*И*звестия СПбГЭТУ «ЛЭТИ»

***СЕРИЯ** «Информатика, управление и компьютерные технологии»*

1’2005

СОДЕРЖАНИЕ

Редактор *Э. К. Долгатов*
Комп. верстка *Е. Н. Паздникова*

ЛР № 020617 от 24.06.98 г.

Подписано в печать 14.05.05 г.
Формат 60x84 1/8.
Бумага офсетная.
Гарнитура "Таймс".
Печать: ризограф.
Печ. л. 7,75.
Тираж 100 экз. Заказ .

Издательство
СПбГЭТУ «ЛЭТИ»

197376, Санкт-Петербург,
ул. Проф. Попова, 5

УДК 621.3.049.77.001.2

О. Л. Гаврикова, С. Э. Миронов

ТЕХНОЛОГИЧЕСКИ ИНВАРИАНТНОЕ ПРОЕКТИРОВАНИЕ СЕМЕЙСТВ БИБЛИОТЕК СТАНДАРТНЫХ ФРАГМЕНТОВ КМОП БИС

Рассматриваются вопросы проектирования библиотек стандартных фрагментов БИС. Предлагается новый подход к решению проблемы повышения плотности кристалла, основанный на создании оптимизированных по площади библиотек ячеек для каждого из макроблоков кристалла.

Виртуальная сетка, сжатие топологии, технологически инвариантное проектирование, библиотеки стандартных фрагментов БИС

Библиотеки стандартных фрагментов КМОП БИС и технологическая инвариантность. Проектирование СБИС базируется на использовании САПР. При этом от свойств и характеристик библиотеки элементов в значительной степени зависят и характеристики проектируемых БИС.

На отечественном рынке появились пакеты проектирования БИС зарубежных фирм, таких, как Synopsys, Menthor Graphics, Cadence, Compass, коммерческие версии которых стоят порядка нескольких сотен тысяч долларов. При этом готовая библиотека топологических фрагментов, разработанная под конкретную технологию, или разработка библиотеки в конкретных проектных нормах оценивается зарубежными фирмами суммой тоже порядка сотен тысяч долларов. Переход на новую технологию с другими проектными нормами требует дополнительных затрат.

В свете изложенного становится очевидной актуальность разработки технологически инвариантной библиотеки стандартных фрагментов [1]. Использование ее в сочетании со стандартными САПР БИС позволит создавать проекты БИС, которые можно реализовать в любой «кремниевой мастерской», в том числе зарубежной, в том числе на самой передовой технологии. Поскольку в настоящее время достаточно четко обозначилась лидирующая роль КМОП-технологии, речь в первую очередь идет об инвариантности в пределах разновидностей технологий этого класса, хотя возможен выход на более широкую область применимости.

Технологически инвариантная библиотека обеспечит:

– накопление топологической проектной информации в программных файлах для обеспечения естественного развития, оперативности корректировки и модификации;

– живучесть проекта в условиях быстро меняющейся технологии;

– возможность переноса топологических проектов фрагментов на любые предприятия различных стран.

Разработка топологии фрагментов библиотеки. Технологически инвариантный метод проектирования библиотек стандартных фрагментов МОП БИС использует символьный сеточный метод проектирования топологии и понятие виртуальной сетки [2].

Виртуальная сетка не является метрической и определяет только взаимное расположение элементов. Точные геометрические размеры элементы и топология в целом приобретают в процессе сжатия, где происходит подстановка описания конкретного состава фо-

тошаблонов и детальных проектных норм. При этом информация о технологии считывается как данные, что позволяет автоматически настраивать топологию фрагмента на любую конкретную технологию из определенного класса. Полученная реализация топологии инвариантна к выходному языку описания топологии и преобразуется к выходной форме с помощью постпроцессора.

В процессе сжатия помимо технологических могут учитываться и ограничения на взаимное расположение элементов топологии фрагментов, задаваемые разработчиком. Эта возможность системы проектирования [2] позволяет разрабатывать сложные фрагменты, согласовывая по габаритам и положению выводов входящие в их состав блоки, и проектировать библиотеки стандартных фрагментов.

Под стандартизацией фрагмента понимается то, что фрагмент входит в библиотеку верхней САПР БИС (в частности, в библиотеку трассировщика топологии) и удовлетворяет стандартным ограничениям последней, таким, как:

– дискретность одного из габаритов фрагмента (например, горизонтального) и дискретность координат выводов вдоль горизонтальных границ фрагмента, что обеспечивает удобство трассировки выводов фрагментов;

– одинаковый габарит фрагмента (например, вертикальный) и одинаковые для всех фрагментов вертикальные координаты границ карманов, горизонтальных шин земли и питания, что позволяет соединять ячейки библиотеки в горизонтальные полосы, разделенные горизонтальными же каналами трассировки.

Таким образом, разработка топологии фрагментов выполняется в три этапа:

1) сжатие всех фрагментов, входящих в состав библиотеки, с учетом только технологических проектных норм (рис. 1, *а*) без учета системных «библиотечных» ограничений на вертикальный габарит, расположение шин питания и границ карманов, дискретность длины ячеек и координат выводов вдоль горизонтальных границ фрагмента;

2) определение параметров топологической модели библиотеки, заключающееся:

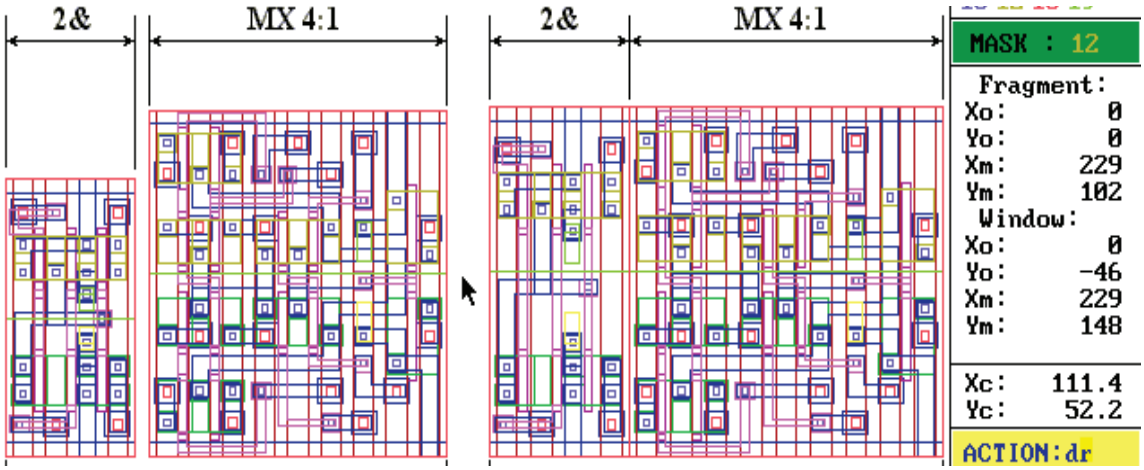
– в определении (путем чтения из соответствующих файлов описания топологии координат определенных линий виртуальной сетки) вертикальных габаритов фрагментов, координат границ карманов, координат шин земли и питания (дискретность второго (горизонтального) габарита задается проектировщиком, исходя из параметров технологии);

– в вычислении на основании полученных данных таких значений этих параметров, которые позволят состыковывать все ячейки библиотеки;

3) сжатие всех фрагментов, входящих в состав библиотеки, с учетом стандартных для библиотеки ограничений (параметров топологической модели, вычисленных на предыдущем этапе) (рис. 1, *б*).

Технологически инвариантные семейства библиотек стандартных фрагментов КМОП БИС. Стандартные фрагменты, из которых методом размещения и трассировки собирается большинство макрофрагментов БИС (как правило, нерегулярные схемы или схемы с малой степенью регулярности), являются основными «строительными блоками» в микроэлектронике. Однако любая стандартизация приводит к снижению качества проектирования, а применительно к микроэлектронике – к снижению плотности упаковки топологии. Этот недостаток в последнее время становится все более и более ощутимым, а

борьба за повышение степени интеграции, всегда бывшая одним из ведущих (помимо повышения быстродействия), сейчас является едва ли не основным направлением в микроэлектронике практически всего мира.



а *б*

Рис. 1

Актуальность поиска методов увеличения плотности кристаллов для России очевидна и связана с недостаточным уровнем технологии в стране, выражающимся в недостаточной плотности упаковки схем, что накладывает серьезные ограничения на сложность кристаллов. Перед передовыми компаниями зарубежных стран такая проблема встала особенно остро в последние годы, когда бурное развитие технологии, позволявшее раньше относительно просто решать проблемы нехватки площади кристалла переходом к меньшим проектным нормам, привело, в конце концов, к тому, что уже сравнительно недалек физический предел, поставленный на пути технологов самой природой.

В результате описанных в предыдущем пункте действий в более простых ячейках библиотек (таких, как инверторы, элементы И-ИЛИ-НЕ и т. п.), где внутренняя разводка гораздо проще, чем в сложных фрагментах (таких, как сумматоры, триггеры и т. п.), часть площади пустует (рис. 1). Исходя из этого целесообразно использовать не одну, единую для всего кристалла библиотеку элементов, а семейства библиотек [3], оптимизированных по площади для каждого из входящих в состав кристалла макроблоков.

Технологически инвариантные семейства библиотек стандартных фрагментов МОП БИС, оптимизированных по площади для каждого из входящих в состав кристалла макроблоков, обеспечат не только возможность переноса топологических проектов фрагментов на любые предприятия различных стран, но и более плотную упаковку и топологии фрагментов, и кристалла в целом, чем единая для всего кристалла библиотека элементов.

Система TLIBR технологически инвариантного проектирования семейств библиотек стандартных фрагментов КМОП БИС. В состав системы проектирования семейств библиотек TLIBR помимо исполняемых модулей входит «глобальная» библиотека, содержащая полный набор технологически инвариантных виртуальных описаний стандартных фрагментов, реальные топологические описания которых получаются путем сжатия с учетом только технологических проектных норм.

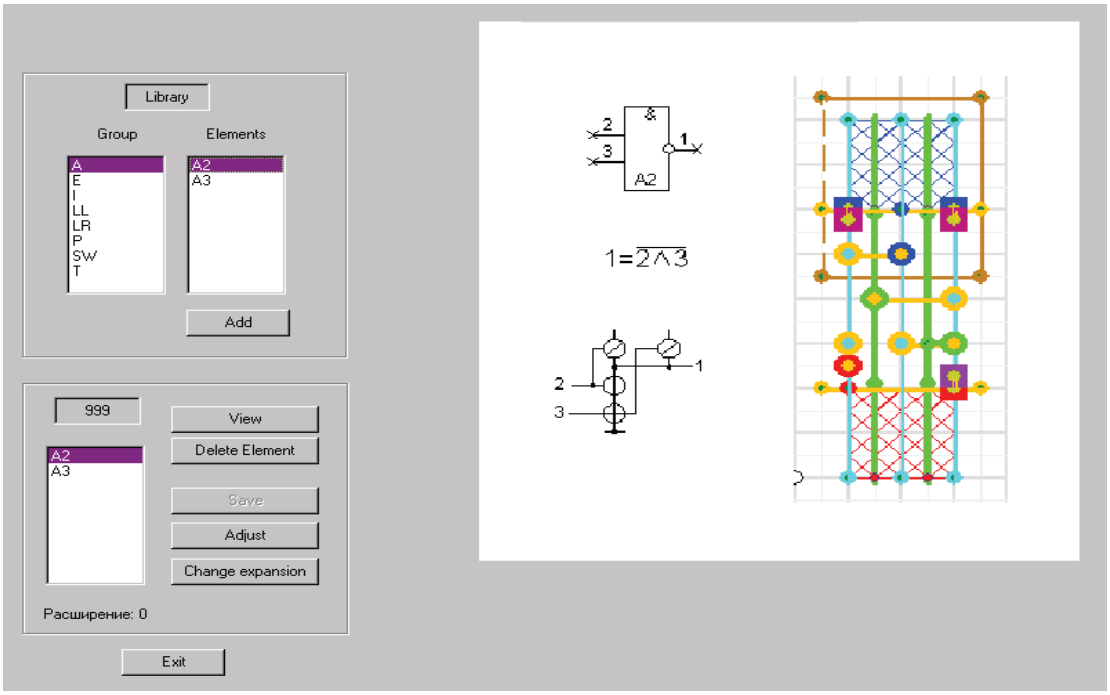


Рис. 2

Проектирование «локальных» библиотек отдельных блоков [3] начинается с выбора и копирования из «глобальной» библиотеки ячеек, используемых в конкретном макрофрагменте, в директорий библиотеки макрофрагмента (что является одной из функций разработанного для проектирования семейств библиотек интерфейса программы TLIBR, соответствующее окно которой приведено на рис. 2). В директории библиотеки макрофрагмента над ними автоматически выполняются действия по определению параметров топологической модели библиотеки и сжатие фрагментов с учетом полученных стандартных для библиотеки ограничений на значения параметров топологической модели.

СПИСОК ЛИТЕРАТУРЫ

1. Зуев И. С., Миронов С. Э. Технологически инвариантная библиотека стандартных фрагментов КМОП БИС // Автоматизация проектирования дискретных систем. CAD DD'97: Материалы 2-й междунар. конф., Минск, 12-14 нояб. 1997 / Институт технической кибернетики НАН Беларуси. Минск, 1997. – Т. 2. – С. 240–245.

2. Технологически инвариантная система проектирования топологии стандартных фрагментов МОП БИС / И. С. Зуев, А. Б. Максимов, С. Э. Миронов, Н. М. Сафьянников // Изв. вузов. Электроника. 2003. – № 3. – С. 63–70.

3. Миронов С. Э. Метод технологически инвариантного проектирования семейств библиотек стандартных фрагментов КМОП БИС / СПбГЭТУ «ЛЭТИ». – СПб., 2003. – 6 с. Деп. в ВИНИТИ 26.08.2003, № 1621-В2003.

O. L. Gavrikova, S. E. Mironov

PROCESS-TOLERANT DESIGN OF THE FAMILIES OF CMOS VLSI STANDARD CELLS LIBRARY

In this article considered the questions of VLSI standard cells library design. New approach to solution the problem of the chip density increase is offered. This approach based on creation of VLSI standard cells library, which optimized by square for each chip macroblock.

Virtual grid, topology compaction, process-tolerant design, VLSI standard cells library

УДК 681.14; 539.107.2

И. В. Матвеева

КВАНТОВЫЕ ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

Обсуждаются наиболее популярные реверсивные квантовые логические элементы, приводятся разные варианты их описания. Основное внимание уделяется контролируемым квантовым преобразователям. Описывается универсальный квантовый вентиль.

Кубит, квантовый регистр, квантовый вентиль, контролируемый вентиль, универсальный квантовый вентиль

Квантовый бит. Основной ячейкой квантового компьютера является квантовый бит, или, сокращенно, *кубит* (*q*-бит) [1], [2]. Это квантовая частица, имеющая два базовых состояния, которые обозначаются $|0\rangle$ [1] и $|1\rangle$. Несмотря на то что кубит может находиться в бесконечно большом количестве «различных суперпозиций», извлечь из него (при измерении) можно только один «классический» бит информации, проецируя состояние квантовой системы на один из базисных векторов и теряя при этом суперпозицию [2] состояний.

Квантовый регистр. Квантовый регистр – цепочка квантовых бит. К базовым состояниям квантового регистра, образованного L кубитами, так же как и в классическом случае, относятся все возможные последовательности нулей и единиц длиной L. Всего может быть 2^L различных комбинаций. Их можно считать записью чисел в двоичной форме от 0 до $2^L - 1$ и обозначать 0, 1, 2, 3, ..., $2^L - 1$. Однако эти базовые состояния не исчерпывают всех возможных значений квантового регистра (в отличие от классического), поскольку существуют еще и состояния суперпозиции. Классического аналога у большинства возможных значений квантового регистра (за исключением базовых) не существует. Говорят, что квантовый регистр может находиться одновременно в нескольких состояниях.

Квантовые вентили. В ходе выполнения вычислений квантовый регистр подвергается преобразованиям, переводящим регистр в новое состояние. Квантовое преобразование (оператор) $\boldsymbol{U}$ над n кубитами может быть представлено матрицей $2^n \times 2^n$. Квантовая механика требует, чтобы преобразование $\boldsymbol{U}$ было унитарным, т. е. $U^{-1} = U^{\mathrm{т}}$, где т – символ транспонирования. Типичное квантовое вычисление – это выполнение над входным значением регистра $|\varphi\rangle$ нескольких унитарных операторов $U_1 U_2 \ldots U_n$ для получения конечного состояния $U_1 U_2 \ldots U_n |\varphi\rangle$.

Минимальная матрица унитарного преобразования имеет размерность 2×2. Схема выполнения однокубитового преобразования показана на рис. 1.

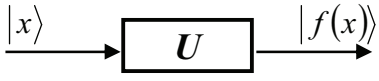


Рис. 1

[1] Состояние квантовой системы и выполняемые над ней преобразования описывают, используя векторы и матрицы, или при помощи бра- и кет-векторов, введенных Дираком в 1958 г.

Независимая переменная $|x\rangle$ представлена в общем случае когерентной суперпозицией базисных состояний $|0\rangle$ и $|1\rangle$: $|x\rangle = a|0\rangle + b|1\rangle$, где a,b – комплексные числа, причем $|a|^2 + |b|^2 = 1$. Унитарный оператор ***U*** воздействует на вектор $|x\rangle$ таким образом, что на выходе формируется результативный вектор $U|x\rangle = U(a|x\rangle + b|x\rangle) = aU|0\rangle = bU|1\rangle = |f(x)\rangle$.

Простые унитарные операции над кубитами называются квантовыми «логическими вентилями» (Deutsch 1985, 1989). В литературе часто встречается другой термин – «гейт» (от английского gate – ворота). По числу задействованных кубит вентили делятся на одно- и многокубитные. Одним из важнейших следствий унитарности квантовых преобразований является реверсивность этих преобразований, т. е. при повторном применении система вернется в исходное состояние. Следовательно, вентили должны быть реверсивными.

Вентиль переводит одно состояние регистра в другое. Действие вентиля на регистр можно записать так: $G|R\rangle = |R'\rangle$. Вентили – линейные операции: $G(|p\rangle + |q\rangle) = G|p\rangle + G|q\rangle$. Произвольный пример действий вентиля: $G\left(\frac{i}{\sqrt{2}}|010\rangle + \frac{1}{2}|001\rangle + \frac{i}{2}|110\rangle\right) = \frac{1}{2}|010\rangle + \frac{1}{2}|101\rangle + \frac{1}{\sqrt{2}}|111\rangle$. Для демонстрации действий вентиля на кубиты обычно используют матричную запись.

Например, если кубит переходит из одного состояния в другое $|0\rangle \to |0\rangle$, $|1\rangle \to \exp(i\omega t)|1\rangle$, то говорится, что по прошествии времени *t* на кубит воздействовали вентилем: $P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$, где $\theta = \omega t$.

В bra/ket-нотации данное выражение записывается как $P(\theta) = |0\rangle\langle 0| + \exp(i\theta)|1\rangle\langle 1|$.

Некоторые элементарные квантовые вентили: $I \equiv |0\rangle\langle 0| + |1\rangle\langle 1|$ – эквивалентность; $X \equiv |0\rangle\langle 1| + |1\rangle\langle 0|$ – НЕ (NOT); $Z \equiv P(\pi)$; $Y \equiv XZ$; $H \equiv \frac{1}{\sqrt{2}}\left[(|1\rangle + |0\rangle)\langle 0| + (|0\rangle - |1\rangle)\langle 1|\right]$ – вентиль Адамара.

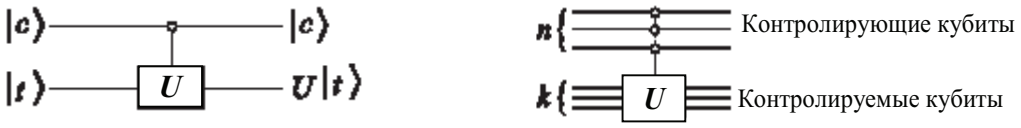
Все они, являясь унитарными операторами, действуют на один кубит и могут быть реализованы посредством какого-либо гамильтониана в уравнении Шредингера. В отличие от классических систем для кубита можно построить неограниченное число вентилей.

Контролируемые (условные) вентили. Из всего множества возможных унитарных операторов наибольший интерес представляет подмножество, которое имеет вид $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U$, где ***I*** – однокубитовая операция эквивалентности, а ***U*** – какой-либо однокубитовый вентиль. Такой вентиль носит название «controlled ***U***» – контролируемый или условный вентиль, поскольку воздействие вентилей ***I*** или ***U*** на второй кубит (кубит цели) контролируется состоянием ($|0\rangle$ или $|1\rangle$) управляющего (контролирующего) кубита. В символьной записи эти вентили записываются в виде CGate, где Gate – это операция с кубитом цели.

Так как область квантовых вычислений бурно развивается на сегодняшний день, в литературе встречаются разные варианты обозначения и изображения вентилей. Графически вентили часто обозначаются кружком или квадратом с цифрой или буквой внутри.

Кубиты представляются горизонтальными нитями. Действие вентиля на кубит показывается путем «нанизывания» вентиля на нужный кубит (или несколько кубит, если это не однобитный вентиль).

Стандартный подход к изображению вентилей «Контролируемое преобразование» – основной блок (непосредственно с обозначением преобразования) отображается на линии того кубита, состояние которого нужно изменить. К управляющим кубитам, состояние которых является условием для выполнения преобразования, подводится линия с точкой, где $\boldsymbol{U}$ – произвольный унитарный оператор (рис. 2, *а*). В более общем виде «Контролируемое преобразование» может быть представлено, как показано на рис. 2, *б*.



а *б*

Рис. 2

В общем случае CNOT($C\,|\,t$) – логический вентиль, в котором целевой кубит t управляется набором кубит C, таких, что $t \notin C$.

Новое состояние целевого кубита t является результатом выполнения операции XOR старого состояния t с состояниями управляющих кубит C, объединенных с помощью операции AND.

Например, вентиль CNOT, показанный на рис. 3, может быть представлен как $\mathrm{CNOT}\left(\left\{x_1, \overline{x_2}, x_3\right\}\,|\,x_4\right)$. Цвет точки ○ или ● показывает, что целевой кубит изменяет свое состояние, когда состояние условного кубита – $|0\rangle$ (инверсное управление) или – $|1\rangle$ (прямое управление) соответственно. В данном примере состояние кубита x_4 изменится, если $x_1 = x_2 = |1\rangle$ и $x_2 = |0\rangle$. Целевой кубит будет изменен согласно операции $x_4 \to x_4 \oplus x_1\overline{x_2}x_3$.

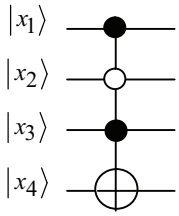


Рис. 3

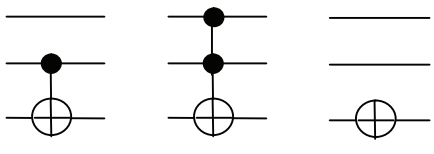
Некоторые специальные случаи вентиля имеют свои собственные имена:

– CNOT с одним управляющим кубитом называется Controlled-Not (CNOT) (рис. 4, *а*);

– CNOT с двумя управляющими кубитами назван вентилем Toffoli (CCNOT) (рис. 4, *б*);

– CNOT без управляющего кубита (рис. 4, *в*) назван NOT-вентиль, в этом случае CNOT(x_i) будет «переброшен» безусловно. Операция NOT графически выглядит как крест в кружке – $\oplus$.

Вентиль «Контролируемое НЕ» (Controlled-Not, CNOT) является квантовым эквивалентом двухвходовой схемы XOR, изменяет состояние одного кубита (контролируемого) на противоположное при условии, что другой (контролирующий) кубит находится в состоянии $|1\rangle$. Преобразование «Контролируемый NOT» (CNOT) может быть определено как $\mathrm{CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$.



а *б* *в*

Рис. 4

Унитарный оператор вентиля: $\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$.

Действие вентиля Controlled-NOT (CNOT) можно записать в виде $|00\rangle \to |00\rangle$, $|01\rangle \to |01\rangle$, $|10\rangle \to |11\rangle$, $|11\rangle \to |10\rangle$. Данный перечень изменения состояний аналогичен таблице истинности классического двоичного логического вентиля.

Другой популярный квантовый контролируемый вентиль носит название вентиль Тоффоли, или Controlled-controlled -NOT (CCNOT).

Унитарный оператор вентиля: $\mathrm{CCNOT} = \left(\begin{array}{cccccc:cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right)$.

Матрица оператора CCNOT характерна для вентилей «Контролируемое преобразование». Нижняя матрица 2×2 описывает преобразование (в данном случае – инвертирование), а верхняя единичная – «контроль», ее размерность зависит от числа контролирующих кубит.

Иногда в литературе вентиль Тоффоли обозначается большой латинской буквой *T* [3]. С помощью *T* можно построить полный набор булевых вентилей, так как операторы NOT и AND конструируются следующим образом:

$$T|1,1,x\rangle = |1,1,\neg x\rangle,\ T|x,y,0\rangle = |x,y,x \wedge y\rangle.$$

Другой популярный вентиль – вентиль Фредкина (Fredkin gate) представляет собой управляемую перестановку CSWAP и может быть определен как

$F = |0\rangle\langle 0| \otimes I \otimes I + |1\rangle\langle 1| \otimes S$, где S – операция перестановки:

$S = |00\rangle\langle 00| + |01\rangle\langle 10| + |10\rangle\langle 01| + |11\rangle\langle 11|$.

Вентиль *F*, как и *T*, достаточен для представления любой комбинаторной схемы:

$$F|x,0,1\rangle = |x,x,\neg x\rangle,\ F|x,y,1\rangle = |x,y \wedge x,y \wedge \neg x\rangle,\ F|x,0,y\rangle = |x,y \wedge x,y \wedge \neg x\rangle.$$

Хотя преобразования *T* и *F* достаточно полны для реализации произвольной булевой схемы, с их помощью нельзя представить произвольное квантовое преобразование. Потребуется еще добавить однобитовые вращения. Можно показать, что любая унитарная матрица размерности $n \times n$ может быть образована путем комбинирования двухкубитовых вентилей CNOT и вентилей вращений одного кубита. Таким образом, данная пара операций может рассматриваться в квантовых вычислениях как универсальная, т. е. путем комбинирования операций вращения и CNOT можно описать операцию контролируемого вра-

щения, являющуюся одиночным универсальным вентилем. Достаточно добавить следующие однобитовые вращения и преобразования сдвига фазы:

$$\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}, \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{-i\alpha} \end{pmatrix}, \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{pmatrix} \text{ для всех } \alpha.$$

Подобные неклассические вращения и сдвиг фазы являются решающим фактором для использования возможностей квантового компьютера.

Описание подобных универсальных вентилей можно найти у Дойча (Deutsch et. a1. 1995), Ллойда (Lloyd 1995), Ди Винченцо (DiVincenzo 1995a) и Баренцо (Barenco 1995) [5] – [7].

Следует отметить, что двухкубитовые вентили достаточны для выполнения квантовых вычислений. Именно поэтому квантовый вентиль является мощным и важным понятием.

В заключение хочется отметить, что квантовые вычисления развиваются чрезвычайно быстро и интенсивно и задача построения квантового компьютера уже выходит из области чисто теоретических умозрительных задач. Несмотря на то что до построения реального квантового компьютера еще далеко, в качестве одного из многочисленных примеров можно сослаться на недавнюю публикацию в Интернете: «Команда профессора Дитера Мешеде (Dieter Meschede) из института прикладной физики университета Бонна (Institut fur Angewandte Physik – http://www.iap.uni-bonn.de/) сумела построить квантовый регистр из нейтральных атомов. Следующий шаг группы – организация с помощью аналогичной технологии квантовых гейтов – аналогов логических преобразований в обычном компьютере. Это может занять два года, говорят в Бонне».

СПИСОК ЛИТЕРАТУРЫ

1. Schumacher B. Quantum coding // Phys. Rev. 1995. A 51. P. 2738–2747.
2. Герасимов В. И., Сафьянников Н. М. Квантовый объект информации // Изв. СПбГЭТУ «ЛЭТИ». Сер. «Управление, информатика и вычислительная техника». 2001. №1. С. 19–22.
3. Доич Д., Экерт А., Лупачини Р. Машины, логика и квантовая физика // Математическое просвещение. 2001. Сер 3, вып. 5. С. 47–60.
4. Стин Э. Квантовые вычисления. Ижевск, 2000.
5. Conditional Quantum Dynamics and Logic Gates / A. Barenco, D. Deutsch, A. Ekert, R. Jozsa // Phys. Rev. Lett. 1995. Vol. 74, № 20. P. 4083-4086.
6. Barenco A. A Universal Two-Bit Gate for Quantum Computation // Proc. R. Soc. London A. Vol. 449. P. 679–683.
7. Deutsch D., Barenco A. and Ekert A. Universality in Quantum Computation. // Proc. R. Soc. London A. Vol. 449. P. 669–677.

I. V. Matveeva

QUANTUM GATES

Most popular reversible quantum logic elements are envisaged, the different variants of their notation are shown. The Main attention is spared controlled quantum converter. The universal quantum gate is described.

Qubit, quantum array, quantum gate, controlled gate, universal quantum gate

УДК 681

П. И. Васькин, Нгуен Доанг Куонг

ИСПОЛЬЗОВАНИЕ МЕТОДОВ ОБНАРУЖЕНИЯ ЗНАНИЙ В БАЗАХ ДАННЫХ С ВРЕМЕННЫМИ РЯДАМИ

Предлагается новый способ для анализа баз данных с временными рядами (БДВР), который позволяет систематизировать накопление статистической информации. Процесс открытия знаний в БДВР включает: фильтрацию данных временных рядов; расчет значения индикатора обобщенного тренда (ИОТ); расчет индикатора тренда самого ИОТ; накопление статистической информации, сортируемой в зависимости от значений ИОТ и его тренда. Подход демонстрируется на базе данных рынка FOREX, которая является базой данных с временными рядами по своей сущности.

Временные ряды, базы данных, обнаружение знаний, обобщенный тренд

Введение. В отличие от статических баз данных базы данных с временными рядами (БДВР) содержат множество записей, в которых некоторые атрибуты или действия связаны с временными метками.

В БДВР рынка FOREX каждая запись включает кроме статических атрибутов, таких, как имя валютной пары, некоторые динамические атрибуты, такие, как цена открытия, закрытия, максимальная и минимальная, относящиеся к определенному интервалу времени. В базах данных такого типа временная размерность является неотъемлемой частью БД и должна быть учтена в методологии обнаружения знаний.

В то время как в статических базах данных обычно ссылаются на сами атрибуты в процессе извлечения знаний, в динамической базе данных ссылаются на свойства, определяемые через атрибуты. Примером такого свойства является тренд, который понимается как угол наклона прямой, аппроксимирующей значения атрибута на определенном отрезке времени.

Сырые данные рынка FOREX зашумлены и трудны для обработки. Атрибуты каждой записи подвержены влиянию различных факторов, затрудняя наблюдение признаков длительного срока. Поэтому необходима предварительная обработка первоначальных сырых данных и работа с преобразованной информацией. Предварительная обработка включает в себя фильтрацию. Существуют несколько различных способов фильтрации данных. Будем использовать для фильтрации вычисление экспоненциальной скользящей средней соответствующих значений обрабатываемого атрибута.

Отметим, что большая часть исследователей традиционно сосредоточивается на извлечении знаний из статических реляционных баз данных. Специальные свойства управляющих систем для временных рядов, в отличие от систем управления базами данных (СУБД), изучались Дрейером и др. [1]. Хан и другие исследовали базы данных с временными рядами для периодических сегментов [2] и частично – для периодических шаблонов [3], используя технику обогащения данных. Их техника (основанная на обогащении данных методами ассоциативных правил [4]) предназначена для открытия временных шаблонов (групп событий, упорядоченных по времени) и, в некоторой степени, временных правил (причинно-следственных отношений между событиями).

Лу и др. изучали фондовый рынок для извлечения межтранзакционных ассоциативных правил [5]. Была представлена формулировка *N*-размерной транзакционной базы дан-

ных. Лу определяет события на временной шкале и пытается извлечь связи между подобными шаблонами событий. Другой метод открытия межсобытийных ассоциаций («эпизодов») описан Маниллой и др. [6]. Попытка представления временных данных временными рядами с нечеткой логикой дана в [7].

Большая размерность входного пространства временных рядов накладывает серьезные ограничения на существующие методы обогащения данных. Поэтому неудивительно, что в литературе невозможно найти какие-либо удовлетворительные результаты предсказания поведения рынка FOREX или других временных явлений аналогичной природы.

Предметная область.

Определение 1. База данных с временными рядами – это набор записей $\left\{r_j^{\Delta t}\right\}_{j=1}^{N(\Delta t)}$, таких, что каждая запись содержит множество атрибутов и значение времени $r_j^{\Delta t} = \left\{s_1, s_2, ..., s_k, d_1, d_2, ..., d_m, t_j^{\Delta t}\right\}$.

Каждый атрибут может быть либо вещественным числом $a_i \in IR$, либо дискретным $a_i \in IN$ и может иметь или не иметь связи со значением времени. Если атрибут связан со значением времени, он называется динамическим, в противном случае – статическим.

Будем рассматривать следующие значения интервала времени $t_j^{\Delta t}$: 1 мин, 5 мин, 15 мин, 1 ч, 4 ч и 1 день.

Определение 2. Атрибутная функция – это функция от времени, элементы которой являются значениями атрибута i в записях, и она отображается как функция от времени $a_i^{\Delta t}(t)$, так что

$$a_i^{\Delta t}(t_\chi) = a^{\Delta t} \in r_j^{\Delta t}, \exists\, t_\chi \in r_j^{\Delta t},$$

где $a_i^{\Delta t}$ – атрибут i во временном ряде с интервалом Δt; $r_j^{\Delta t}$ – j-я запись из базы данных во временном ряде с интервалом Δt; t_χ – момент времени, связанный с этой записью.

Определение 3. Свойство определяется на интервале времени $[t_1, t_2]$, если некоторая атрибутная функция $a_i^{\Delta t}(t)$ может быть аппроксимирована другой функцией от времени, т. е. $\varphi^{\Delta t}(t)$.

Говорят, что $\varphi^{\Delta t}$ и ее параметры являются свойством $a_i^{\Delta t}(t)$ на интервале $[t_1, t_2]$. Например, если $\varphi^{\Delta t}(t) = \alpha_i^{\Delta t} t + \beta_i^{\Delta t}$ на некотором интервале, можно сказать, что в этом интервале функция $a_i^{\Delta t}(t)$ имеет наклон $\alpha_i^{\Delta t}$, где наклон – это свойство, извлеченное из $a_i^{\Delta t}(t)$ на некотором интервале.

Определение 4. Изменение свойства между двумя соседними интервалами является событием.

Предварительная обработка баз данных с временными рядами. Предварительная обработка баз данных с временными рядами заключается в фильтрации сырых данных.

Как уже упоминалось, база данных с временными рядами рынка FOREX очень зашумлена. Цена каждой сделки зависит от случайных событий, но можно также предположить наличие долгосрочного тренда. Предварительная обработка сырых данных состоит в получении очищенных данных с небольшим (насколько это возможно) аддитивным шумом.

Предположим, что сырые данные $a_{\text{raw}}(n)$ состоят из сигнала долгосрочного тренда $a(n)$ и шума $e(n)$ аддитивной природы, т. е. $a_{\text{raw}}(n) = a(n) + e(n)$.

Операция очистки состоит в нахождении $\hat{a}(n)$ как приближения долгосрочного сигнала $a(n)$. Для очистки данных, как правило, используют операцию фильтрации низкой частоты, т. е. операцию, которая исключает волны высокой частоты (относящиеся в наибольшей мере к шуму) и оставляет только низкочастотные волны (относящиеся в наибольшей мере к долгосрочному сигналу).

Имеется достаточно много операций фильтрации низкой частоты в области времени и частоты: простая скользящая средняя, взвешенная скользящая средняя и т. д. Следует отметить, что независимо от специалистов по обработке сигналов технические аналитики рынка FOREX подошли к использованию техники сглаживания для фильтрации сигналов.

Подавляющее число технических аналитиков сегодня предпочитают экспоненциальное сглаживание, которое заключается в нахождении экспоненциального скользящего среднего (ЭСС). С точки зрения отношения к новым данным, генерируемым рынком, ЭСС представляет собой идеальный компромисс между повышенной чувствительностью взвешенного скользящего среднего и заметным отставанием простого скользящего среднего. В отличие от многих других техник усреднения ЭСС следует за трендом текущих данных более гладко, с минимальным количеством скачков и наименьшим запаздыванием [8].

С точки зрения вычислений, ЭСС также представляет немало выгод пользователю: данный метод предполагает небольшое число расчетов и не слишком сложные манипуляции с данными. Для получения новых значений экспоненциального скользящего среднего необходимы численные значения, относящиеся только к двум периодам: самому последнему периоду необработанных данных и предшествующему периоду экспоненциального скользящего среднего. Кроме того, метод вычисления ЭСС позволяет избежать текущих ошибок, связанных с потерей несущественных и малозначимых данных.

Существует несколько подходов к вычислению экспоненциального скользящего среднего. Для его расчета воспользуемся следующей формулой: $\text{EMA} = \frac{1}{T}C + \frac{T-1}{T}\text{Ep}$, где EMA – ЭСС текущего периода; C – цена закрытия текущего периода; Ep – ЭСС предшествующего периода; T – число периодов ЭСС (период ЭСС).

Для каждой валютной пары и всех рассматриваемых интервалов времени (1 мин, 5 мин, 15 мин, 1 ч, 4 ч, 1 день) определим три ЭСС с периодами 8, 13 и 55. Значения периодов ЭСС взяты из последовательности Фибоначчи.

ЭСС с периодом 55 будем использовать для оценивания статической составляющей индикатора обобщенного тренда (долгосрочный тренд). Динамическую составляющую обобщенного тренда будем оценивать с помощью экспоненциальных скользящих средних с периодами 8 и 13.

Индикатор обобщенного тренда. Изменение рыночных цен происходит в форме тренда. Характер тренда определяется многими переменными. На движение цен влияют перемены, происходящие на уровне соответствующих фундаментальных экономических факторов. Таким образом, в момент возникновения тренда причина его появления не все-

гда очевидна. Определить обстоятельства, способствовавшие рождению конкретного тренда, не всегда удается даже по прошествии значительного временного периода. Технические индикаторы рынка – это инструмент идентификации тренда и изменений в нем; причины формирования тренда не могут быть определены с их помощью.

Известные технические индикаторы рынка, как правило, ограничены рамками конкретной валютной пары и конкретного интервала времени, даже когда индикатор является комбинацией нескольких других индикаторов.

Предлагаемый технический индикатор рынка не выходит за рамки одной валютной пары, но в его основе лежат 18 экспоненциальных скользящих средних для 6 интервалов времени: 1 мин, 5 мин, 15 мин, 1 ч, 4 ч и 1 день. Из общепринятых в техническом анализе рынка FOREX интервалов времени в данный список не включены следующие интервалы: 10 мин и 30 мин. При выборе интервалов времени мы руководствовались следующим соображением: интервалы должны отличаться друг от друга примерно в одинаковое число раз (в данном случае в 3…6 раз).

Пусть $c_{ij}^{\Delta t}$ – атрибутная функция, соответствующая цене закрытия валютной пары i в j-й период по интервалу Δt. Обозначим через $e_{i8}^{\Delta t}(t)$ ($e_{i13}^{\Delta t}(t)$, $e_{i55}^{\Delta t}(t)$) аппроксимирующую ее функцию в виде экспоненциальной скользящей средней с периодом 8 (13, 55), а через $u_{i8}^{\Delta t}(t)$ ($u_{i13}^{\Delta t}(t)$, $u_{i55}^{\Delta t}(t)$) – свойство атрибутной функции, которое определяется как угол наклона аппроксимирующей функции (значение производной). Тогда индикатор обобщенного тренда (ИОТ) будем рассчитывать по формуле

$$I(t)=\sum_{\Delta t}\left(\frac{u_{i55}^{\Delta t}(t)}{\max\limits_{t'<t}\left|u_{i55}^{\Delta t}(t')\right|}+\frac{u_{i8}^{\Delta t}(t)-u_{i13}^{\Delta t}(t)}{\max\limits_{t'<t}\left|u_{i8}^{\Delta t}(t')-u_{i13}^{\Delta t}(t')\right|}\right),$$

где $\Delta t\in\{1\text{ мин},5\text{ мин},15\text{ мин},1\text{ ч},4\text{ ч},1\text{ день}\}$.

Пусть $ie_{i8}^{5\text{ мин}}(t)$ ($ie_{i13}^{5\text{ мин}}(t)$, $ie_{i55}^{5\text{ мин}}(t)$) – экспоненциальная скользящая средняя индикатора обобщенного тренда с периодом 8 (13, 55), вычисленная для пятиминутного интервала, а $iu_{i8}^{5\text{мин}}(t)$ ($iu_{i13}^{5\text{мин}}(t)$, $iu_{i55}^{5\text{мин}}(t)$) – угол ее наклона. Тогда индикатор тренда самого ИОТ будем вычислять по формуле

$$IT(t)=\frac{iu_{i55}^{5\text{ мин}}}{\max\limits_{t'<t}\left|iu^{5\text{ мин}}{}_{i55}(t')\right|}+\frac{iu_{i8}^{5\text{ мин}}(t)-iu_{i13}^{5\text{ мин}}(t)}{\max\limits_{t'<t}\left|iu_{i8}^{5\text{ мин}}(t')-iu_{i13}^{5\text{ мин}}(t')\right|}.$$

Успешная торговля на рынке FOREX зависит от умения трейдера прогнозировать поведение рынка и принятия решений в реальном времени. Пятиминутный интервал является приемлемым временем принятия решения. За этот промежуток трейдер должен проанализировать значения ИОТ, тренда ИОТ и накопленную статистическую информацию.

Формирование гистограмм распределения. Индикатор обобщенного тренда может принимать значения в диапазоне [–12; +12], а индикатор тренда ИОТ – в диапазоне [–2; +2]. В любой момент времени t, с точностью до пятиминутного интервала, состояние рынка FOREX для рассматриваемой валютной пары будем характеризовать парой $\{I(t),IT(t)\}$. Разделим каждый

из диапазонов на равные промежутки, например: $\{-12; -10; -8; -6; -4; -2; 0; 2; 4; 6; 8; 10; 12\}$ и $\{-2; -1,6; -1,2; -0,8; -0,4; 0; 0,4; 0,8; 1,2; 1,6; 2\}$.

Тогда получаем 143 (13×11) возможных канонических состояния рынка, в которых ИОТ и его тренд принимают значения, указанные выше. Произвольное состояние рынка будем раскладывать на комбинацию четырех канонических. Рассмотрим конкретный пример. Пусть рынок находится в состоянии, когда значение индикатора обобщенного тренда равно – 4,2, а значение тренда ИОТ равно 0,9. Это состояние рынка будет выражаться через следующие канонические состояния: $\{-6; 0,8\}$, $\{-6; 1,2\}$, $\{-4; 0,8\}$ и $\{-4; 1,2\}$.

Обозначим через $\mu_{rz}^{r'z'}$ принадлежность состояния рынка $\{r'; z'\}$ каноническому состоянию $\{r; z\}$. Она вычисляется по формуле $\mu_{rz}^{r'z'} = \left(1 - \frac{|r - r'|}{\Delta r}\right) \times \left(1 - \frac{|z - z'|}{\Delta z}\right)$, где Δr – шаг шкалы для значений ИОТ, а Δz – шаг шкалы для значений тренда ИОТ.

В рассматриваемом примере имеем: $\mu_{-6;0,8}^{-4,2;0,9} = 0,075$; $\mu_{-6;1,2}^{-4,2;0,9} = 0,025$; $\mu_{-4;0,8}^{-4,2;0,9} = 0,675$; $\mu_{-4;1,2}^{-4,2;0,9} = 0,225$.

Для каждого канонического состояния рынка, которое встречалось хотя бы один раз, формируем три гистограммы распределения, а именно: гистограмму максимальных цен, гистограмму минимальных цен и гистограмму цен закрытия.

Во-первых, поясним, почему отсутствует гистограмма цен открытия. В гистограммах фиксируются не сами цены сделок, а их разница по отношению к цене открытия. Таким образом, гистограмма цен открытия вырождается в одно значение 0 с вероятностью 1.

Во-вторых, оговорим длительность интервала, в течение которого будем фиксировать максимальную и минимальную цену, а в момент его завершения – цену закрытия. На наш взгляд, стратегия игры на рынке FOREX, в основе которой лежит технический анализ, должна ориентироваться на позиции, открываемые на относительно небольшой срок (1…2 ч), поэтому в качестве длительности интервала, используемого при построении гистограмм, возьмем 1 ч. Предполагаем, что после каждого часа с момента открытия позиции трейдер либо ее закроет, либо примет решение продлить нахождение в рынке еще на 1 ч.

В-третьих, изложим позицию относительно формы извлечения знаний. Многие исследователи пытаются сформулировать знания в виде правил. Но даже правила с нечеткой логикой слишком грубы для выражения сугубо вероятностной природы движения цен на рынке FOREX. Для этих цен абсолютно только одно правило: «В любой момент времени цена может пойти в любую сторону». Поэтому правильным подходом к данной предметной области, на наш взгляд, является систематизация в формировании статистической информации и использование процедуры принятия решения на основе собранной статистической информации.

Таким образом, зная в момент времени t значения $I(t)$ и $IT(t)$, в момент времени $t + 1$ ч можно иметь всю необходимую информацию для пополнения гистограмм. Остается только организовать процесс регистрации соответствующих цен и процесс занесения новой информации в гистограммы распределения.

Процедура принятия решения. В момент времени t на основании значений $I(t)$ и $IT(t)$ строим гистограммы распределения максимальных, минимальных цен и цен закрытия путем объединения соответствующих гистограмм канонических состояний рынка с использованием коэффициентов принадлежности.

Гистограмма цен закрытия используется для принятия решения о вступлении в рынок, а также о том, какого типа позицию необходимо открывать: длинную (покупка) или короткую (продажа). Если математическое ожидание цены закрытия близко к нулю, то открывать позиции в данный момент не стоит. Если математическое ожидание цены закрытия больше нуля, можно открыть длинную позицию (совершить сделку покупки лота рассматриваемой валютной пары). Если математическое ожидание цены закрытия меньше нуля, то можно открыть короткую позицию (совершить сделку продажи лота).

Гистограмма распределения максимальных цен используется для формирования ордера на снятие прибыли при открытии длинной позиции. При открытии короткой позиции она нужна для формирования ордера на фиксацию убытков.

Гистограмма распределения минимальных цен используется для формирования ордера на снятие прибыли при открытии короткой позиции. При открытии длинной позиции она помогает формировать ордер на фиксацию убытков.

СПИСОК ЛИТЕРАТУРЫ

1. Dreyer W., Kotz Dittrich A., Schmidt D. Research Perspectives for Time Series Management Systems // Data Mining and Knowledge Discovery. 1994. Vol. 23, № 1. P. 10–15.
2. Han J., Gong W., Yin Y. Mining Segment-Wise Periodic Patterns in Time-Related Databases // Proc. 1998 Int. Conf. On Knowledge Discovery and Data Mining (KDD'98), New York City, NY, Aug. 1998. P. 214–218.
3. Han J., Dong G., Yin Y. Efficient Mining of Partial Periodic Patterns in Time Series Database // Proc. 1999 Int. Conf. on Data Engineering (ICDE'99), Sydney, Australia, March 1999.
4. Srikant R., Agrawal R. Mining Quantitative Association Rules in Large Relational Tables // Proc. ACM-SIGMOD 1996 Conf. on Management of Data, Montreal, Canada, 1996.
5. Lu H., Han J., Feng L. Stock Movement Prediction and N-Dimensional Inter-Transaction Association Rules // Proc. of 1998 SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98), Seattle, Washington, June, 1998. P. 12:1–12:7.
6. Manilla H., Toivonen H., Verkamo A. I. Discovery of Frequent Episodes in Event Sequences // Data Mining and Knowledge Discovery. 1997. Vol. 1, № 3. P. 259–289.
7. Tsai C., Wu S. A study for Second Order Modeling of Fuzzy Time Series // Proc. of 1999 IEEE International Fuzzy System Conf., Aug., 1999. P. 719–725.
8. Энциклопедия технических индикаторов рынка / Пер с англ. 2-е изд. М.: Альбина Бизнес Букс, 2004.

P. I. Vaskin, Nguyen Doan Cuong

KNOWLEDGE DISCOVERY TECHINQUE FOR TIME SERIES DATABASES

In this paper we introduce a new method for analyzing Time Series Databases (TSDB) which allows to systematize the accumulation of statistical information. The process of knowledge discovery in TSDB includes: filtering of time series data, calculation of generalized trend indicator (GTI) value, calculation of GTI's own trend indicator, accumulation of statistical information that is sorted by GTI and its trend values. We demonstrate our approach on a FOREX market database which is a Time Series Database by its nature.

Time Series, databases, knowledge discovery, generalized trend

УДК 331.101.1 + 004.5

П. И. Падерно, Кванг До Хонг

ОЦЕНКА КАЧЕСТВА ПРОГРАММНЫХ ПРОДУКТОВ С ПОМОЩЬЮ МЕТОДА АНАЛИЗА ИЕРАРХИЙ

Проведен анализ понятия эффективность прикладных программ. Предложено оценивать эффективность прикладных программ с помощью метода анализа иерархий. Рассмотрены два подхода к комплексированию мнений экспертов и приведены результаты эксперимента (опроса пользователей относительно трех типов программных средств).

Прикладные программы, метод анализа иерархий, комплексирование мнений экспертов, коэффициенты значимости, факторы, предпочтения конкретных типов пользователей

Рынок программного обеспечения с каждым днем становится все более и более разнообразным. В любой области применения, будь то полиграфия, бухгалтерия или конструкторские работы, существует большое количество программ, однотипных с точки зрения целевого использования, но сильно различающихся как функциональными возможностями, удобством использования и легкостью освоения, так и стоимостью. Однако пользователи, работающие над схожими задачами и использующие похожие программы, могут иметь различные индивидуальные пожелания, и их требования к тем или иным характеристикам и возможностям этих программ могут очень сильно различаться. Поэтому большое количество программных продуктов (ПП), не столь мощных и многофункциональных, как лидеры, также находят своего потребителя, поскольку и стоят, как правило, дешевле.

Все частные свойства (показатели) ПП разбиты на три группы: внутрисистемные свойства, знание которых необходимо в основном только высококвалифицированному пользователю, показатели назначения, отражающие степень адекватности самого объекта предъявляемым к нему требованиям, а также свойства (показатели), отражающие удобство работы с объектом для конкретного пользователя (группы пользователей).

Данный подход позволяет представить эффективность ПП в виде кортежа $E = \langle S, N, H \rangle$, где S – внутрисистемные свойства (группа показателей), обеспечивающие эффективную работу ПП на системном уровне; N – свойства (показатели) назначения, отражающие степень адекватности самого ПП предъявляемым к нему требованиям, направленным на достижение конкретной цели; H – свойства (показатели), отражающие удобство работы с ПП для конкретного пользователя (от термина Human factors).

Оценка эффективности (качества) ПП базируется на использовании метода анализа иерархий (МАИ). Каждая из групп свойств (показателей) верхнего уровня иерархии S, N, H в свою очередь разбивается на несколько подгрупп свойств (показателей) следующего уровня. Операция разбиения показателей на группы показателей более низкого уровня продолжается на каждом уровне иерархии. Процесс построения иерархии показателей продолжается до конечных показателей эффективности нижнего уровня, не все из которых могут быть оценены строгими математическими методами.

Полученная иерархическая модель эффективности ПП включает номенклатуру показателей, которая значительно шире номенклатуры стандартов ИСО-9000.

Коэффициенты значимости показателей эффективности на каждом из уровней иерархии вычисляются *на основе словесных высказываний (предпочтений) пользователей* по специально разработанному алгоритму. Процесс вычисления коэффициентов значимости показателей различных уровней иерархии продолжается до конечных показателей эффективности нижнего уровня.

Для определения весов (важностей) конечных характеристик (показателей нижнего уровня) для конкретного пользователя (групп пользователей) экспертами заполняются матрицы сравнений для всех взаимосвязей каждого из элементов с элементами более низкого уровня. Обработка набора таких матриц позволяет получить значения весов каждого из частных критериев эффективности по всем конкретным характеристикам.

Комплексирование с учетом весовых коэффициентов дает возможность не только получить оценки эффективности ПП для конкретных групп пользователей, но и выбрать ПП, наиболее эффективный (отвечающий требованиям) для типовых групп пользователей, т. е. определить объект, наиболее подходящий для конкретного пользователя.

Для экспериментальной оценки характеристик эффективности ПП были выбраны следующие программные продукты, широко представленные на рынке программ, достаточно известные и часто используемые различными пользователями: Adobe Photoshop 7.0; Компас 3D-LT 5.11; Netscape Navigator 7.2.

К проведению эксперимента были привлечены (в качестве экспертов) студенты 5-го курса специальности АСОИУ, достаточно хорошо подготовленные в области использования разнообразных ПП. После их опроса в зависимости от степени знакомства и владения вышеперечисленными программными продуктами они были разделены на 3 группы: начинающие (23 человека); полупрофессионалы (6 человек); профессионалы (3 человека).

Каждой из этих групп была представлена структура свойств ПП, наиболее соответствующая уровню знаний и умений (квалификации) экспертов (применительно к данному ПП). Далее по мнению экспертов были составлены наборы соответствующих матриц и проведена оценка значимости отдельных составляющих качества.

Результаты эксперимента, полученные после комплексирования мнений экспертов относительно значимости показателей верхнего уровня, приведены в таблице.

Комплексирование мнений экспертов. Введем следующие обозначения: n – количество экспертов; B_i – вектор, полученный в результате обработки мнений i-го эксперта по методу анализа иерархий до операции нормирования; α_i – коэффициенты значимости экспертов, причем $\sum_{i=1}^{n}\alpha_i = 1$.

Первый подход – средневзвешенное арифметическое:

$$\alpha_1 B_1 + \alpha_2 B_2 + ... + \alpha_n B_n = C_H,\ D_H = \left\{ c_{hi} \Big/ \sum_{i=1}^{k} c_{hi} \right\}_{i=1}^{k},$$ где c_{hi} – элементы вектора C_H.

Второй подход – средневзвешенное геометрическое:

$$\left(B_1\right)^{\alpha_1} \bullet \left(B_2\right)^{\alpha_2} \bullet ... \bullet \left(B_n\right)^{\alpha_n} = C_G,\ D_G = \left\{ c_{gi} \Big/ \sum_{i=1}^{k} c_{gi} \right\}_{i=1}^{k},$$ где c_{gi} – элементы вектора C_G; знак • означает произведение векторов (матриц) в смысле Адамара (поэлементное); возведение в степень также происходит в смысле Адамара.

Необходимо заметить, что выполнение неравенства $c_{gi} > c_{hi}$ означает, что для данного *i*-го частного показателя качества мнения экспертов *были более согласованы*, чем для большинства остальных случаев.

В таблице обозначены факторы: X_1 – эффективность, X_2 – удобство эксплуатации ППП, X_3 – стоимость.

Программный продукт		Пользователи					
		Начинающий		Полупрофессионал		Профессионал	
		D_h	D_g	D_h	D_g	D_h	D_g
Adobe Photoshop 7.0 *Начинающий: 8* *Полупрофесс.: 4* *Профессионал: 2*	X_1	0,42274	0,41916	0,53625	0,52781	0,38289	0,37625
	X_2	0,08902	0,08792	0,09961	0,13964	0,08754	0,08653
	X_3	0,48824	0,49302	0,36423	0,33255	*0,52957*	*0,53724*
Компас-3D LT 5.11 *Начинающий: 13* *Полупрофесс.: 2* *Профессионал: 2*	X_1	*0,79333*	*0,79232*	*0,7545*	*0,7562*	0,7316	0,7255
	X_2	*0,18396*	*0,18623*	0,2234	0,2223	*0,2345*	*0,2398*
	X_3	0,02271	0,02145	*0,0221*	*0,0215*	*0,0339*	*0,0347*
Netscape Navigator 7.2 *Начинающий: 6* *Полупрофесс.: 4* *Профессионал: 2*	X_1	*0,38932*	*0,39414*	*0,62707*	*0,63828*	0,42615	0,43852
	X_2	0,32251	0,31725	0,21786	0,21361	0,42329	0,42729
	X_3	*0,28817*	*0,28861*	0,15507	0,14811	0,15056	0,13419

Результаты:

1. Результаты, выделенные в таблице курсивом, указывают на то, что мнения экспертов по поводу этих показателей были согласованы между собой сильнее, нежели для других показателей в данной экспертизе.

2. Достаточно единодушное мнение всех категорий экспертов по поводу стоимости программного продукта Компас-3D LT 5.11 (третья строка второго раздела таблицы) обусловлено его низкой стоимостью и, соответственно, сказалось на значениях остальных показателей (завышенных).

3. Различное отношение неквалифицированных пользователей к сходным частным показателям качества обусловлено варьированием иерархических структур показателей для разных ПП.

Вывод. При проведении дальнейших экспериментов с целью выявления предпочтений конкретных групп пользователей необходимо предъявлять всем группам пользователей *единую* иерархию показателей качества и выбирать для эксперимента ПП, близкие по стоимости.

Pavel I. Paderno, Quang Do Hong

ESTIMATION QUALITY SOFTWARE PRODUCTS BY MEANS OF THE METHOD ANALYSIS HIERARCHIES

The analysis concept efficiency of application programs is lead. It is offered to estimate efficiency of application programs by means of the method analysis hierarchies. Two approaches to interconnecting opinions of experts are considered and results of experiment (interrogation of users concerning three types of software) are adduced.

Application programs, method analysis hierarchies, interconnecting opinions of experts, coefficients values, factors, preferences of specific types users

УДК 681.3

О. А. Биниенко

БАЙЕСОВСКИЕ СЕТИ В ЗАДАЧЕ УПРАВЛЕНИЯ ПРЕДВЫБОРНОЙ КАМПАНИЕЙ

Рассмотрена экспертная система для решения задачи управления предвыборной кампанией. Подробно описывается основной модуль управления – модуль байесовских сетей.

Выборы депутатов, управление ведением предвыборной кампании, применение байесовской сети доверия

Процесс управления, байесовские сети доверия как средство решения задачи управления. Процесс управления ведением предвыборной кампании можно разделить на три этапа: формирование стратегии, разработка тактики и собственно управление избирательными кампаниями (оперативное).

На этапе формирования стратегии вырабатываются цели и задачи предвыборной кампании, определяется критерий выполнения этих целей и задач. Существует множество доступных стратегий, и суть формирования стратегии состоит в формировании этого множества и выборе оптимальной стратегии перед началом предвыборной кампании. Оптимальной считается та стратегия, которая максимизирует или минимизирует некоторый критерий. Разработка тактики – это выбор способов и приемов, которыми достигаются стратегические цели. Оперативное управление заключается в корректировке стратегии и тактики и принятии решений, детализирующих стратегию, по ходу ведения кампании. Оно включает набор действий по корректировке предвыборной кампании при получении новой оперативной информации или при изменении оперативной обстановки, реагирование на не предсказуемые изначально события, например форс-мажорные обстоятельства, исправление совершенных ошибок.

Экспертная система содержит подсистему управления ведением предвыборной кампании, которая состоит из модулей управления [см. лит.]. В экспертной системе имеются модули для решения частных задач управления и по крайней мере один модуль управления универсального назначения. Экспертная система имеет открытую архитектуру, что позволяет добавлять модули управления. В настоящий момент система имеет только один модуль для решения частных задач управления – модуль распределения финансовых средств. В качестве математического аппарата, лежащего в основе модуля управления универсального назначения, выбраны байесовские сети доверия. Опишем этот модуль подробнее.

Структура байесовской сети управления ведением предвыборной кампании. Подсистема прогнозирования исхода выборов позволяет получить параметры прохождения для различных комбинаций значений атрибутов (понятие атрибутов и их классификация описаны в [см. лит.]). Некоторые из этих параметров прохождения представляют собой вероятности. Эти параметры прохождения, найденные подсистемой прогнозирования исхода выборов, могут быть непосредственно использованы для построения набора вершин и связей байесовской сети и получения вероятностей в ней.

В байесовской сети есть вершины, соответствующие атрибутам кандидатов в депутаты и параметрам прохождения. Атрибуты кандидатов в депутаты влияют на параметры прохождения, поэтому данные вершины могут быть соединены друг с другом как непосредственно, так и через другие (внутренние) вершины.

Байесовская сеть содержит 7 множеств, или групп, вершин: 2 множества вершин значений атрибутов, 2 множества вершин скрытых параметров, множество вершин параметров прохождения, множество вершин форс-мажорных обстоятельств и множество вершин принимаемых решений [см. лит.]. Множества вершин предназначены для того, чтобы упорядочить структуру байесовской сети. Существуют алгоритмы, по-разному обрабатывающие вершины в зависимости от того, к какой группе вершина принадлежит. Причинно-следственные связи могут существовать только между определенными вершинами, причем принадлежность вершин к группам определяет, может ли быть создана связь между этими вершинами.

Построение байесовской сети. Байесовская сеть всегда строится для конкретного участия кандидата в депутаты в конкретной избирательной кампании. Это не исключает возможности добавления вершин о других кандидатах в депутаты, но атрибуты берутся и параметры прохождения рассчитываются всегда для данного кандидата. Информация о том, какой это кандидат и какая кампания, сохраняется и загружается вместе с байесовской сетью. Операции, которые пользователь может производить с байесовской сетью, объединены в группы. Имеются 4 группы операций: заполнение байесовской сети по прогнозу исхода выборов, заполнение и коррекция байесовской сети в ручном режиме, вычисление с помощью байесовской сети и изменение байесовской сети на этапе оперативного управления.

Существует 2 режима работы с байесовской сетью: режим редактирования сети и режим исполнения сети. Режиму исполнения соответствует группа операций вычисления с помощью байесовской сети, а режиму редактирования – остальные группы операций.

Рассмотрим группы операций подробнее и конкретизируем каждую из них.

Заполнение байесовской сети в ручном режиме. Все операции по редактированию байесовской сети можно подразделить на операции, производимые в автоматическом или полуавтоматическом режиме, и операции в ручном режиме. В ручном режиме пользователь может создать новую сеть либо отредактировать сеть, полученную при выполнении других операций редактирования. Редактирование сети в ручном режиме состоит из таких элементарных операций, как создание новой вершины, модификация параметров вершины (например, названия и стоимости принятия решений), удаление имеющейся вершины, создание новой связи, допустимое при условии попадания вершин в нужные группы, удаление связи, ручное редактирование таблиц безусловной и условной вероятностей. При создании и удалении связей изменяется таблица условных вероятностей (ТУВ) вершин, в которые входят данные связи.

При создании и удалении связи ТУВ меняется таким образом, чтобы сохранить максимум информации о вероятностях, которые были до данного изменения. Поясним на примерах.

Пусть все вершины в сети бинарные (могут принимать два значения: 1 и 2) и были вершины A, B и C, причем вершина A – предок вершины C. Пусть вершина C имеет ТУВ, соответствующую табл. 1.

Таблица 1

A	1	2
$P(C = 1)$	0.4	0.7

Пусть добавилась связь от B к C. Изменение ТУВ показано в табл. 2.

Таблица 2

A	1	1	2	2
B	1	2	1	2
$P(C = 1)$	0.4	0.4	0.7	0.7

Другими словами, обе вероятности ($P(C|A) = 0.4$ и $P(C|\overline{A}) = 0.7$) сохранились. Изменим ТУВ так, как показано в табл. 3.

Таблица 3

A	1	1	2	2
B	1	2	1	2
$P(C = 1)$	0.3	0.6	0.5	0.8

Уберем связь между вершинами B и C. Изменения ТУВ показаны в табл. 4.

Таблица 4

A	1	2
$P(C = 1)$	0.45	0.65

Новые вероятности рассчитываются через старые: $0.45 = (0.3 + 0.6)/2$; $0.65 = (0.5 + 0.8)/2$. Другими словами, $P(C|A)$ и $P(C|\overline{A})$ принимают значения среднего арифметического двух вероятностей, соответствующих $P(C|A)$, или двух вероятностей, соответствующих $P(C|\overline{A})$.

Перейдем к рассмотрению операций по редактированию байесовской сети, которые проходят в автоматическом или полуавтоматическом режиме. Это значит, что за один раз создается не одна вершина или связь, а целая группа вершин и связей.

Заполнение байесовской сети по прогнозу исхода выборов. Заполнение байесовской сети по прогнозу исхода выборов объединяет все действия, которые производятся с использованием результатов работы подсистемы прогнозирования исхода выборов и ее отдельных модулей [см. лит.]. В версии 1.0 экспертной системы используются результаты обучения модуля прогнозирования № 1 (использующего аппарат математической статистики), а именно статистические данные о способствовании того или иного значения атрибута прохождению кандидата в депутаты. На каждом шаге генерируется одно правило. В версии 1.0 экспертной системы могут использоваться следующие правила:

если кандидат проходит, то атрибут *a* попадает в категорию *n*;

если кандидат набирает долю голосов больше или меньше *x*, то атрибут *a* попадает в категорию *n*;

если кандидат занимает место больше или меньше *x*, то атрибут *a* попадает в категорию *n*;

если атрибут *a* попадает в категорию *n*, то кандидат проходит;

если атрибут *a* попадает в категорию *n*, то кандидат набирает долю голосов больше или меньше *x*;

если атрибут *a* попадает в категорию *n*, то кандидат занимает место больше или меньше *x*.

По каждому правилу создаются две вершины: вершина-предок в группе параметров прохождения и вершина-потомок во второй группе значений атрибутов. Одна вершина принадлежит к группе значений атрибутов, другая – к группе параметров прохождения.

Числа в ТУВ равны значениям вероятностей в правилах. Например, предположим, что выбрано правило “Если атрибут ‘образование’ попадает в категорию 1, то кандидат проходит”, а категория 1 для образования – это высшее образование. Пусть вероятность прохождения кандидатов в депутаты, имеющих высшее образование (атрибут “Образова-

ние” = “Высшее”) составляет 20 %, а вероятность прохождения кандидатов в депутаты, не имеющих высшего образования, составляет 10 %. Тогда генерируется пара вершин: вершина-предок “Кандидат имеет высшее образование” и вершина-потомок “Кандидат прошел в депутаты” со следующей таблицей условных вероятностей (табл. 5).

Таблица 5

Сведения о кандидате	Образование = “Высшее”	Образование ≠ “Высшее”
Кандидат прошел	0.2	0.1
Кандидат не прошел	0.8	0.9

Оценки вероятностей в ТУВ вычисляются как отношения. Например, в рассматриваемом примере вероятности равны отношению количества прошедших или не прошедших кандидатов, имеющих или не имеющих заданное значение атрибута (в данном случае высшее образование), к общему количеству кандидатов в депутаты, имеющих или не имеющих высшего образования. Аналогичные отношения имеются и для пяти других правил.

Изменение байесовской сети на этапе оперативного управления. Неотъемлемой частью процесса оперативного управления является реагирование на появление новых факторов, существование которых изначально не известно. В связи с этим байесовская сеть доверия строится таким образом, чтобы можно было в любой момент времени добавить к сети новую гипотезу и присвоить ей некоторую вероятность, которую обозначим как *P*. При этом все параметры сети изменятся определенным образом в зависимости от *P*.

В версии 1.0 экспертной системы изменение байесовской сети на этапе оперативного управления включает построение вершин и связей на основе информации о примененных избирательных технологиях и форс-мажорных обстоятельствах. Рассмотрим использование информации об избирательных технологиях более подробно.

На этапе оперативного управления обязательно должно использоваться поступление новых свидетельств, которые соответствуют информации, например, о примененных избирательных технологиях. Вся информация об избирательных технологиях, в частности, названия используемых избирательных технологий, сведения о вершине, по отношению к которой рассчитывается эффективность технологии, и ее состояние, название новой вершины в байесовской сети доверия, названия и значения атрибутов, эффективность технологии, заносится в базу данных на этапе ее редактирования. Атрибуты в данном случае – это атрибуты кандидата в депутаты, избирательного округа или избирательной кампании [см. лит.], т. е. обстоятельства, при которых была применена технология и которые влияют на эффективность ее использования. Эффективность лежит в интервале от –1 до 1.

Для каждой избирательной технологии создается новая вершина и связь, соединяющая эту вершину с вершиной, по отношению к которой подсчитывается эффективность.

Рассмотрим пример. Пусть название новой вершины в базе данных – B и имеется только одна избирательная технология, эффективность которой приводится по отношению к первому состоянию бинарной вершины D и равна 0.6. Это означает, что эффективность технологии показывает, насколько применение технологии при данных значениях атрибутов способствует тому, чтобы событие, соответствующее вершине D байесовской сети, приняло первое значение. Тогда создается новая вершина B в группе принимаемых решений и связь, соединяющая вершины B и D. Эффективность подсчитывается как среднее арифметическое эффективностей применения технологий с теми же значениями атри-

бутов, что и у того кандидата, для которого строится байесовская сеть. Эта эффективность используется при модификации значений в ТУВ.

Пусть в рассматриваемом примере вершина D зависит только от вершины A и имеет ТУВ следующего вида (табл. 6).

Таблица 6

A	1	2
P(D = 1)	0.65	0.4

ТУВ после изменения байесовской сети на этапе оперативного управления показана в табл. 7.

Таблица 7

A	1	1	2	2
B	1	2	1	2
P(D = 1)	0.26	0.86	0.16	0.76
P(D = 2)	0.74	0.14	0.84	0.24

Первое состояние вершины B означает, что принято решение не использовать технологию, второе состояние вершины B означает, что принято решение использовать технологию. Эффективность равна 0.6, и разница между прежней условной вероятностью и нулем или единицей уменьшилась на 60 %. Поскольку эффективность рассчитывается по отношению к первому состоянию вершины D (это состояние вершины D является наилучшим), $P(D = 1 \mid B = 2) > P(D = 1)$ (эффективность положительна, и применение технологии, соответствующей вероятности вершины B, увеличило вероятность первого состояния вершины D), $P(D = 1 \mid B = 1) < P(D = 1)$ (неприменение технологии уменьшило вероятность первого состояния вершины D). Новые вероятности вычисляются через старые: 0.26 = 0.65 – 0.6 (0.65 – 0); 0.86 = 0.65 + 0.6 (1 – 0.65); 0.16 = 0.4 – 0.6 (0.4 – 0); 0.76 = 0.4 + + 0.6 (1 – 0.4).

Расчет байесовской сети. Все рассмотренные операции производятся в режиме редактирования байесовской сети. Теперь рассмотрим режим вычисления. В режиме вычисления пользователь задает критерий успешности участия кандидата в депутаты в избирательной кампании. В версии 1.0 экспертной системы этот критерий соответствует одной из вершин в группе параметров прохождения, и выбор критерия сводится к выбору вершины. Нужно максимизировать вероятность того, что выбранная вершина окажется в требуемом состоянии (например, вероятность того, что кандидат наберет более 30 % голосов). Пользователь вводит свидетельства (состояния вершин) и запускает алгоритм вычисления.

В режиме вычисления байесовская сеть перебирает наборы совместно принимаемых решений, отбрасывая те наборы, которые не удовлетворяют введенным ограничениям, и выдает на выходе ту комбинацию решений, которая максимизирует вероятность оптимального состояния выбранной вершины (параметра прохождения). В версии 1.0 экспертной системы рассматривается один тип ограничений – ограничения по бюджету. Каждому решению соответствует цена – стоимость исполнения этого решения. Также имеется общий бюджет кампании, и стоимость исполнения принятых решений не должна превышать общий бюджет кампании.

Экспериментальное испытание подсистемы управления ведением предвыборной кампании. Подсистема управления была опробована на выборах в местное самоуправление 2004 г. Система рекомендовала разумные решения, исполнение которых позволило

участвующим в эксперименте кандидатам в депутаты войти в пятерку лучших кандидатов в депутаты, при том что именно 5 лучших кандидатов проходили в депутаты по данному округу. Это означает, что испытание системы прошло успешно.

СПИСОК ЛИТЕРАТУРЫ

Биниенко О. А. Система прогнозирования исхода выборов депутатов и управления процессом предвыборной кампании // Изв. СПбГЭТУ “ЛЭТИ”. Сер. “Информатика, управление и компьютерные технологии”. 2004. Вып. 2. С. 63–67.

O. A. Binienko

THE ELECTIONEERING MANAGEMENT AND ITS REALIZATION IN THE EXPERT SYSTEM

The electioneering management process for candidates in deputies in Russia is described. For The expert system contains the subsystem of electioneering management for solving the management task. The modules of management in this subsystem are described.

Deputies election, electioneering management, using of Bayesian network, distribution of finances

УДК 37.014.1

А. В. Горячев, Н. Е. Новакова

ПРАГМАТИЧЕСКИ ПЕРЕСТРАИВАЕМЫЕ АРХИТЕКТУРЫ КОМПЬЮТЕРНЫХ СРЕД МНОГОПРОФИЛЬНОЙ ПОДДЕРЖКИ УЧЕБНОГО ПРОЦЕССА

Рассматривается процесс установки программного обеспечения и операционных систем на компьютеры в учебных классах. Анализируются различные технологии, позволяющие работать с приложениями в учебном процессе.

Перестраиваемые архитектуры, компьютерный класс, учебный процесс, образ операционной системы, удаленная установка приложений, виртуальные машины, терминальный сервис

Использование компьютеров в условиях вузов, особенно готовящих студентов по информационным технологиям, характеризуется очень большим спектром прикладного и системного программного обеспечения (ПО), изучаемого и используемого в ходе учебного процесса. При этом в течение семестра разные группы работают с самым разным ПО, а учебные классы, в которых проводится работа с этим ПО, очень часто не могут быть специализированы ввиду их малочисленности и децентрализованности управления ими.

Обычно работа с ПО производится в учебных классах институтского, факультетского или кафедрального подчинения, но проблемы по установке и дальнейшей поддержке необходимого ПО, как правило, одинаковы. В самом общем виде их можно охарактеризовать так:

– несовместимость ПО между собой и операционными системами;

– ограниченные аппаратные ресурсы;

– принципиальная невозможность установки ПО с высокими требованиями к ресурсам;

– большое количество приложений, усложняющее управление компьютером и увеличивающее время восстановления ПО в случае аварийных ситуаций.

Все это ставит задачу организации оперативной установки ПО и управления им с минимальными затратами времени и организационных ресурсов.

Возможны следующие пути решения обозначенных проблем:

– централизация управления компьютерными классами и их использования;

– применение технологий дистанционной установки приложений на локальные компьютеры и контроля за ними;

– применение технологий дистанционной установки на локальные компьютеры операционных систем с предустановленным программным обеспечением;

– применение технологий дистанционной установки на локальные компьютеры операционных систем с последующей дистанционной установкой ПО;

– использование технологии виртуальных машин;

– применение терминальных служб.

Централизованное управление компьютерными классами возможно только при очень высокой централизации управления всем учебным процессом и эффективно только тогда, когда одни и те же курсы читаются на многих кафедрах, что дает возможность эффективно загрузить класс, настроенный под совершенно конкретную задачу. Такая ситуация встречается очень редко.

Технология ***дистанционной установки приложений*** имеет следующие особенности:

1. Эффективна только при одной операционной системе на компьютере.

2. Требует отслеживания «взаимозависимости» приложений.

3. Позволяет организовать различную настройку одного приложения для разных пользователей, однако требует для этого немало усилий.

4. Как правило, не критична к конфигурации аппаратных средств, так как это «проблемы» ОС (кроме требований приложения, например – объем ОЗУ).

5. Требует обязательного ограничения возможностей пользователей по отношению к операционной системе, конфигурация которой не должна изменяться.

6. Требует обязательной установки в среде ОС программных агентов, обеспечивающих доставку приложений.

7. При высокой динамике (необходимости смены в течение дня нескольких комплектов ПО) требует наличия эффективной системы оперативного удаления приложений с рабочих станций и четкой политики по определению события, к которому «привязывается» установка приложений.

8. Не использует режим одновременной передачи информации (multicast), что приводит к резкому возрастанию сетевого трафика в случае одновременной доставки одного и того же приложения на большое количество рабочих станций.

Дистанционная установка операционных систем с предустановленным программным обеспечением характеризуется следующими особенностями:

1. Позволяет создавать сконфигурированные «образы» рабочего места, настроенного на решение конкретного набора задач конкретными способами.

2. Для каждого образа может использоваться наиболее подходящая ОС.

3. Требует организации централизованного хранения данных на серверах. Это устраняет необходимость управления дисковым пространством рабочих станций.

4. Может использовать режим одновременной передачи информации (multicast), что сокращает сетевой трафик за счет использования только одного потока информации в сети.

5. Наиболее эффективно работает с установленными в среде ОС агентами, что позволяет постоянно контролировать состояние ОС и выполнять задачи по «снятию» или «заливке» образов дистанционно в любой момент времени.

При использовании дистанционной установки операционных систем появляются следующие проблемы:

• Большой объем каждого образа и большое количество образов приводят к необходимости выделения существенного объема дискового пространства на сервере для хранения образов и значительному времени доставки образов.

• Невозможно создавать «множественные конфигурации» отдельных приложений (одно и то же приложение, но с разными настройками).

• Требуется максимально идентичная конфигурация аппаратных средств.

Раздельная установка на компьютеры пользователей операционных систем и последующая дистанционная установка отдельным процессом программного обеспечения сочетает в себе достоинства и недостатки обоих методов.

В случае, когда даже на одном занятии требуются различные конфигурации ОС и ПО, причем конфигурации различаются по пользователям, а не по компьютерам, в выигрышном положении оказываются программные системы, использующие для хранения своей информации и информации о настройках рабочих мест каталоги сетевых ресурсов типа Novell eDirectory или Microsoft Active Directory [1]. Это позволяет интегрировать систему установки и настройки конфигурации рабочей станции и приложений с единым каталогом сетевых ресурсов, в которой пользователь выполняет аутентификацию, однозначно ассоциируя себя с учетной записью каталога.

Технология виртуальных машин (ВМ) предоставляет возможность запуска операционной системы в среде приложения базовой («host») ОС [2]–[4]. Например, операционная система базовой машины – Windows XP, в ее среде запускается ПО виртуальных машин, а уже оно, в свою очередь, предоставляет возможность запускать гостевые («guest») ОС, в качестве которых может использоваться Linux, Novell NetWare или другая ОС Microsoft.

Особенности технологии:

– дает возможность ***одновременной*** работы на одном компьютере с несколькими (как правило, 2–3) различными операционными средами;

– моделирует сетевое взаимодействие, что позволяет на компьютере без сетевых карт «проиграть» взаимодействие по сети двух виртуальных компьютеров;

– сохраняет свою информацию в виде небольшого количества файлов Host-операционной системы. Это позволяет легко копировать эти файлы с сервера на рабочие станции. Таким образом, легко выполнить клонирование виртуальной машины – файлы базовой ВМ с сервера копируются на рабочую станцию с установленным ПО виртуальных машин, а затем в настройках ПО виртуальных машин создается новая ВМ на основе скопированных файлов;

– позволяет легко уничтожить «следы всех трудов» – удаляются все файлы из каталога виртуальной машины и на их место записываются с сервера новые. Это дает возможность никак не ограничивать прав пользователя на ресурсы локальной ОС;

– позволяет гибко взаимодействовать с Host-операционной системой, или используя его сетевой адаптер и эмулируя его копию (при этом виртуальный компьютер при взаимодействии по сети неотличим от реального), или организовав на основе базового компьютера точку трансляции сетевых адресов, получая возможность взаимодействия с внешней сетью, будучи невидимой для остальных узлов сети;

– дает возможность запоминания состояния виртуальной машины в нужный момент времени и отката к этому состоянию в случае возникновения проблем;

– позволяет гибко работать с аппаратными ресурсами как с реальными (CDROM, Floppy, Sound и т. д.), так и эмулировать их (можно эмулировать CDROM с помощью ISO образа или создавать любое необходимое количество «жестких дисков»), эмулируя конфигурации, порой трудно создаваемые физически. Например, в среде виртуальных машин на одной физической машине без дополнительных физических ресурсов можно создать конфигурацию, достаточную для установки кластера надежности.

Наиболее заметный недостаток – требовательность к ресурсам, среди которых особо критическими являются размер оперативной памяти и быстродействие процессора. Недостаток этих ресурсов выражается в заметно сниженной производительности. Для применения механизма отката требуется дополнительное место на жестком диске для сохранения текущего состояния.

Применение терминальных служб. История этого механизма восходит к эпохе мэйнфреймов.

Основная задача терминального сервиса – работа с уже установленными на сервере приложениями в условиях четких ограничений по возможностям. При этом используются ресурсы именно сервера, а в качестве одного из видов клиента может использоваться Web-обозреватель, что находит в последнее время все более широкое применение.

Преимущества:

• низкие требования к ресурсам рабочей станции: развернуть клиента можно на любой ОС из линейки Microsoft (даже для DOS), а для работы Web-клиента потребуется всего лишь «не самый старый» Web-обозреватель;

• отсутствие влияния на конфигурацию рабочей станции – для работы потребуется только установка небольшого клиента, а в некоторых случаях (Windows XP) он автоматически устанавливается при установке ОС;

• абсолютная «непривязанность» к конкретной рабочей станции – сеанс, начатый на одной рабочей станции, может быть закончен на другой;

• минимальное время подготовки к работе – сервис доступен сразу после загрузки ОС рабочей станции;

• возможность запускать приложения, которые требуют ресурсов, отсутствующих на рабочей станции (малый объем оперативной памяти);

• возможность работы с несколькими сеансами сразу;

• единая аутентификация требует жестко определять режимы доступа.

Наиболее эффективным, безусловно, может стать сочетание нескольких методов, так как некоторые из них совместимы с другими, а ряд из них имеют даже интегральные реализации, например технология виртуальных серверов, в которой на базовом сервере запускается некоторое количество виртуальных машин, а доступ к ним со стороны клиентов осуществляется только по технологии терминального доступа.

В связи с этим первой задачей при выборе схемы управления ПО в учебном классе будет классификация приложений по возможной среде их запуска. Нужно сразу определить, какие приложения потребуют локальной установки, какие могут быть установлены

на виртуальных машинах, а к каким возможно организовать доступ через терминальный сервис. После этого нужно решить как разворачивать операционные системы и устанавливать на них локальные приложения.

Очень важно, что большинство решений базируется на современных сетевых технологиях [4]. Это значит, что в современном учебном процессе без них не обойтись.

СПИСОК ЛИТЕРАТУРЫ

1. Вишневский А. В. Windows Server 2003. Для профессионалов / Центр «Традиция». 2004.

2. Brian Wared. The Book of VMware: The Complete Guide to VMware Workstation. No Starch Press; 1 edition, 2002.

3. Anthony T. Mann. The Rational Guide To: Microsoft Virtual PC 2004. Rational Press. 2004.

4. BrianMadden. Citrix MetaFrame XP: Advanced Technical Design Guide, Second Edition. BrianMadden.com. 2003.

5. Bill Holtsnider, Brian D. Jaffe. IT Manager's Handbook: Getting Your New Job Done. Morgan Kaufmann; 1st edition. 2000.

N. E. Novakova, A. V. Goryachev

PRAGMATICALLY RECONFIGURABLE ARCHITECTURES OF MULTIPLE-DISCIPLINE EDUCATION SUPPORT OF COMPUTER ENVIRONMENTS PROCESS

Process of program and operating system installation in the computer classes. Analyze differ technologies to exploite application education process

Reconfigurable Architecture, Computer Class, Education Process, Operating System Image, Remote Installation of Application, Virtual Machines, Terminal Service

УДК 519.87

Д. М. Выборнов

РАСЧЕТ СТАЦИОНАРНОГО РЕЖИМА НЕЛИНЕЙНЫХ СХЕМ

Рассматривается методика машинного расчета нулевого уровня электронных схем, основанная на решении описывающих схему нелинейных уравнений. Эффективность данной методики достигается благодаря направленному использованию специфики решаемой задачи и адаптации процесса, последовательных приближений к характеру нелинейных характеристик компонентов.

Стационарный режим, линеаризация, нелинейные компоненты, итерационный метод Ньютона, компактная обработка разреженных матриц, символьный и численный этапы

Линеаризация характеристик нелинейных компонентов. Машинный расчет нулевого уровня электронных схем сводится к численному решению описывающих схему нелинейных уравнений. Если для простейших электронных схем отыскание корней этих уравнений с помощью распространенных итерационных методов не вызывает затруднений, то расчет нулевого уровня многокаскадных схем с обратными связями представляет собой весьма трудоемкую задачу. Сложность структуры современных электронных схем заставляет искать такие подходы к расчету нулевого уровня, которые отличались бы высокой скоростью сходимости и некритичностью к выбору начальных условий.

Пусть имеется электронная схема произвольной структуры, многополюсные компоненты которой (транзисторы, диоды, интегральные подсхемы и т. д.) заданы [1]:

• линейные – произвольно, в том числе и смешанной системой параметров;

• нелинейные – функциональными зависимостями вида

$$P_i = f(Q_1, Q_2, ..., Q_j, ..., Q_n), \quad (1)$$

где $P_i(Q_i)$ – второстепенные (основные) переменные, которые в зависимости от выбранного способа описания многополюсника определяют напряжение или ток.

Если выбрано некоторое исходное состояние многополюсника, характеризуемое совокупностью основных переменных

$$Q_i^{(k)} = Q_1^{(k)}, Q_2^{(k)}, ..., Q_n^{(k)},$$

то можно разложить нелинейные функции вида (1) в ряд Тейлора в окрестностях точки $Q_i^{(k)}, k = 0, 1, 2, ...$ Отбрасывая члены высшего порядка малости, получим систему линеаризованных уравнений, *i*-е из которых выглядит так:

$$P_i = P_i^{(k)} + \sum_{(j)} \frac{\partial P_i}{\partial Q_j}\Big|_{Q_i^{(k)}} (Q_j - Q_j^{(k)}),$$

где $P_i^{(k)} = f(Q_1^{(k)}, Q_2^{(k)}, ..., Q_n^{(k)})$. Введя следующие обозначения линеаризованных параметров многополюсника [1]:

$$S_{wi}^{(k)} = P_i^{(k)} + \sum_{(j)} \frac{\partial P_i}{\partial Q_j}\Big|_{Q_i^{(k)}} Q_j^{(k)},$$

$$w_{ij}^{(k)} = \frac{\partial P_i}{\partial Q_j},$$

получим уравнение автономного многополюсника:

$$P_i = \sum_{(j)} w_{ij}^{(k)} Q_j + S_{wi}^{(k)} (i, j = 1, 2, ..., n). \quad (2)$$

Здесь $w_{ij}^{(k)}$ и $S_{wi}^{(k)}$ – соответственно *неавтономный* и *автономный параметры* многополюсника, определяемые выбранными координатами рабочей точки.

Итак, линеаризация исходных уравнений (1) приводит электронный прибор к *автономному многополюснику* [2], который, в свою очередь, можно представить в виде соединения неавтономного многополюсника и задающих автономных источников. Следовательно, задача математического описания схемы с нелинейными компонентами сводится к задаче описания схемы с автономными многополюсниками и может быть решена на основании методики прямого образования матриц анализируемой схемы. Применение указанной методики позволяет получить обобщенное уравнение анализируемой схемы в смешанном координатном базисе [1]:

$$\Psi(Q) = M^{(k)} Q + S_w^{(k)} = 0, \quad (3)$$

где Q – m-мерный вектор-столбец базисных переменных; $M^{(k)}$ – квадратная матрица коэффициентов m-го порядка; $S_w^{(k)}$ – m-мерный вектор-столбец автономных параметров и задающих источников, характеризующих линеаризованную электронную схему.

При расчетах на мини-ЭВМ целесообразно использовать метод узловых напряжений, позволяющий записать уравнения (2) и (3) в виде:

$$I_i = \sum_{(j)} w_{ij}^{(k)} U_j + J_{yi}^{(k)}, (i, j = 1, 2, ..., n),$$

$$\psi(X) = W^{(k)} X + J_y^{(k)} = 0.$$

Здесь $W^{(k)}$ – матрица узловых проводимостей схемы; X – вектор узловых напряжений; $J_y^{(k)}$ – задающий вектор, элементами которого являются автономные параметры $J_{yi}^{(k)}$ и токи J задающих источников; $w_{ij}^{(k)}$ – неавтономный параметр многополюсного компонента.

Итерационная процедура. Метод Ньютона предусматривает построение итерационного процесса на основе вектора начального приближения $Q^{(k)}$ в соответствии с рекуррентным соотношением [1]

$$Q^{(k+1)} = Q^{(k)} - J^{-1}(Q^{(k)})\Psi(Q^{(k)}), \tag{4}$$

где $J(Q^{(k)})$ – матрица Якоби, вычисленная при $Q^{(k)}$; $\Psi(Q^{(k)})$ – вектор невязок по базисным переменным $Q^{(k)}$.

При практической реализации метода Ньютона соотношение (4) преобразуется к виду:

$$J(Q^{(k)})\Delta Q^{(k)} = -\psi(Q^{(k)}), \tag{5}$$

$$Q^{(k+1)} = Q^{(k)} + \Delta Q^{(k)}, \tag{6}$$

где $\Delta Q^{(k)}$ – вектор корректирующих поправок.

На каждом k-м шаге процесса приближений эта система линейных уравнений решается одним из методов алгебры относительно $\Delta Q^{(k)}$. Основные препятствия, которые приходится преодолевать при машинной реализации схемы итераций (5) и (6), заключаются в следующем [1]:

1) вектор начального приближения $Q^{(k)}$ должен быть достаточно близок к значению корня, в противном случае сходимость процесса не гарантируется;

2) отсутствие контроля и управления сходимостью повышает требования к выбору $Q^{(k)}$ и сужает область притяжения корня.

Соотношение (4) допускает следующее схемотехническое истолкование. Вектор невязок показывает, как удовлетворяются записанные в виде

$$\Psi(Q^{(k)}) = M^{(k)} Q^{(k)} + S_w^{(k)} \tag{7}$$

обобщенные законы Кирхгофа для сходящейся последовательности итераций $Q^{(0)}, Q^{(1)}, Q^{(2)}, ..., Q^{(k)}$. Смешанная матрица $M^{(k)}$ исходного уравнения является по существу якобианом системы функции $\Psi(Q^{(k)})$ в точке $Q = Q^{(k)}$:

$$J(Q^{(k)}) = \Psi'(Q)\Big|_{Q^{(k)}} = M^{(k)}. \tag{8}$$

После подстановки соотношений (7) и (8) в уравнение (4) получаем компактное и удобное при машинном анализе выражение

$$Q^{(k+1)} = -J^{-1}(Q^{(k)})S_w^{(k)}, \tag{9}$$

которое в базисе узловых напряжений примет вид

$$X^{(k+1)} = -W^{-1}(X^{(k)})I_Y^{(k)}.$$

Взаимосвязь итерационной формулы (9) и исходного матричного уравнения (3) свидетельствует о том, что на каждом k-м шаге процесса Ньютона вычислению очередного приближения $Q^{(k+1)}$ можно сопоставить расчет электронной цени, структура схемных связей которой остается неизменной, в то время как параметры нелинейных автономных многополюсников $w_{ij}^{(k)}$ и $S_{wi}^{(k)}$ (и, следовательно, элементы $M^{(k)}$, $S_w^{(k)}$) подлежат обновлению в соответствии с текущим состоянием вектора $Q^{(k)}$.

Исследуемая электронная цепь представляется, таким образом, последовательностью линейных эквивалентов, каждый из которых можно считать адекватным исходной нелинейной цепи на k-м шаге процесса приближений. Отметим, что организация расчетов по итерационной формуле (9) имеет преимущество перед классической схемой (4): она избавляет от необходимости определять для каждого приближения элементы невязок $\Psi(Q)$ и, главное, предоставляет благоприятные возможности для управления сходимостью процесса вычислений.

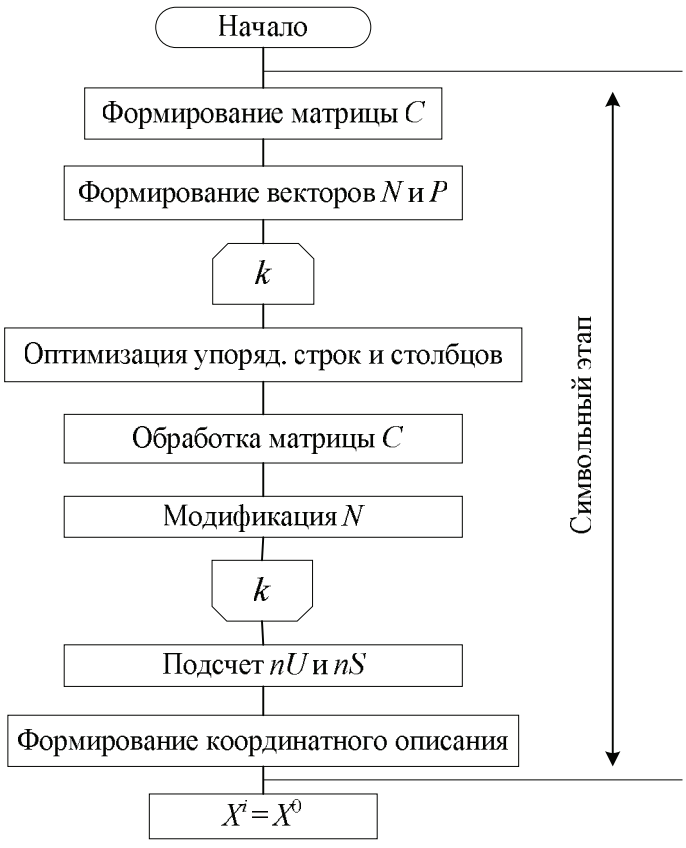


Рис. 1

При расчете данного режима возможно применение метода компактной обработки разреженных матриц. Это достаточно быстрый и эффективный метод, при правильном подходе к которому можно ускорить расчет схемы за счет сокращения количества шагов в алгоритме. Данный метод состоит из двух этапов – символьного и численного [3].

На символьном этапе (рис. 1) формируется *индексно-структурная матрица* [3], с помощью которой можно удалить из всех компактных схем описания массивы, в которых записаны координаты ненулевых элементов, а также осуществить предварительную обработку схемы с целью построения ее портрета.

На численном этапе (рис. 2) осуществляется непосредственный расчет схемы.

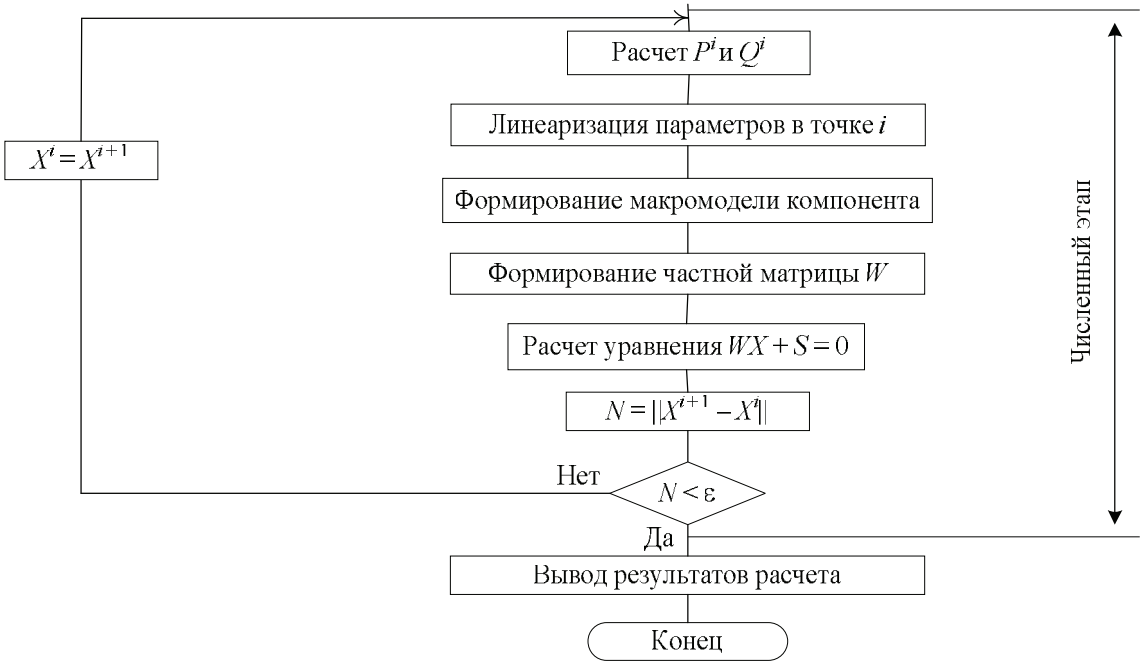


Рис. 2

Возможность получения априорных оценок работоспособности разрабатываемых устройств существенна прежде всего для схем в микроэлектронном исполнении, поскольку исследование последних на ЭВМ является экономически единственно выгодным способом их проектирования – интегральная технология исключает традиционный этап физического макетирования.

СПИСОК ЛИТЕРАТУРЫ

1. Анисимов В. И., Дмитревич Г. Д., Ежов С. Н. Автоматизация схемотехнического проектирования на мини-ЭВМ. Л.: Изд-во Ленингр. ун-та, 1983.

2. Влах И., Сингхал К. Машинные методы анализа и проектирования электронных схем. М.: Радио и связь, 1988.

3. Писсанецки С. Технология разреженных матриц. М.: Мир, 1988.

Vybornov Dmitry

CALCULATION OF A STATIONARY MODE OF NONLINEAR CIRCUITS

Machine calculation of a zero level of electronic circuits is reduced to the numerical decision of the nonlinear equations describing the circuit. Efficiency of a design procedure of a zero level considered further is reached due to directed use of specificity of a solved problem and adaptation of process, consecutive approximations to character of nonlinear characteristics of components.

Stationary mode, linearization, nonlinear components, an iterative method of Newton, compact processing of rarefied matrixes, symbolical and numerical stages

УДК 004.657

В. А. Кудинов

ОБ ОДНОМ ПОДХОДЕ К АВТОМАТИЗИРОВАННОМУ ПРОЕКТИРОВАНИЮ БАЗ ДАННЫХ

Рассматривается подход к проектированию баз данных, при котором задача небольшой сложности может быть решена пользователем (заказчиком) будущей системы.

Базы данных, проектирование баз данных, разработка приложений для работы с базами данных

Неотъемлемой частью многих задач, решаемых при помощи компьютерных технологий, являются базы данных (БД). Проектированием БД обычно занимается группа разработчиков комплекса программного обеспечения, решающего поставленную задачу. Для успешной реализации проекта непосредственному процессу проектирования предшествует анализ предметной области, который должен выполняться совместно с заказчиком. После создания самой БД требуется разработка оболочки, обеспечивающей доступ к данным. В целом процесс разработки оказывается достаточно трудоемким и дорогостоящим. С развитием рынка программного обеспечения и значительным увеличением спроса на системы, использующие БД, предпринимаются различные попытки снизить затраты на разработку БД и оболочки, поскольку сам процесс проектирования и построения различных систем во многом повторяется, а значит, может быть частично автоматизирован.

Постановка задачи. Рассматривается один из подходов к автоматизированному проектированию баз данных, позволяющий выполнять процесс проектирования пользователям будущей системы без привлечения специалистов в области баз данных и программистов. Для этого необходимо разработать программный продукт – систему проектирования баз данных, отвечающую следующим требованиям:

1. Система позволяет построить некоторую модель, соответствующую поставленной задаче, причем описанную в терминах данной предметной области.

2. Система должна обладать наглядным средством (средствами) визуализации проектируемой модели, позволяющим получить подробную информацию о каждом элементе модели и связях между ними.

3. При работе с первичной моделью система должна осуществлять контроль на соответствие модели основным принципам построения реляционных БД: минимизация избыточности данных, обеспечение нормализации, предотвращение наиболее распространенных логических ошибок со стороны проектировщика.

4. Система должна транслировать первичную модель в логическую модель данных, а затем и в физическую базу данных.

5. Система должна быть совместима с различными СУБД, позволяя пользователю выбирать ту или иную СУБД для трансляции логической модели данных в физическую БД.

6. Система должна предоставлять возможность редактирования первичной модели, логической модели данных и физической БД вне зависимости от стадии реализации задачи.

7. Система должна предоставлять пользователю доступ к физической базе данных в виде некоторой программной оболочки.

В зависимости от реализации поставленной задачи к системе проектирования могут быть предъявлены дополнительные требования. Например, по возможности использовать отдельные функции той или иной СУБД, не являющиеся стандартными, однако повышающие эффективность использования выбранной СУБД.

Решение поставленной задачи. Решение задачи по созданию системы проектирования БД следует начать с разработки инструмента проектирования и визуализации проектируемой модели.

Первичная модель, которую будет строить пользователь, хотя и является прототипом будущей модели данных, не является четкой логической моделью данных. Для перехода к последней служит программное обеспечение. Для пользователя создаваемая им модель должна быть проста для понимания и максимально приближена к понятиям его предметной области, оставаясь в то же время универсальной для многих предметных областей. Такое решение может обеспечить ввод понятий *классов* и *атрибутов классов*. Пользователь, описывая в среде проектирования новый класс, фактически будет подразумевать под ним класс из своей предметной области. Перечисление атрибутов класса и их типов будет описывать все существенные для данной задачи характеристики этого класса. Также потребуется и описание связей между классами. На основе информации о классах, их атрибутах и связях будет осуществлен переход от начальной модели к логической модели данных. Процедуру перехода от логической к физической модели данных можно формализовать и выполнять в автоматическом режиме.

Поскольку контроль модели на соответствие реляционной структуре выполняется программным обеспечением, это приводит ко многим ограничениям в отношении структуры проектируемой системы. Среди основных:

• ограничение по глубине связей между элементами системы;

• ограниченный набор видов связей между элементами системы;

• жесткие отношения «один-к-одному», «один-ко-многим» и «многие-ко-многим» между элементами системы;

• невозможность использовать некоторые типы данных (некоторые модификации стандартных типов);

• невозможность использовать некоторые нестандартные приемы при построении БД, присущие отдельным СУБД.

Стоит отметить, что система принципиально не может предотвратить некоторые ошибки, связанные с нормализацией данных, поскольку часто по описанию объекта системы его невозможно проверить на соответствие 2-й или 3-й нормальной форме. В этом случае структура будет не самой рациональной из возможных, что, однако, не повредит работе будущей системы в целом.

Остановимся на способе визуализации проектируемой структуры системы. В профессиональных средах проектирования баз данных для визуализации логической модели данных широко применяются так называемые ER-диаграммы. Суть отображения структуры БД в виде ER-диаграммы сводится к следующему: как правило, каждая сущность (объект системы) отображается на диаграмме в виде прямоугольника (поля), разбитого на строки. В первой строке (заголовке) стоит название объекта, а в строках ниже перечисляются атрибуты данного

объекта. Пользователь может располагать объекты на диаграмме в удобном для него порядке. Часто связи между объектами можно задать простым перетаскиванием атрибута одного объекта на поле другого, при этом указав тип отношения между связываемыми объектами. Редактирование самого объекта и его атрибутов также должно быть доступно при обращении к соответствующему полю на диаграмме. В данном случае ER-диаграмма также может быть применима для описания системы классов, их атрибутов и связей.

Альтернативным способом визуализации является расположение объектов системы и их атрибутов в виде иерархического дерева. На первом уровне такой структуры располагаются объекты системы (в данном случае – классы), на втором – их атрибуты. Связи между классами также логично представить в виде атрибутов специального типа. Третьим уровнем дерева может служить расширенное описание каждого атрибута, включающее как его основные свойства, которые относятся к логической модели данных, так и дополнительные (способ отображения при визуализации структуры, дополнительная обработка данных при вводе и извлечении из БД и др.). Такой способ визуализации является более компактным по сравнению с ER-диаграммой, но не обладает такой же наглядностью в отношении связей между элементами системы.

Далее необходимо рассмотреть вопрос о трансляции описываемой структуры в физическую структуру базы данных. В случае работы с сетевыми СУБД (MS SQL Server, Oracle, Interbase, Firefox и др.) наилучшим вариантом будет создание не только самой базы данных, но и так называемого *скрипта*: последовательности команд на языке SQL, результатом выполнения которых будет готовая база данных. При этом следует учесть особенности различных диалектов языка SQL для конкретных СУБД. В случае же локальных баз данных (Paradox, dBase) создание скрипта теряет смысл, поскольку в этих типах БД отсутствует собственный компилятор языка SQL, а доступ к данным осуществляется через стандартный для ОС Windows механизм ODBC либо посредством решений сторонних производителей (например, Borland Database Engine). Стоит отметить особое удобство применения описываемого подхода к разработке локальных баз данных, поскольку редактирование уже созданной физической структуры БД будет сводиться к редактированию логической модели системы.

В случае же сетевых баз данных вносить изменения в структуру БД можно как при помощи начального средства проектирования, так и при ручном редактировании ранее сгенерированного скрипта. Не исключен и вариант загрузки скрипта в стороннее средство для работы с той или иной СУБД. Возможность самостоятельного построения модели данных пользователем не исключает последующие этапы согласования и корректировки этой модели со специалистами. Здесь важно отметить, что начального этапа проектирования структуры системы и настройки некоторых ее параметров уже достаточно для начала работы с системой. Необходимо только выполнить трансляцию модели в СУБД назначения (это программное обеспечение сделает в автоматическом режиме), а способ доступа к данным будет рассмотрен далее.

Помимо проектирования структуры необходимо решить задачу создания оболочки для доступа к БД. Под оболочкой подразумевается программа (или набор программ), обеспечивающая необходимый доступ к данным (ввод, просмотр и редактирование данных). Часто при разработке программных продуктов с использованием БД такую оболочку приходится создавать с нуля, несмотря на то что в наиболее популярных средах проектирования присут-

ствуют специальные компоненты и решения именно для разработки такого рода программ. Создание такой оболочки в Microsoft Access будет менее трудоемким процессом по сравнению с профессиональными средами проектирования (MS Visual C++, Borland Delphi, Borland C++ Builder). Создание формы и элементов интерфейса для ввода, редактирования и просмотра данных ускоряется, во-первых, за счет четкой визуализации этого процесса, а во-вторых, за счет так называемых мастеров, которые генерируют готовые формы и элементы интерфейса для наиболее распространенных ситуаций. Известная среда проектирования БД ERwin имеет встроенные средства для генерирования кода программ, обеспечивающих доступ к данным. Но и в данном случае имеются в виду наиболее стандартные ситуации.

В рассматриваемом случае нужно учесть неподготовленность пользователя к самостоятельной разработке программы доступа к данным, поэтому предлагается разработать универсальную оболочку, динамически перестраиваемую относительно логической модели данных. С одной стороны, такое решение может оказаться не самым удобным, особенно если необходимо выделять некоторые элементы интерфейса на фоне остальных либо при вводе данных осуществлять дополнительную проверку и обработку. С другой стороны, при помощи дополнительных параметров, задаваемых для каждого атрибута каждого класса в первичной проектируемой модели системы, можно будет настраивать форму для работы с данными так, чтобы сделать ввод и просмотр информации максимально удобным. В итоге перед пользователем такой системы вообще не будет стоять задача разработки оболочки для работы с данными. Ему потребуется лишь настроить формы таким образом, чтобы это соответствовало поставленной задаче. Аналогично случаю разработки модели системы в данном случае удается избежать привлечения специалистов для разработки оболочки доступа к данным, предлагая пользователю лишь настройку параметров уже готовой оболочки.

При описанном подходе к проектированию БД и разработке программной оболочки возникнут трудности с переходом от логической модели данных к физической БД, поскольку потребуется обеспечить доступ к данным независимо от того, в какую СУБД транслировалась модель системы. Другими словами, необходимо предусмотреть совместимость оболочки со всеми поддерживаемыми СУБД. Самым простым и логичным решением этой задачи может быть использование языка SQL (стандартизированный язык запросов к базе данных). Любая СУБД на сегодняшний день поддерживает стандартные команды SQL, дополняя их набором собственных специализированных команд. Объединенный набор команд часто называется диалектом языка SQL для той или иной СУБД. Также стоит отметить, что каждая СУБД имеет свои типы данных, часто несовместимые между разными диалектами. Иногда стандартные типы данных, оговоренные для базового языка SQL, некоторыми СУБД трактуются по-разному. Все эти особенности должны быть учтены в разрабатываемой системе проектирования. Как вариант можно не использовать некоторые специализированные типы данных, ограничившись стандартными: целочисленными, числовыми с плавающей точкой и символьными строками. С другой стороны, можно пойти по более сложному пути, который позволит использовать некоторые преимущества отдельных СУБД: настроить оболочку для работы с данными под конкретные СУБД с их типами данных. Выбор же СУБД будет осуществляться перед построением логической модели, так чтобы заранее предоставить пользователю доступ к расширенным возможностям выбранной СУБД, в том

числе ее типам данных. При трансляции в физическую структуру и перенастройке оболочки доступа к данным все пользовательские настройки будут учтены, и в итоге можно будет более эффективно использовать ту или иную СУБД по сравнению с однообразным подходом к работе со всеми СУБД одновременно.

Заключение. При выборе подхода к проектированию системы с использованием базы данных пользователь (заказчик) должен определить сложность разрабатываемой системы и, исходя из этого, выбрать средство проектирования. Для большой и сложной системы, над которой работает группа разработчиков и которая впоследствии требует непрерывной поддержки, идеальным средством проектирования является среда ERwin, включающая в себя инструмент проектирования структуры данных на основе ER-диаграммы, встроенные средства разработки оболочки доступа к данным и многое др. К сложным задачам следует отнести сетевые задачи с большим числом клиентских подключений, задачи, в которых требуется высокая производительность системы (системы банковского учета, бронирования билетов, управления крупным производством и др.). Для решения более простых задач (локальные БД, сетевые БД с небольшим числом клиентских подключений) можно воспользоваться описанной системой проектирования. Такой подход позволит решать стоящие перед пользователем задачи без привлечения специалистов как в области баз данных, так и в области программирования, поскольку среда проектирования автоматически сгенерирует физическую базу данных на основе созданной пользователем модели, а универсальная оболочка обеспечит доступ к данным.

V. A. Kudinov

ABOUT ONE APPROACH TO THE DATABASE AUTOMATED DESIGN

Consideration of the approach to computer-aided engineering of databases, by means of which not very complicated task can be accomplished by the client of the workable software.

Databases, Database engineering, Database-based application design

УДК 625.571

Д. И. Батищев, С. Е. Власов, В. В. Балашов

ИССЛЕДОВАНИЕ МЕТОДОВ АВТОМАТИЗИРОВАННОЙ ТРАССИРОВКИ ОДНОСЛОЙНЫХ СТРУКТУР

Рассматриваются пути решения проблемы автоматизированной трассировки специализированных микроэлектронных устройств, реализуемых по технологии с одним слоем коммутации. Выбрано направление для разработки алгоритма трассировки подобных структур.

Автоматизированная трассировка, трассировка однослойных структур

Проблема трассировки однослойных структур. Несмотря на развитие многослойных устройств, для изделий специального назначения (СВЧ, аналого-цифровые устройства) актуальным остается вопрос их реализации одним слоем коммутации. Одной из серьезных проблем является автоматизированное проектирование топологии этих изделий. Это связано с тем, что современные зарубежные САПР БИС (в частности, САПР Cadence [1]) не рассчитаны на трассировку подобных структур, а также предназначены для работы минимум с двумя коммутационными слоями. Поэтому требуется разработка собственного алгоритма автоматизированной трассировки для однослойной коммутации.

Рассмотрим один из возможных типов специализированного микроэлектронного устройства – аналого-цифровой БМК (АБМК), реализованный по технологии с одним слоем металла. Роль второго трассировочного слоя выполняют вставки из поликремния, имеющие зафиксированные контактные окна.

В аналоговой части кристалла располагаются различные аналоговые элементы (операционные усилители, компараторы, ключи, две группы резисторов). В другой части кристалла расположены заготовки для формирования цифровых элементов. По периметру расположены элементы ввода-вывода (цифровые, аналоговые и мощные буфера).

Аналоговая и цифровая части кристалла различаются по структуре, размеру элементов и плотности их размещения.

Цифровые элементы имеют меньшие размеры, чем аналоговые, и расположены регулярно на поверхности кристалла. Аналоговые элементы значительно превосходят по размерам цифровые элементы, расположены нерегулярно, выводы некоторых аналоговых элементов не попадают в единую сетку трассировки.

Кристалл выполнен по технологии с ортогональным расположением горизонтальных и вертикальных трасс, т. е. проводимые трассы всегда располагаются под прямым углом друг к другу.

Постановка задачи трассировки. Алгоритм должен автоматически реализовывать максимально возможное число трасс. Это должно быть основным критерием при выборе метода трассировки. Вторым критерием должна являться скорость работы алгоритма, а третьим – минимизация по суммарной длине трасс. Аналоговая и цифровая части кристалла должны обрабатываться алгоритмом трассировки как единое целое.

Так как все области контактов вскрыты с помощью окон в оксиде [2], алгоритм не должен проводить связи в областях свободных (неиспользуемых) контактов.

Выбор направления для разработки алгоритма трассировки. Особенность всех алгоритмов трассировки, по которым трассы прокладываются последовательно одна за другой, заключается в том, что конфигурации всех построенных соединений сразу жестко фиксируются. Проведение новых соединений зависит от ранее выполненных, которые рассматриваются как преграды и ограничивают проведение новых трасс [3].

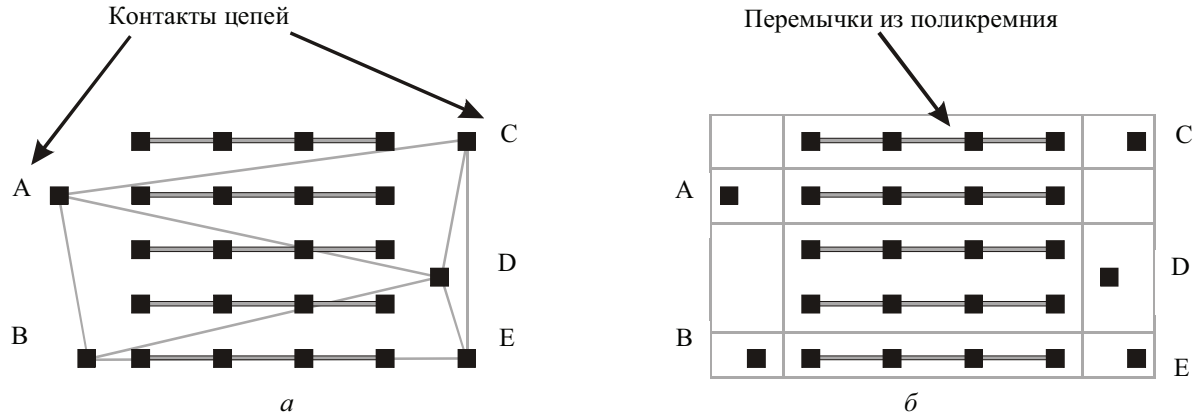
Использование метода гибкой трассировки позволяет избавиться от жесткой фиксации трасс на трассировочном поле. Подобная концепция гибкой трассировки рассмотрена в [4]. Основу методов составляют два положения: распространение волны по макродискретам и отсутствие фиксации («гибкость») трасс в пределах макродискретов на первом этапе трассировки. Ребра макродискретов (граней) соединяют особые точки (выводы размещенных на конструктивном узле элементов, угловые точки запретных зон).

Процесс трассировки выполняется в два этапа. На первом (макротрассировка) происходит распространение волны по макродискретам. В случае построения пути трасса фиксируется с точностью до макродискретов, ребра которых она пересекает. За время проведения макротрассировки определяется положение трасс относительно друг друга и особых точек с учетом метрических характеристик конструктивного узла. На этапе микротрассировки осуществляется построение точной конфигурации трасс – их фиксация на дискретном рабочем поле. Рисунок топологии, получаемый после микротрассировки, содержит ломаные линии, составленные из сегментов любых направлений (не только ортогональных и диагональных), поэтому возникает необходимость применения специальных процедур сглаживания или ортогонализации для придания трассам более технологичного вида.

В работах [4], [5] отмечается, что при использовании метода гибкой трассировки проблема определения очередности разводки цепей стоит менее остро.

Выбор типа макродискретов. При гибкой трассировке наиболее часто применяются прямоугольные и треугольные грани. Первые чаще всего используются при регулярной структуре рабочего поля, вторые – в случае существенно нерегулярных структур.

Преимущество использования треугольных граней (рис. 1, *а*) заключается в том, что общее число граней на дискретном рабочем поле (ДРП) будет значительно меньше, чем в случае использования прямоугольных граней (рис. 1, *б*).



а *б*

Рис. 1

Меньшее количество макродискретов увеличивает скорость работы алгоритма трассировки соединений, однако использование треугольных макродискретов имеет ряд недостатков:

1) сложнее учитывать поликремниевые вставки и, следовательно, вычислять пропускную способность макродискретов. В этом случае необходимо разрабатывать более сложный алгоритм трассировки по макродискретам;

2) после микротрассировки участки цепей получаются проведенными под разными углами, и необходим дополнительный алгоритм их ортогонализации.

Использование прямоугольных граней имеет следующие преимущества:

1) поликремниевые вставки можно выделить в отдельные макродискреты. Таким образом проще учесть пропускную способность макродискретов при работе алгоритма трассировки соединения (на рис. 2 приведен вариант прямоугольного макродискрета с двумя поликремниевыми вставками и пример прохождения цепей);

2) после завершения этапа микротрассировки цепи проще приводить к ортогональному виду.

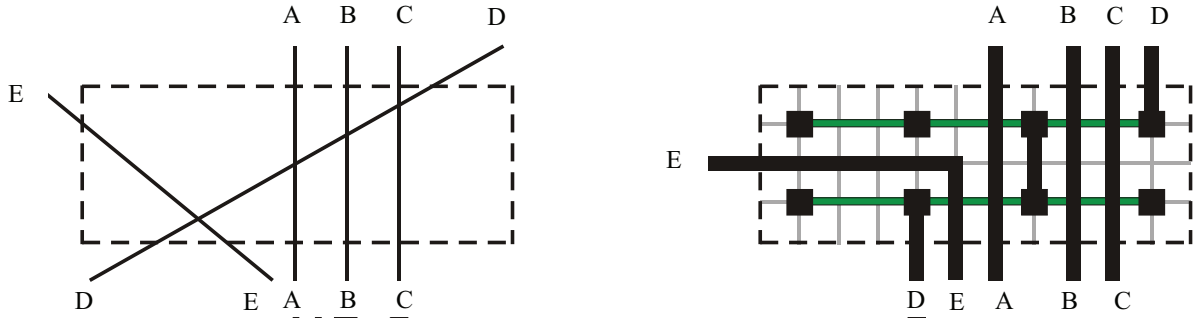


Рис. 2

Исходя из всего изложенного, АБМК предпочтительнее покрывать прямоугольными макродискретами.

Выбор метода для разработки алгоритма макротрассировки. Методы глобальной трассировки слабо разработаны и освещены в литературе, хотя на этапе глобальной трассировки решается вопрос о возможности реализации заданной схемы в виде конкретной топологии БИС [6]. Наиболее подходящий для трассировки однослойных структур алгоритм глобальной трассировки приведен в [6].

Основой этого алгоритма является процедура соединения очередной пары точек цепи с учетом критериев качества глобальной трассировки: минимума суммарной длины соединения и равномерности загрузки каналов БМК. При этом используется распространение меток на модели коммутационного пространства по волновому алгоритму [3], [6] – [9].

Поскольку алгоритм рассчитан на регулярную структуру кристалла с программируемыми контактными окнами, для АБМК можно использовать только основные принципы распространения волны и учета ресурса перемычек, остальные положения алгоритма необходимо адаптировать к нерегулярной структуре АБМК с постоянными контактными окнами. Также можно использовать различные методы ускорения работы алгоритма, поскольку основное время уходит на распространение волны по макродискретам.

Для ускорения выполнения этапа распространения волны можно использовать различные методы – метод выбора начальной точки [8], метод встречной волны [7] – [9], метод построения связывающих деревьев [7], метод специальной раскраски ДРП [7]. Следующая группа методов эффективна только в начале работы программы трассировки, ко-

гда рабочее поле не загружено трассами – ограничение области распространения волны [7], дискретный лучевой алгоритм, или метод прицеливания [3], [7], [8], лабиринтный метод трассировки (метод одностороннего ветвления) [9].

Метод последовательного сокращения ограничений на разметку ДРП в 2...5 раз снижает затраты машинного времени по сравнению с разметкой на полном ДРП для каждой цепи без ухудшения качества трассировки [7].

Несмотря на то что использование метода гибкой трассировки снижает важность порядка трассировки цепей, на этапе макротрассировки это остается важной проблемой. Для определения порядка трассировки цепей предполагается воспользоваться эволюционно-генетическим подходом [5], [10].

Выбор метода для разработки алгоритма микротрассировки. Исходной информацией для этого этапа служат данные, полученные после «макротрассировки». Для каждой макроячейки необходимо определить точное местоположение участков трасс, проходящих через нее. Для решения этой задачи можно использовать основные принципы канальной трассировки, когда в пределах одной ячейки необходимо распределить участки цепей по каналам (см. рис. 2).

Принципы канальной трассировки изложены в [3], [6], [8], [9]. Интерес представляет второй этап канальной трассировки – «локальная трассировка». Наиболее подходящий алгоритм трассировки канала с одним переменным слоем коммутации приведен в [11], однако необходимо дорабатывать положения алгоритма для технологии с постоянными контактными окнами.

Основная трудность на этом этапе – учет выводов, не попадающих в глобальную сетку трассировки.

Заключение. Исходя из особенностей конструкции разрабатываемых в НИИИС кристаллов, наиболее приемлемым для трассировки АБМК является метод гибкой трассировки с прямоугольной формой ячеек.

На этапе макротрассировки алгоритм распространения меток на модели коммутационного пространства целесообразно разрабатывать, основываясь на методах волновой трассировки. Для определения порядка трассировки цепей предполагается воспользоваться эволюционно-генетическим подходом.

На этапе микротрассировки возможно использовать основные принципы канальной трассировки, когда в пределах одной ячейки необходимо распределить участки цепей по каналам.

Разработанные алгоритмы реализовываются на языке С++. Реализованный трассировочный модуль планируется встроить в коммерческую САПР БИС.

СПИСОК ЛИТЕРАТУРЫ

1. Электроника: Наука, Технология, Бизнес. 2003. №5.

2. Микроэлектроника: Учеб. пособие для втузов: В 9 кн. / Под ред. Л. А. Коледова. Кн. 3. Базовые матричные кристаллы и программируемые логические матрицы / Пономарев М. Ф., Коноплев Б. Г. М.: Высш. шк., 1987.

3. Абрайтис Л. Б. Автоматизация проектирования топологии цифровых интегральных микросхем. М.: Радио и связь, 1985.

4. Базилевич Р. П. Алгоритмические методы гибкой трассировки межсоединений / АН УССР. Институт кибернетики. Киев, 1979.

5. Власов С. Е. Разработка системы проектирования гибких полиимидных носителей на базе геометрических методов трассировки: Автореф. дис. … канд. техн. наук / Нижегородский гос. техн. ун-т. Н. Новгород, 1995.

6. Автоматизация проектирования матричных КМОП БИС / А. В. Назаров, А. В. Фомин, Н. Л. Дембицкий и др.; Под ред. А. В. Фомина. М.: Радио и связь, 1991.

7. Быстродействующие матричные БИС и СБИС. Теория и проектирование / Б. Н. Файзулаев, И. И. Шагурин, А. Н. Кармазинский и др.; Под общ. ред. Б. Н. Файзулаева и И. И. Шагурина. М.: Радио и связь, 1989.

8. Конструирование ЭВМ и систем: Учеб. для вузов. 2-е изд., перераб. и доп. М.: Высш. шк., 1989.

9. Проектирование СБИС / М. Ватанабэ, К. Асада, К. Кани, Т. Оцуки; Пер. с япон. М.: Мир, 1988.

10. Батищев Д. И. Генетические алгоритмы решения экстремальных задач: Учеб. пособие / Под ред. акад. АЕН Я. Е. Львовича; Воронеж. гос. техн. ун-т; Нижегородский гос. ун-т. Воронеж, 1995.

11. Сердобинцев Е. В., Сорока Д. В., Тихонов А. И. Математическая модель и алгоритм трассировки канала с одним переменным слоем коммутации // Электронная техника. Сер. 3. Микроэлектроника. 1989. №2. С. 41–44.

D. I. Batischev, S. E. Vlasov, V. V. Balashov

RESEARCH OF METHODS OF THE AUTOMATED SINGLE-LAYERED STRUCTURES TRACING

A problem of the automated tracing of the specialized microelectronic devices sold on technology with one layer of switching is considered. The direction for algorithm of tracing of similar structures development is chosen.

Automated tracing, single-layered structures tracing

УДК 658.512.011.56(043.3)

Найрат Самер

МОДЕЛИРОВАНИЕ СИСТЕМ НА ОСНОВЕ МЕТОДОВ ДИАКОПТИКИ

Рассматриваются алгоритмы и способы организации программного обеспечения для моделирования больших систем на основе методов диакоптики, т. е. декомпозиции систем. Задача решается на основе математического описания схемы в виде блочных окаймленных матриц.

Моделирование, диакоптика, декомпозиция, блочная матрица, подсистема

Необходимость моделирования больших систем выдвигает актуальную задачу разработки методов, позволяющих повысить эффективность вычислительного процесса как с точки зрения объема занимаемой памяти, так и с точки зрения скорости выполнения вычислительных операций. Одним из способов решения такой задачи является использование методов диакоптики, т. е. методов решения систем уравнений по частям. Пусть моделируемая система может быть разбита на m подсистем и имеет, например, вид, изображенный на рис. 1.

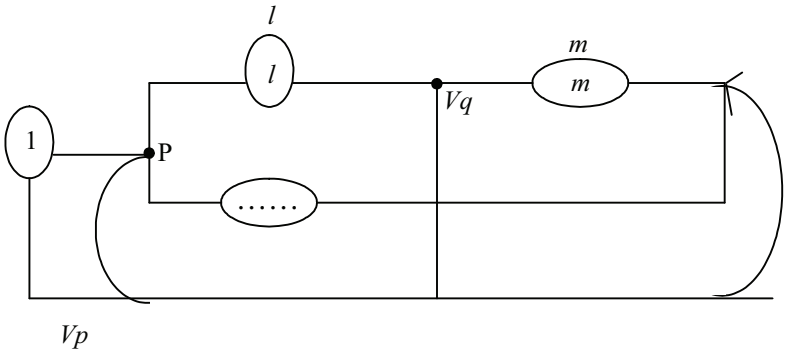


Рис. 1

Для каждого узла такой схемы можно ввести узловой потенциал, определяющий его состояние (потенциалы V_q, V_p для l-й подсистемы – рис. 2).

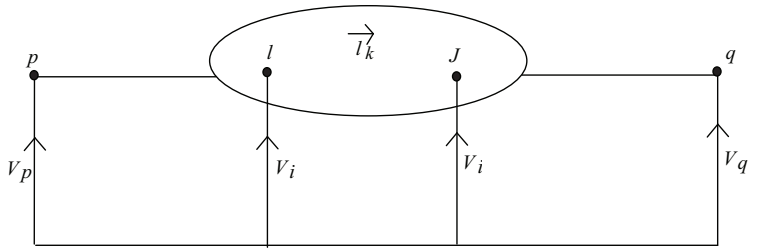
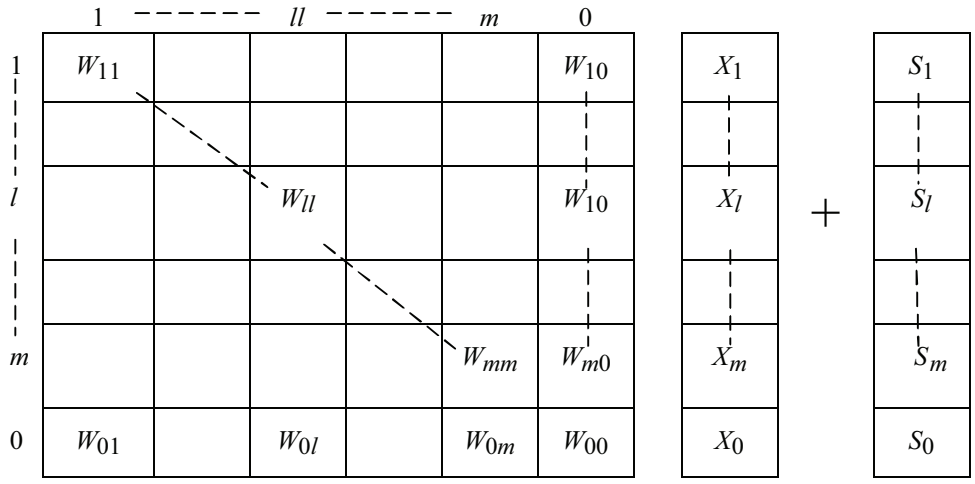


Рис. 2

Образуем из всех введенных узловых потенциалов вектор узловых потенциалов $V_0 = [..., V_p, ..., V_q, ...]^t$. Выделим в каждой подсистеме внутренние узловые потенциалы V_i, V_j и составим из них для каждой подсистемы вектор $V_k = [\ldots, V_i, \ldots, V_j]^t$. Составим далее общий вектор всех переменных системы X и включим в него узловые потенциалы узлов, в которых подсистемы соединяются между собой, а также все внутренние узловые потенциалы подсистемы. Общая структура такого вектора имеет блочный вид

$X = [X_1^t, \ldots, X_l^t, \ldots, X_m^t, X_0^t]^t$, где $X_0 = V_0$, $X_l = [V_l^t, I_l^t]^t$.

При этом вектор $I_l = [..., i_k, ...]^t$ является вектором внутренних последовательных полюсных переменных l-й подсистемы. С учетом принятой структуры блочного вектора X матричное уравнение всей системы в расширенном базисе узловых потенциалов может быть записано в виде



Приведенное уравнение содержит окаймленную блочную матрицу, для которой могут быть записаны уравнения для отдельных подсистем, при этом для каждой подсистемы можно записать матричное уравнение в виде

$$W_{ll} X_l + W_{l0} X_0 + S_l = 0,\ l = 1, \ldots, m.$$

Отсюда можно выразить вектор переменных всех подсистем X_l:

$$X_l = -W_{ll}^{-1} W_{l0} X_0 - W_{ll}^{-1} S_l .$$

Если ввести обозначения $\overline{W_{l0}} = W_{ll}^{-1} W_{l0}$, $\overline{S_l} = W_{ll}^{-1} S_l$, то выражение для вектора переменных подсистем может быть представлено в виде

$$X_l = \bar{W}_{l0} X_0 - \bar{S}_l,\ l = 1,\ \ldots,\ m.$$

Для определения входящего в это выражение вектора X_0 составим уравнение для всей системы в целом, используя последнюю строку общего матричного уравнения

$$\sum_{l=1}^{m} W_{0l} x_l + W_{00} X_0 + S_0 = 0. \qquad (1)$$

Подставляя в (1) значения X_l, получим

$$\sum_{l=1}^{m} (W_{0l} \bar{W}_{l0} X_0 - W_{0l}\ \overline{S_l}) + W_{00} X_0 + S_0 = 0. \qquad (2)$$

Введем обозначения

$$\bar{W}_{00} = W_{00} - \sum_{l=1}^{m} W_{0l} \bar{W}_{l0}\ ,\ \bar{S}_0 = S_0 - \sum_{l=1}^{m} W_{0l} \bar{S}_l\ .$$

С учетом введенных обозначений выражение (2) может быть записано в следующем виде:

$$\bar{W}_{00} X_0 + \bar{S}_0 = 0. \qquad (3)$$

Полученное матричное уравнение позволяет найти вектор переменных связи подсистем. Решая уравнение (3), получим $X_0 = \bar{W}_{00}^{-1} \bar{S}_0$.

Подставляя значения переменных связи в каждое уравнение подсистем

$$X_l = \bar{W}_{l0} X_0 - \bar{S}_l,\ l = 1,\ \ldots,\ m, \qquad (4)$$

получим значения всех внутренних переменных.

Таким образом, в результате проведенной декомпозиции определены все переменные системы, при этом вместо решения общего уравнения $WX + S = 0$ матрица которого имеет большой порядок, используется обращение матриц W_{ll} и W_{00}, порядок которых путем выбора соответствующего числа подсистем и узлов связи может быть сделан сколь угодно малым.

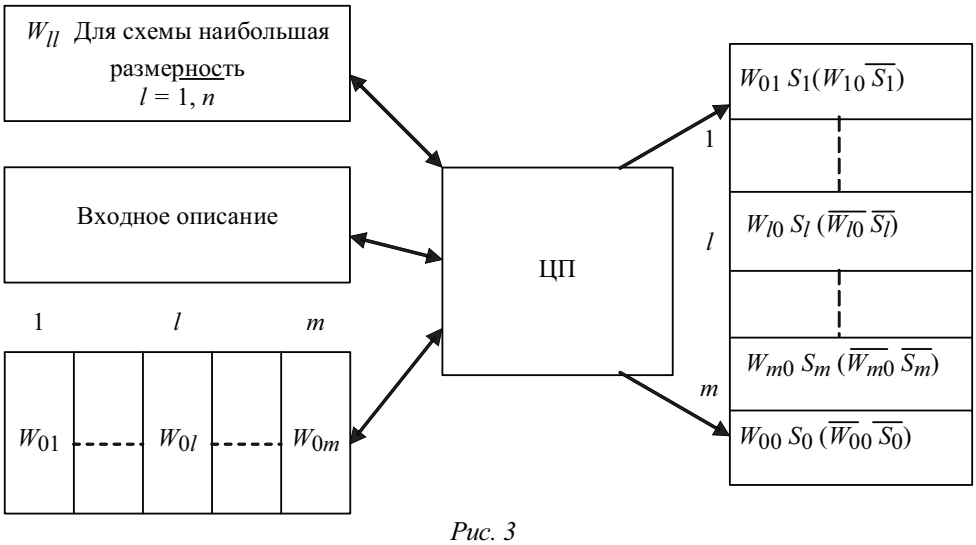


Рис. 3

В соответствии с разработанным алгоритмом моделирования систем на основе блочных окаймленных матриц можно построить следующую структуру организации вычислительного процесса (рис. 3).

Процесс моделирования в соответствии с принятой структурой разделяется на следующие этапы:

1. На основе введенной пользователем информации для каждой подсистемы формируются массивы параметров блочных матриц $W_{ll}, W_{l0}, W_{0l}, W_{00}, S_0, S_l$.

2. Для каждой подсистемы вычисляются $\bar{W}_{l0}$ и $\bar{S}_l$ на основе алгоритма Гаусса–Жордана. В результате матрица W_{ll} преобразуется в единичную диагональную матрицу, а матрицы W_{l0} и S_l преобразуются соответственно в $\bar{W}_{l0}$ и $\bar{S}_l$. Матрицы $\bar{W}_{l0}$ и $\bar{S}_l$ запоминаются для каждой подсистемы вместо матриц W_{l0} и S_l, при этом матрицы W_{l0} и S_l после выполнения хранить не требуется.

3. Рассчитывается вектор переменных связи X_0. На основе матриц $W_{00}, S_0, W_{0l}, W_{l0}, \bar{S}_l$ вычисляются матрицы $\bar{W}_{00}$ и $\bar{S}_0$, и на основе алгоритма Гаусса решается уравнение $\bar{W}_{00}\bar{X}_0 + \bar{S}_0 = 0$, что позволяет определить вектор X_0.

4. Рассчитываются внутренние переменные отдельных подсистем. Внутренние переменные каждой подсистемы вычисляются на основе полученных на предыдущих этапах матриц $\bar{W}_{l0}$, $\bar{S}_l$ согласно (4) последовательно для каждой подсистемы.

Таким образом, указанная последовательность вычисления позволяет минимизировать объем занимаемой памяти и сводит задачу решения уравнений высоких порядков к последовательному решению уравнений более низких порядков.

Для оценки эффективности предложенной процедуры моделирования систем на основе методов диакоптики введем показатель

$$a = M/M_1,$$

где M – объем памяти, необходимой для расчета с использованием предложенного диакоптического подхода, M_1 – объем памяти, требуемый для расчета обычными методами.

Если система разделена на m подсистем, число переменных в каждой подсистеме равно n_l, а число переменных связи составляет n_0, то эффективность метода диакоптики может быть определена выражением

$$\alpha = (n_0 + \sum_{l=1}^{m} n_l)^2 / (\max_l n_l^2 + 2n_0^2 \sum_{l=1}^{m} n_l + n_0^2).$$

Для численной оценки экономии памяти за счет использования диакоптического подхода предположим, что система разделена на 10 одинаковых подсистем и число переменных в каждой подсистеме равно числу переменных связи, т. е. $n_l = n_0$. Для этого случая получим $\alpha = (1 + 10)^2 \; n_l^2 / (1 + 20 + 1) n_l^2 = 5{,}5$.

Таким образом, выигрыш в памяти за счет выбранного способа организации вычислительного процесса равен 5,5. Очевидно, что при увеличении числа подсистем выигрыш в памяти будет увеличиваться еще в больше степени.

Для оценки эффективности предлагаемого диакоптического подхода по быстродействию отметим, что время решения системы уравнений методом Гаусса определяется числом мультипликативных операций и пропорционально величине $n^3/3$, где n – число решаемых совместных уравнений.

В соответствии с приведенной ранее структурной схемой вычислительного процесса, эффективность по быстродействию определяется выражением

$$\alpha = ((n_0 + \sum_{l=1}^{m} n_l)^n a) / (\sum_{l=1}^{m} n_l^3 a + n_0^3 \sum_{l=1}^{m} n_l + n_0^3 a), \quad (5)$$

где $a = 1/3$.

Для случая, когда моделируемая система разбита на 10 одинаковых подсистем и число переменных в каждой подсистеме равно числу переменных связи, т. е. $n_l = n_0$, получим

$$\alpha = (1+10)^3 1/3 n_l^3 /(10+1/2+10+1/3) n_l^3 = 30.$$

Таким образом, эффективность по быстродействию при использовании методов декомпозиции увеличивается в 30 раз. Из структуры выражения (5) следует, что с увеличением числа подсистем эффективность по быстродействию будет возрастать еще в больше степени.

Nyrat Samer

SIMULATION SYSTEMS ON THE BASE OF METHODS DECOMPOSITION

In this paper are considered algorithms and methods organization programs support for simulation big systems on the base of methods decomposition, that's meaning decomposition system. The problem solved on the base of mathematical description scheme in type of the blocked bordered matrix.

Simulation, decomposition, block matrix, subsystem

УДК 519.6

А. Н. Борзых

ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ МЕТОДОВ РАСЧЕТА МАКСИМАЛЬНОГО СОБСТВЕННОГО ЗНАЧЕНИЯ СИММЕТРИЧНОЙ МАТРИЦЫ

Рассматривается алгебраическая проблема собственных значений. Предлагаются эффективные программные реализации, позволяющие ускорить степенной метод и метод бисекции для поиска максимального собственного значения симметричной матрицы. Даются результаты численных экспериментов.

Алгебраическая проблема собственных значений, степенной метод, метод бисекции, спектр матрицы, максимальное собственное число

Во многих инженерных задачах возникает необходимость расчета максимального собственного значения симметричной матрицы. Например, в задачах теории управления максимальные собственные значения позволяют получить информацию об устойчивости системы, в задачах физики – определить собственные частоты колебаний, в задачах математической статистики – произвести редукцию пространства признаков.

Независимо от области применения полученных результатов перед исследователем возникает задача выбора того или иного вычислительного алгоритма. Основными критериями выбора обычно являются время процессорного счета, точность получаемого результата и устойчивость алгоритма к матрицам с особыми свойствами (например, плохо обусловленным).

Из работ [1]–[12] можно получить исчерпывающую информацию о теоретическом обосновании и асимптотических оценках скоростей сходимости различных алгоритмов, однако вопрос о программной реализации и, тем более, о сравнении реальных вычислительных затрат различных алгоритмов лежит за рамками математических исследований. С другой стороны, именно этот вопрос прежде всего интересует инженера-программиста.

В статье рассматривается одна из наиболее востребованных задач на собственные числа – задача вычисления максимального собственного значения симметричной матрицы. Представляются результаты численных экспериментов и даются рекомендации по практическому выбору алгоритма в зависимости от априорных свойств исходной матрицы. Даются приемы, позволяющие осуществить эффективную программную реализацию и обеспечить получение результата с любой заданной точностью.

Классическим методом решения поставленной задачи является ***степенной метод***. Напомним, что суть метода заключается в построении итерационной последовательности векторов $\boldsymbol{x}_i$:

$$\boldsymbol{x}_{i+1} = A\boldsymbol{x}_i,$$

в которой начальный вектор $\boldsymbol{x}_0$ выбирается произвольным образом. Из работ [1, гл. 9], [2, гл. 5], [3, гл. 6], [4, гл. 3], [5, § 30], [7, гл. 11], [8, § 4.2], [10, § 4.4.1], [11, § 23], [12, § 5.2.2] известно, что при $i \to \infty$ вектор $\boldsymbol{x}_i$ сходится по направлению к собственному вектору, соответствующему максимальному по модулю собственному числу. Чтобы осуществить программную реализацию данного метода, необходимо ответить на ряд дополнительных вопросов:

1. По какой формуле оценивать собственное число?

2. По какому критерию останавливать итерационный процесс?

3. Можно ли быть уверенным, что результат получен с заданной точностью?

4. Когда и как нормировать вектор $\boldsymbol{x}_i$, чтобы предотвратить переполнение машинной арифметики?

5. Какова техника получения максимального и минимального по значению, максимального и минимального по модулю собственных значений?

Предлагается следующая практическая реализация.

Ввиду симметрии исходной матрицы наиболее эффективным способом оценки приближенного значения собственного числа можно считать метод скалярного произведения:

$$\lambda_i \approx \frac{(\boldsymbol{x}_i, \boldsymbol{x}_i)}{(\boldsymbol{x}_i, \boldsymbol{x}_{i-1})}.$$

В работах [2, § 30], [8, § 4.2] показано, что в случае симметричных матриц данный метод дает почти двукратное сокращение числа итераций.

Однозначной рекомендации по выбору критерия остановки итерационного процесса не существует. Наиболее популярным и простым способом является оценка стабилизации приближенного собственного числа на соседних шагах алгоритма:

$$\text{если } \left|\lambda_i - \lambda_{i-1}\right| < \varepsilon \text{, то завершить работу.}$$

Очевидно, что применение такого критерия не дает никакой информации о точности получаемого результата и, в принципе, может приводить либо к остановке алгоритма «раньше срока», если ε выбрано слишком большим, либо к чрезмерно долгой его работе, если ε очень мало.

Рассмотрим иной критерий остановки, лишенный данного недостатка. Введем понятие *вектора невязки* $\boldsymbol{r}$:

$$\boldsymbol{r} = A\boldsymbol{x}^* - \lambda^* \boldsymbol{x}^*,$$

где $\boldsymbol{x}^*$ и λ^* – приближенные значения собственного вектора и собственного числа на текущем шаге алгоритма.

Покажем, что для любых $\boldsymbol{x}^*$ и λ^* существует $\lambda \in \sigma(A)$, такое, что

$$\left|\lambda - \lambda^*\right| \le \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{x}^*\|}. \qquad (1)$$

Пусть P – ортогональная матрица собственных векторов симметричной матрицы A, а D – диагональная матрица ее собственных значений:

$$A = P^{-1} D P, \quad D = \operatorname{diag}(\lambda_1, ..., \lambda_n), \quad PP^{\mathrm{T}} = I.$$

Будем считать, что $\lambda^* \notin \sigma(A)$. Тогда матрица $(A - \lambda^* I)$ является невырожденной и имеет обратную. Запишем вектор $\boldsymbol{x}^*$ следующим образом:

$$\begin{aligned}\boldsymbol{x}^* &= (A - \lambda^* I)^{-1}(A - \lambda^* I)\boldsymbol{x}^* = (A - \lambda^* I)^{-1}(A\boldsymbol{x}^* - \lambda^* I\boldsymbol{x}^*) = \\ &= (A - \lambda^* I)^{-1}\boldsymbol{r} = (P^{-1}DP - \lambda^* I)^{-1}\boldsymbol{r} = P(D - \lambda^* I)^{-1}P^{-1}\boldsymbol{r}.\end{aligned}$$

Тогда для оценки его нормы имеем:

$$\begin{aligned}&\left\|\boldsymbol{x}^*\right\| \le \|P\| \cdot \left\|(D - \lambda^* I)^{-1}\right\| \cdot \left\|P^{-1}\right\| \cdot \|\boldsymbol{r}\|, \\ &\left\|\boldsymbol{x}^*\right\| \le \underbrace{\operatorname{cond}(P)}_{=1} \cdot \left\|(D - \lambda^* I)^{-1}\right\| \cdot \|\boldsymbol{r}\|, \\ &\left\|\boldsymbol{x}^*\right\| \le \max_i \left|\frac{1}{\lambda_i - \lambda^*}\right| \cdot \|\boldsymbol{r}\|, \\ &\min_i \left|\lambda_i - \lambda^*\right| \le \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{x}^*\|},\end{aligned}$$

что доказывает неравенство (1).

Таким образом, при использовании для остановки итерационного процесса критерия

$$\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{x}^*\|} < \varepsilon$$

получаемое собственное значение имеет погрешность, гарантированно не превышающую заданное ε.

Для предотвращения переполнения машинной арифметики необходимо «время от времени» производить нормировку вектора $\boldsymbol{x}_i$:

$$\boldsymbol{x}'_i = \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|}.$$

Сложность такой операции составляет $O(n)$ и при больших размерностях почти не влияет на время счета алгоритма (асимптотическая сложность одного его шага определяется матрично-векторным умножением и составляет $O(n^2)$). Тем не менее, для небольшого повышения быстродействия на матрицах малых размерностей данную операцию можно выполнять не на каждом шаге, а, например, через каждые k шагов. Заметим, что операции расчета λ_i и проверки критерия остановки также можно выполнять лишь через каждые k шагов. Эксперименты показывают, что выбор $k = 10$ дает вполне приемлемые результаты.

Известно, что степенной метод позволяет вычислять максимальное по модулю собственное значение, а для расчета минимального по модулю собственного значения разработан метод обратных итераций (см. [8, § 4.3], [1, гл. 9], [2, гл. 5]). В данной работе проводилось вычисление максимального не по модулю, а по значению собственного числа, что обеспечивалось за счет предварительного сдвига спектра матрицы в положительную полуплоскость на величину L1 нормы матрицы (максимум сумм модулей элементов по строкам):

$$A' = A + \|A\|_1 I.$$

Очевидно, что для вычисления минимального по значению собственного числа можно сдвинуть спектр матрицы в обратную сторону:

$$A' = A - \|A\|_1 I.$$

Рассмотрим другой алгоритм расчета максимального собственного значения – метод ***бисекции*** (см. [6, § 44], [10, § 5.3.4]). В книгах по методам вычислений данный метод освещается в меньшей степени, чем степенной метод, хотя по результатам численных экспериментов в большинстве случаев показывает значительно лучшее время процессорного счета.

Идея алгоритма основывается на использовании теоремы Сильвестра об инерции, позволяющей определить число отрицательных, положительных и нулевых собственных значений заданной матрицы A'. Взяв в качестве A' матрицы $(A - aI)$ и $(A - bI)$, можно для любых a и b узнать количество собственных значений матрицы A, заключенных на интервале $[a, b]$. Применяя рекурсивное деление интервала $[a, b]$ пополам, можно локализовать любое собственное значение с любой заданной точностью.

Рассмотрим оптимальную реализацию этого метода для отыскания максимального по значению собственного числа симметричной матрицы. Во-первых, исходную матрицу следует привести к трехдиагональной форме, удовлетворяющей условию

$$\forall i, j : |i - j| > 1 \quad \Rightarrow \quad a_{ij} = 0 .$$

Известно, что такое приведение всегда возможно, и наиболее оптимальным алгоритмом его осуществления считается метод отражений Хаусхолдера (см. [1, гл. 5, § 29, 30], [2, § 51]). Сложность алгоритма – $O(n^3)$, что при больших n определяет основную часть вычислительных затрат метода бисекции.

Далее вводится функция $\mathrm{Negcount}(A, z)$, вычисляющая количество собственных значений матрицы A, меньших, чем z (терминология заимствована из [10]). Если $a_i, i = 1, ..., n$ – элементы главной диагонали A, а $b_i, i = 1, ..., (n-1)$ – элементы под и над главной диагональю, то значение $\mathrm{Negcount}(A, z)$ совпадает с количеством отрицательных элементов последовательности d_i, определенной по рекурсивной формуле:

$$\begin{aligned} d_1 &= a_1 - z, \\ d_i &= (a_i - z) - \frac{b_{i-1}^2}{d_{i-1}}, \quad i = 2, ..., n. \end{aligned} \tag{2}$$

Спецификой расчета именно максимального собственного значения является то, что информация о количестве собственных значений, меньших, чем z, является избыточной и для реализации алгоритма достаточно введения лишь булевской функции $\mathrm{IsPos}(A, z)$, дающей TRUE, если существует хотя бы одно собственное значение, большее, чем z, и FALSE – в противном случае. Тогда вычислительная схема алгоритма имеет вид:

```
while(b-a>eps) {
    z=(a+b)/2;
    if(IsPos(A,z)) a=z;
    else b=z;
}
```

Для реализации функции $\mathrm{IsPos}(A, z)$ также предлагается использовать значения d_i, определенные по формуле (2), но останавливать их расчет, когда встречается хотя бы одно положительное значение d_i, сразу возвращая TRUE. В противном случае, когда все они отрицательны, – возвращать FALSE.

Проведенные эксперименты показывают, что время выполнение $\mathrm{IsPos}(A, z)$ в среднем на 25…30 % меньше времени выполнения $\mathrm{Negcount}(A, z)$.

Другой особенностью эффективной реализации метода бисекции является «удачный» выбор начального интервала $[a, b]$, которому гарантированно принадлежит максимальное собственное значение. Предлагается следующий метод.

Выполним преобразование подобия трехдиагональной симметричной матрицы A, которое обеспечит неотрицательность всех внедиагональных элементов матрицы:

$$A' = D^{-1} A D, \quad \text{где } D = \mathrm{diag}\left(1, \frac{a_{12}}{|a_{12}|}, \frac{a_{12} a_{23}}{|a_{12} a_{23}|}, \ldots, \frac{a_{12} \ldots a_{n-1,n}}{|a_{12} \ldots a_{n-1,n}|}\right).$$

Произведем сдвиг спектра матрицы A', который обеспечит неотрицательность также и диагональных элементов:

$$A'' = A' - cI \text{, где } c = \min_i(a'_{ii}).$$

В итоге получена неотрицательная матрица A'' (все элементы неотрицательны), спектр которой смещен относительно спектра матрицы A на известную величину c. Из работ [5, § 18.9], [9, § 9.2] известно, что для максимального собственного значения $\lambda_{\max}$ неотрицательной матрицы A справедливо:

$$\min_i(p_i) \le \lambda_{\max} \le \max_i(p_i), \quad p_i = \sum_j a_{ij}.$$

Тогда в качестве начальных значений a и b можно взять:

$$a = \min_i(p_i), \quad b = \max_i(p_i),$$

что дает весьма хорошую начальную локализацию максимального собственного числа для любой симметричной матрицы.

Представим результаты ***численных экспериментов*** по определению времени процессорного счета рассмотренных ранее алгоритмов. Для проведения сравнительного исследования были сгенерированы случайные симметричных матрицы различных размерностей от 10 до 100. Для генерации элементов матриц использовалось четыре закона распределения случайной величины:

1. Равномерный закон распределения на интервале [0,1] (рис. 1).
2. Равномерный закон распределения на интервале [–1,1] (рис. 2).
3. Нормальный закон распределения с мат. ожиданием 0 и дисперсией 1 (рис. 3).
4. Нормальный закон распределения с мат. ожиданием 0.5 и дисперсией 1 (рис. 4).

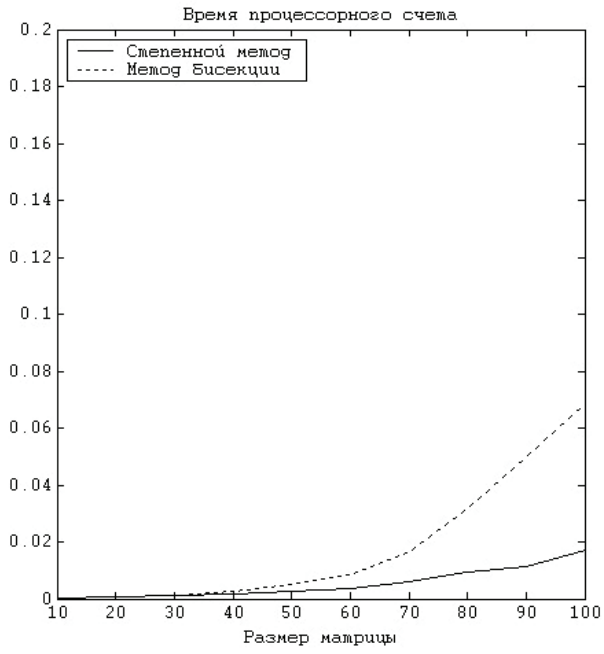


Рис. 1

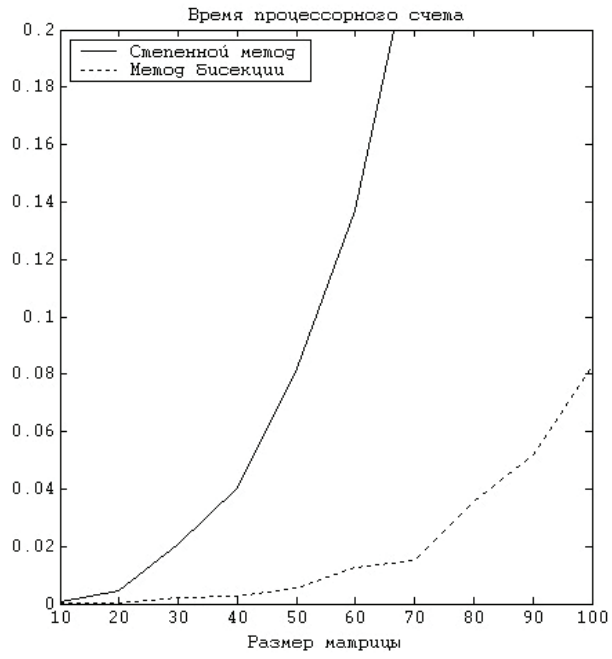


Рис. 2

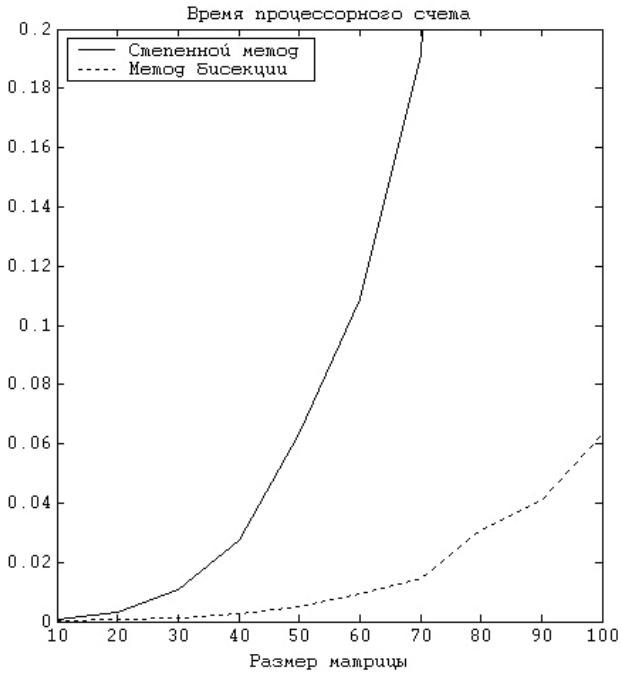


Рис. 3

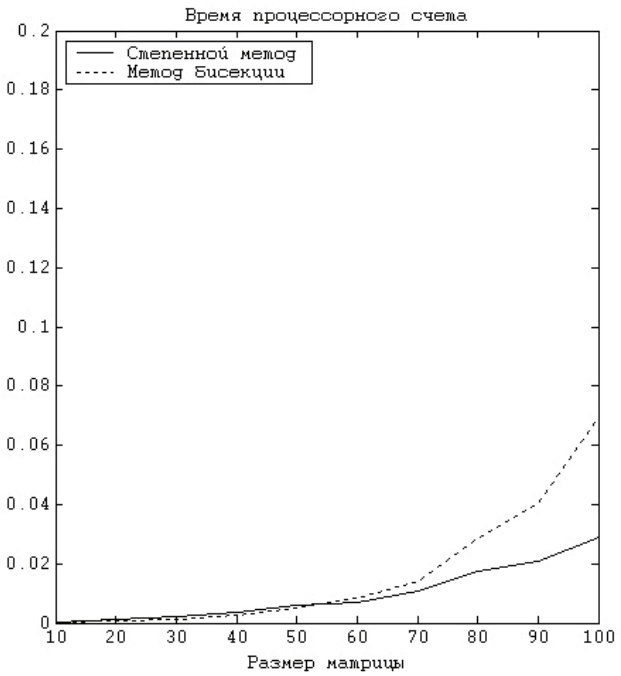


Рис. 4

Представленные на графиках результаты получены путем усреднения 20 прогонов. По оси ординат отложено абсолютное время процессорного счета на машине Pentium 300. Константа ε в степенном методе и методе бисекции – 0.001. Программный код разработан на языке C++ и наиболее качественно оптимизирован для всех использованных операций.

Из результатов видно, что время процессорного счета метода бисекции практически не зависит от способа генерации тестовой матрицы. Действительно, оно складывается из времени приведения к трехдиагональной форме и времени итерационного процесса. Время приведения к трехдиагональной форме зависит лишь от размерности задачи и составляет $O(n^3)$, а время итерационного процесса зависит еще от начальной длины интервала $[a,b]$ и константы требуемой точности ε:

$$T \approx O(\log_2 \frac{b-a}{\varepsilon} n).$$

Степенной метод значительно менее «предсказуем», так как скорость его сходимости зависит от отношения $\frac{|\lambda_1|}{|\lambda_2|}$, где λ_1 и λ_2 – наибольшие по модулю собственные значения, о которых «заранее» ничего сказать нельзя.

На рис. 5–8 показаны гистограммы распределения собственных значений матриц, генерируемых по законам 1, 2, 3 и 4 соответственно. По оси абсцисс – собственное значение, по оси ординат – частота его появления в спектре. Размерность матриц – 200.

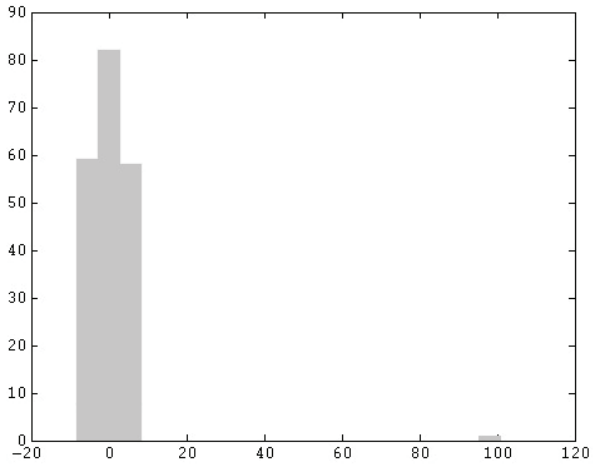


Рис. 5

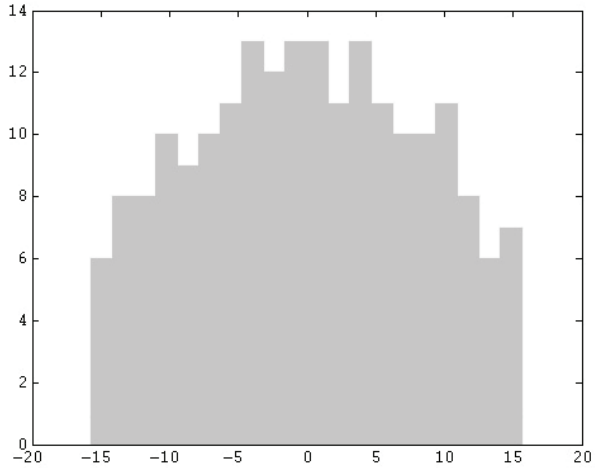


Рис.6

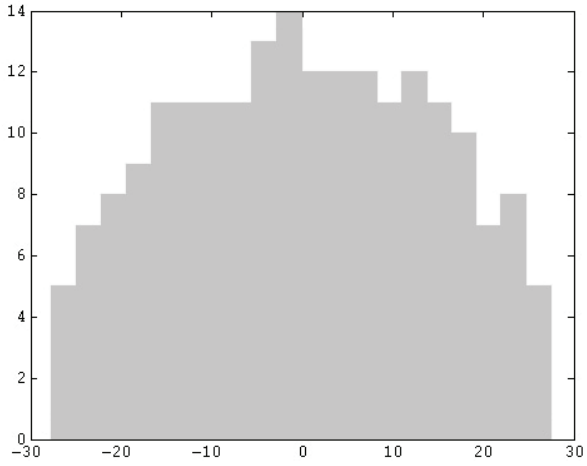


Рис. 7

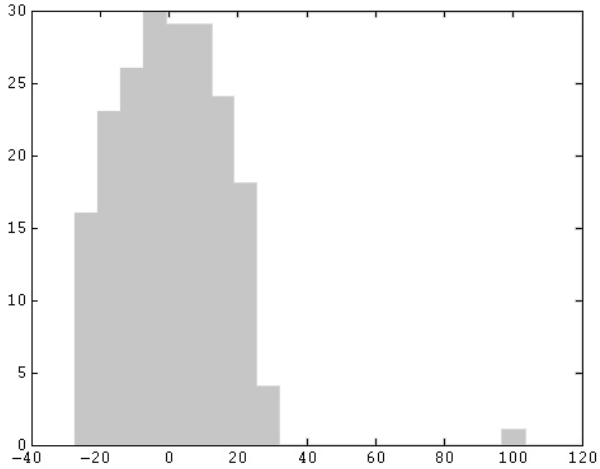


Рис. 8

Интересно отметить, что способы генерации матриц 3 и 4 практически «одинаковы», но порождают совершенно различные спектральные гистограммы. Если в матрицах «вида 3» половина элементов положительна и половина отрицательна, то в матрицах «вида 4» примерно 69 % элементов положительны и примерно 31 % – отрицательны. Несмотря на это получаемая спектральная гистограмма очень близка к той, что соответствует 100 % положительных элементов (матрицы «вида 1»).

Данные соображения позволяют сделать выбор в пользу одного из алгоритмов. Если исходная матрица является неотрицательной и ее элементы имеют «более-менее» случайную природу, то, конечно, предпочтительнее степенной метод. Также если есть какая-либо априорная информация, позволяющая судить о хорошей изоляции максимального собственного значения от остальных собственных значений (см. рис. 5 и 8), то степенной метод, скорее всего, будет эффективнее. Если же об исходной матрице нет никакой информации или, тем более, известно, что ее спектр распределен «более-менее» равномерно, то предпочесть следует метод бисекции. Он всегда и гарантированно даст точное решение за вполне «предсказуемое» время.

Автор выражает большую благодарность своему научному руководителю, заведующему кафедрой высшей математики-2 Санкт-Петербургского государственного электротехнического университета «ЛЭТИ», д-ру физ.-мат. наук, проф. Коточигову Александру Михайловичу.

СПИСОК ЛИТЕРАТУРЫ

1. Уилкинсон Дж. Х. Алгебраическая проблема собственных значений / Пер. с англ. В. В. Воеводина, В. Н. Фаддеевой. М.: Наука, 1970.
2. Фаддеева В.Н. Вычислительные методы линейной алгебры. М., Л.: Гос. изд-во техн.-теорет. лит., 1950.
3. Бахвалов Н. С., Жидков Н. П., Кобельников Г. М. Численные методы. М.: Наука, 1987.
4. Икрамов Х. Д. Несимметричная проблема собственных значений. М.: Наука, 1991.
5. Воеводин В. В., Кузнецов Ю. А. Матрицы и вычисления. М.: Наука, 1984.
6. Воеводин В. В. Вычислительные основы линейной алгебры. М.: Наука, 1977.
7. Хейгеман Л., Янг Д. Прикладные итерационные методы / Пер. с англ. А. Ю. Еремина, И. Е. Капорина; Под ред. Ю. А. Кузнецова. М.: Мир, 1986.
8. Вержбицкий В. М. Численные методы. М.: Высш. шк., 2000.
9. Ланкастер П. Теория матриц / Пер. с англ. С. П. Демушкина. М.: Наука, 1982.
10. Деммель Дж. Вычислительная линейная алгебра / Пер. с англ. Х. Д. Икрамова. М.: Мир, 2001.
11. Волков Е. А. Численные методы. М.: Наука, 1982.
12. Самарский А. А., Вабищевич П. Н., Самарская Е. А. Задачи и упражнения по численным методам. М.: Эдиториал УРСС, 2000.

A. N. Borzyh

COMPUTATIONAL COMPLEXITY OF METHODS FOR FINDING EXTREME EIGENVALUE OF SYMMETRIC MATRIX

The algebraic eigenvalue problem is considered. Effective software implementations that accelerate the iteration method and the bisection method for finding extreme eigenvalue of symmetric matrix are proposed. Experimental results are reported.

Algebraic eigenvalue problem, iteration method, bisection method, matrix spectrum, extreme eigenvalue

УДК 618.3

А. В. Иванов, В. М. Лачинов, А. О. Поляков

РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЕЙ ГЛОБАЛЬНОГО ХРАНЕНИЯ НАУЧНЫХ ТЕКСТОВ

Рассматриваются механизмы построения формальных характеристик устойчивости и изменчивости метаструктуры как системы взаимодействующих БД. Формирование текущей семантики хранилища данных рассматривается как процесс динамического взаимодействия БД, составляющих метаструктуру и экспертные части системы.

Метаструктура, производная по структуре, технология хранения научных текстов, формирование семантики

В [1] была дана постановка работ в самом общем виде – предлагалась технология хранения как процесс самостоятельной реструктуризации хранилища, существующего в структуре метамашины. Это означает, что технология реализована в автоматизированном режиме с участием групп экспертов и «коллективного эксперта», т. е. того, что называют «научной общественностью». Было показано:

а) проблема создания технологи хранения может быть поставлена только и именно как глобальная, поскольку она эквивалентна проблеме формирования текущей семантики научного языка;

б) в силу той же причины должна ставиться задача создания именно автоматизированной системы управления. Такая постановка задачи создания автоматической системы строго равнозначна созданию «искусственного интеллекта» (ИИ), эквивалентна выявлению и учету «общественного сознания» научного сообщества.

Напомним, что в среде апологетов ИИ сама проблема возможности существования ИИ не рассматривается, произвольно принят постулат о неизбежности создания ИИ и его неизбежном будущем превосходстве над естественным интеллектом, будь то индивидуальный разум или «коллективное сознание» в масштабе всей цивилизации. Не рассматривается также адекватность или даже какая-нибудь соотносимость «естественного интеллекта» и ИИ, причем ни как феномена, ни как на уровне соотносимости механизмов, реализующих эти феномены.

Такая ситуация утвердилась со времени первых работ и появления самого термина ИИ. Впрочем, за последнее время стали появляться и некоторые изменения в таком подходе. Так возник целый ряд работ, например [2], пытающихся убедить читателя в следующем: «Коль скоро мы не можем адекватно представить себе, что такое семантика и семиотика и каковы механизмы их формирования, то и не надо, будем заниматься тем, что доступно на уровне манипулирования синтаксическими описаниями в рамках постулированной семантики».

Другими словами, задача формирования семантики явно заменяется задачей комбинирования «частных семантик» при априори заданном содержимом «семиотического котла», т. е. множества значений научных терминов и их отношений. Понятно, что это разные, более того, разнопорядковые задачи. Задача манипулирования «тем, что есть», например, задача генной инженерии и проблема возникновения жизни (почему и как она возникла?) – это разные задачи, это самоочевидно и признается всеми.

Однако в отношении задач структурирования хранилищ текстов такого очевидного понимания и, тем более, сформированного мнения нет. Налицо подмена проблемы феномена самоструктуризации другой задачей, тем, «что мы умеем решать». В [1] сформулированы условия, при которых возможно некоторое формальное введение понятий об изменчивости и устойчивости структуры и сформулировано понятие «производной по структуре» как некоторого аналога производной от функции. Кратко напомним суть постановки проблемы.

Глобальное хранилище научных текстов по определению есть открытая система, т. е. формирующая самое себя под воздействием потока поступающих текстов и потока контекстов – потока суждений «коллективного разума научной общественности» о том, каковы объем, семантика и отношения научных понятий. Т. е. открытая система характеризуется как изменчивостью, постоянным накоплением текстов и контекстов (оценочных определений различного уровня), так и устойчивостью, коль скоро она имеет главной целью свое существование как нечто целостное. В соответствии с феноменологией открытых систем, следуя общему направлению [3], полагаем, что формально феномены неограниченной изменчивости и одновременной устойчивости «как целого» могут быть совмещены с помощью введения понятия метаструктуры открытой системы. Метаструктура представляется как иерархия трех взаимосвязанных уровней структур – так называемая функционально полная метаструктура. При этом уровни постоянно изменяются, «растут» как структуры под воздействием входных потоков данных (могут расти потенциально неограниченно), но метаструктура в целом как совокупность метауровней имеет единственной целью сохранение своей целостности. Другими словами, как бы ни изменялись отдельные уровни метаструктуры, общий ее вид и правила взаимодействия уровней остаются неизменными.

Фундаментальным свойством метаструктуры, ее «метаструктурности» является то, что каждый из метауровней устроен по принципу той же иерархической триады, по принципу матрешки. Т. е. хранилище текстов представляется как триадная иерархия метауровней

$$S = \{S_г \rightarrow S_м \rightarrow S_{тек}\},$$

где $S_г$ – структурный уровень текстов; $S_м$ – структурный уровень метатекстов; $S_{тек}$ – структурный уровень гипертекстов.

Такое представление подсказывается как феноменологией открытых систем [3], т. е. всегда и повсеместно наблюдается в структуре открытых систем, так и тем, что это минимальная метаструктура в которой можно дать формализацию «изменчивости» и «устойчивости» структуры.

Впрочем, триадную организацию систем можно весьма очевидно наблюдать в процессе формирования естественного языка человека. Будем считать «первичным текстом» поток элементарных ощущений. Некоторая совокупность элементарных ощущений формируется в слово, например, по схеме «нечто некоторой формы, некоторого размера, достаточно тяжелый, достаточно твердый» и т. д. – скорее всего, «камень», хотя, строго говоря, такого рода схема в реальности может насчитывать огромное число «контрольный понятий» и все равно приводить к ошибке. Кстати, именно поэтому картофелеуборочный комбайн довольно успешно собирает камни с полей.

Строго говоря, не имеет значения, насколько точно формулируются эти первичные ощущения и каков их «полный набор» для каждого понятия «слова», важна лишь общая характеристика процесса и неважны строгие количественные характеристики. Совокупность первичных ощущений, в конце концов, замыкается в «слово», обозначение некоторого предмета или действия.

Синтаксис языка соединяет слова в структуру предложения, т. е. структуру, отображающую некоторое текущее отношение между понятиями. Т. е. общая метаструктура современных языков на самом нижнем уровне имеет вид триады структур, раскрывающих структуру первичного текста $S_{п.т}$:

$$S_{п.т}. => \{\{\text{предложения}\} \leftrightarrow \{\text{слова}\} \leftrightarrow \{\text{первичные ощущения}\}\}.$$

Весьма характерно, что во всех современных языках структуры типа {{совокупность предложений} = повествование} логически отделены как более высокий уровень метаструктуры, представляют собой уже следующий метаструктурный уровень.

Обратим внимание на то, как определяется «верх» $S_{п.т}$. Синтаксическая структура «предложение» логически фиксирует текущее, одномоментное значение некоторого отношения между двумя или более «словами» (понятиями). Т. е. рассмотренная триада именно то, что позволяет фиксировать некоторую устойчивость на заданный момент описываемого изменчивого отношения. Само «значение отношения» становится своего рода «вторичным словом», «метасловом» или «метапонятием», единицей следующего структурного уровня.

Синтаксическая структура предложения как бы «опредмечивает», «делает конкретной вещью» некоторое значение отношения между «первичными понятиями». Следующий уровень языка $S_{мт}$ (метауровень) представляется триадой

$$S_{мт} => \{\{\text{предметные области – науки}\} \leftrightarrow \{\text{темы}\} \leftrightarrow \{\text{повествования}\}\}.$$

Характерно, что структурное замыкание уровня $S_{п.т}$ «опредмечивание отношения» задает как бы по рекурсии и метаструктуру следующего уровня $S_{мт}$ (метатекстов) и мета-

структуру языка в целом. Так, еще сохранились целые группы языков «непредметных» (эскимосский и др.), где центральным в предложении является не предмет, а действие. В этих языках не сложилась синтаксическая структура предложения, фразы как отдельная структурная единица. В этих языках весьма длинное повествование представляет собой «одно очень длинное предложение». Т. е. там, где не предусмотрено определение метаструктуры, «факт наличия следующего, отдельного уровня, не происходит и практического выделения структуры. Повествование складывается в единую, непрерывную последовательность действий, следующих друг за другом».

Таким образом, чтобы «понятие следующего уровня, метапонятие» могло сформироваться, должна быть задана метаструктура как «наличие уровней, следующих вверх по иерархии, но уже логически самостоятельных (целостных единиц)». Здесь, на примере замыкания структуры предложения видно как синтаксический механизм, ограничивая цепочку взаимосвязей предметов и отношений (взаимодействий), порождает «семантическую единицу следующего уровня».

Но также очевидно, что «семантической» единицей предложение является лишь в контексте некоторого данного повествования и данной темы, в другом контексте оно может стать бессмысленным набором слов или иметь другую семантику. Понятно, что «функция вхождения в контекст» существенно разрывна и нелинейна – одно и то же предложение может иметь смысл в различных контекстах, бывают фразы, подходящие к очень широким и разнообразным контекстам – так называемые сложные синтагмы. Т. е. очевидно, что механизм синтаксического замыкания фразы не единственный, он не функционален без другого механизма – проверки, подходит ли именно такое замыкание к данному повествованию и данной теме. Работают два взаимодополняющих механизма: замыкание фразы (синтез метапонятия) и проверка на вхождение этого метапонятия в повествование и тему.

Второй механизм, «анализ на вхождение» требует для своего функционирования явного определения уровня $S_{\text{мт}}$ на некоторой физически существующей области памяти системы (чтобы было с чем сравнивать). Но заметим сразу же, что в этом процессе не обнаруживается никакого прямого или косвенного указания на то, как должна быть устроена область памяти для $S_{\text{мт}}$ – то ли это декларативная структура, то ли чисто ссылочная. А это будет важно при попытке реализации системы, способной выполнять рассмотренные функции. Этот вопрос решается на основе анализа процесса выяснения вхождения предложения в некоторую тему. Для этого нужно построить все деревья вывода для всех тем, содержащих вхождения данного предложения и выбрать те из них, для которых производные по структуре

$$D^1S, D^2S \to \min.$$

Очевидно, что в общем случае таких деревьев будет несколько, т. е. конкретное предложение может быть отнесено к нескольким темам.

Вот здесь-то и должен проявиться автоматизированный характер создаваемой системы, конкретный человек – эксперт должен решить: либо задана одна конкретная тема, либо окончательное соотнесение должно быть выполнено при анализе дальнейших предложений (в пределе – всего повествования).

Заметим, что при определенных условиях эта процедура построения деревьев вывода и поиска минимума производных всегда конечна, даже в том случае, если структура $S_{\text{п.т}}$ на нижнем уровне входящих в нее структур работает потенциально бесконечно. А именно:

а) все входящие структуры метаструктуры устроены самоподобно, например, как системы B*-деревьев;

б) все входящие структуры существуют как физические.

Тогда искомую процедуру можно построить чисто механически как перечисление всех систем всех входящих B*-деревьев по всем уровням структур.

Разумеется, среди входящих B*-деревьев будут повторяющиеся, которым достаточно, например, присваивать индексы в порядке их построения. При указанных условиях этот процесс всегда конечен, более того, число шагов предвычислимо [4].

Фактически, при указанном ранее типе организации метаструктуры как иерархической совокупности самоподобных баз данных (БД) должна быть выполнена процедура «сквозного разыменования» по всей иерархии БД, причем должны быть отобраны те деревья, листьями которых является самый нижний уровень, т. е. для них процедура разыменования завершима. Те деревья, для которых процедура не заканчивается, просто выбрасываются. Выполнение процедуры разыменования по метаструктуре как по единой БД назовем процедурой тривиализации, поскольку результатом является тривиальное (по дереву) соотнесение первичных данных с темой.

Вопрос о том, как, какими средствами организовать эту процедуру, в частности, как быть с проблемой выхода на полный перебор, целесообразно вынести в отдельную, самостоятельную публикацию.

В связи с этим далее сосредоточимся на следующем моменте. В результате тривиализации метаструктуры получим ее состояние на некоторый момент времени t_0. Будем считать время некоторой свободной переменной, заданной в троичной ультраметрической шкале {«до», «сейчас», «после»}. Т. е. величина физического времени присутствует чисто условно, фактически это изменение не «во времени», а «в контексте», важен лишь факт, что изменение происходит не мгновенно. Значит, необходимо сравнить тривиализированные метаструктуры в моменты времени:

$t_0 - \Delta t$ – что было «до»;

t_0 – что было «зафиксировано последним»;

$t_0 + \Delta t$ – что происходит в «текущий момент».

Соответственно, надо построить два «супердерева разности»:

$$\Delta S\,(-\Delta t) = \Delta S\,(t_0) - \Delta S\,(t_0 - \Delta t),$$

$$\Delta S\,(+\Delta t) = \Delta S\,(t_0 + \Delta t) - \Delta S\,(t_0).$$

«Дерево – разность» – это перечисление всех отличающихся супердеревьев, появившихся за момент Δt в результате поступления очередной порции текстов и контекстов.

Подчеркнем, что здесь рассматривается общий случай, когда сообщение поступает на все уровни метаструктуры, например:

$$\Delta S = \begin{Bmatrix} S_{\text{ГТ}}(t_0) - S_{\text{ГТ}}(t_0 - \Delta t) \\ S_{\text{МТ}}(t_0) - S_{\text{МТ}}(t_0 - \Delta t) \\ S_{\text{П.Т}}(t_0) - S_{\text{П.Т}}(t_0 - \Delta t) \end{Bmatrix}$$

Другими словами, поступает не только текст на уровне $S_{\text{п.т}}$, например прямое или косвенное указание – «эта фраза относится к другой (обозначение) теме», но и гипертекст также, например, указание «это надо рассматривать в такой-то (обозначение) картине Мира». При этом (обозначение) может быть и пустым – «неизвестной» темой.

Введенная ранее оценка «производной по структуре» *DS* оказывается, таким образом, отношением числа узлов по уровням структуры ΔS.

Проводя аналогию с производной функции многих переменных можно заметить, что *DS* аналогична некоторой оценке модуля вектора полной производной, но именно оценке, а не величине. Числовое значение *DS* имеет смысл только для некоторой локальной области «локального состояния» метаструктуры *S*. На этом сходство с полной производной функции многих переменных (ФМП) и заканчивается. Фундаментальными отличиями от ФМП являются следующие:

а) нет никакой априорной связи между значениями Δt и *DS*, хотя в каждом конкретном случае такая зависимость явно наблюдается;

б) производная *DS* является как бы производной по направлениям, причем не ортогональным и тем более не ортонормированным.

Последнее очевидно из того, что уже априори не заданы все первичные ощущения (элементарные сигналы), заданы не все «слова» (термины), определены не все предметные области (могут возникать новые) и т. д. Т. е. это еще раз показывает, что *DS* имеет смысл только как локальная оценка и по времени, и по пространству термов. Смысл локальности оценки состоит в том, что нельзя из временной последовательности событий делать однозначный вывод об их логической взаимосвязи – «после не означает вследствие». Т. е. отношение последовательностей наблюдаемых событий и последовательностей их логической связи в общем случае нетранзитивно.

Нетранзитивность представления «в целом» является своеобразной платой за целостное представление о Мире. Фактически процедура построения «областей непрерывности», производных *DS* означает, что «экспериментируя с гипотезами» находят некоторые области транзитивности, в которых из временной последовательности событий допустимо делать выводы об их логической взаимосвязи.

Заметим – чисто механически, просто потому, что в данной «картине Мира», данной предметной области и данном «повествовании» получается некоторая связная во времени последовательность тривиализованных метаструктур *S*. Фактически построение разностной структуры ΔS является «частной ортогонализацией структур в пределах доступных фактов». При этом выделяется как бы минимальная согласующая структура.

В сравнении со всеми известными подходами новым здесь является то, что впервые вводится эффективное определение семантики как такого разбиения термов и их взаимосвязей, которое обеспечивает наибольшую устойчивость метаструктуры *S* как целого.

Здесь это определение дается как набор процедур над определенными совокупностями взаимодействующих баз данных. Разумеется, нельзя не поставить вопрос о сходимости, причем конечной сходимости этого набора процедур. Данный круг вопросов определенно представляет собой целую значительную область исследований, что было ясно еще со времен работ Ухтомского и Анохина [5], [6], с тех пор как поставлен вопрос о це-

лостном образе восприятия Мира и способах оперирования такими представлениями. Предлагается наиболее простое решение – во-первых, целостное должно быть устойчивым, поэтому ранее и предложен способ формального определения этого типа «устойчивости» и набор процедур «анализа на устойчивость». Многие вопросы феноменологического характера правомочности этого подхода рассмотрены в [3], [7] и ряде других работ, но, повторяем, это целое отдельное научное направление, не меньше.

Здесь же важно то, что существует некоторый, например предложенный здесь набор эффективных процедур «построения семантики». Некоторым доказательством правомочности предложенного здесь подхода является уже то, что наука существует в современном виде, т. е. сложилась именно как процесс формирования специализированных научных языков в рамках процесса существования естественного языка. Далее естественно поставить вопрос о том, возможно ли эффективное построение рассмотренных здесь процедур в рамках современных информационных технологий.

Такое решение реализуемо как развитие HTML-технологий и технологий, основанных на формализации языка «М» [8]. Необходимо построить систему эффективного взаимодействия априори неопределенного числа неортогональных баз данных. Инструментом, адекватным задаче, может быть подход, предложенный в метасистеме qWord [9], однако требуется существенная доработка подхода как на уровне идеологии проекта, так и на уровне инструментария, чему и будет посвящена следующая публикация.

СПИСОК ЛИТЕРАТУРЫ

1. *Иванов А. В.* Проблемы создания технологии глобального хранилища научных текстов // Изв. СПбГЭТУ «ЛЭТИ». Сер. «Информатика, управление и компьютерные технологии». 2004. Вып. 2. С. 68–75.

2. *Шерстюк Ю. М.* Основы метауправления функциональностью в информационных системах. СПб.: Геликон Плюс, 2000.

3. *Лачинов В. М., Поляков А. О.* Метамашина. СПб.: Изд-во СПбГПУ, 2003.

4. *Кнут Дональд Э.* Искусство программирования. Т. 3 . Сортировка и поиск. 2-е изд. М: Издательский дом «Вильямс», 2000.

5. *Ухтомский А. А.* Очерк физиологии нервной системы // Соч. Т. 4. Л.:Наука, 1945.

6. *Анохин П. К.* Биология и нейрофизиология условного рефлекса. М.: Медицина, 1968.

7 *Поляков А. О., Лачинов В. М., Тукабаев П. Т.* Управление в сложных биотехнических системах // Вестник МГТУ им. Н. Э. Баумана. Сер. «Естественные науки». 2004. № 2. С.74–89.

8 *Долженков А. Н., Поляков А. О.* Создание промышленных информационных систем на основе М-технологии // Доклады IV СПб международной конференции «Региональная информатика 95». СПб.: Изд-во СПОИСУ, 1995.

9 *Веселов В., Долженков А.* Технология XML – новая страница в развитии СУБД // Открытые системы. 2001. № 2.

A. V. Ivanov, V. M. Lachinov, A. O. Polyakov

REALIZATION OF AUTOMATIC SYSTEM OF TECHNOLOGY MANAGEMENT OF GLOBAL STORAGE OF SCIENTIFIC TEXST

The mechanisms of building of formal characteristics of stability and changeability of metastructure as a system of interacting data bases are considered. Forming of current semantics of data storage is considered as a process of dynamic interaction of data bases, consisting of the metastructure and expert parts of the system.

Metastructure, derivative by structure, technology of saving scientific of texts, shaping of semantic