учитывается зависимость между параметрами транзакций.

Полученные результаты, прежде всего, говорят о том, что на результат классификации транзакций сильно влияет метод представления входных данных. Как правило, распределение значений параметров транзакций не подчиняется какому-то конкретному закону распределения, поэтому использование в качестве входных данных непосредственно значений параметров транзакций при их классификации дает плохой результат. Использование разработанного механизма преобразования значений параметров транзакций действительно оказалось эффективным для данной тестовой выборки.

Список литературы

1. **EMV** Will Push Card Fraud To Next Weakest Link [Электронный ресурс] // The payment news and resource center (Epaynews): [web-сайт]. 09.11.2005. http://www.epaynews.com/index.cgi?survey = &ref = browse&f = view&id = 11341361006222 15 212&block (02.02.2007).

2. **Неделку Е.** Осторожно, идет охота за карточками [Электронный ресурс] //Биржа: [web-сайт]. 06.09.2005. < http://gazeta.birga.od.ua/article/1616 > (02.02.2007).

3. **Рубенштейн Т. Б., Мирошкина О. В.** Пластиковые карты. М.: Гелиос АРВ, 2002.

4. **Brause R., Langsdorf T., Hepp M.** Neural Data Mining for Credit Card Fraud Detection // Proc. of the 1 lth IEEE International Conference on Tools with Artificial Intelligence, Evanston, IL. 1999. P. 103—106.

5. **Cahill M., Chen F., Lambert P., Pinheiro J. C, Sun D. X.** Detecting Fraud in the Real World // Chapter in Handbook of Massive Datasets / Ed. J. Abello, P. Pardalos and M. Resende. New York: Kluwer Press, 2000.

6. **John G. H., Langley P.** Estimating continuous distributions in Bayesian classifiers // Proc. of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo, 1995.

7. **Chan, Philip K.; Stolfo, Salvatore J.** Toward scalable learning with non-uniform class and cost distribution: A case study in credit card fraud detection // Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining, o. AAAI Press, 1998.

8. **Lam, Wai;** Bacchus, Fahiem. Learning Bayesian belief networks: An approach based on the MDL principle // Computational Intelligence. 1994. Vol. 10. № 4. P. 269—293.

9. **Suzuki J.** Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique // Proceedings of the international conference on machine learning, Bally, Italy, 1996.

УДК 622:51.002:131.3

С. Н. Гончаренко, канд. экон. наук, доц.,
Московский государственный
горный университет

Информационные технологии формирования альтернативных моделей инвестирования горно-промышленных проектов

Рассмотрены возможности применения информационных технологий для привлечения внешних инвестиций в проекты развития горно-добывающих предприятий. Обоснована необходимость разработки и проведения комплекса мероприятий, направленных на привлечение отечественных и зарубежных инвестиционных ресурсов. Показаны результаты оценки и анализа инвестиционной привлекательности горно-промышленных проектов.

Развитие инвестиционных процессов становится все более важной проблемой управления промышленным предприятием. Российская кредитно-финансовая система функционирует в условиях неустойчивой рыночной конъюнктуры при деформированной структуре кредитно-финансового рынка, слабой законодательной и нормативной поддержке государственного регулирования. Наблюдается тенденция недоверия иностранных инвесторов к слабо развитой инвестиционной инфраструктуре, что является одной из причин кризисного состояния инвестиционной сферы. Кроме того, основные фонды промышленных предприятий на текущий момент достаточно сильно устарели и требуют значительных объемов капитальных вложений за счет привлечения как отечественных, так и зарубежных инвестиций. При этом многие инвестиционные проекты не находят своего реального применения в связи с отсутствием четких экономических механизмов финансирования. Из этого следует, что необходимо создание дополнительных экономических рычагов и методик, способствующих продвижению и воплощению реальных инвестиционных проектов.

Отсутствие опыта в области привлечения инвестиций с использованием современных механизмов, форм и методов инвестирования не позволяет горным предприятиям в полной мере привлекать необходимые инвестиционные ресурсы, что еще больше усугубляет их тяжелое финансовое состояние.

Поэтому формирование структуры инвестирования проектов горно-добывающих предприятий является актуальной проблемой, решение которой позволит обеспечить необходимый приток инвестиционных ресурсов в проекты. В этих целях необходимо создание информационной технологии выбора альтернативных моделей инвестирования горно-промышленных проектов добывающих предприятий, которая позволит:

- обосновать перечень и определить пороговые значения ключевых параметров и агрегированных показателей инвестиционных проектов для определения возможности привлечения внешних источников финансирования;
- сформировать структуру инвестирования, основанную на моделировании денежных потоков с учетом взаимоотношений участников проекта в ходе его реализации;
- обеспечить эффективность использования собственных средств, основанную на определении соотношения показателей рентабельности основных фондов и цены привлеченных для их создания инвестиционных ресурсов.

В МГГУ была разработана такая информационная технология, суть которой заключается в формировании альтернативных моделей инвестирования горно-промышленных проектов и выборе их оптимальной структуры. В основе разработанной информационной технологии лежат методы и алгоритмы, на основе которых осуществляется моделирование показателей денежных потоков с учетом взаимоотношений всех возможных участников. Выбор оптимальной структуры инвестирования горно-промышленных проектов добывающих предприятий состоит из четырех основных этапов:

1. Определение и оценка ключевых параметров и агрегированных показателей инвестиционного проекта.

2. Формирование исходного множества приемлемых источников инвестирования проектов.

3. Построение альтернативных структур инвестирования проектов.

4. Формирование альтернативных моделей инвестирования и оптимальной структуры инвестиционных потоков.

Этап 1. Определение ключевых параметров и оценка агрегированных показателей инвестиционного проекта

В рамках этапа (рис. 1) проводится оценка коммерческой эффективности инвестиционного проекта, определяются его ключевые параметры и агрегированные показатели для дальнейшего определения возможности привлечения внешних инвестиционных источников. Определение ключевых параметров и агрегированных показателей инвестиционного проекта базируется на разрабо-

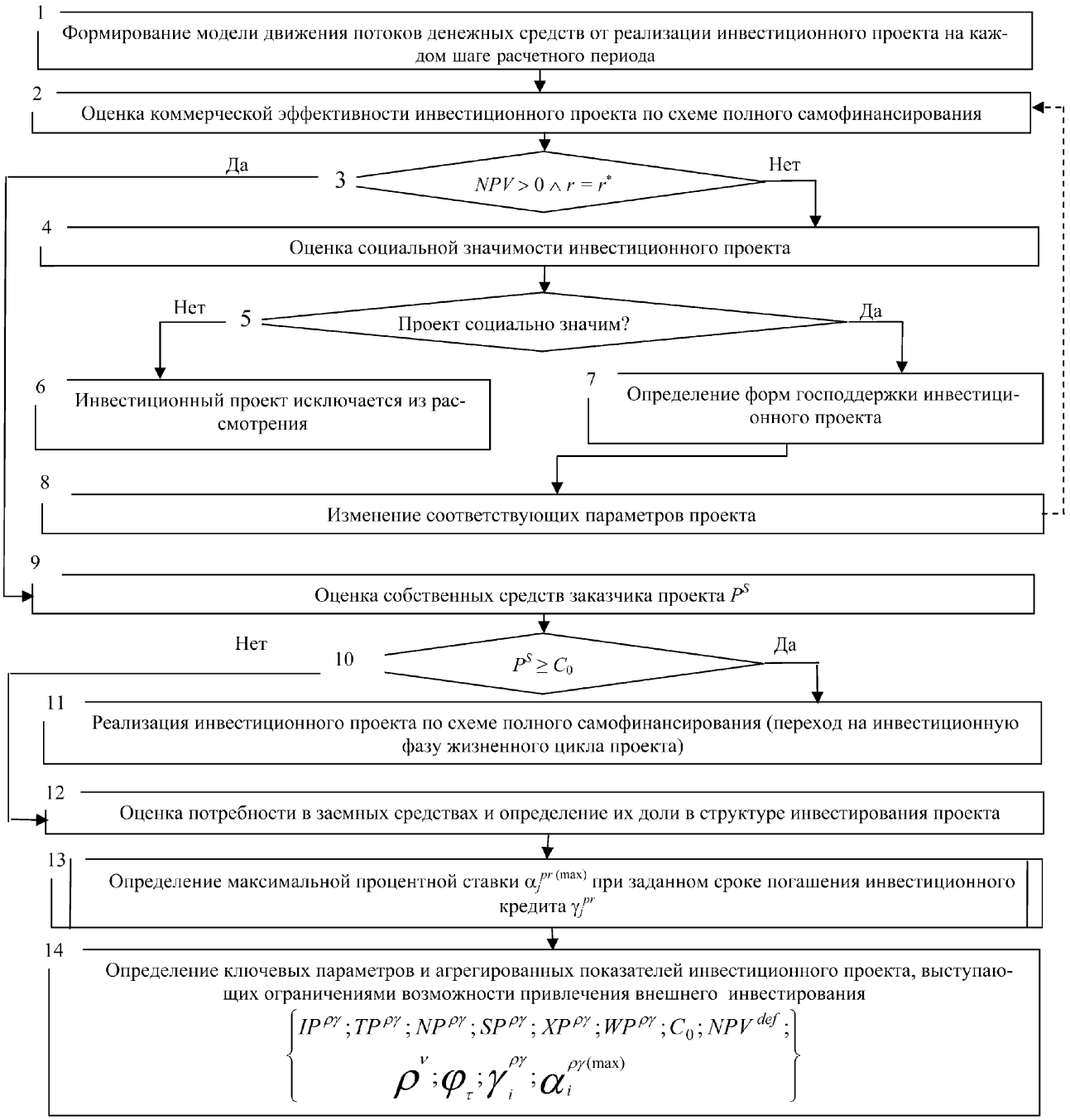


Рис. 1. Определение ключевых параметров и оценка агрегированных показателей инвестиционного проекта:
r — норма дисконта; r^* — безрисковая коммерческая норма дисконта; P^S — сумма собственных средств; C_0 — размер капитальных вложений; $\alpha^{pr(\max)}$ — максимальная процентная ставка по кредиту; γ^{pr} — срок погашения кредита; P^v — размер внешнего инвестирования; φ_τ — сумма чистых поступлений (без учета выплат по кредитам); *IP* — инвестирование проектов в отрасли промышленности; *TP* — инвестирование проектов определенного типа; *NP* — назначения инвестиционных проектов; *SP* — степень проработки проекта; *XP* — технологическая структура инвестиционных проектов; *WP* — характер воспроизводства основных фондов инвестиционных проектов

танной модели движения потоков денежных средств от реализации проекта с учетом специфики горно-добывающей отрасли промышленности и экономических, технологических и организационных аспектов деятельности предприятия. Формирование множества качественных ключевых параметров осуществлялось по результатам экспертных оценок. Количественные ключевые параметры и агрегированные показатели выделяются из базовой модели движения денежных потоков по степени влияния на показатели инвестиционной деятельности предприятия.

На этом этапе было установлено, что основными ключевыми параметрами, определяющими приемлемость привлечения внешних источников инвестирования, являются *отраслевая принадлеж-*

Ключевые параметры и агрегированные показатели горно-промышленного проекта

Ключевые параметры инвестиционного проекта			
Качественные			
Ключевой параметр	Обозначение	Значение параметра	
Отраслевая принадлежность инвестиционного проекта	IP^{pr}	Отрасль промышленности	
Тип инвестиционного проекта	TP^{pr}	Модернизация. Расширение. Техническое перевооружение	
Назначение инвестиционного проекта	NP^{pr}	Производственное назначение	
Степень проработки инвестиционного проекта	SP^{pr}	Предынвестиционные исследования	
Технологическая структура инвестиционного проекта	XP^{pr}	Приобретение оборудования; строительно-монтажные работы	
Характер воспроизводства основных фондов инвестиционного проекта	WP^{pr}	Простое, расширенное воспроизводство	
Количественные			
Ключевой параметр	Обозначение	Расчетная формула	Граничные условия
Срок реализации проекта	T	$T = \begin{cases} T_1 = \{t_1^i,\ t_2^i,\ t_3^i,\ ...,\ t_n^i\} \\ T_2 = \{t_1^0,\ t_2^0,\ t_3^0,\ ...,\ t_n^0\} \\ T_3 = \{t_1^f,\ t_2^f,\ t_3^f,\ ...,t_n^f\} \end{cases}$	$\lambda_1 + \max\{\lambda_2, \lambda_3\} \leqslant T\lambda_4$ $\lambda_1 = \min\{-1(t_1 - \gamma_j)\},\ \lambda_1 \geqslant 0$
Общая сумма инвестиций в проект	C_0	$C_0 = \sum\limits_{t=0}^{T} MI_t = \rho^v + \rho^s$	$C_0 \geqslant MI_{1t} + MI_{2t} + MI_{3t} + MI_{4t} + MI_{5t} + MI_{6t}$
Агрегированные показатели			
Агрегированный показатель	Обозначение	Расчетная формула	Граничные условия
Чистый дисконтированный доход	NPV^{def}	$NPV^{def} = \sum\limits_{t=0}^{T} PV_t^{def}$	$NPV^{def} \geqslant 0,\ r = r'$
Необходимый размер внешнего заемного финансирования	ρ^v	$\rho^v = C_0 - \rho^s$	$\rho^v \geqslant \{MI_{1t} + MI_{2t} + MI_{3t} + MI_{4t} + MI_{5t} + MI_{6t}\} - \rho_t^s$
Сумма чистых поступлений (без учета выплат по кредитам)	φ_τ	$\varphi_\tau = PO_t - MO_t$	$\varphi_\tau > 0,\ \varphi_\tau \to \max$
Срок погашения кредитов (первое приближение)	γ_1^{pr}	$\gamma^1 = \frac{\rho^v}{\varepsilon_\tau}$, при $\varepsilon_\tau = \text{const}$	$\varepsilon_\tau < \varphi_\tau$ $\gamma^1 \leqslant T$
Срок погашения кредитов (второе приближение)	γ_j^{pr}	$\gamma_j^{pr} = \gamma_j^{pr} + 1$	$\varepsilon_\tau < \varphi_\tau$ $\gamma_j^{pr} \leqslant T$
Максимальное значение процентной ставки	$\alpha_i^{pr(\max)}$	Итеративный подбор значения показателя	$\varepsilon_\tau < \varphi_\tau$ $\gamma_j^{pr} \leqslant T$

Условные обозначения в таблице: λ_1 — период времени, прошедший с момента инвестиционных вложений до момента принятия фондов на баланс предприятия; λ_2 — срок амортизации основных фондов; λ_3 — срок амортизации нематериальных активов; λ_4 — срок полного погашения кредита; γ — момент принятия на баланс j количества оборудования; t^* — момент осуществления первых инвестиционных вложений; MI_1 — затраты на закупку и поставку оборудования; MI_2 — затраты на проведение проектных работ; MI_3 — затраты на увеличение оборотного капитала; MI_4 — затраты на строительно-монтажные работы; MI_5 — затраты на приобретение нематериальных активов; MI_6 — затраты на пусконаладочные работы; ε — размер выплат по кредитам.

ность проекта, его тип, назначение, степень проработки, технологическая структура, характер воспроизводства основных фондов, срок реализации и общая сумма инвестиций в проект. В обоснованной системе агрегированных показателей инвестиционного проекта основными элементами являются *необходимое значение внешнего заемного финансирования, размер чистых поступлений от реализации проекта, срок погашения кредитов и максимальное значение процентной ставки по кредитам.*

Оценка значений агрегированных показателей осуществляется по разработанной модели наличного оборота. Модель позволяет провести оценку экономической эффективности реализации проекта и определить необходимые объемы внешнего инвестирования, а также установить максимальную процентную ставку, при которой гарантируется полное погашение привлеченных ресурсов в заданные заказчиком сроки. Результаты проведенной оценки экономической эффективности инвестиционных проектов являются основанием либо для рекомендации их к реализации (NPV > 0), либо для отклонения указанных проектов (NPV < 0). В случае, если экономически несостоятельный инвестиционный проект признается социально значимым и эффективным с позиции социально-экономических последствий его реализации, то определяются возможности применения различных форм его государственной поддержки, которые позволят повысить экономическую эффективность данного инвестиционного проекта.

На основе сопоставления размера необходимых инвестиций в проект и собственных средств заказчика, вкладываемых в проект, определяется потребность во внешнем инвестировании проекта. В случае, когда собственных средств заказчика оказывается достаточно для покрытия необходимых инвестиционных вложений, дальнейшая реализация проекта осуществляется по схеме полного самофинансирования.

Для определения максимальной процентной ставки, при которой в заданные заказчиком сроки обеспечивается приток наличности, достаточной для погашения основной суммы долга и начисленных процентов, разработан следующий алгоритм (модуль 13, см. рис. 1), включающий:

- разбиение расчетного периода на равные временные отрезки, продолжительность которых для проектов горнодобывающих предприятий, как правило, соответствует одному году;
- определение методом итеративного перебора значения процентной ставки, при которой гарантируется погашение привлеченных ресурсов в срок, равный периоду реализации проекта;
- последовательное сокращение срока погашения привлеченных ресурсов и определение соответствующего значения максимально возможной процентной ставки.

Цикл повторяется до тех пор, пока не будет достигнут срок, за который становится невозможным погашение привлеченных инвестиционных ресурсов даже при минимально возможной процентной ставке.

Результатом проведенных на первом этапе исследований является сформированная система ключевых параметров и агрегированных показателей инвестиционного проекта, используемых для определения возможности привлечения внешних источников предоставления инвестиционных ресурсов (см. таблицу).

Этап 2. Формирование исходного множества приемлемых источников инвестирования проектов

На этапе 2 (рис. 2) осуществляется:

- формирование множества источников предоставления инвестиционных ресурсов;
- определение набора критериев приемлемости каждого конкретного источника для инвестирования проекта;
- отбор источников по определенным критериям и их классификация по форме участия в инвестировании проекта.

В рамках реализации этапа формируется множество источников предоставления инвестиционных ресурсов в проекты промышленного инвестирования $IN' = \{in'1, in'2, in'3, ..., in'n\}$, и на основе предложенного набора критериев приемлемости источников для инвестирования проектов определенного типа (PF_{in}, IP_{in}, TP_{in}, соответственно: осуществление кредитования на принципах проектного финансирования; принадлежность объекта инвестирования к отрасли промышленности; тип инвестиционного проекта) проводят их отбор и формируют множество источников инвестирования проектов определенного типа. Далее полученное множество источников классифицируют по предложенному признаку — форме участия в инвестировании проекта. По результатам классификации множество источников инвестирования разбиваются на два подмножества: источники, осуществляющие непосредственное инвестирование проекта (инвестиционные банки и компании, международные инвестиционные организации и т. д.), и косвенные участники процесса инвестирования (гаранты, поручители и т. д.).

Для всех источников первого подмножества, $IN_p = \{in_1^p, in_2^p, in_3^p, ..., in_n^p\}$, определяются условия предоставления инвестиционных ресурсов, а для всех участников второго подмножества, $IN_k = \{in_1^k, in_2^k, in_3^k, ..., in_n^k\}$, — условия участия в инвестиционном проекте.

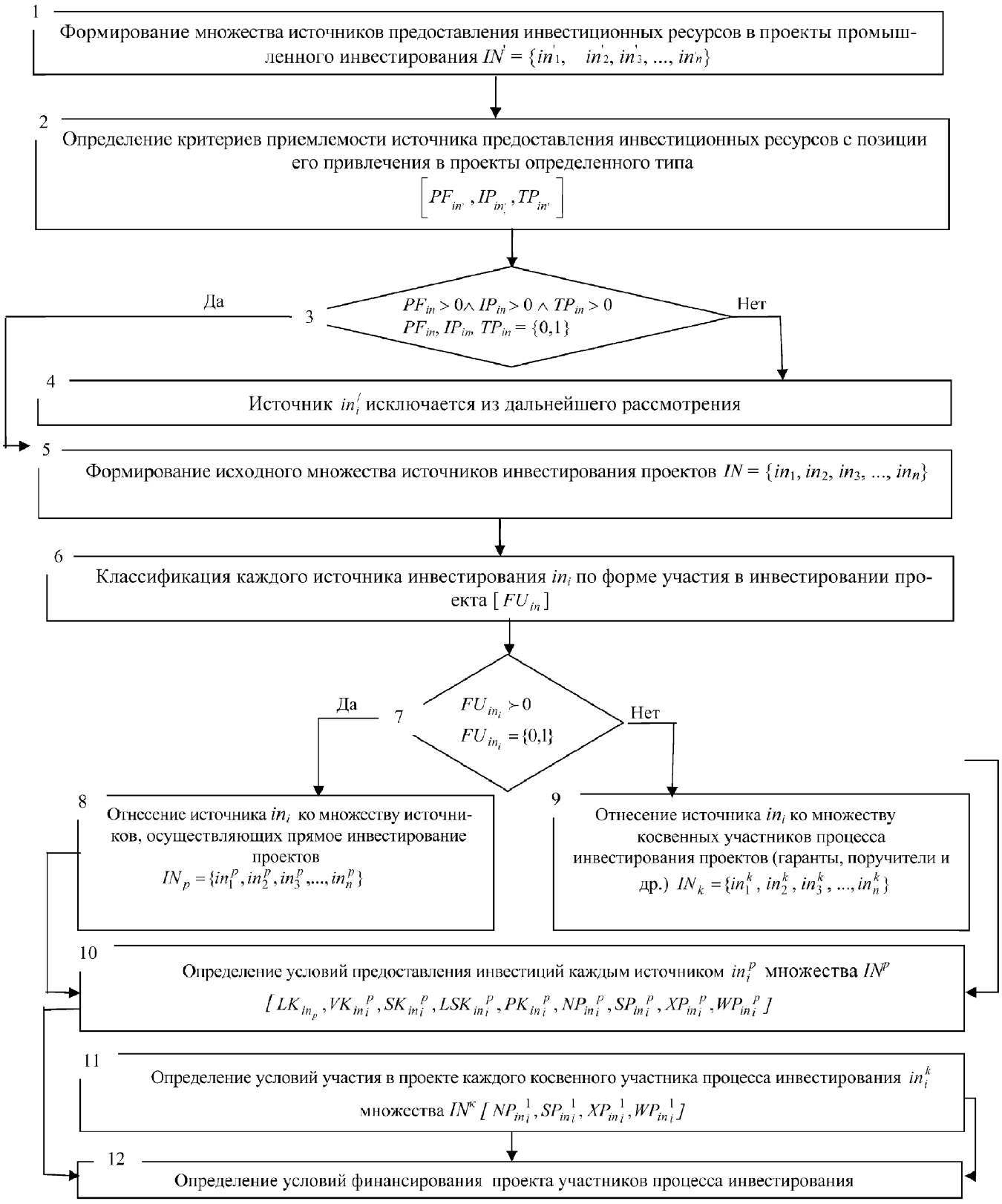


Рис. 2. Формирование исходного множества источников инвестирования проектов:
кредитование на принципах ПФ; *IP* — инвестирование проектов, реализуемых в определенной отрасли; *TP* — инвестирование проектов определенного типа; *LK* — лимит заимствования; *VK* — максимальная величина кредита; *SK* — максимальный срок предоставления кредита; *LSK* — льготный срок по выплате процентов и основного долга; *PK* — процентная ставка по кредиту; *NP* — назначение проекта; *SP* — степень проработки проекта; *XP* — технологическая структура проекта; *WP* — характер воспроизводства основных фондов проекта

Этап 3. Построение альтернативных структур инвестирования проектов

На этапе 3 (рис. 3) формируется множество альтернативных структур инвестирования проекта, представляющих собой набор источников с соответствующими объемами предоставляемых инвестиционных ресурсов.

Формирование альтернативных структур инвестирования основано на разработанной процедуре итеративного перебора источников предоставления инвестиционных ресурсов множества $IN_p = \{in_1^p, in_2^p, in_3^p, ..., in_n^p\}$ с учетом их приемлемости для инвестирования конкретных проектов при определенных ключевых параметрах и оцененных агрегированных показателях.

Разработанный алгоритм позволяет провести оценку и выбор приемлемых источников инвестирования при варьировании значений показателя срока погашения привлеченных инвестиционных ресурсов и соответствующих значений кредитных ставок. Ограничения определяются условиями предоставления инвестиций каждым конкретным источником. Алгоритм предусматривает последовательное выполнение следующих операций.

Блок 1. Сопоставление значения максимальной процентной ставки, соответствующего минимальному сроку погашения привлеченных инвестиционных ресурсов, и значения процентной ставки каждого источника предоставления инвестиционных ресурсов множества $IN_p = \{in_1^p, in_2^p, in_3^p, ..., in_n^p\}$.

Блок 2. Оценка приемлемости источника инвестирования проекта на базе сопоставления ключевых параметров и условий предоставления инвестиционных ресурсов данным источником.

Блок 3. Определение объема инвестирования данным источником (согласно лимиту заимствования) и предварительная оценка возможности погашения привлеченных ресурсов в установленные сроки:

$$p^{\nu}_{in_i^p} = VK_{in_i^p}, \text{ при } \frac{\sum_{r=n}^{\wp} CF_t^{def}}{p^{\nu}_{in_i^p}} \geqslant \nu, \qquad (1)$$

где $p^{\nu}_{in_i^p}$ — размер инвестиционного кредита, привлекаемого из источника in_i^p, млн руб.; $VK_{in_i^p}$ — размер инвестиционного кредита, предоставляемого источником in_i^p, млн руб.; CF_t^{def} — значение финансового итога от реализации проекта в дефлированных ценах на шаге t, млн руб.; n — шаг привлечения кредита, ед.; $\wp$ — шаг полного погашения привлеченного кредита, ед.; ν — коэффициент покрытия задолженности, ед.

Результатом проведенных на этапе 3 исследований является сформированное множество альтернативных структур инвестирования, определенный набор источников и предоставляемых ими инвестиционных ресурсов.

Этап 4. Формирование альтернативных моделей инвестирования и оптимальной структуры инвестиционных потоков

Формирование альтернативных моделей инвестирования проектов осуществляется на базе множества альтернативных структур инвестирования $SI_p = \{si_1^p, si_2^p, si_3^p, ..., si_n^p\}$ с привлечением необходимых косвенных участников процесса инвестирования $IN_k = \{in_1^k, in_2^k, in_3^k, ..., in_n^k\}$, выступающих в роли гарантов, поручителей и страховщиков и определяемых условиями предоставления инвестиционных ресурсов прямыми (непосредственными) источниками инвестирования проектов (рис. 4).

С учетом определенного набора необходимых косвенных участников процесса инвестирования для каждой альтернативной структуры разрабатывается модель инвестирования, описывающая процессы движения денежных средств в ходе реализации проекта в соответствии с договорными отношениями между всеми его участниками:

$$MI^{\text{ext}} = \{mi_1^{\text{ext}}, mi_2^{\text{ext}}, mi_3^{\text{ext}}, ..., mi_n^{\text{ext}}\}. \qquad (2)$$

По результатам оценки экономической эффективности реализации инвестиционного проекта осуществляется выбор оптимальной структуры инвестиционных потоков по критерию максимизации значения чистого дисконтированного дохода от реализации проекта:

$$SI_{\text{opt}} = \{si | si \in SI \;\&\; si = \arg\max f(si)\},$$
$$f(si) = NPV \;\&\; SI = \{si_1, si_2, si_3, ..., si_n\}. \qquad (3)$$

Таким образом, разработанная информационная технология выбора альтернативных моделей инвестирования проектов горно-добывающих предприятий позволит решить следующие задачи:

- оценить реализуемость проекта при каждом альтернативном варианте структуры инвестирования, т. е. провести проверку соответствия каждого варианта всем имеющимся ограничениям — химико-технологическим, экономическим, экологическим, социальным и пр.;
- дать оценку абсолютной эффективности проекта, характеризующейся превышением оценки совокупного результата над оценкой совокупных затрат. При отрицательной абсолютной эффективности проект, как правило, исключается из дальнейшего рассмотрения.

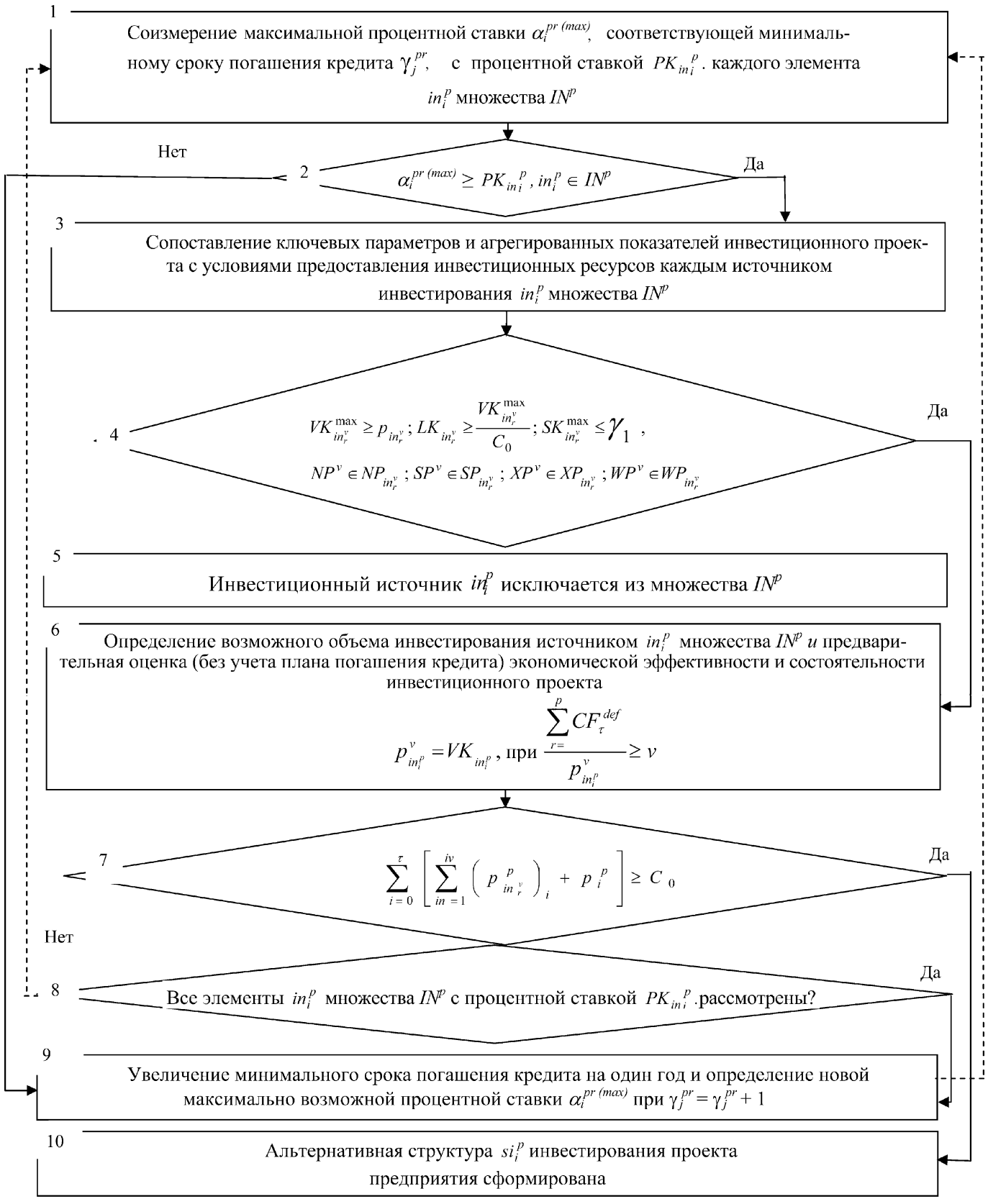


Рис. 3. Формирование альтернативных структур инвестирования проектов

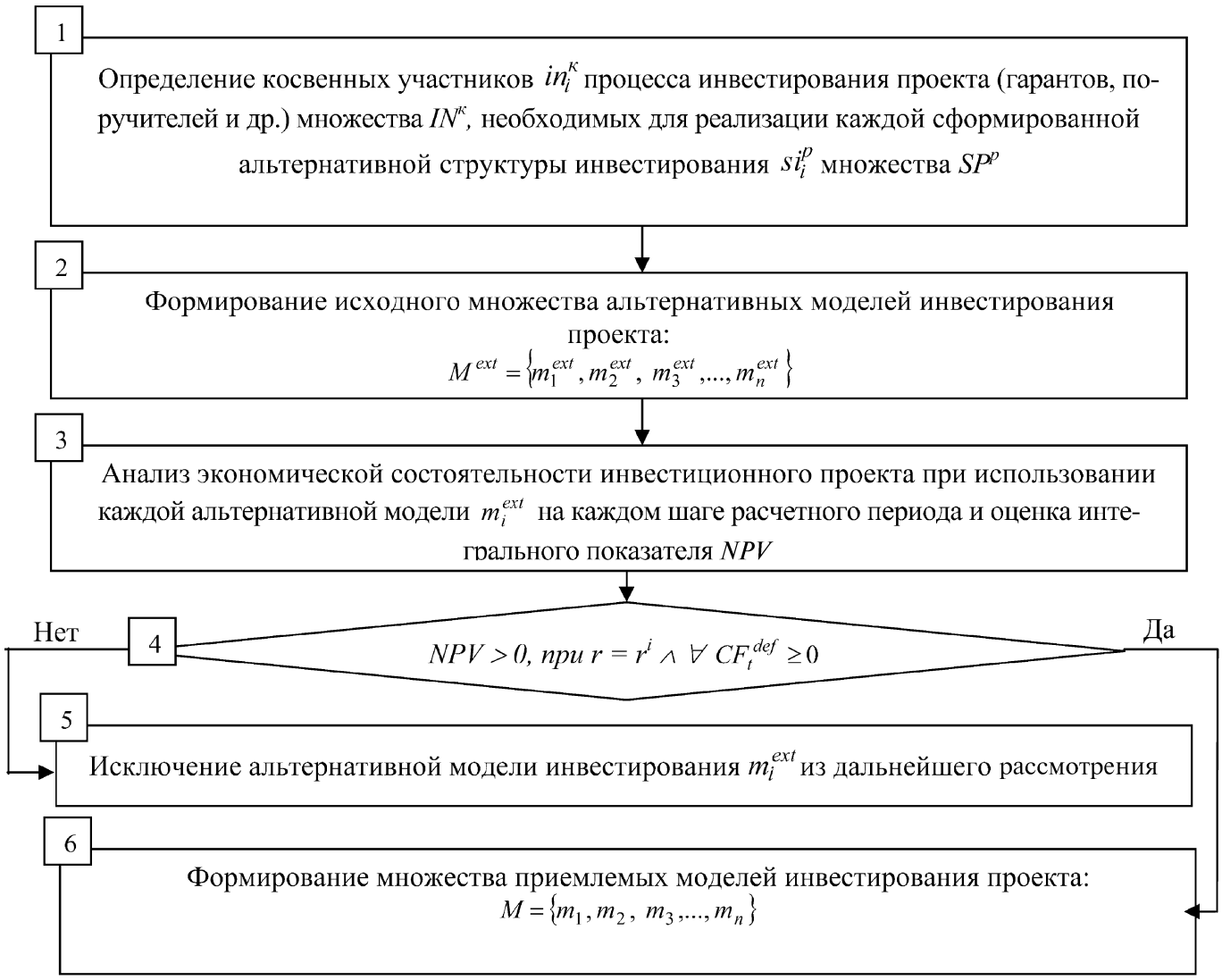


Рис. 4. Формирование множества альтернативных моделей инвестирования

Исключением являются проекты действующего предприятия, имеющие социально-экономическую направленность, когда все альтернативные возможности, в том числе и возможность не осуществлять проект, имеют отрицательную абсолютную эффективность;

- провести оценку сравнительной эффективности, осуществляемую по совокупности качественных критериев проекта. Она дает возможность определения структуры предпочтений моделей инвестирования. Оценка сравнительной эффективности проводится, как правило, на множестве альтернативных моделей инвестирования проекта, применительно к проектам, реализуемым на действующих предприятиях. В этих случаях сопоставляются программы развития предприятия, предусматривающие различные варианты используемых моделей инвестирования;
- осуществить оптимизацию и выбор из множества структур инвестирования наиболее эффективной при определенных ограничениях. Как правило, в качестве ограничений выступает лимит кредитования различных источников финансирования.

Список литературы

1. **Блех Ю., Гетце У.** Инвестиционные расчеты. Модели и методы оценки из инвестиционных проектов. Калининград: Янтарный сказ, 1997.

2. **Бочаров В. В.** Финансово-кредитные методы регулирования рынка инвестиций. М.: Финансы и статистика, 1993.

3. **Беренс В., Хавренек П. М.** Руководство по оценке эффективности инвестиций. М.: Инфра-М, 1995.

4. **Воронцовский А. В.** Инвестиции и финансирование: Методы оценки и обоснования. СПб.: Изд-во С.-Петербургского университета, 1998.

5. **Идрисов А. Б., Картышев С. В., Постников А. В.** Стратегическое планирование и анализ эффективности инвестиций. М.: Филинъ, 1997.

6. **Инвестиционное** проектирование. Практическое руководство по экономическому обоснованию инвестиционных проектов / Под ред. С. И. Шумилина. М.: Финстат-информ, 1995.

Поздравляем юбиляра!

Профессору кафедры вычислительной математики, сотруднику Института проблем информационной безопасности МГУ им. М. В. Ломоносова, доктору физико-математических наук

Валерию Александровичу ВАСЕНИНУ,

известному ученому, крупному специалисту в области математического, алгоритмического и программного обеспечения высокопроизводительных информационно-вычислительных систем и сетей, члену редколлегии журнала "Информационные технологии" исполняется 60 лет.

Талантливый ученый и преподаватель, Валерий Александрович является автором ряда актуальных исследований в области разработки и внедрения средств обеспечения безопасности информационных технологий и противодействия кибертерроризму, включая вопросы компьютерной безопасности в сети Интернет и информационной безопасности объектов национально-телекоммуникационной инфраструктуры.

Высокий профессионализм, широкая эрудиция, большая трудоспособность, чуткость и отзывчивость снискали ему большое уважение учеников и коллег.

Дорогой Валерий Александрович! Сердечно поздравляем Вас с юбилеем и желаем Вам крепкого здоровья и новых творческих успехов в Вашей научной и педагогической деятельности!

Редколлегия и редакция журнала.

CONTENTS

Адрес редакции:

107076, Москва, Стромынский пер., 4/1

Телефон редакции журнала **(495) 269-5510**

E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Художник *В.Н. Погорелов*.
Технический редактор *О. А. Ефремова*. Корректор *О. А. Шаповалова*

Сдано в набор 17.12.2007. Подписано в печать 18.01.2007. Формат 60×88 1/8. Бумага офсетная. Печать офсетная.
Усл. печ. л. 10,78. Уч.-изд. л. 11,98. Заказ 79. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати, телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Отпечатано в ООО "Подольская Периодика"
142110, Московская обл., г. Подольск, ул. Кирова, 15