МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М.В. ЛОМОНОСОВА

Факультет вычислительной математики и кибернетики

ПРОГРАММНЫЕ СИСТЕМЫ И ИНСТРУМЕНТЫ

Тематический сборник

№ 6

Под общей редакцией
чл.-корр. РАН Л.Н. Королева

МОСКВА
2005

УДК 519.6+517.958
ББК 22.19
П75

Печатается по решению Редакционно-издательского совета факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова

П75 **Программные системы и инструменты:** Тематический сборник факультета ВМиК МГУ им. М. В. Ломоносова: № 6/ Под ред. Л.Н. Королева. – М: Издательский отдел факультета ВМиК МГУ (лицензия ИД №05899 от 24.09.2001г.), 2005. – 240 с.

ISBN 5-89407-243-3

В данный сборник включены научные работы и сообщения, связанные с общими вопросами программирования и информатики, касающиеся, в том числе проблем образования, исследований и разработок, связанных с анализом экспериментальных данных, имитационным моделированием систем реального времени, сетевой обработкой, а также с описанием некоторых инструментальных систем, которые могут оказаться полезными.

В этих публикациях нашли отражение исследования и разработки в области создания программных систем, выполненные учеными, аспирантами и студентами факультета. Большая часть результатов доложена на Ломоносовских Чтениях 2005 года.

Статьи сборника будут интересны студентам, аспирантам и специалистам в области разработки прикладных программных систем с использованием новых информационных технологий.

УДК 519.6+517.958
ББК 22.19

ISBN 5-89407-243-3

СОДЕРЖАНИЕ

---

[1] Работа выполнена при поддержке гранта РФФИ 05-01-00948 и гранта президента РФ НШ- 1689.2003.1.

[2] Работа выполнена при поддержке грантами РФФИ 05-01-00744а и РФФИ 03-01-00745.

[3] Работа выполнена при поддержке гранта РФФИ 05-01-00744а

[4] Работа выполнена при поддержке гранта РФФИ 05-07-90238.

[1] Работа выполнена при частичной поддержке РФФИ, грант № 04-01-00556.

[2] Работа поддерживается грантами РФФИ 03-01-00745, РФФИ 06-01-00691.

[3] Работа выполнена при поддержке гранта РФФИ № 05-07-90238.

СБОРНИК

" Программные системы и инструменты"

От редколлегии:

В предлагаемом читателю тематическом сборнике публикуются статьи, посвященные в основном описаниям инструментальных программных систем, разработанных авторами публикаций, относящихся к методам обработки экспериментальных данных, имитационному моделированию, проблемам сетевой обработки, некоторым вопросам теоретического программирования.

В нем также публикуются статьи и заметки общего характера, касающиеся разделов информатики, тесно связанных с программированием.

Редколлегия

Раздел I

Общие вопросы программирования и информатики

Корухова Ю.С.

Дедуктивный синтез программ с использованием волновых правил[1]

Введение

В работе рассматривается задача синтеза программ по их формальным спецификациям. Спецификация описывает соотношение между входными и выходными данными программы, то есть формулирует условия задачи, которую требуется решить. При дедуктивном подходе спецификация рассматривается как математическая теорема, для которой требуется построить конструктивное доказательство. То есть утверждая существование объекта, удовлетворяющего заданным условиям, мы должны предъявить способ его построения. Программа, в отличие от спецификации, обязательно должна содержать алгоритм вычисления результата.

В работе [10] предложен метод дедуктивных таблиц, применимый для синтеза функциональных программ. Согласно этому методу, доказательство теоремы-спецификации проводится с помощью специальных правил вывода, каждому из которых сопоставлен определенный шаг построения функциональной программы. Метод позволяет синтезировать три базовые конструкции функциональной программы: применение функции, условное выражение и рекурсию. Авторами метода доказано, что построенная таким образом программа удовлетворяет всем условиям, заданным в спецификации, то есть нет необходимости в дальнейшем проводить верификацию программы. В

[1] Работа выполнена при поддержке гранта РФФИ 05-01-00948 и гранта президента РФ НШ- 1689.2003.1

связи с этим представляет интерес проведение синтеза автоматически, который позволил бы автоматически получать программы, соответствующие заданным спецификациям. Однако при проведении синтеза в дедуктивной таблице на каждом шаге может быть применимо несколько правил вывода, и даже для простых задач возникает очень большой перебор. Для его сокращения требуются дополнительные эвристики. В данной работе предлагается использовать волновые правила [7] при построении рекурсивной ветви функции, чтобы определить план проведения доказательства. Этот подход был реализован в системе автоматического синтеза функциональных программ АЛИСА.

В работе приводится краткое описание синтеза программ с помощью метода дедуктивных таблиц, далее описываются основные понятия, связанные с использованием волновых правил, и представляется способ применения волновых для синтеза программ в дедуктивной таблице. В заключении приведено сравнение с известными системами синтеза и сформулированы полученные результаты.

Синтез программ в дедуктивной таблице

Синтез программы проводится на основе формальной спецификации. Мы будем использовать язык спецификаций, основанный на конструкциях логики предикатов первого порядка, содержащий арифметические операции, а также предикаты и функции для работы со списками и целыми числами (head – первый элемент списка, tail – "хвост" списка, number – предикат, определяющий, является ли его аргумент числом и т.д.) В качестве примера рассмотрим спецификацию функции, которая вычисляет целочисленный квадратный корень заданного неотрицательного целого числа:

$$\langle sqrt(b)\rangle <= \text{find } \langle z\rangle \text{such that if } b\geq 0 \text{ then } (z^2\leq b \text{ and } b<(z+1)^2) \quad (1)$$

В спецификации указано, что для заданного b требуется найти такое z, которое удовлетворяет условиям, записанным после ключевых слов such that. Такой спецификации соответствует теорема существования

$$\forall b\ \exists\ z \text{ if } b\geq 0 \text{ then } (z^2\leq b \text{ and } b<(z+1)^2) \quad (2)$$

Чтобы синтезировать функцию sqrt(b), мы строим конструктивное доказательство теоремы (2) (доказательство существования z). Если теорему удастся доказать, то будет получен алгоритм вычисления z, который затем непосредственно переводится в текст программы.

Доказательство проводится в специальной структуре, называемой дедуктивной таблицей (см. таблицу 1).

Утверждения	Цели	f(a)
A_1		
...		
A_n		
	G_1	t_1
	...	
	G_m	t_m

Таблица 1. Структура дедуктивной таблицы

Каждая строка таблицы содержит логическое выражение, являющееся либо утверждением, либо целью, и может содержать терм в выходной колонке. Таблица является наглядной формой записи выражения

if $A_1 \wedge ... \wedge A_n$ then $G_1 \vee ... \vee G_m$

где $A_1, ..., A_n$ известные аксиомы о предметной области, а $G_1, ..., G_m$ – цели, хотя бы одну из которых необходимо доказать. Выходная колонка используется для построения синтезируемой функции. Если задана спецификация нескольких функций, результаты которых зависят друг от друга, то синтез таких функций проводится одновременно, а в таблице возникает несколько выходных колонок, но они все обрабатываются аналогичным образом, поэтому в дальнейшем будем рассматривать таблицу с одной выходной колонкой.

Из определения дедуктивной таблицы вытекают следующие свойства.

- Свойство двойственности: утверждение любой строки может быть перенесено с отрицанием в колонку целей, а цель, также с отрицанием, - в колонку утверждений.
- Свободные переменные в разных строках являются независимыми и при необходимости могут быть переименованы.
- Логическое выражение (утверждение или цель) любой строки можно заменить на эквивалентное выражение (например, if A then B else C на $(A \wedge B) \vee (\neg A \wedge C)$).
- Если в выходной колонке строки находится свободная переменная, не содержащаяся в утверждении (или цели) этой строки, то эта переменная может быть удалена из выходной колонки.

Перед началом синтеза в колонке утверждений таблицы могут содержаться аксиомы, описывающие свойства известных предикатов и функций. В качестве цели для доказательства добавляется теорема-спецификация, а в выходную колонку записывается переменная, существование которой утверждает эта теорема. Для спецификации (1)

в таблицу, содержащую аксиомы, будет добавлена новая цель (см. таблицу 2).

Утверждения	Цели	sqrt(b)
...		
$0^2=0$		
	if $b\geq 0$ then $(z^2\leq b \wedge b<(z+1)^2)$	z

Таблица 2. Исходная таблица для синтеза функции sqrt(b)

Доказательство цели осуществляется с помощью дедуктивных правил, которые добавляют новые строки в таблицу. Правила применяются до тех пор, пока не будет получена тождественно истинная цель (или тождественно ложное утверждение). В выходной колонке строки, содержащей такую цель (соответственно, утверждение) окажется терм, соответствующий синтезированной функции, который непосредственно переводится в текст функциональной программы. Так как в дальнейшем будут рассматриваться именно функциональные программы, вместо термина "программа" мы будем иногда употреблять термин "функция", подразумевая, что функциональная программа представляет собой описание некоторой функции.

Мы будем рассматривать синтез программ на языке Лисп, используя то его подмножество, которое сохраняет функциональный стиль программирования, то есть не содержит присваиваний и операторов цикла.

Рассмотрим основные дедуктивные правила, используемые для синтеза программ.

Правило расщепления. Утверждение, содержащее конъюнкцию (цель, содержащая дизъюнкцию), может быть расщеплено на составные части. Выходной терм полученных строки дублирует выходной терм исходной строки.

Правило резолюции. В таблицу, содержащую строки 1 и 2, может быть добавлена строка 3 (см. таблицу 3).

1:		$G_1[P]$	s
2:		$G_2[Q]$	t
3:		$G_1\lambda[\text{false}] \wedge G_2\lambda[\text{true}]$	if $P\lambda$ then $t\lambda$ else $s\lambda$

Таблица 3. Применение правила резолюции.

Здесь G_1 и G_2 – выражения, не имеющие общих свободных переменных (если это не так, следует переименовать переменные), а P, Q – их подвыражения, которые могут быть унифицированы некоторой подстановкой λ (т.е. $P\lambda \equiv Q\lambda$). Заменив все вхождения $P\lambda$ в $G_1\lambda$ на false, получаем $G_1\lambda$[false], заменив все вхождения $Q\lambda$ в $G_2\lambda$ на true, получаем

$G_2\lambda$[true], после чего добавляем новую строку с указанной целью и условным выходным термом.

Замечания о выходном терме:

1. Если $s\lambda=t\lambda$, то упрощаем условный терм: if $P\lambda$ then $t\lambda$ else $t\lambda \equiv t\lambda$
2. Если одна из строк (например, G_2) не имеет выходного терма, то вместо условного терма в качестве выхода новой строки записывается $s\lambda$ - выход второй строки.
3. Если у обеих строк нет выходных термов, то и у новой строки его не будет.

Правило замены эквивалентных термов. В таблицу, содержащую строки 1 и 2 (см. табл. 4), может быть добавлена строка 3:

1:		$G_1[P=Q]$	s
2:		$G_2[R]$	t
3:		$G_1\lambda[false] \wedge G_2\lambda<Q\lambda>$	if $(P=Q)\lambda$ then $t\lambda$ else $s\lambda$

Таблица 4. Применение правила замены эквивалентных термов.

Здесь G_1 и G_2 – выражения, не имеющие общих свободных переменных (если это не так, следует переименовать переменные); P, Q и R - термы, причем в цель G_1 входит формула P=Q, а терм R, входящий в G_2, может быть унифицирован с P подстановкой λ (т.е. $P\lambda=R\lambda$.). Заменив все вхождения $(P=Q)\lambda$ в $G_1\lambda$ на false, получаем $G_1\lambda$[false], заменив некоторые вхождения $R\lambda$ в $G_2\lambda$ на $Q\lambda$, получаем $G_2\lambda<Q\lambda>$. После этого добавляем в таблицу новую строку с указанной целью и условным выходом. (См. в правиле резолюции замечания о выходном терме).

Применив описанные дедуктивные правила можно построить нерекурсивную ветвь функции. Для синтеза нерекурсивной ветви функции sqrt на основе спецификации и аксиомы a1 (см. таблицу 2) в дедуктивную таблицу могут быть добавлены новые строки (см. табл. 5).

a1	$0^2=0$		
g0		if $(b\geq 0)$ then $(z^2\leq b$ and $b<(z+1)^2)$	z
g1		$\neg (b\geq 0)$	
g2		$z^2\leq b$ and $b<(z+1)^2$	z
g3		$0\leq b$ and $b<(0+1)^2$	0
g4		$b<1$	0

Таблица 5. Построение нерекурсивной ветви функции sqrt

Здесь строки g1 и g2 – результат расщепления исходной цели g0, строка g3 получена после применения правила замены эквивалентных термов в a1 и g2, резолюция g3 и g1 порождает строку g4 (нерекурсивная ветвь функции sqrt). Строка, соответствующая нерекурсивной ветви функции sqrt(b), имеет следующий смысл: если будет доказано, что b<1 (цель), то в результат функции – 0 (выходной терм).

Для синтеза рекурсивного обращения к функции потребуется применение доказательства по индукции.

Добавление гипотезы индукции. В таблицу, содержащую цель g0 может быть добавлена гипотеза индукции h (см. таблицу 6).

	Утверждения	Цели	f(b)
g0		Q[b,z]	z
h	if $x <_{wf} b$ then Q[x,f(x)]		

Таблица 6. Добавление гипотезы индукции.

Варианты гипотез индукции получаются при выборе конкретного wf-отношения $<_{wf}$ (well founded). Отношение является wf-отношением, если в рассматриваемой теории нельзя построить бесконечно убывающую последовательность x_1, x_2, x_3 ... , такую что $x_2 <_{wf} x_1$, $x_3 <_{wf} x_2$, ... В области лисповских списков примером такого отношения может служить отношение между «хвостом» списка (tail) и самим списком. Для неотрицательных целых чисел в качестве wf-отношения можно рассматривать обычное отношение <.

После добавления в таблицу гипотезы индукции в таблице появляется терм, содержащий обращение к синтезируемой функции, таким образом появляется рекурсивное обращение. Фактически на этом этапе требуется доказать шаг индукции: установить истинность логического следствия гипотеза индукции → заключение, получив при этом в выходной колонке терм, содержащий рекурсивную ветвь функции. Наличие правила индукции позволяет строить рекурсивные программы.

Построение пути доказательства с помощью волновых правил

Описанные дедуктивные правила позволяют синтезировать все базовые конструкции функциональной программы и сформировать структуру программы динамически, в процессе синтеза, что является преимуществом метода. Однако на каждом шаге синтеза могут оказаться применимыми несколько дедуктивных правил, что порождает огромный перебор вариантов даже для синтеза простых программ. Задачи, решения которых могут быть построены за 20-30 шагов (шагом будем считать одно применение правила к двум строкам таблицы), требуют просмотра более 30000 вариантов. Для более сложных примеров автоматический синтез с помощью метода дедуктивных таблиц оказывается неприемлим на практике, поэтому необходимы дополнительные эвристики, позволяющие сократить перебор. В

качестве такой эвристики предлагается использовать волновые правила (rippling) [7].

В процессе построения рекурсивной функции возникает доказательство по индукции, при котором необходимо показать, что из гипотезы индукции следует заключение. На практике очень часто гипотеза и заключение индукции синтаксически похожи. Этот факт и предлагается использовать: выделяются общие части (основа) гипотезы и заключения индукции, отмечаются различия и для доказательства проводятся только те преобразования, которые не меняют основу, а различия уменьшают. Для спецификации (1) шаг индукции выглядит следующим образом:

$$(b-1\geq 0) \text{ and } ((\mathrm{sqrt}(b-1))^2 \leq b-1) \text{ and } (b-1 < (\mathrm{sqrt}(b-1)+1)^2) \rightarrow$$

$$(\underline{\lfloor b-1 \underline{+1} \rfloor} \geq 0) \text{ and } (z^2 \leq \underline{\lfloor b-1 \underline{+1} \rfloor}) \text{ and } (\underline{\lfloor b-1 \underline{+1} \rfloor} < (z+1)^2) \qquad (3)$$

Здесь подчеркнуты фрагменты выражения, составляющие различия между гипотезой и заключением. Такие фрагменты называются волновыми фронтами. В формуле (3) b является константой, так как это входной параметр функции, а z – переменная, которая может быть сопоставлена с произвольным термом, поэтому она не включена в волновой фронт. Волновой фронт может содержать только терм целиком, поэтому общие фрагменты гипотезы и заключения, оказавшиеся внутри волнового фронта образуют волновую дыру и считаются входящими в основу. Например, в выражении $(\underline{\lfloor b-1 \underline{+1} \rfloor} \geq 0)$ основой считается выражение $b-1 \geq 0$.

Преобразование заключения индукции к гипотезе осуществляется с помощью волновых правил – правил переписывания, в которых выделены фронты волн и основы. Эти правила формируются из аксиом, которые описывают свойства объектов, участвующих в доказательстве. Расстановка волновых фронтов производится с помощью алгоритма унификации различий, описанного в [6]. Идея алгоритма состоит в том, что выражения просматриваются слева направо и те их фрагменты, которые не удается унифицировать, объявляются волновым фронтом, остальные считаются входящими в основу. Например, из аксиомы $x = y \rightarrow x+c=y+c$ может быть получено правило переписывания

$$\underline{\lfloor x \underline{+c} \rfloor} = \underline{\lfloor y \underline{+c} \rfloor} \Rightarrow x=y \qquad (4)$$

Здесь и далее символом $\rightarrow$ будет обозначаться логическое следствие, а символ $\Rightarrow$ будет обозначать переписывание. Заметим, что при переписывании доказательство ведется в "обратном" направлении, от заключения индукции к гипотезе, поэтому направление переписывания оказывается противоположным по отношению к логическому следствию в аксиоме.

После расстановки волновых фронтов должны выполняться два условия: основы левой и правой части нового правила переписывания

совпадают, а волновые фронты перемещаются так, чтобы на каждом шаге уменьшить различия. Существует три основных способа перемещения волновых фронтов: перемещение на более высокий уровень выражения (если его представить в виде дерева), перемещение в направлении к свободной переменной, либо волновые фронты, присутствующие в левой части правила переписывания не входят в его правую часть, как, например, в правиле (4). Применение волновых правил, перемещающих волновые фронты первым способом, позволяет получить внутри волнового фронта целиком пример гипотезы индукции и использовать ее в доказательстве. Во втором случае свободная переменная, будучи окруженной волновым фронтом, заменяется на новую переменную, поглощая таким образом волновой фронт и уменьшая различие между гипотезой индукции и заключением. В третьей ситуации, при исчезновении волнового фронта, также происходит уменьшение различия.

Применение волновых правил имеет два существенных преимущества по сравнению с использованием обычных правил переписывания. Во-первых, волновые правила преобразуют только те подвыражения, которые содержат различия, основа при таком переписывании не изменяется. Во-вторых, так как на каждом шаге применения правила происходит уменьшение различий, процесс доказательства обязательно завершается.

При проведении синтеза в дедуктивной таблице волновые правила применяются следующим образом. Для всех известных аксиом формируются волновые правила. Эти правила достаточно построить один раз, в момент появления аксиомы в системе. Затем доказательство шага индукции проводится с помощью известных волновых правил, а номера соответствующих им аксиом сохраняются как путь доказательства. После этого проводится синтез программы в дедуктивной таблице в соответствии с найденным путем. То есть при применении дедуктивных правил последовательно участвуют аксиомы из найденного пути, а остальные варианты применения правил не рассматриваются. Результатом этого этапа синтеза является построение рекурсивной ветви функции. Применение правила резолюции для строк таблицы, соответствующих рекурсивным и нерекурсивным ветвям функции, завершает синтез.

Заключение

В работе рассмотрен подход к автоматизации синтеза функциональных программ на основе метода дедуктивных таблиц. Метод дедуктивных таблиц применялся для синтеза различных

программ "вручную", например, была показана возможность синтеза с помощью этого метода некоторых алгоритмов сортировки [12] и алгоритма унификации [11]. Автоматизированный синтез с помощью метода дедуктивных таблиц был реализован в [8] и [5]: выполнение конкретного дедуктивного правила осуществляла система, а выбор правила, которое будет применено, на каждом шаге доказательства делал пользователь системы, либо явно указывая это правило [8], либо написав заранее тактику доказательства для конкретной задачи [5]. Для проведения синтеза автоматически были необходимы дополнительные эвристики, позволяющие определить порядок применения дедуктивных правил.

Волновые правила были разработаны для построения доказательств [7]. Их применение для синтеза программ осложняется тем, что при синтезе объект, доказательство существования которого проводится, не известен во время доказательства и его свойства не могут быть описаны с помощью аксиом, на основе которых формируются волновые правила. В системах, применяющих волновые правила для автоматизированного синтеза, использовались шаблоны, задающие структуру программы, подходящий вариант выбирался и конкретизировался в процессе доказательства [9]. Класс программ, которые можно было синтезировать с помощью таких систем, был ограничен библиотекой заданных шаблонов программ. Однако использование волновых правил позволяло существенно сократить перебор вариантов при проведении доказательства. Это преимущество было использовано при синтезе программ с помощью метода дедуктивных таблиц. Предложенный подход был реализован в системе автоматического синтеза программ АЛИСА [2]. Некоторые результаты синтеза представлены в таблице 7.

спецификации	количество вариантов применения правил		
	полный перебор	с волновыми правилами	необхо димое
<div(i,j),mod(i,j)><=find <y,z> such that if (i≥ 0) ∧ (j>0) then (i=y*j+z) ∧ (z ≥ 0) ∧ (z<j) *вычисление результата деления нацело и остатка*	30300	149	20
<front(a),last(a)><= find <h,t> such that if ¬(a =()) then (tail(t)=()) ∧ (a=h ° t)	7200	28	15

разделение списка на "начало" и последний элемент			
<sqrt(b)><=find <z> such that if b≥ 0 then (z^2<=b) and (b<$(z+1)^2$) and (z ≥ 0) *целочисленный квадратный корень*	31000	76	19

Таблица 7. Некоторые результаты синтеза программ в системе АЛИСА

В работе [9] отмечалось, что система синтеза, использующая волновые правила, не синтезировала функции разделения списков на части. Одна из таких задач была решена системой АЛИСА (см. таблицу 7, пример <front(a),last(a)>). Помимо преимуществ предложенного подхода следует отметить, что в настоящий момент система не анализирует эффективность синтезированной программы, то есть поиск вариантов продолжается до тех пор, пока какое-либо решение не найдено. Построение более эффективной программы эквивалентной данной (трансформационный синтез [1]) может быть рассмотрено как самостоятельная задача. Использование волновых правил для поиска пути доказательства в дедуктивной таблице позволило сократить перебор вариантов на два порядка при автоматическом синтезе простых программ и сделать возможным синтез более сложных.

Литература

1. Большакова Е.И., Мальковский М.Г. Автоматический синтез программ. Издательство Московского университета, 1987
2. Корухова Ю.С., Пильщиков В.Н. Система дедуктивного синтеза программ // Искусственный интеллект, Наука і освіта, Донецк, №2, 2002: с. 451-459.
3. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М., Наука, 1983
4. Armando A., Smaill A., Green I. Automatic Synthesis of Recursive Programs: The Proof-Planning Paradigm // Automated Software Engineering, 6(4), 1999: 329-356
5. Ayari A., Basin D. A Higher-Order Interpretation of Deductive Tableau // Journal of Symbolic Computation, 31(5), 2001: 487-520
6. Basin D., Walsh T. Difference Unification // Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1993: 116-122

7. Bundy A., Basin D., Hutter D., Ireland A. Rippling: Meta-level Guidance for Mathematical Reasoning. Cambridge University Press, 2005

8. Burback R. et al. Using the Deductive Tableau System // Macintosh Educational Software Collection, Chariot Software Group, 1990

9. Kraan I., Basin D., Bundy A. Middle-Out Reasoning for Synthesis and Induction // Journal of Symbolic Computation, 16(1-2), 1996: 113-145

10. Manna Z., Waldinger R. Fundamentals of Deductive Program Synthesis // IEEE Transactions on Software Engineering, 18(8), 1992: 674-704

11.Nardi, D. Formal Synthesis of a Unification Algorithm by the Deductive-Tableau Method // Journal of Logic Programming , 7,1989: 1-43

Traugott, J. Deductive Synthesis of Sorting Programs // Journal of Symbolic Computation, vol.7, 1989: 533-572

Попова Е.А.

Разработка адаптивной системы машинного обучения на основе аппарата деревьев решений

Введение.

В последнее время проблеме адаптивного машинного обучения посвящается много работ, обзор некоторых можно найти в [1]. Это связано с приложениями в области создания интелектуальных роботов и навигационных систем [2],[3], систем автоматизированного анализа данных в реальном времени [3], [4],[5], в приложениях систем модели данных (Data Mining) [6]. В случаях дискретных данных, имеющих различный смысл, и пропущенные данные наиболее успешное обучение осуществляется на основе деревьев решений [7]. Если обучающая база данных состоит из строк признаков снабженных меткой класса, то деревья решений позволяют классифицировать данные, вычислить порог классификации и вероятность успешной классификации. При поступлении в компьютерную систему новых данных ее поведение должно измениться, т.е.она должна адаптироваться к новым данным. На языке обучения это означает, что требуется переобучение системы. Методы переобучения интенсивно разрабатываются в [8], [9],[10]. Одним из важных аспектов адаптивного обучения является хранение в памяти прошлой информации о поведении системы и новой информации, поступающей с сенсоров. Для систем реального времени, работающих с огромными списками данных, требуется разработка методов хранения необходимой информации о прошлом поведении системы и методов быстрой классификации новых данных.

В настоящей работе делается акцент на двух аспектах проблемы. Первый связан с тем, что обученная на примерах система должна самостоятельно принять решение о появлении новых данных, т.е. относящихся к некоторому новому классу. Второй аспект связан с тем, что старые навыки поведения не должны быть потеряны, а должны быть сохранены в памяти в сжатой форме, доступной для реконструкции прошлого поведения. В работе предлагается алгоритм адаптации, основанный на деревьях решений. Основное преимущество выбранного метода состоит в том, что он позволяет хранить информацию о прошлом поведении системы в сжатом виде последовательности принимаемых решений в ответ на данные сенсоров. Вероятности же переходов, хранимые в памяти, позволяют следить за частотой появления принципиально новых, плохо классифицируемых (по старому) примеров и если новые примеры поступают часто (выше некоторого порога по вероятности), то принимать решение о

дообучении системы. В этом случае считается, что новые данные относятся к новому классу и для классификации строится новое дерево.

1. Алгоритм обучения системы на тренировочном наборе данных. Вычисляемые и хранимые признаки поведения.

Алгоритм обучения на тренировочном наборе данных соответствует известному алгоритму С4.5 Р.Куинлена [11]

Пусть нам задано некоторое обучающее множество Т, содержащее образы, каждый из которых характеризуется М признаками (атрибутами независимые переменные). Пусть через $C_1,C_2,...C_k$ обозначены метки классов (зависимая переменная, целевая переменная). Обозначим вектор признаков через

$$\mathbf{X}^p = \{ x_1,x_2,..x_i,..x_M \mid C_k \}^p$$

где р- номер примера, к - номер класса, x_i^p i -ый признак р-го примера. Узлом дерева является узел проверки, а листом - узел решения. Условие в узле будем называть тестом. Множество Т содержит примеры, относящиеся к разным классам. Цель построения дерева - разбить множество Т на некоторые подмножества T_i которые будут состоять в основном их примеров относящихся к одному классу.

Вначале выбирается наиболее значимый атрибут для разбиения. Выбранный атрибут должен разбить множество так, чтобы получаемые подмножества состояли из объектов, принадлежащих к одному классу максимально приближены к этому, т.е. количество образов-примесей в каждом из этих множеств было как можно меньше.

Мы используем теоретико-информационный критерий для выбора наиболее значимого атрибута. Для различных классов и различных примеров каждый признак x^i упорядочивается в порядке возрастания величины. Вычисляются средние значения соседних величин признака:

$$x_i^{\bullet} = (x_i + x_{i+1})/2$$

Эти значения $x_i^{\bullet}$ являются границами разбиения на подмножества T_i по признаку x_i, i=1,...M-1. Пусть $N_{k,i}$ число примеров из T_i относящихся к одному классу C_k . Тогда вероятность того, что случайно выбранный пример из множества T_i будет принадлежать к классу C_k равна $P_{k,i}=N_{k,i}/N_i$, где N_i есть полное число образов в множестве T_i. Информация, содержащаяся в выборе $x_{i,k}$ равна $I_{k,i}=-\log_2 P_{k,i}$, а средняя информация, приходящаяся на такой выбор, есть энтропия:

$$H(x_i,T_i) = -\sum_{k=1}^{N} \{ P_{k,i} \cdot \log_2 P_{k,i} \} \cdot \frac{N_i}{N} \qquad (1)$$

Обозначим через $H(x_i,T)$ энтропию полного множества T при анализе по признаку x_i. Тогда критерием выбора признака будет минимальное значение

$$G(X) = H(x_i,T) - H(x_i,T_i) \qquad (2)$$

по всем признакам. Итак, множества $T_1,T_2,...T_n$ получены при разбиении исходного множества T по признаку x_i Выбирается атрибут, дающий максимальное значение по критерию (1)

Пусть выбран признак x^i для всех примеров. Теперь разбиваем множество T на два подмножества путем вычисления порога по выбранному признаку. Для этого вычисляется пороговое значение признака $x_{i,th}$ из набора x_i^* из условия максимизации энтропии (1), т.е. $\max_{x_i^*} H(x_i^*,T_i)$. Исходное множество примеров разбивается на два подмножества $T_{x_i}^L$ и $T_{x_i}^R$ по критерию:

$$x_i < x_{i,th}, \qquad (3)$$

множество $T_{x_i}^L$ и

$$x_i > x_{i,th}, \qquad (4)$$

множество $T_{x_i}^R$.

Далее рассматриваем множество $T_{x_i}^L$. Делаем проверку, если в $T_{x_i}^L$ находятся примеры одного класса, то получено решение и алгоритм останавливается. Если в $T_{x_i}^L$ присутствуют примеры других классов, то продолжаем разбиение каждого из полученных множеств $T_{x_i}^L$ и $T_{x_i}^R$. Пусть анализируем дальше $T_{x_i}^L$, в котором выбираем наиболее информативный признак для следующего разбиения, используя указанную выше процедуру. Причем признак x_i исключается из соревнования. Пусть выбран признак x_j для следующего разбиения. Из условия максимизации энтропии вычисляется порог $x_{j,th}$ по признаку x_j и множество $T_{x_i}^L$ оказывается разбитым на два : T^{LL} и T^{LR} по признаку x_j. Если полного разделения образов не получилось, то разбивается множество $T_{x_i}^R$ на два множества T^{RL} и T^{RR}. Таким образом, используя два признака, мы разбили исходное множество на 4 подмножества T^{LL}, T^{LR}, T^{RL} и T^{RR}. Далее процедура повторяется. В данной работе мы используем следующий критерий остановки алгоритма как ограничение глубины дерева : когда проведено разделение множества примеров по всем признакам алгоритм останавливается. После остановки алгоритма принимается решение об отнесении образов к классу. Считается вероятность принадлежности примера (образа) к классу

Классификация новых данных происходит с помощью дерева и выдается ошибка классификации. Пусть N_k число примеров из T^{LR} относящихся к одному классу C_k. Тогда вероятность того, что случайно выбранный пример из множества T^{LR} будет принадлежать к классу C_k равна $P_k=N_k/N_{LR}$, где N_{LR} есть полное число образов в множестве T^{LR}. Если $P_k > P_{th}$, где P_{th} заранее выбранное пороговое значение вероятности классификации. Например, если число признаков равно двум, то дерево содержит три узла (пороговые значения для разбиения) и ссылку на потомков. Если остановки не произошло по ходу разбиения, то дерево содержит четыре листа, как решения принятые в четырех подмножествах множества образов. Эти пороги x_{th}, указатели и листы (P_{th}) должны содержаться в памяти для классификации образов.

Основа программы адаптивного обучения тестировалась на известной базе данных [12] [13] и результаты сравнивались с результатом работы C4.5. Результаты сравнения приведены в таблице 1, где показана ошибка классификации.

Source	DataSet	Dimensions	C4.5	IRID3	Adaptive System
			Accuracy Size(Rules)	Accuracy Size(Rules)	Accuracy Size(Rules)
(2)mbase	Iris	150x5	98.0 5	98.0 4	98.3 5
(1)car/	Cars	1728x6	89.0 15	86.0 19	87.0 15
(1)wine/	Wine	170x13	95.0 6	91.0 12	95.6 6
(1)glass/	Glass	214x10	76.2 12	65.4 30	72.9 13

В таблице приведен процент правильно распознанных примеров. Таблица показывает, что результаты работы программы адаптации на одинаковых тестовых базах данных не уступает известным программам.

2. Алгоритм классификации и адаптации к новым данным.

Предположим, что имеется некоторый временной процесс поступления новых данных в систему. Цель системы классифицировать новые данные, т.е. отнести их к одному из известных классов или к новому классу.

На рис.1 показана структура адаптивной системы

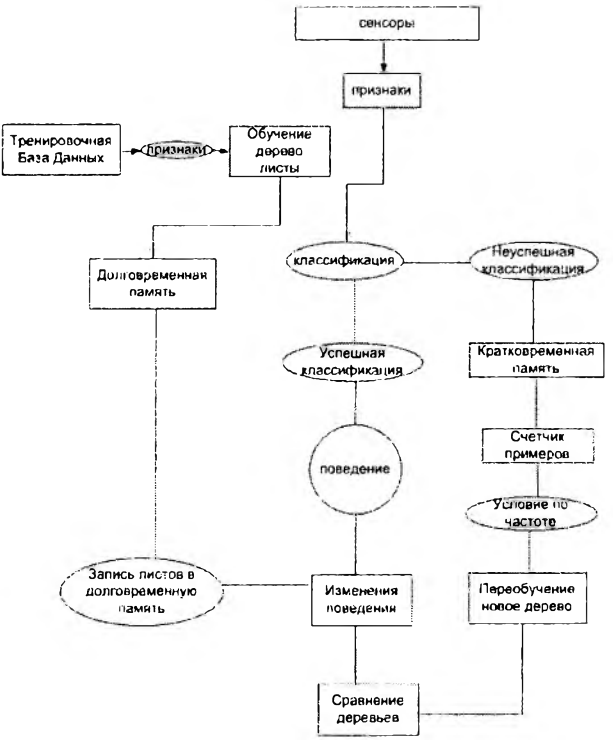


Рисунок 1: Структура адаптивной системы

На каждом шаге по времени, в процессе классификации вычисляется ошибка классификации. Если ошибка меньше порога, устанавливаемого обученным деревом, то принимается решение об отнесении новых данных к одному из известных классов и поведение системы происходит по старым правилам. Если ошибка больше установленного порога, то новый вектор признаков заносится в кратковременную память и хранится в ней определенное количество (задаваемое в алгоритме) шагов по времени. Если новые данные также показывают плохую классификацию по прошлому, то они также заносятся на определенный период в кратковременную память. Поведение системы сохраняется в течение накопления новых данных. Если частота появления новых плохо классифицируемых данных превосходит определенный задаваемый порог, то принимается решение о дообучении системы. При этом считается, что новые данные

принадлежат к одному новому классу. Дообучение состоит в построении нового дерева решений и вычислении нового порога, разделяющего новое количество классов. После этого признаки поведения системы с новыми данными заносятся в долговременную память. Процедура повторяется и в течении всего времени наблюдений система обладает новыми качествами классификации и возможностями нового поведения.

Опишем теперь алгоритм адаптации. В начале основная память системы представляет собой дерево решений, построенное по тренировочной базе данных. Далее на вход системы подаются новые данные в виде набора признаков

$$\mathbf{X}^{new} = \{ x_1, x_2, .. x_i, .. x_M \parallel C_k \}^{new}$$

Дерево решений классифицирует его таким образом, что если мы пришли к листу 'noclass' (т.е. дерево не отнесло данные ни к какому классу),то по алгоритму построения мы должны выбрать наиболее часто встречающийся класс у родительского узла этого дерева и листу присвоить этот класс. При аккредитации так и происходит, только при этом запоминается количество таких примеров. Для каждого примера запоминается его ошибка классификации, вычисляемая по следующему правилу:

$$\varepsilon = \frac{\sum_i (x_i - node_{avi})^2}{\sum_i (node_{thi} - node_{avi})^2} + P(w)$$

где P(ω)- вероятность классификации, которая изменяется при отсечении ветвей и объединении узлов, x_i, $node_{avi}$, $node_{thi}$-значение соответствующего атрибута входного вектора на пути от корня к листу, среднее отклонение значений атрибута от порогового по всему обучающему множеству и пороговое значение для атрибута в данном узле. Устанавливается порог допустимого отношения

$$\varepsilon_c = \frac{N_c}{N_{tot}} \qquad (6)$$

при определенном количестве N_c уже поступивших на вход векторов образов. При выходе за это пороговое значение система начинает свое переобучение. Процесс переобучения состоит в извлечении правил из дерева, хранящегося в основной памяти, которые в свою очередь записываются в долговременную память. При этом уже хранятся только условия для принятого решения в виде узла и ассоциированных с ним гипотез . Дерево стирается из памяти и строится новое на основе базы данных поступивших на вход системы за этот период. Здесь встает вопрос о том, каким классам все-таки

принадлежат те вектора признаков, которые не были распознаны деревом? Предлагаемый алгоритм выбора класса таков: для каждого входного вектора существует ошибка распознавания, определенная в (5). Тогда для каждого такого вектора анализируются все отклонения от пороговых значений в узлах пути от корня до класса и выбирается наименьшее значение, т.е. вероятность того, что была выбрана другая ветвь при сравнении $x_i < node_{th}$ будет наибольшая. Далее вычисляется отношение

$$R = \frac{(node_{th} - x_i)^2}{(node_{th} - \max/\min_{val})^2}$$

Если же это отношение меньше установленного изначально порогового значения, тогда вычисляется ошибка распознавания при выборе другого пути и сравнивается с ранее полученной ошибкой. В зависимости от того какая из них больше и будет принято решение о принадлежности вектора к классу, изначально выбранному, или же тому, который был получен в результате другого пути по дереву. Иначе, просто присваивается класс, полученный при аккредитации.

Итак, в долговременной памяти находится набор узлов в ассоциированных с ним гипотез. Такое представление легко для понимания и не содержит лишней информации, а именно, тех узлов дерева которые могут не участвовать при классификации, но содержатся на пути от корня к листу. После обновления основной памяти (построения нового дерева, извлечения правил) происходит дальнейшая аккредитация, только с учетом хранимой в долговременной памяти информации. Если дерево основной памяти не может классифицировать вектор, тогда идет сразу же обращение в узлы долговременной памяти. Если же не удается классифицировать пример, то действуем по алгоритму описанному выше.

3. Эксперименты с задачей о навигации робота.

В качестве модельной задачи была выбрана задача управления мобильным роботом. У робота имеется 5 датчиков, отслеживающих состояние среды и расположенных следующим образом: один слева, справа, сзади и два датчика впереди. Описание такого робота можно найти в [14]. В каждый момент времени с датчиков поступают данные о нахождении другого объекта в окружении (значения датчиков меняются от 0-300). В [14] задача ставится как объезд роботом всех препятствий. В отличие от [14] мы предполагаем, что робот имеет возможность подпрыгивать на определенную высоту. Теперь новая трехмерная задача навигации состоит в том, что робот должен не просто объезжать препятствие, попавшееся на пути, но при необходимости перепрыгивать

через него. Множество решений, которые может принять робот таково: поворот налево (turn left), поворот направо(turn right),вращение, поворот на угол(rotate), прыжок(jump), движение вперед(forward).Обозначим это допустимое множество через M_d. Так как робот должен не просто уходить от препятствия на пути, а огибать его, то кроме значений датчиков нами был добавлен угол поворота робота, относительно начального положения при котором он двигался вперед. Также, если высота препятствия допустима (определяется значением датчика), то робот может перепрыгнуть через него. Таким образом, атрибутами базы данных являются датчики робота (их всего 5), каждый из которых определяет расстояние до объекта, а классами исходной базы данных будут решения, которые может принять робот после обработки информации с датчиков (мн-во M_d).

Example	x_0	x_1	x_2	x_3	x_4	x_5	Decision
ex_n	0- 300	0-300	0-300	0-300	0-300	0,90l,90r,180	M_d

Формирование начальной базы данных происходило уже на знаниях тех пороговых значений датчиков, которые говорили о минимально допустимом расстоянии до препятствия, при котором робот мог двигаться по направлению к препятствию. Однако обучение происходило на определенных интервалах значений датчиков из допустимых,

$$x_{i,min} - x_{i,max}$$

где $x_{i,min}$ и $x_{i,max}$ минимальное и максимальное значение величины на сенсоре. С целью последующего дообучения робота, т.е. чтобы возникали ситуации, когда поведение робота не было определено базой данных (в этом случае путь от корня к листу приводил к неопределенному классу или к листу с большей ошибкой классификации). Обозначим возможные пороговые значения, отвечающие за минимальное расстояние до объекта, при котором возможно движение вперед как $x_{0,th}, x_{1,th}, x_{2,th}, x_{3,th}, x_{4,th}$. Тогда обучающую выборку будем формировать по такому принципу : случайный выбор 1-го признака из диапазона ($x_{i,min}, x_{i,th}$), для всех остальных 4-х атрибутов, отвечающих за данные с датчиков, случайное значение берется из диапазона ($x_{i,min}, x_{i,max}$), где $x_{i,min}, x_{i,max}$- минимальное и соответственно максимальное значение данного атрибута; значение нечислового атрибута (в нашем случае это поворот относительного начального положения движения:0,90l,90r,180) и решение из набора допустимых решений M_d. Таким образом, роботу задаются условия для принятия решения в виде записей в обучающие вектора случайных значений из

указанных интервалов. В результате поведение робота было определено для всех возможных ситуаций, кроме следующих:

$$x_0 > x_{0,th}, x_1 < x_{1,th}, x_2 < x_{2,th}, x_3 > x_{3,th}, x_{4,min} < x_4 < x_{4,max}, 90l \quad (8)$$

$$x_0 > x_{0,th}, x_1 < x_{1,th}, x_2 < x_{2,th}, x_3 > x_{3,th}, x_{4,min} < x_4 < x_{4,max}, 90r \quad (9)$$

$$x_0 > x_{0,th}, x_1 < x_{1,th}, x_2 < x_{2,th}, x_3 > x_{3,th}, x_{4,min} < x_4 < x_{4,max}, 180 \quad (10)$$

$$x_0 > x_{0,th}, x_1 < x_{1,th}, x_2 < x_{2,th}, x_3 > x_{3,th}, x_{4,min} < x_4 < x_{4,max}, 0 \quad (11)$$

В качестве первоначальной обучающей выборки было взято 240 случайных векторов, значения которых генерировались случайным образом в определенных выше интервалах (память робота в начале работы системы). Далее последовательно на вход подавались векторы, неформально разбитые на 2 группы из M и P векторов таким образом, чтобы было возможно проследить за адаптацией принятого решения для неопределенных в обучающей выборке ситуаций, а именно: первые M векторов группы формировались случайным образом, т. е. из допустимых интервалов значений датчиков, эти векторы могли представлять неопределенное решение. Следует напомнить, что под неопределенным решением, мы понимаем классифицированный с большой ошибкой вектор, либо отнесенный деревом решений к неопределенному классу, а остальные P векторов случайным образом генерировались из выше написанных неопределенных в обучающей базе данных интервалов. Таким образом, входные векторы представляли собой множество

$$x_1, x_2, \ldots, x_M, x_{M+1}, x_{M+2}, \ldots, x_K, x_{K+1}, x_{K+2}, .., X_{K+Kp}, \ldots x_P \quad (12)$$

где вектор $X_{K+Kp}, \ldots X_P$ формируется из случайных значений ранее неопределенного интервала, а $x_{M+1}, x_{M+2}, \ldots, x_K$; $x_{K+1}, x_{K+2}, ..$ случайным образом генерируемые значения из интервалов, которые уже представлялись при аккредитации.

На рис.2 показана структура адаптивной системы показана зависимость ошибки распознавания системы от числа новых примеров подаваемых на вход адаптивной системы: (а) На этом рисунке можно проследить как меняется ошибка распознавания системы(т.е. число тех входных векторов, которые система либо относит к "noclass", либо распознает с очень плохой ошибкой) в зависимости от количества примеров, которые подавались на вход системы в порядке, описанном

выше. На графике можно проследить всплеск, который говорит о появлении нераспознанных системой входной векторов и далее постепенную адаптацию к новому решению. Как только ошибка превосходит заданный порог, система начинает свое переобучение, на графике эти моменты обозначены вертикальной прямой. После переобучсния можно видеть постепенное затухание графика ошибки, которое и свидетельствует об адаптации к новому решению 1. (b) Можно проследить ту же зависимость только к другому решению (т.е. на вход подавались значения векторов из других неопределенных интервалов),адаптация к решению 2

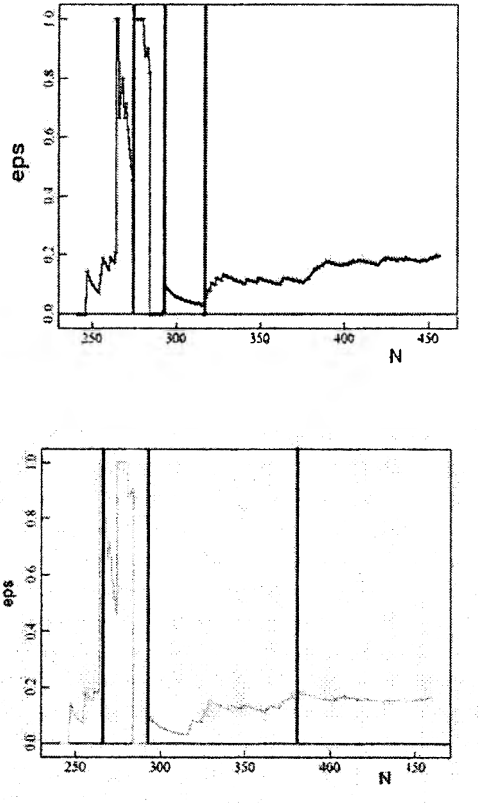


Рисунок 2: Зависимость ошибки распознавания от числа новых примеров подаваемых на вход адаптивной системы: (a)адаптация к решению 1 (b) адаптация к решению 2 .

На рис.3 зависимость ошибки распознавания от числа новых примеров при адаптации к нескольким решениям (1,2,3,4).Это говорит о

том, что все входное множество векторов, подаваемых системой для аккредитации разбито на несколько групп, каждая из которых отвечает за новое решение. Таким образом система постепенно адаптируется сразу к нескольким решениям (значения генерятся одновременно из нескольких неопределенных при обучении интервалов значений датчиков). Как только ошибка превосходит пороговое значения, система переобучается (всплески на рисунке).

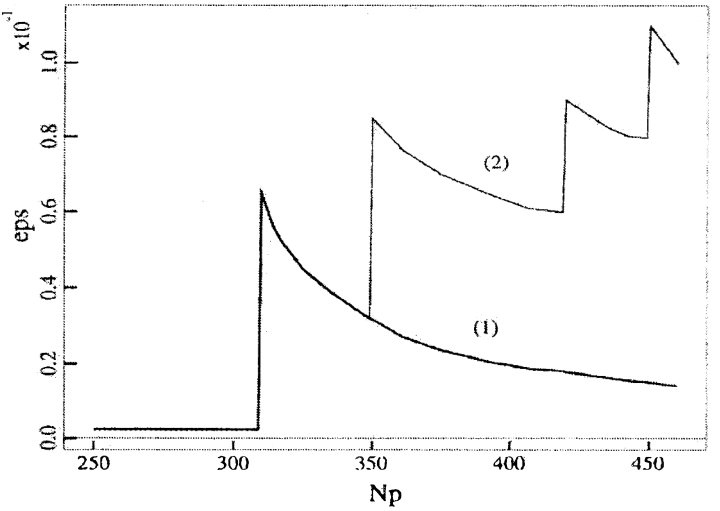


Рисунок 3: Зависимость ошибки распознавания от числа новых примеров при адаптации к нескольким решениям (1,2,3,4)

4. Заключение.

В работе предложен алгоритм адаптации системы к новым условиям на основе деревьев решений. Показано, что переход от прошлого поведения к будущему может быть эффективно построен на основе выбранных признаков поведения хранимых в долговременной и кратковременной памяти системы. Основным является то, что эволюция системы происходит на основе хранимых признаков - данные-решение которые позволяют эффективно принимать решение об адаптации. Приведенные эксперименты показывают эффективность адаптации с помощью аппарата деревьев решений.

Работа выполнена при поддержке гранта РФФИ-

Список литературы

1. R.Mitra,J.Basak Methods of case adaptation: A survey:Research Articles/ International Journal of Intelligent Systems, V.20,Issue 6 (June 2005) pp.627-645 , 2005

2. Robotics: control,sensing, vision, and intellegence Robot inellegence and task planning .*BOOK* .-2004.-640 c.

3. Basak J. Online Adaptive Decision Trees .- *Neural Computation* ,2004, V.16, N9, pp. 1959-1981.

4. Pagallo Adaptive Decision Tree Algorithms for learning from examples// .*in PhD Thesis, University of California at Santa Cruz, CA,1990*

5. J.F.D.Addison, S.Wermter/ Methods for integrating memory into Neural Networks In Condition Monitoring *6th IASTED Int.Conf.Artificial Intelligence and soft Computing (ASC-2002)* July 17-19 , 2002, Banff, Alberta, Canada

6. Data Mning and Knowledge Discovery/ Editors: Yike Guo,Robert Grossman, V.3,N3,1999,pp.237-261

7. J.Gama, Functional Trees, . *Machine Learning, Kluwer Academic Press* , pp 219-250, Vol.55(3) 2004

8. H. Pistori, J.J.Neto Desion Tree Induction using Adaptive FSA .*clei electronic jornal* .-2004 Vol.6;Number1:Paper4

9. Adaptive Lerning Sistem - ALES Laboratory of Artificial Intelligence and Computer scince, University of Porto, Porto, Portugal .*www.liacc.up.pt/ jgama/ales/ales.html* .-2005..

10.N.Indurkhya and S.M.Weiss Estimating Performance for Voted Desion Trees IBM Reseach Division Technical Report RC-21199 In Intelligent Data Analysis (IDA), 1999 .*IBM* .-1999.

11.J.Ross Quinlan C4.5 : Programs for Machine learning *Kaufmann Publishers*, 1993

12.ftp:// ftp.ics.uci.edu/pub/machine-learning-databases/

13.http://www.quantlet.org/mdbase/

14.http://www.ics.uci.edu/mlearn/MLRepository.html University of California, Department of Information and Computer Science

Малышко В. В., Морозов В. А.

Об одном подходе к решению задачи сопоставления с образцом на основе алгоритма Rete[1]

Введение

В системах искусственного интеллекта, основанных на знаниях, важной компонентой является логический вывод. От его эффективности зачастую зависят возможности системы по решению поставленных перед нею задач. Некоторые реализации логического вывода (например, в продукционных системах) основаны на сопоставлении с образцом.

В настоящее время практическое значение имеют несколько алгоритмов сопоставления, один из которых – алгоритм Rete [1]. В данной статье предлагаются способы модификации этого алгоритма с целью повышения скорости сопоставления путём исключения избыточных проверок и сокращения среднего количества проверок, выполняемых в процессе сопоставления.

Основные понятия продукционных систем

Продукционная система (см. [2]) состоит из множества продукций, интерпретатора и рабочей памяти. Данные в рабочей памяти хранятся в виде отдельных элементов, характеризуемых своим классом и набором значений атрибутов этого класса. Продукция состоит из двух частей: левой части – антецедента – и правой – консеквента. Антецедент описывает условия на содержимое рабочей памяти, при выполнении которых продукция применима, и представляет собой набор шаблонов элементов рабочей памяти. Шаблон элемента содержит указание на класс элемента и ограничения на атрибуты. Основные виды ограничений – это сравнения с константами и с переменными. Переменные могут принимать любые значения и используются для задания связей между различными атрибутами. Упорядоченный набор элементов рабочей памяти удовлетворяет антецеденту, если выполнены следующие условия: каждый элемент в наборе удовлетворяет соответствующему ему по порядку шаблону и переменные с одинаковыми именами во всех шаблонах связаны с одинаковыми значениями. Пара (<продукция>, <упорядоченный набор элементов

[1] Данная работа поддержана грантом РФФИ 05-01-00948.

рабочей памяти, удовлетворяющих её левой части>) называется активацией продукции.

В консеквенте продукции указываются действия, которые предписывает выполнить данная продукция в случае удовлетворения её антецедента, такие как добавление и удаление элементов рабочей памяти, завершение работы системы и т. п.

Цикл работы интерпретатора продукционной системы состоит из следующих шагов:

1. *Сопоставление.* На этом шаге интерпретатор определяет множество активаций продукций элементами, находящимися в текущий момент в рабочей памяти (так называемое конфликтное множество).
2. *Разрешение конфликтов.* Из конфликтного множества выбирается одна активация; если множество пусто, интерпретатор завершает свою работу.
3. *Выполнение действий.* Интерпретатор производит действия, указанные в правой части продукции, активация которой была выбрана. Если среди выполненных действий не было завершения работы системы, происходит переход к шагу 1.

Из всех действий интерпретатора наиболее сложным и ресурсоёмким является сопоставление. Рассмотрим его более подробно.

Алгоритм сопоставления Rete

Алгоритм Rete [1], разработанный Чарльзом Форджи, не требует итерации по множествам продукций и элементов рабочей памяти, что обеспечивает достаточно эффективную работу при большом количестве продукций и большом объёме рабочей памяти.

Практика использования продукционных систем показывает, что в большинстве случаев на каждой итерации работы интерпретатора изменяется лишь незначительная часть рабочей памяти. Поэтому сопоставитель, использующий алгоритм Rete, не просматривает каждый раз всю рабочую память, а принимает на вход информацию об изменениях в рабочей памяти и определяет, какие изменения необходимо произвести в конфликтном множестве, чтобы оно соответствовало новому состоянию рабочей памяти. Описания изменений рабочей памяти передаются сопоставителю в виде так называемых токенов. Токен представляет собой упорядоченную пару, первым элементом которой является тег, а вторым – упорядоченный набор элементов рабочей памяти. На самом деле, токены, подаваемые сопоставителю на вход, содержат только один элемент рабочей памяти (добавляемый или удаляемый), но данное выше более общее

определение понадобится при дальнейшем описании алгоритма Rete, поскольку токены используются и внутри сопоставителя. В большинстве реализаций алгоритма Rete используется только два тега: «+» и «–».Тег «+» означает добавление в рабочую память. Тег «–» означает удаление из рабочей памяти.

Основной компонентой сопоставителя является специальная сортирующая сеть – Rete-сеть – компилируемая по левым частям продукций. Далее описаны процессы компиляции сети и её использования при осуществлении сопоставления.

Сопоставляя элементы рабочей памяти шаблонам в левых частях продукций, сопоставитель проверяет соблюдение некоторых ограничений. Все ограничения, задаваемые в левых частях продукций, можно разделить на две категории: внутриэлементные ограничения и межэлементные ограничения. Внутриэлементные ограничения относятся к атрибутам одного элемента. Их можно разделить на два вида: нахождение значения атрибута в заданном отношении с константой и нахождение значений двух атрибутов в заданном отношении. Ограничения второго вида возникают при использовании одной переменной несколько раз в одном шаблоне элемента. Межэлементные ограничения связывают значения атрибутов различных элементов. Они возникают при использовании одной переменной в нескольких шаблонах элементов в одной продукции.

Рассмотрим следующий фрагмент левой части продукции, состоящий из двух шаблонов элементов:

<Goal Type="Simplify" Object="{N}"/>

<Expression Name="{N}" Arg1="{X}" Op="*" Arg2="{X}"/>

Здесь определены следующие внутриэлементные ограничения:

значение атрибута Type элемента класса Goal равно "Simplify",

значение атрибута Op элемента класса Expression равно "*",

значения атрибутов Arg1 и Arg2 элемента класса Expression равны,

а также межэлементное ограничение:

значение атрибута Object элемента класса Goal равно значению атрибута Name элемента класса Expression.

Компилятор Rete-сети строит сеть, соединяя между собой узлы, проверяющие соблюдение ограничений. Обрабатывая левую часть очередной продукции, компилятор начинает с внутриэлементных ограничений. Он определяет множество внутриэлементных ограничений, заданных в каждом шаблоне, (называемое α-множеством) и строит для этого шаблона линейную последовательность узлов (называемых α-узлами), выходящую из классового узла, соответствующего классу шаблона, и заканчивающуюся узлом α-памяти (см. [3]). Каждый α-узел проверяет соблюдение одного ограничения. В узле α-памяти хранятся элементы рабочей памяти,

удовлетворяющие всем ограничениям соответствующего α-множества. Закончив с внутриэлементными ограничениями, компилятор строит узлы (называемые β-узлами), проверяющие соблюдение межэлементных ограничений. У β-узлов имеется по два входа, так что они соединяют два пути Rete-сети в один. Первый β-узел соединяет выходы узлов α-памяти для первых двух шаблонов продукции, второй β-узел соединяет выход первого с выходом узла α-памяти для третьего шаблона и так далее. Каждый β-узел проверяет соблюдение всех межэлементных ограничений, определённых для соединяемых им элементов. Наконец, после β-узлов, компилятор строит специальный терминальный узел, соответствующий рассматриваемой продукции. Этот узел присоединяется к последнему β-узлу. Общая структура Rete-сети представлена на Рис. 1.

Классовые узлы являются входами в сеть. Они получают на

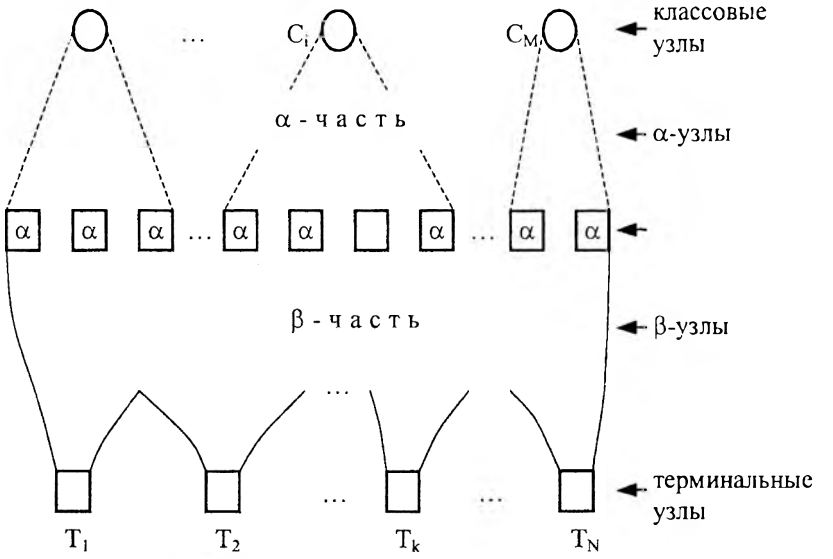


Рис. 1. Общая структура Rete-сети.

вход токены, передаваемые сопоставителю и содержащие элементы соответствующих классов, и рассылают их копии всем своим потомкам. У потомков классовых узлов – α-узлов – имеется по одному входу и одному или более выходов. Каждый α-узел проверяет соблюдение одного ограничения и посылает токены, прошедшие проверку, всем потомкам. β-узлы сравнивают токены, пришедшие на различные входы, и соединяют их в более длинные токены, если они удовлетворяют межэлементным ограничениям, определённым в левой части продукции. Проверки, производимые при проходе токенов по сети,

гарантируют, что терминальные узлы получают на вход только токены, содержащие наборы элементов, удовлетворяющие левой части соответствующей продукции. Терминальные узлы посылают вовне сопоставителя информацию о необходимых изменениях конфликтного множества.

Сопоставитель должен сохранять информацию между итерациями цикла работы интерпретатора, чтобы не просматривать каждый раз всю рабочую память. В Rete-сетях вся необходимая информация хранится в узлах α-памяти и β-узлах. В каждом таком узле хранятся копии наборов элементов рабочей памяти, содержавшихся в токенах, пришедших на вход узла. Наборы элементов хранятся до тех пор, пока не будут удалены, как описано ниже.

Тег токена указывает на то, каким образом необходимо изменить информацию, хранящуюся в узлах сети, при обработке данного токена. Терминальные узлы используют теги для определения того, необходимо ли добавить новую активацию в конфликтное множество или удалить из него уже имеющуюся там активацию. При обработке «+»-токена активация добавляется, а при обработке «–»-токена активация удаляется. Узлы α-памяти и β-узлы используют теги, чтобы определить, как им следует изменить свои памяти. Если токен имеет тег «+», соответствующий набор элементов добавляется в память, а если тег «-», то набор из памяти удаляется. Кроме того, при создании нового токена в β-узле его тег берётся равным тегу только что пришедшего в узел токена.

Задача компиляции эффективной Rete-сети

Задача компилятора Rete-сети заключается в том, чтобы по левым частям продукций построить сеть, задающую порядок проведения проверок, осуществляемых над каждым добавляемым или удаляемым из рабочей памяти элементом, для выяснения, удовлетворяет ли он ограничениям, определённым в левых частях продукций. При этом из всех допустимых сетей желательно построить сеть как можно более высокого качества.

Качество сети определяется средним временем работы интерпретатора, использующего данную сеть, на различных начальных состояниях рабочей памяти, которое, в свою очередь, зависит от среднего числа производимых операций сравнения при добавлении или удалении одного элемента рабочей памяти. Ход выполнения продукционной программы, очевидно, может сильно зависеть от начального состояния рабочей памяти, поэтому для разных наборов начальных данных оптимальными могут оказаться различные сети.

Однако поскольку оптимизация производится безотносительно конкретного начального состояния рабочей памяти, необходимо построить сеть, которая была бы «хорошей» в среднем по всем начальным данным.

Rete-сеть можно разбить на две части. Первая (назовём её α-часть) содержит внутриэлементные проверки, позволяющие для произвольного элемента каждого класса установить, в какие узлы α-памяти этот элемент должен быть помещён. Вторая часть (β-часть) проверяет соблюдение межэлементных ограничений, соединяя элементы из узлов α-памяти (см. Рис. 1).

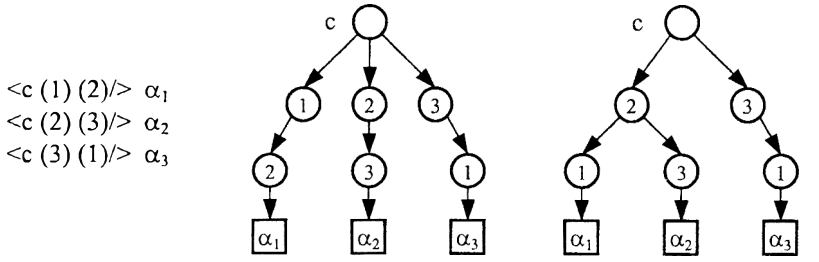
Множества классовых узлов, узлов α-памяти и терминальных узлов для каждой продукционной программы определяются однозначно. Поэтому задачу компиляции Rete-сети можно разбить на независимые подзадачи: генерацию α-части и генерацию β-части сети.

Генерация α-части сети

Как уже упоминалось выше, каждый узел α-памяти относится к одному и только одному классу. Таким образом, α-часть Rete-сети разбивается на непересекающиеся составляющие, каждая из которых связана со своим классовым узлом и может быть сгенерирована независимо от остальных.

Алгоритм генерации одной такой составляющей α-части Rete-сети (назовём её α-деревом) должен принимать на вход множество всех α-множеств, относящихся к заданному классу, и выдавать структуру, определяющую порядок выполнения проверок соблюдения в произвольном элементе данного класса ограничений из α-множеств. Показателем качества такой структуры является среднее число проверок (операций сравнения), которые будут произведены для произвольного элемента данного класса.

Рассмотрим шаблоны элементов, изображённые на Рис. 2(а). Цифрами в круглых скобках условно обозначены внутриэлементные ограничения, α_1, α_2 и α_3 – обозначения соответствующих α-множеств. Простейший способ построения α-дерева – построить для каждого узла α-памяти отдельную ветвь, не связанную с остальными ветвями, соединяющую классовый узел и узел α-памяти и содержащую проверки соответствующего α-множества в произвольном порядке. На Рис. 2(б) представлено дерево для рассматриваемого примера, построенное таким способом.



(а) Шаблоны элементов класса c (б) Простейшее дерево (в) Оптимальное дерево

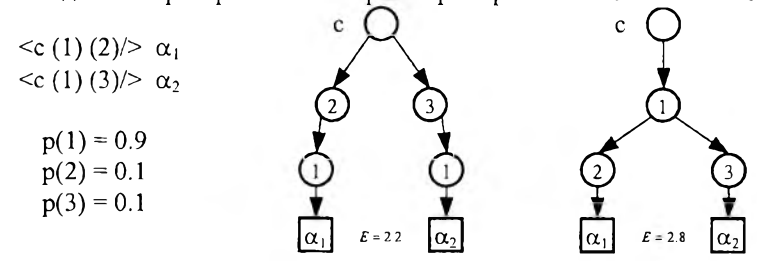
Рис. 2. Пример построения α-дерева для класса c.

Недостатки такой структуры начинают проявляться когда α-множества для класса пересекаются (а это бывает почти всегда). В этом случае одна и та же проверка, расположенная на нескольких ветвях, будет применяться к одному и тому же элементу несколько раз. Так если в примере на Рис. 2(б) в классовый узел придет токен с элементом, для которого выполнены все ограничения 1–3, то будет произведено 6 проверок, то есть каждая проверка будет произведена дважды над одним элементом.

Очевидным способом улучшения такой структуры является вынесение общих для нескольких ветвей проверок наверх и объединение нескольких узлов в один. В примере на Рис. 2 можно поменять местами на левой ветви проверки 1 и 2 и объединить узлы, выполняющие проверку 2, в один узел. Результирующая структура показана на Рис. 2(в). Легко видеть, что это дерево является также минимальным по числу узлов для данного примера. При этом избавиться от повторных проверок не удаётся: это дерево в худшем случае произведёт 5 проверок вместо 3. Итак, избыточные (повторные) проверки являются неотъемлемым свойством такого рода α-деревьев.

Для того чтобы иметь возможность вычислить основной показатель качества α-дерева – среднее число производимых проверок для произвольного элемента – необходимо ввести вероятности выполнения каждой проверки. При этом для разных проверок эти вероятности будут, вообще говоря, разными. Так, например, разумно предположить, что вероятность равенства значения некоторого атрибута фиксированной константе существенно меньше вероятности неравенства этой константе. В дальнейшем будем считать, что для каждой проверки задана фиксированная вероятность её выполнения (разумеется, приблизительная). Наиболее адекватные значения вероятностей для разных типов проверок можно подобрать экспериментально.

Как уже было сказано, общим методом улучшения структуры α-дерева является объединение одинаковых узлов на разных ветвях. Этот метод приводит к уменьшению количества узлов в дереве, но в условиях, когда у каждой проверки есть своя вероятность выполнения, далеко не всегда приводит к уменьшению среднего количества производимых проверок. Рассмотрим пример на Рис. 3. На Рис. 3(а)



(а) Шаблоны элементов и вероятности успеха проверок

(б) Дерево с минимальным средним числом проверок

(в) Дерево с минимальным числом узлов

Рис. 3. Пример, показывающий, что уменьшение количества узлов не всегда приводит к уменьшению среднего числа производимых проверок.

показаны шаблоны элементов, а также вероятности успешного выполнения каждой из трёх проверок. Дерево на Рис. 3(в) получено из дерева на Рис. 3(б) вынесением узлов, проверяющих условие (1), наверх и объединением их в один узел. Полученное дерево проверяет каждое условие только один раз (повторные проверки отсутствуют). Но если мы посчитаем среднее число производимых проверок, то для дерева на Рис. 3(б) мы получим

$$E = 1 + p(2) \cdot 1 + 1 + p(3) \cdot 1 = 2.2,$$

тогда как для дерева на Рис. 3(в) получаем

$$E = 1 + p(1) \cdot 2 = 2.8.$$

Здесь мы сталкиваемся с другой эвристикой улучшения качества α-деревьев: вынесением наверх проверок с наименьшими вероятностями успеха. Действительно, если проверка закончилась неудачно, то все проверки, лежащие ниже данной отсекаются. Эта эвристика, очевидно, очень часто противоречит эвристике объединения узлов, поскольку объединяемые узлы сначала нужно переместить наверх, в то время как они могут иметь большие вероятности успеха, чем другие узлы на тех же ветвях.

Наконец, расширим понятие избыточной проверки. До этого момента об избыточности проверки шла речь только тогда, когда одна и та же проверка производилась над одним и тем же элементом несколько

раз. Но, предположим, успешно выполнилась проверка на равенство значения атрибута некоторой константе. Из этого можно сделать вывод, что проверки на неравенство значения этого атрибута той же константе или на равенство другой константе закончатся неудачно. Значит, все эти проверки можно назвать избыточными. В общем случае, проверка, производимая над некоторым элементом рабочей памяти, является избыточной, если её результат предопределён проверками, уже произведёнными над тем же элементом.

Итак, мы приходим к необходимости использования другой, более совершенной, структуры, определяющей порядок применения проверок. Эта структура должна позволять задавать оптимальный порядок проверок при полном отсутствии избыточности.

Введём дополнительные обозначения. Единственным входом нашего алгоритма является множество всех α-множеств, относящихся к рассматриваемому классу. Обозначим это множество Λ:

$$\mathrm{A} = \{\alpha_i\}_{i=1}^{k}.$$

Задачу нахождения оптимального порядка проведения проверок, которую нам нужно решить, обозначим Π(A). Введём также обозначение для всего множества различных проверок, которые потенциально необходимо осуществить:

$$U(\mathrm{A}) = \bigcup_{\alpha_i \in \mathrm{A}} \alpha_i .$$

Общая схема решения задачи такова:

1. Выбираем проверку $a^* \in U(\mathrm{A})$, которую мы будем выполнять первой (в любом случае, некоторая фиксированная проверка всегда будет выполняться первой).
2. Для случая успешного выполнения проверки устанавливаем последовательность дальнейших проверок равной решению задачи $\Pi(\mathrm{A} + a^*)$, где
$$\mathrm{A} + a^* = \left\{\alpha' \middle| \alpha' = \alpha \setminus \left\{a \in \alpha \middle| a^* \Rightarrow a\right\}, \alpha \in \mathrm{A} \setminus \left\{\alpha_i \middle| \exists a \in \alpha_i : a^* \Rightarrow \neg a\right\}\right\}$$
3. Для случая неуспешного выполнения проверки устанавливаем последовательность дальнейших проверок равной решению задачи $\Pi(\mathrm{A} - a^*)$, где
$$\mathrm{A} - a^* = \left\{\alpha' \middle| \alpha' = \alpha \setminus \left\{a \in \alpha \middle| \neg a^* \Rightarrow a\right\}, \alpha \in \mathrm{A} \setminus \left\{\alpha_i \middle| \exists a \in \alpha_i : \neg a^* \Rightarrow \neg a\right\}\right\}$$
4. Если $U(\mathrm{A}) = \varnothing$, то нужно послать рассматриваемый токен всем $\alpha_i \in \mathrm{A}$.

На множество A можно посмотреть следующим образом: если для некоторого $\alpha_i \in \mathrm{A}$ выполнены все проверки $a \in \alpha_i$, то необходимо

послать рассматриваемый токен узлу α-памяти, соответствующему α_i. Пункт 2 гласит, что в случае, если проверка a^* выполнена успешно, то необходимо удалить из A все α_i, содержащие проверки, невыполнение которых следует из выполнения a^*, а из всех оставшихся α_i удалить те проверки, выполнение которых следует из выполнения a^*. То есть, если одна из проверок в α_i не выполнена, мы удаляем α_i из A, поскольку в соответствующий узел α-памяти рассматриваемый токен не должен попасть ни при каких условиях. Если же одна из проверок в α_i выполнена, то её нужно просто удалить из α_i, уменьшив тем самым количество условий, соблюдение которых нужно проверить, прежде чем можно будет послать токен в соответствующий узел α-памяти.

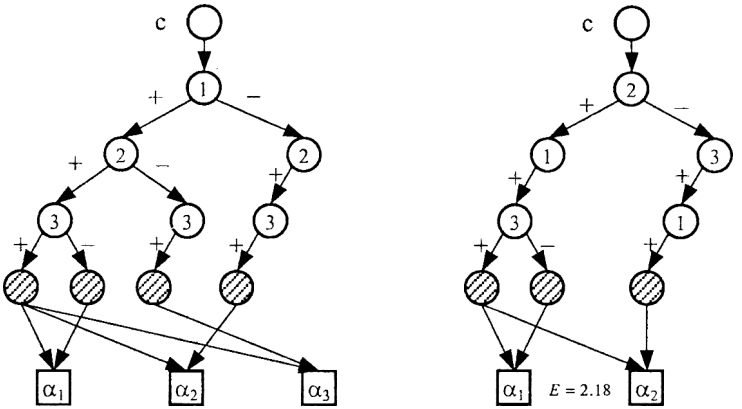
Если некоторое α_i оказалось пусто, это значит, что все проверки в этом множестве прошли успешно и токен можно послать узлу α-памяти. На каждом шаге мы сводим исходную задачу к двум задачам меньшей размерности (с меньшим по мощности $U(\mathrm{A})$), поскольку, как минимум, сама проверка a^* не будет присутствовать ни в $U(A+a^*)$, ни в $U(A-a^*)$.

Получающийся порядок проверок можно представить в виде дерева. Каждый внутренний узел такого дерева будет производить одну проверку и иметь не более двух потомков: один соответствует успешной проверке (then-потомок, он присутствует всегда), а другой - неуспешной (else-потомок, он может и отсутствовать). Узел производит проверку над пришедшим токеном и, в зависимости от результата проверки, посылает токен либо then-потомку, либо else-потомку.

У классового узла один потомок – проверяющий узел. Кроме того, необходимо ввести узлы нового типа – назовём их распределяющими узлами. Когда после очередной проверки не нужно больше производить никаких проверок, а остаётся только послать токен некоторым узлам α-памяти (пункт 4 в вышеизложенной схеме), соответствующим потомком проверяющего узла становится распределяющий узел. Он просто посылает пришедший ему на вход токен всем своим потомкам – узлам α-памяти.

После невыполнения некоторой проверки может оказаться, что токен не нужно посылать ни в один узел α-памяти. В таком случае проверяющий узел просто не будет иметь else-потомка.

На Рис. 4 представлены примеры таких if-then-else деревьев для рассмотренных ранее наборов шаблонов.



(а) Дерево проверок для примера на Рис. 2.

(б) Дерево проверок для примера на Рис. 3.

Рис. 4. Примеры if-then-else деревьев, задающих порядок проверок.

Очевидно, такая схема полностью исключает избыточные проверки, поскольку в каждом узле имеется информация обо всех произведённых к данному моменту проверках. Также нетрудно видеть, что для любого классического α-дерева, фиксировав некоторый порядок обхода (например, в глубину, потомки перебираются слева направо), можно построить if-then-else дерево (возможно с избыточными проверками), реализующее тот же порядок осуществления проверок.

До сих пор остался нерассмотренным важнейший вопрос выбора проверки a^*. Обозначим математическое ожидание числа производимых проверок при оптимальном решении задачи $\Pi(\mathrm{A})$ как $E(\mathrm{A})$. Тогда, выбрав в качестве первой проверки при решении $\Pi(\mathrm{A})$ некоторую проверку a, получим, что лучшее математическое ожидание, которого мы сможем добиться, оптимальным образом решив $\Pi(\mathrm{A}+a)$ и $\Pi(\mathrm{A}-a)$, будет равно

$$E = 1 + p(a)\cdot E(\mathrm{A}+a) + (1-p(a))\cdot E(\mathrm{A}-a).$$

Поэтому при наилучшем выборе a^* получим

$$E(\mathrm{A}) = \min_{a\in U(\mathrm{A})} \{1 + p(a)\cdot E(\mathrm{A}+a) + (1-p(a))\cdot E(\mathrm{A}-a)\},$$
$$a^* = \arg\min_{a\in U(\mathrm{A})} \{1 + p(a)\cdot E(\mathrm{A}+a) + (1-p(a))\cdot E(\mathrm{A}-a)\}. \qquad (1)$$

Применение формул (1) даёт оптимальный порядок проверок, но представляет, по сути, полный перебор, требующий количества действий порядка $|U(\mathrm{A})|!$. Поэтому эти формулы применимы только для задач относительно небольшой размерности. Для задач с большим

числом проверок необходимо использовать приближённые, но полиномиальные по сложности методы.

Интуитивно разумным кажется выбор проверки a^*, минимизирующий математическое ожидание числа проверок, которые потенциально необходимо будет произвести после данной, (то есть максимизирующей среднее число проверок, вычёркиваемых из множества $U(\mathrm{A})$):

$$a^* = \arg \min_{a \in U(\mathrm{A})} \left\{ p(a) \cdot \left| U(\mathrm{A}+a) \right| + (1-p(a)) \cdot \left| U(\mathrm{A}-a) \right| \right\}. \qquad (2)$$

Сравнивая (1) и (2) легко видеть, что формула (2) основана на аппроксимации

$$E(\mathrm{A}) \approx C \cdot \left| U(\mathrm{A}) \right|, \quad 0 < C \le 1, \qquad (3)$$

где точное значение константы C не влияет на выбор a^*.

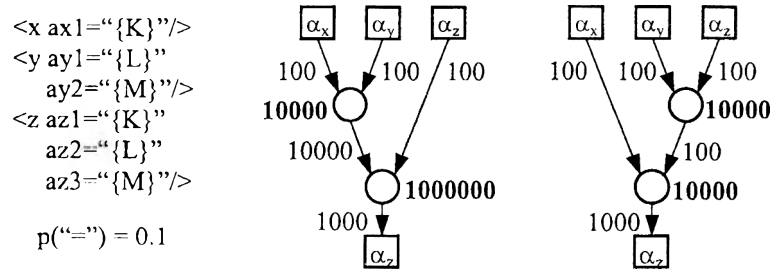
Дальнейшее совершенствование изложенного подхода связано с получением формул, промежуточных между (1) и (2). Общим методом получения таких формул является построение более точных оценок $E(\mathrm{A})$. Очевидно, что увеличение точности оценок будет связано и с увеличением их вычислительной сложности. В качестве одного из универсальных методов повышения точности оценок необходимо выделить просчёт с использованием формул (1) и (3) на глубину > 1: делаем несколько шагов с использованием формул (1), после чего применяем оценку (3). При глубине просчёта, равной 1, получим формулу (2), при глубине, стремящейся к бесконечности, – точную формулу (1).

Генерация β-части сети

Если задача генерации α-части сети разбивается на подзадачи по классам, то задача генерации β-части – по продукциям. Пусть мы рассматриваем продукцию, в левой части которой содержится n шаблонов. Необходимо построить (n–1) β-узлов, соединяющих в некоторой последовательности выходы узлов α-памяти, соответствующих этим n шаблонам. Критерием качества β-части сети является среднее число попыток соединения (сравнений двух токенов для определения возможности их соединения) во всех β-узлах сети.

Поскольку операция соединения является коммутативной и ассоциативной [4], соединение можно проводить в любом порядке. В статье [1] рассматривается только простейший порядок соединения: сначала соединяются узлы α-памяти, соответствующие первым двум

шаблонам в продукции, затем выход получившегося узла соединяется с



(а) Левая часть продукции (б) Левоассоциативная структура (в) Оптимальная структура

Рис. 5. Пример, показывающий неоптимальность левоассоциативной структуры соединений.

узлом α-памяти третьего шаблона и т. д.

Этот порядок чаще всего далёк от оптимального. На Рис. 5(а) приведён пример левой части продукции. Предположим, что за время работы программы в каждый из узлов α-памяти, связанных с шаблонами на Рис. 5(а), пришло по 100 токенов с операциями добавления. Предположим также, что вероятность совпадения значений двух атрибутов из разных элементов равна 0.1. На Рис. 5(б) представлена левоассоциативная, а на Рис. 5(в) – правоассоциативная структура соединений. Около каждой дуги указано количество токенов, прошедших по этой дуге, а около каждого β-узла – количество попыток соединения в данном узле. В данном случае изменение левоассоциативной структуры на правоассоциативную уменьшает число попыток соединения с 1010000 до 20000.

Введём для каждого β-узла s следующие параметры:

Token(s) – число токенов, сформированных данным узлом и переданных его потомкам, за всё время работы программы;

Memory(s) – среднее число соединений элементов, хранящихся в памяти, присоединенной к выходу данного узла;

Test(s) – число попыток соединения, произведённых данным узлом за время работы программы;

Ratio(s) – процент удачных соединений в данном узле.

Обозначим непосредственных предшественников узла s как l и r. Тогда нетрудно доказать следующие соотношения для этих параметров:

1. $Test(s) = Token(l) \cdot Memory(r) + Token(r) \cdot Memory(l)$.

2. $Token(s) = Test(s) \cdot Ratio(s)$.

Поскольку узлы α-памяти также могут быть непосредственными предшественниками β-узлов, для них тоже будем рассматривать параметры *Token* и *Memory*.

Предположим, в каждый классовый узел за время работы программы пришло N токенов, где N – достаточно большое число. Тогда для некоторого узла α-памяти α_i можно воспользоваться оценкой:

$$Token(\alpha_i) \approx N \cdot \prod_{a \in \alpha_i} p(a).$$

Значение параметра *Memory*, вообще говоря, зависит от соотношения числа токенов с операцией добавления и токенов с операцией удаления. Поэтому можно просто принять:

$$Memory(s) \approx M \cdot Token(s), \quad 0 < M \le 1.$$

При этих предположениях можно пытаться найти соединяющую структуру с минимальной суммой $Test(s)$ по всем β-узлам s. Однако это потребует очень большого перебора, да и оптимальность полученной структуры будет весьма условной, вследствие условности сделанных предположений. Вместо этого мы будем стремиться построить просто «неплохую» структуру и избежать заведомо невыгодного порядка соединения, пользуясь эвристическими соображениями.

Как показывает практика, обычно выгоднее сначала строить те β-узлы, для которых $Token(s)$ мало. Это объясняется тем, что количество токенов, порождаемых узлом, входит как множитель в количество проверок, проводимых всеми потомками данного узла. Хорошей иллюстрацией этой идеи является Рис. 5, где было выгоднее сначала соединить узлы α-памяти для шаблонов y и z именно потому, что получающийся таким образом β-узел порождал меньше токенов.

В условиях предположения, сделанного о параметре *Memory*, получаем следующую формулу:

$$\begin{aligned} Token(s) &= Test(s) \cdot Ratio(s) = \\ &= \big(Token(l) \cdot Memory(r) + Token(r) \cdot Memory(l)\big) \cdot Ratio(s) = \\ &= 2M \cdot Token(l) \cdot Token(r) \cdot Ratio(s). \end{aligned}$$

Мы уже оценили $Token(\alpha_i)$, а $Ratio(s)$ можно посчитать для любого β-узла, перемножив оценки вероятностей выполнения элементарных проверок, которые необходимо произвести в узле s.

Теперь можно сформулировать алгоритм построения β-части сети для одной продукции:

1. Возьмём в качестве множества J множество шаблонов элементов в левой части продукции. С каждым шаблоном свяжем соответствующий ему узел α-памяти.
2. Находим минимум по всем неупорядоченным парам (x, y) элементов J значения выражения $Token(l) \cdot Token(r) \cdot Ratio(s)$, где l и r – узлы, связанные с x и y соответственно, s – потенциальный β-узел, соединяющий l и r. Пусть (x*, y*) – пара, на которой достигается минимум.
3. Удаляем x* и y* из J. Строим β-узел, соединяющий узлы, связанные с x* и y*. Добавляем в J пару (x*, y*) и связываем с ней только что построенный узел.
4. Если в J остался один элемент, то требуемая структура построена, иначе – переходим к 2.

Заключение

В ходе анализа алгоритма Rete было выявлено, что при использовании классической структуры α-части Rete-сети во многих случаях невозможно избежать проведения избыточных проверок. В связи с этим предложена другая структура α-части сети, позволяющая полностью исключить избыточные проверки. Кроме того, для сокращения среднего числа проверок, производимых в процессе сопоставления, предложены оптимизирующие алгоритмы компиляции α- и β-части Rete-сети. На основе предложенных решений реализован компилятор Rete-сети для прототипной продукционной системы.

Литература

1. Forgy C. L. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Matching Problem // Artificial Intelligence. 1982. 19. N 1. P. 17–37.

2. Люгер Дж. Ф. Искусственный интеллект. Стратегии и методы решения сложных проблем. М.: Вильямс, 2003. – Разд. 5.3. – С. 196–212.

3. Scales D. Efficient matching algorithms for the SOAR/OPS5 production system. Technical Report KSL-86-47. Stanford: Stanford University, Dept. of Computer Science, Knowledge Systems Laboratory, 1986. 50 p.

4. Schor M., Daly T., Lee H. S., Tibbits B. Advances in Rete pattern matching // Proceedings of the Fifth National Conference on Artificial Intelligence. Menlo Park, California: AAAI Press, 1986. P. 226–232.

Громыко В. И., Мальковский М.Г.,
Кузина Л.Н. (МГУ), Симакин А.Г. (РУДН)

Обучающие системы «компьютерного» образования в высшей школе

Абстракт

В статье [1] рассмотрена проблема «компьютерного» образования в высшей школе. Л.Королев, обращаясь к подготовке специалистов в области компьютерных наук, приводит доводы о недопустимости сокращения объема математической подготовки и особом значении фундаментальности в этой подготовке.

Данная статья развивает тему фундаментальности, доводя исследование до определения математических тем базового обучения информатике студентов высшей школы. В этой связи обсуждается также неизбежность использования специально реализуемой обучающей системы (интеллектуальной по Д.Поспелову [2]) и предлагается подходящий для возникших условий метод обучения.

Работа выполнена при финансовой поддержке Российского Фонда Фундаментальных Исследований (грант 0301-00339а).

Хотя потребность в достоверном факте и естественна для человека, но все же она есть его умственный недостаток. /Б. Рассел/

...абсолютно безупречно работающая машина не может обладать интеллектом. Об этом свидетельствуют ряд теорем, которые, однако, ничего не говорят о том, каким уровнем должна обладать машина, не претендующая на безошибочность и безупречность работы. /А. Тьюринг/

...нет правила, из которого можно было бы выводить разумное применение правил /И. Кант/

Третья техническая революция

Современное состояние рационального знания характеризуется разнообразием приложений, связанных с экспансией компьютера в различные предметные области. При этом возросшая мощь его инструментальных систем (ИС) позволяет получать для предметной области (ПО): или, в лучшем случае, модель конструктивной обработки; или, в худшем случае, информационную поддержку для исследований. Таким образом, на любом уровне применения компьютера каждый принужден к синтезирующей деятельности, как в собственной ПО, так и в математических областях, в связи с «математичностью» ИС. В этом и

состоит третья научно-техническая революция (3-НТР) — внедрение в деятельность человека ИС, работающих со знанием [3]. Итак, злободневен переход от системно-информационной науки к системно-информационной культуре. Синонимом 3-НТР является понятие «век систем». Оно противостоит термину «век машин».

Классификация обучающих систем

В связи с требованиями 3-НТР, классификация ведется со стороны учащегося, т.к. считаем, что сегодня от любой ИС следует требовать ее действий по обеспечению «взросления» пользователя в отношении ее уровней сложности. Короче, должен присутствовать не только блок справки, но и блок обучения, причем адаптивный к пользователю. Тогда историческая цепочка развития обучающей системы (ОС) такая: тренажер, экспертная система, рабочее место (РМ), интеллектуальная обучающая система (ИОС).

Тренажер [4] соответствует профессиональному обучению (ПрОб). Необходимо «покрыть» ПО. Метод обучения оптимистический: не можешь — научим; не хочешь — заставим. Похоже на учебный процесс в школе с учащимися переходного возраста.

Экспертная система [4], по-прежнему, соответствует ПрОб, но уже на трудоемкой ПО. Метод обучения рассудочный: надлежит подстелить «соломку» в известных учителю трудных местах. Адаптивность есть, но в отношении иерархии сложностей ПО. Так учат в высшей школе отдельному предмету.

РМ [5,6] соответствует требованию на синтезирующий характер деятельности пользователя. Учащийся рассматривается как субъект обучения, поэтому он нуждается в РМ для обучения.

ИОС функционирует посредством блока, обеспечивающего адаптивность системы к учащемуся в отношении его интеллектуального развития при обучении на заданных ПО. Этот блок реализуется на основании представлений о способах познания рода и человека. Разработчику блока следует ясно представлять — как фиксируются смыслы в культуре и затем наследуются каждым за счет интеллектуального развития.

С точки зрения предъявленной классификации современные ИС ограничиваются богатой справкой и специальным обеспечением выделенных социально классов пользователей (шаблоны). Т.е. они пребывают, чаще всего, на уровне экспертной системы. Таковы издержки общества потребления.

Современные ОС на узких ПО достигают уровня РМ. Создание ИОС [2] ведется в научно-исследовательских проектах [7]. Эти исследования определяются контекстом новых реалий: системно-информационная культура, в которой пребывает каждый; современный

рационализм, фиксирующий единство «надприродной» духовности (человека) и природы (системно-информационной для человека), реально требующей от нас соответствовать ИС. В обучении, в связи с сопутствующими системам объемностью и сложностью знания, это приводит к необходимости подключения *самоорганизации* и *свободы развития* учащегося. Легким следствием этого является обнадеживающая тенденция – востребованная самоорганизация противоречит примитивной социальной организации общества потребления. Труднее заметить, что на этом пути уже преодолеваются также границы классического рационализма, который выстроен на детерминистической позиции нашего соответствия природе — «припоминание» по Платону [8].

Сложность проблемы обучения и необходимость ИОС

Сложность проблемы определяется уже не только необходимостью профессиональной подготовки, но и дополнительно обеспечением условий передачи смысла синтезирующего характера — на уровне применения теорий в межпредметной деятельности учащегося [9]. Поэтому мышление в «век систем» — это обладание адаптивностью уже в отношении открытого к развитию (мета-адаптивность) понимания посредством самоорганизации на базе мировоззренческого теоретического знания. Самоорганизация здесь возникает поскольку нашу онтологическую приспособленностью к развитию приходится серьезно нагружать в возрасте старшеклассника и студента. Соответствующее обучение нуждается в ИОС, т.к. надо обеспечить помощь в нахождении учащимся учебных материалов для приобщения к сложным его интеллекту абстракциям. При этом ИОС должны работать в условиях неполного, в то же время, объемного и открытого знания о ПО, учащемся, методе познания, причем с целеполаганием по развитию сознания учащегося (познай себя в отношении возможностей в рациональном) на базе его работы со знанием. Так конкретизируется нами ИОС [10,11] на основе предлагаемой Д.Поспеловым классификации проблем в ИИ, называемой им 10 «горячей точкой».

«Компьютерное» образование в высшей школе

Деятельность любого выпускника высшей школы на рабочем компьютерном месте требует основательной системной рациональной культуры. В этих условиях возрастают требования к базовым курсам программирования, математики, информатики в отношении формирования мировоззрения учащегося по межпредметным связям.

Еще раз подчеркнем, сложность задачи обучения серьезно возрастает, т.к. следует заниматься применением теорий, а значит обеспечить представлениями об их границах. К тому же особый характер системной культуры требует сбалансированного предъявления слова (теории) и дела (конструктивной деятельности). Итак, следует заниматься мировоззрением учащегося в ракурсе конструктивной деятельности. Впрочем, последнее обстоятельство не столько увеличивает объем изучаемого, сколько является решающим фактором в обучении, т.к. культура века систем внедряет эту деятельность непринужденно и повсеместно.

Интегрально необходимое в высшей школе уже наличествует в представлениях:

[*от программирования*] о языках как языках спецификации, посредством понимания различных стилей программирования [12,13];

[*от информатики*] об объектной технологии, посредством программирования объектно-ориентированного (ООП) и документно-ориентированного (ДОП);

[*от математики*] об алгебраических аспектах объектной технологии, а также математических средствах сравнения систем (например, язык категорий).

Но теория и практика даются профессионально и, как следствие, автономно. К тому же курсы растянуты во времени. Отсюда межпредметные связи отданы на откуп прозорливости учащегося. По этому поводу Л.Королев утверждает «В то же время преподавание высшей математики для компьютерщиков следует изменить, не жалея времени главным образом на фундаментальные концепции, на конструктивные доказательства фундаментальных результатов, оставляя в стороне, быть может, некоторые детали типа прагматических рецептов» [1]. Ясно, что это означает:

правильно отобрать фундаментальность;

фундаментальность должна быть на стыке непрерывности и дискретности [14];

обеспечить учебным материалом эту фундаментальность на первых курсах;

обеспечить усвоение (подъемность) выбранной фундаментальности на первых курсах.

Слово: язык второй грамотности — основа фундаментальности

А.Ершов, фиксируя уже наступившую в 80-90гг возможность приобщения всех к рациональной культуре, выдвинул лозунг «информатика – вторая грамотность». Усилим эту позицию для выбора фундаментальности, полагая необходимым проявить уже состоявшееся

в рациональной культуре понятие «язык второй грамотности». В самом общем понимании, язык второй грамотности образует язык рационализма, достигший через единство программирования (ООП), математики (алгебраические системы), информатики (представление знания в ИС) уровня использования и исследования систем.

Исходим из принципа: основные механизмы опыта заложены в языке. Поэтому мышление приравнивается к осмысленному употреблению языка, а мысль – к осмысленному предложению. Поэтому язык второй грамотности наблюдается в таком ряду обобщений:

1) нулевой язык образует социальный язык общения;
2) язык первой грамотности образует естественный язык, прагматика которого состоит в концептуальном охвате мира;
3) язык второй грамотности образует рациональный язык (математические обобщения являются образующими подсознания), необходимый для повышения точности при системном взгляде на мир, при системном существовании в нем, требовании на его системную организацию.

Уточнимся. Информатика рассматривается как развитие рациональной культуры, т.е. развитие математики до конструктивного предъявления сложных (многопараметрических) ИС на основе конструирующих возможностей программирования. Обобщенное представление предмета информатики тотчас же переводит традиционную математику в «реальность», на которой выстраиваются абстрактные методы конструктивных разделов. Точнее, в математике взаимодействие «реальной» и абстрактной частей характеризуется новой связностью: во-первых, за счет их согласования в связи с конструктивностью /*Д. Кнут*/; во-вторых, за счет формирования на этом пути языковых математических моделей /*Д. Гильберт*, *А. Марков*/. Прагматически сегодня симбиоз математики и информатики проявляется во взаимодействии разделов конструктивной математики с нашими навыками конструирования, которые через РМ перерастают традиционное программирование на ЯП и превращаются в ДОП и ООП, приобщающих всех к системам. Поэтому, например, вычисление следует рассматривать с точностью до разнообразных вычислителей, включая вопросы разрешимости и полиномиальной вычислимости.

Уровень развития	Культурный филогенез рационального	Культурный онтогенез интеллекта
Онтология	Конструктивность	Язык
Реальное	Математика Программирование	Финитные методы Вычислитель (компьютер)
Идеальное	Информатика	ОО системы

Табл. 1. Динамика рациональной культуры и интеллекта

Заключаем, в рациональной культуре усиливается «надприродная» духовная составляющая сознания, но на основе ее материальной сути – языка. Таким образом, в информатике есть синтезирующий стержень, который противостоит тенденции уподобления человека (профессиональный идиотизм) машине, и стержень составляют ДОП и ООП в сложных объектно-ориентированных системах.

В качестве обоснования представленной позиции приведем две таблицы, фиксирующих историческое развитие рациональной культуры рода (культурный филогенез) через ее отношение к субъекту (культурный онтогенез). В табл.1 приведены в динамике три уровня интеллекта человека, соответствующие состояниям рациональной культуры. Онтологический уровень обеспечивается культурой на «подсознательном»; реальный фиксируется учебной деятельностью; идеальный добывается самоорганизацией посредством интеллектуальных прорывов. Это грубая линейная схема, использующая расхожее представление о бытии и сознании (вспомним про еще недавно часто используемый философами основной философский вопрос). Реально, из-за разнообразия рациональных областей (а в них понятий), состояние учащегося размыто по всем этапам (тришкин кафтан). В табл.2 представлены интеллектуальные составляющие, формирующие мировоззрение учащегося в соответствии с 3-НТР. Подчеркнем, что в век машин было проще, т.е. культивировался метафизический детерминизм, который обеспечивал большинству комфорт на аналитическом уровне в границах профессиональной области при функционировании в соответствии с распространенным мировоззрением.

Прагматика	Интеллектуальные образующие		
	синтезирующая	аналитическая	конструктивная
математика для машин	геометрия порядок язык логики	алгебраическая геометрия решетки формальная теория	вычислимость и эффективная вычислимость определимость
информатика для систем	программирование ДОП, ООП	стили программирования (синтаксис, семантика, прагматика) алгебраические системы	языки программирования языки спецификаций

Табл. 2. Динамика интеллектуальных образующих

Если максимально абстрактно представить динамику фундаментальности, замешанную на синтезе математики, программирования и информатики, то получится весьма определенная схема:

НАМ→АМ→САМ

причем реальностью для учащегося следует считать **АМ** для представления теорий, от которых ему надлежит развиться до знания языков представления теорий – **САМ**.

НАМ (наивный аксиоматический метод) – это состояние рационального знания, характеризующееся определенностью моделируемого (наивность), но уже с аксиоматической упаковкой (доказательство). Это состояние типа «Начал Евклида». Состояние **АМ** (аксиоматический метод) характеризуется утратой определенности реального моделирования за счет возросшей абстрактности, приведшей к осознанию большой пользы продуктивных (для интеллекта) моделей (преодоление тирании внешнего мира). Целеполагание по отношению к фундаментальности задается понятием **САМ** (современный аксиоматический метод).

САМ = теория моделей + выводимость (вычислимость)
в неполных формальных теориях,
теория моделей (определимость) = универсальная алгебра +
язык (алгебра) логики.

Заметим, что в связи с ООП особо выделены неполные теории.

На основании изложенного интеллектуальное восхождение в концептуальном происходит:

- на «реальном» для учащегося представлении – **НАМ** в отношении понятий число, геометрия, доказательство, вычисление по формулам;
- на представлении о повышенной точности за счет абстрактности – **АМ** в отношении алгебраических структур и систем;
- при движении к языковой сути наших представлений – **САМ** как проявление формальных теорий, включая ООП и ДОП.

Очертим границы языка второй грамотности, опираясь на золотой фонд концептуальности. При этом надеемся обеспечить условия для преодоления мифа: рациональные методы объясняют, а гуманитарные – служат пониманию.

Следует выделять следующие разделы (представления).

1. ограниченность средств (формальные теории);
   теоремы: Биркгоффа, Геделя-Мальцева (компактности), Геделя (о неполноте), Сколема-Левенгейма (спуска, подъема);
   парадокс Рассела, слабая логика 1-го порядка, логика 2-го порядка.
2. замкнутые (категоричность) алгебраические представления, в том числе чистые теоремы существования;
   теоремы: Фробениуса (тела), Понтрягина (непрерывные тела), Стоуна (булеан), Кэли (представление групп), Фреге (исчисление предикатов), Линдстрема (исчисление предикатов), Гильберта – аксиоматика геометрии, тезис Черча (формализация алгоритмов).
3. неполные теории и наследуемость;
   теоремы: Коэна, принцип Кэли.
4. выводимость и вычислимость;

теоремы: Поста, Черча, Маркова (проблема Туэ-Дена), Новикова (неразрешимость в группах), Клини (рекурсии), Райса (нумерация); классические неразрешимости (циркуль и линейка).

5. эффективная вычислимость, сложность;
   теоремы: Кука, Блюма.
6. вычислители и данные;
   теоремы: Эрбрана (перечислимость), Кодда, Черча-Россера, Робинсона (резолюция).

Получаем впечатляющий индекс понятий:

- от программирования — сортировка, поиск, переборные задачи (универсальные); грамматика, автомат, модель (объектно-ориентированный, лисп, комбинатор, пролог, марков, процесс) вычислителя; абстрактный тип данных (АТД), модели данных;
- от математики — диагональный метод, структура, (под, фактор) алгебра, (гомо, изо, гомсо) морфизм, формальная теория, модель (теория, представление), категория (функтор);
- от информатики — геделизация, разрешимость, перечислимость, сводимость (класс, полнота), функция (универсальная), логика (классическая, модальная).

Дело: ИОС как средство синтезирующей деятельности учащегося

В системно-информационный период к интеллекту (3-НТР) на первый план выдвигаются требования в отношении созидания, основанного на синтезе, экспансионизме, телеологии (в определенном смысле, на априорной синтезирующей индуктивности) и только во вторую очередь задействуются анализ, редукционизм, детерминизм (дедукция). Это связано с повышенным требованиям к системному мышлению.

Программистский разум (конец XX в.) = <навыки конструирования на основе ЯП, ← системный аналитик →, навыки спецификации на основе стилей программирования>

Системологический разум (XXI в.) = <навыки конструирования на основе ООП, ← *САМ* →, навыки спецификации на основе **САМ** >

САМ, как интеллектуальное состояние учащегося, нами называется *САМ* (синтезирующий априорный мастер) и мастер должен представлять:

САМ = {лингвистические модели для вычислений, инверсный характер традиционной математики (обогащение общих структур), сравнение систем (язык категорий)}

«Синтезирующий» означает, что мастер действует в межпредметных областях. «Априорный» фиксирует тенденцию рациональной культуры вынести сложности интеллектуального устройства учащегося на

онтологический уровень. При таких представлениях остро чувствуется недостаток традиционного обучения, заботящегося об интеллекте только через трудности ПО. Чтобы справиться с проблемой, приходится привлечь достижения научного познания и антропологии для формирования универсального обучения (УнОб), являющееся надстройкой над ПрОб. Обнаруживается сложный контекст:

[*от философии*] мышление, рассудочное, разумное, способность суждения, реальное, идеальное, предрассудочное, здесь-бытие, герменевтический круг, знание, понимание, истина, фальсификация, интеллигибельные вещи, конкретное, общее, особенное, рациональное, иррациональное, символическое, антропология, системно-информационная культура, искусство,…;

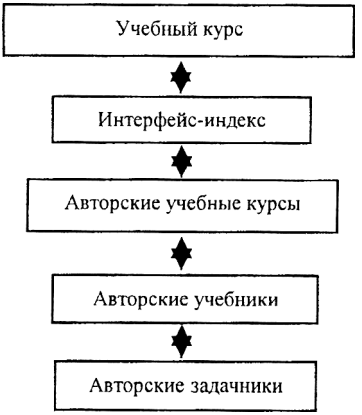
[*от психолоии*] мышление, интеллект, язык, сознание, бессознательное, ассимиляция, аккомодация, саморегуляция, генетическое, медиативное,…;

[*от обучения*] РО, знание-умение-навыки (ЗУН), умение-навыки-знание (УЗН), самооргаизация, свобода учиться, мыслительный акт, экология разума,….

Рассмотрим **универсальное обучение.**

Для какой цели учить? На этот главный вопрос мы уже ответили. Повторимся, злободневен переход от системно-информационной науки к системно-информационной культуре. УнОб нацелено справиться с вызовом времени.

Чему учить? Первая сложность: заключается в объемности ПО, т.к. необходимо охватить синтез математики, программирования, информатики. Сложность преодолевается не написанием нового фундаментального курса, а специальным взаимодействием множества существующих профессиональных авторских и учебного курсов. Такая организация учебного материала соответствует УнОб, связанному непосредственно с проблемой самоорганизации и свободы развития учащегося.



Иерархия документов предметной области

Единицу учебного курса образует проблема-задание для практикума по программированию. Оно соответствует усложненному заданию традиционного практикума в высшей школе. Так как задание выполняется в современной ОО-системе, то его характер должен быть нацелен на «взросление» учащегося в отношении усложняющихся ОО-средств системы. Методический материал по системе, организованный для его исследования учащимся, образует часть методического материала учебного курса. Проблема-задание объемно, иерархично, поэтому схоже с проектом и реализуется в течение года. Его составными частями являются пример-проблемы. Пример-проблемы являются задачами с богатством межпредметных связей, которые обеспечивают концептуальное видение учащимся предмета либо его части. В идеале, для каждого понятия, существенного в рамках курса, должна существовать пример-проблема. Пример-проблемы служат источником для обнаружения в профессиональных авторских курсах: во-первых, частных задач; во-вторых, методических материалов. Для этого отношение понятий авторских курсов подчинены связям головных понятий, заданных в учебном курсе (посредством отображения *корреспонденции* [15]).

Кого учить? Вторая сложность УнОб сложнее первой. Л.Королев утверждает: «Строгость мышления, умение находить алгоритмы решения, используя систематику классической математики, позволит специалисту, *обладающему такими способностями*, обеспечить свой приоритет в освоении всего того нового, что будет появляться в компьютерной науке, позволит ему *творить* это *новое*». Мною выделено курсивом — за какие качества учащегося надлежит сегодня сражаться в обучении. На повестке «новое», а ему соответствует деятельность в «незнаемом». Вот и выявилась суть

современной несостоятельности ПрОб – обеспечивает ученика ответами. УнОб преодолевает границы ПрОб – обеспечивает на основе ответов формулировку вопросов, являющихся базой саморазвития учащегося. ПрОб отвечает за ЦЕЛЕсообразность, определяемое локальным знанием, а УнОб – за ЦЕЛОсообразность, определяемое мировоззрением.

Вообще говоря, гуманитарное знание всегда было озабочено целесообразностью, но не смогло противостоять агрессивному социальному развитию («возрастание к гуманности», по Гердеру, не состоялось). Экология разума – острая болезнь человечества, которая преодолевается на рациональном пути, т.к. здесь затребовано и обеспечено средствами интеллектуальное развитие рода. Мыслительные процессы обретают точность, если приобретают языковую форму, допускающую вывод (использование аналитики, как средства упаковки). После развития математики до теории моделей (универсальная алгебра + логика), затем до формальных теорий (обеспечивающих языковую аналитическую обработку), следует признать, что мышление добывает устойчивость за счет нового опыта в рациональных языках, обслуживающих программирование и информатику. В этом новом качестве, связанным с системной организацией, рационализм называется эволюционным, т.к. системную сложность (при быстрой динамике развития) человек может покрыть только за счет самоорганизации. Именно при таком взгляде может быть сформулирован тезис о происходящем изменении подсознательного стиля мышления (мышление как творчество). Исторически во времена изменения подсознательного математика и познание начинает затрагивать всех. Сегодня такое время.

На этих основаниях, как аналогия и развитие кантовского понятия «синтетически априорное», введена цель обучения – самоорганизация *САМ.* За это состояние учащегося борется УнОб.

Как учить? УнОб включается как составная часть традиционного учебного процесса. ИС именуется **ИОС-Flint**. Ее интеллектуальный блок действует в соответствии с априорной и динамической моделями учащегося. Эти модели, используемые для интеллектуального развития учащегося, следуют из здравого смысла, но с удовлетворением обнаруживаем сходство с самой общей моделью самоорганизации синергетики [16]. Учащийся (взаимодействует со знанием) рассматривается как нелинейная система, которую следует продвинуть к увеличению сложности.

Во-первых, выбраны естественные классы учащегося, т.к. «самосознание индивида всего лишь легкий трепет в замкнутых токах исторической жизни» (Г.-Г.Гадамер). Ими являются:

- ПРАГМАТИК — состояние фиксируется пользовательским аспектом предмета, заключенным в средствах ДОП и ООП;

- ПРОФЕССИОНАЛ — состояние фиксируется профессиональным аспектом предмета, заключенным в представлениях о программировании на классических стилях ЯП (с упором на модели типов);
- УНИВЕРСАЛ — состояние фиксируется теоретическим аспектом предмета, заключенным в математических понятиях определимости, эффективной вычислимости, ИС;
- НАЧИНАЮЩИЙ — состояние фиксируется системным аспектом школьного курса математики, заключенным в НАМ, АМ для проявления понятия АТД;

Во-вторых, в хаосе сложности для учащегося (как условия интеллектуального прорыва) возникновение порядка специфично и уникально. Отсюда предоставление учащемуся свободы развития. Эта свобода состоит в предоставлении учащемуся учебного материала на основе его выбора собственного пути из множества формируемых системой путей (восхождение по обобщениям к абстрактному, схождение к конкретному), соответствующих целостности учебного курса.

ИОС-Flint действует в соответствии с методом обучения **ГРОМ**. Аббревиатура означает - герменевтики [17] и развивающего обучения (РО [18]) мастер. Мастер устроен для интегрированного предмета (включая границы составляющих его теорий) и поэтому функционирует на разведенных смыслах понятий: ЗНАТЬ (ПрОб) и ПОНИМАТЬ (УнОб). Метод является продвижением метода РО, занятого концептуальным развитием учащегося в средней школе (сосредотачиваясь на теории), в высшую школу. В высшей школе, когда учащийся уже пребывает в «режиме с обострением», следует использовать знания для интеллектуальных прорывов. Надлежит способствовать целостному восприятию предметов для установления межпредметных связей. Для этого привлекается герменевтическая теория познания, которая распространяет познание на каждого посредством понимательной деятельности. Ее ключевые принципы естественны: надлежит обеспечить вхождение в смысл на основе прежде всего предсуждения учащегося; особая роль циклического процесса в отношении части и целого (конкретного и общего). Формируемые ИОС пути как раз связывают части (авторские курсы) и целое (учебный курс). Отметим, что напряжение создания ПО для РО, имеющееся в средней школе, в высшей снимается за счет богатства авторских курсов и предоставления свободы развития. Для ГРОМ инвариантом является ПОНИМАНИЕ учащегося. Это означает, что единицей учебного действия ИОС-Flint является путь, отвечающий за взаимодействие целого и частей.

Итак реализуемая ИОС-Flint не контролирует знания, а помогает учащемуся, обеспечивая непрерывное обучение, причем

разрывное во времени. Грубо говоря, система обеспечивает учащегося «нелинейным» взаимодействием с учебным материалом различных авторских курсов в рамках: конкретной цели учебного курса; общей цели – или интеллектуальные прорывы, или восстановление, или понимание в отношении обобщающих смыслов понятий.

Адаптивность учебного материала добывается исследованием на нем *отношения толлерантности* [15], формируемой системой на основе динамической модели учащегося.

Заключение

Во-первых, разум способен на большее, чем ему обычно дают шанс достичь в традиционной образовательной системе. Во-вторых, ИС обеспечивают обстановку сложности, конструктивная культура ее вбирает. Этим мы можем воспользоваться посредством ИОС.

Наконец, последнее. Провозглашенный гуманитариями «переход от мира науки к миру жизни» посредством синергетики достиг лона науки. И это обнадеживает.

Литература

1. Королев Л.Н. «Компьютерное» образование, состояние дел и перспективы. //Вестник Московского университета. Серия 15, ВМиК, 1992, №2.

2. Поспелов Д.А. Десять «горячих точек» в исследованиях по ИИ.//Интеллектуальные системы Том 1. М.: МГУ, 1996.

3. Левитин К.Е. Прощание с АЛГОЛом. М.: Знание, 1989.

4. Карлащук В.И. Обучающие программы. М.: Солон-Р, 2001.

5. Средства дистанционного обучения. Методика, технология, инструментарий. С-Пб.: БХВ-Петербург, 2003.

6. Интеллектуальное компьютерное место учащегося для обучения информатике. Грант РФФИ 000100713, 2000-2002, рук. Трифонов Н.П., исп. Громыко В.И.

7. НИТ и базовое обучение информатике в высшей школе. Грант РФФИ 030100339, 2003-2005, рук. Мальковский М.Г., исп. Громыко В.И.

8. Прокл. Комментарий к первой книге «начал» Евклида. М.: Греко-латинский кабинет, 1994.

9. Громыко В.И. Эволюция разума к ноосфере (роль информатики). //Синергетика. Труды семинара. Том 7. М.: МГУ, 2004.

10.Громыко В.И., Мальковский М.Г. Интеллектуальные обучающие системы для базового обучения информатике. Научно-методический семинар по информатике «Актуальные проблемы информатики в современном российском образовании». М.: МГУ, 2004.

11.Громыко В.И.,... Интеллектуальные обучающие системы для базового обучения информатике (реализация). Научно-методический семинар по информатике «Актуальные проблемы информатики в современном российском образовании». М.: МГУ, 2004.

12.М. Брой. Информатика. Основополагающее введение. Ч.-1,2,3,4. М.: Диалог-МИФИ, 1996-98.

13.Брой М., Румпе Б. Введение в информатику: сборник задач. Структурированное собрание задач с образцами решений. М.: Научный Мир, Диалог-МИФИ, 2000.

14.Graham R.L., Knuth D.E., Patashnik O. Конкретная математика. М.: Мир, 1998.

15.Шрейдер Ю.А. Равенство, сходство, порядок. М.: Наука, 1971.

16.Николис Г., Пригожин И. Познание сложного. М.: УРСС, 2003.

17.Г.-Г.Гадамер. Актуальность прекрасного. М.: Искусство, 1991.

18.Давыдов В.В. Теория развивающего обучения. М.: ИНТОР, 1996.

Ершов Н.М.

Бисекция ориентированных графов

Введение

Задача разрезания графа на подграфы является классической задачей в теории оптимизации параллельных вычислений [1,2]. В простейшей своей форме эта задача носит название задачи бисекции графа и формулируется как поиск такого разбиения взвешенного графа на два равных по весу подграфа, которое минимизирует суммарный вес дуг, связывающих вершины из разных подграфов. В последние десять лет было разработано несколько эффективных приближенных методов решения этой задачи, таких как спектральные [3] и многоуровневые методы [4]. В настоящее время написан ряд систем, реализующих данные методы, среди которых наиболее известными являются системы Metis [5] и Chaco [6], позволяющие в разумное время проводить бисекцию графов с числом вершин порядка 10^5–10^6.

Наряду с классической задачей бисекции существуют и различные ее разновидности, одной из которых является задача бисекции ориентированных графов. Как известно, ориентированные ациклические графы допускают разложение вершин по уровням, так что для любых двух вершины, лежащих на одном уровне, не существует ориентированного пути, ведущего из одной вершин в другую. Если такой граф представляет собой граф алгоритма, то смысл распределения вершин графа по уровням состоит в том, что операции алгоритма, соответствующие вершинам одного уровня, не зависят друг от друга и поэтому могут выполняться параллельно. В этом случае задача бисекции формулируется следующим образом. Для данного взвешенного ориентированного ациклического графа требуется найти такое разбиение его на два подграфа, которое 1) разбивает каждый уровень данного графа на равные по весу части; 2) и минимизирует суммарный вес дуг, связывающих вершины из разных подграфов. Видно, что в такой постановке задача бисекции отличается от классической постановки только понятием сбалансированности разбиения. Если в классической постановке сбалансированность понимается глобально, то в задаче бисекции ориентированных графов сбалансированность рассматривается по уровням исходного графа, т.е. в некотором смысле локально. К сожалению, адаптировать вышеупомянутые методы, решающую классическую задачу бисекции, для решения задачи бисекции ориентированных графов оказывается или

достаточно сложно или даже невозможно, так как в них существенно используется та самая глобальность в понимании сбалансированности разбиения. Тем не менее, в настоящей работе предлагается метод, который за счет специальной модификации заданного ориентированного графа сводит задачу бисекции этого графа к классической задаче бисекции, для решения которой оказываются применимы все вышеупомянутые методы и программные системы. Работа предлагаемого метода иллюстрируется результатами численных экспериментов.

Постановка задачи бисекции в классической форме

Пусть задан взвешенный граф $\Gamma = <V,N,\sigma,E,M,\omega>$, в котором

- V – множество вершин;
- N – количество вершин;
- $\sigma = \{\sigma_i \in \mathbf{N}, i = 1,...,N\}$ – вектор весов вершин;
- $E \subseteq V \times V$ – множество дуг;
- M – количество дуг;
- $\omega = \{\omega_i \in \mathbf{N}, i = 1,...,M\}$ – вектор весов дуг.

Бисекцией графа Γ назовем разбиение множества его вершин на два непересекающихся подмножества. Бисекцию будем задавать вектором x, имеющим элементы вида

$$x_i = \begin{cases} 1, & \text{если } i\text{-ая вершина принадлежит 1-ому подмножеству} \\ -1, & \text{если } i\text{-ая вершина принадлежит 2-ому подмножеству} \end{cases}$$

Множество всех таких векторов обозначим за X.

Назовем бисекцию сбалансированной, если суммарные веса двух подмножеств вершин, определяемых этой бисекцией, примерно равны. Строгое понятие сбалансированности определяется следующим образом. Заметив, что скалярное произведение (σ, x) определяет разность весов двух частей бисекции, примем за меру сбалансированности квадрат этого произведения

$$\beta(x) = (\sigma, x)^2 = x^T \Sigma x. \qquad (1)$$

Здесь Σ – матрица, определяемая соотношением $\Sigma = \sigma\sigma^T$. Таким образом, назовем бисекцию x сбалансированной, если вектор x доставляет минимум квадратичной форме $\beta(x)$.

Размером разбиения назовем суммарный объем дуг, соединяющих вершины из разных подграфов. Как известно [1], эта величина выражается с помощью другой квадратичной формы

$$\delta(x) = \frac{1}{4} x^T A x \qquad (2)$$

Здесь A – матрица Лапласа графа Γ, имеющая элементы вида

$$a_{ij} = \begin{cases} -\omega_k, \text{ если } i\text{-ая и } j\text{-ая вершины соединены } k\text{-ой дугой,} \\ 0, \text{ если } i\text{-ая и } j\text{-ая вершины не соединены дугой,} \\ \text{степени } i\text{-ой вершины, если } i = j. \end{cases}$$

Таким образом, задача бисекции неориентированного графа в классической постановке звучит так: *для заданного графа найти сбалансированное разбиение минимального размера.*

Отметим, что эта задача является по сути задачей минимизации двух квадратичных форм (1) и (2) на множестве X

$$\begin{cases} \beta(x) = x^T \Sigma x \to \min \\ \delta(x) = \frac{1}{4} x^T A x \to \min \end{cases} \qquad (3)$$

притом, что минимизация первой формы является основным критерием: разбиение должно быть прежде всего сбалансированным, а уже потом – быть минимального размера.

Постановка задачи бисекции для ориентированного графа

Чтобы формализовать понятие по уровневой сбалансированности, необходимого для постановки задачи бисекции в ориентированном случае, обозначим через L число уровней в графе и введем в рассмотрение матрицу B размера $L \times N$, задающую распределение вершин графа по уровням. Ее элементы определяются следующим образом

$$b_{li} = \begin{cases} \sigma_i, \text{ если } i\text{-ая вершина принадлежит } l\text{-ому уровню,} \\ 0, \text{ если } i\text{-ая вершина не принадлежит } l\text{-ому уровню.} \end{cases}$$

Нетрудно убедиться, что для заданного вектора бисекции x l-ая компонента вектора Bx определяет разность суммарных весов вершин двух подграфов разбиения, принадлежащих l-ому уровню. Просуммировав квадраты компонент этого вектора, мы и получим искомую оценку по уровневой сбалансированности:

$$\tilde{\beta}(x) = (Bx)^2 = (Bx, Bx) = (x, B^T Bx) = (x, \Lambda x) = x^T \Lambda x,$$

где матрица $\Lambda = B^T B$ есть матрица размера $N \times N$, определяемая только графом Γ, т.е. не зависящая от разрезания x. Элементы этой матрицы определяются следующим образом

$$\lambda_{ij} = \begin{cases} \sigma_i\sigma_j, & \text{если } i\text{-ая и } j\text{-ая вершины лежат на одном уровне,} \\ 0, & \text{если } i\text{-ая и } j\text{-ая вершины лежат на разных уровнях.} \end{cases}$$

Видно, что матрица Λ получается из матрицы Σ обнулением всех элементов, соответствующих парам вершин с разных уровней.

Теперь объединим критерии сбалансированности и минимизации числа дуг, попадающих в разрезание. А именно, будем искать минимум следующего функционала

$$J(x) = \mu\delta(x) + \eta\tilde{\beta}(x) \to \min \qquad (4)$$

где μ и η два параметра. Их значения можно выбирать, например, из физических соображений, так чтобы отношение μ/η соответствовало отношению скорости передачи данных к производительности процессоров вычислительной системы. Либо, можно сделать отношение μ/η много большим единицы, тогда второе слагаемое в функционале будет играть роль штрафа.

Подставляя теперь вместо δ и $\tilde{\beta}$ их выражения, придем к представлению функционала J в виде квадратичной формы

$$J(x) = \frac{\mu}{4}x^T Ax + \eta x^T \Lambda X = x^T Cx,$$

где C есть матрица, равная $\frac{\mu}{4}A + \eta\Lambda$, определяемая графом Γ, и не зависящая от разрезания x, причем для ее недиагональных элементов справедливо следующее представление:

$$c_{ij} = \begin{cases} \frac{\mu}{4}a_{ij} < 0, & \text{если } i\text{-ая и } j\text{-ая вершины связаны дугой,} \\ \eta\lambda_{ij} > 0, & \text{если } i\text{-ая и } j\text{-ая вершины лежат на одном уровне,} \\ 0, & \text{в противном случае.} \end{cases}$$

Сведение задачи бисекции ориентированного графа к классической задаче бисекции

Представим теперь матрицу C в виде суммы трех матриц $\tilde{A}$, $\eta\Sigma$ и D. Сначала к матрице C добавим и вычтем матрицу $\eta\Sigma$:

$$C = C - \eta\Sigma + \eta\Sigma = \tilde{C} + \eta\Sigma, \text{ где } \tilde{C} = C - \eta\Sigma.$$

Так как все положительные недиагональные элементы матрицы C равны соответствующим элементам матрицы $\eta\Sigma$, то у матрицы $\tilde{C}$ все

недиагональные элементы будут либо отрицательными, либо нулевыми. Матрицу $\tilde{C}$ в свою очередь представим в виде суммы матриц $\tilde{A}$ и диагональной матрицы D, так чтобы каждый диагональный элемент матрицы $\tilde{A}$ равнялся модулю суммы всех других элементов, стоящих в той же строке. Таким образом, матрица C представляется в виде суммы

$$C = \tilde{A} + \eta\Sigma + D.$$

Отметим, что матрица $\tilde{A}$ по построению является матрицей Лапласа некоторого графа $\tilde{\Gamma}$, получаемого из графа Γ добавлением дуг, связывающих вершины, лежащие на разных уровнях (и не связанных дугой в графе Γ).

Соответственно, функционал J теперь можно записывать в виде суммы

$$J(x) = x^T Cx = x^T \tilde{A}x + \eta x^T \Sigma x + x^T Dx,$$

причем матрицы $\tilde{A}$, Σ и D, входящие в эту сумму, определяются только графом и не зависят от разрезания x. Легко показать, что последнее слагаемое в функционале $J(x)$ при $x \in X$ является константой (зависящей только от графа Γ), и следовательно, оно не влияет на расположение минимумов этого функционала. Из оставшихся двух слагаемых сформируем следующую оптимизационную задачу

$$\begin{cases} x^T \Sigma x \to \min \\ x^T \tilde{A} x \to \min \\ x \in X \end{cases} \qquad (5)$$

Очевидно, что поставленную таким образом задачу можно рассматривать как задачу бисекции неориентированного графа $\tilde{\Gamma}$, определяемого матрицей Лапласа $\tilde{A}$. Вместе с тем понятно, что в общем случае задачи (3) и (5) не будут эквивалентны, так как при постановке последней задачи было проведено разбиение одного критерия $J(x) \to \min$ на два, один из которых является старшим. Однако, в силу того, что нашей целью является построение приближенного решения задачи (4), то следует ожидать, что на достаточно широком классе графов решение задачи (5) будет хорошим приближением и в качестве решения задачи (4).

Таким образом, предлагается следующий подход к разбиению ориентированного графа. Граф модифицируется вышеописанным способом. Затем производится бисекция модифицированного графа, для этого предполагается использовать систему Chaco. Алгоритм, реализованный в Chaco, гарантирует, что найденное решение будет локально не улучшаемым. Однако в силу того, что задачи (3) и (5) не эквивалентны, не гарантируется, что это разбиение будет соответствовать локальному минимуму функционала $J(x)$. Стандартным

подходом в этой ситуации является использование некоторой финишной процедуры, которая должна найти локальный минимум близкий к найденному разбиению. С этой целью был разработан специальный метод минимизации квадратичной формы на множестве X – метод наискорейшего спуска.

Метод наискорейшего спуска

Рассмотрим задачу минимизации на множестве X квадратичной формы $J(x) = x^T Ax$ с симметричной матрицей A, имеющей нулевую диагональ. Для этого введем понятие локальной окрестности вектора x, которую определим как множество всех векторов, отличающихся от x значением ровно одной компоненты. Рассмотрим теперь некоторый вектор $\tilde{x}$ из локальной окрестности вектора x. Пусть эти векторы отличаются в k-ой компоненте, т.е. $\tilde{x}_i = x_i$ для всех $i \neq k$ и $\tilde{x}_k = -x_k$. Нетрудно показать, что величина $J(\tilde{x})$ выражается через известное значение $J(x) = \sum_{i=1}^{N} a_{ij} x_i x_j$ следующим образом:

$$J(\tilde{x}) = J(x) - 4\sum_{i=1}^{N} a_{ik} x_i x_k .$$

Если ввести в рассмотрение вектор $y = Ax$, то получим, что изменение величины $J(x)$ при изменении на противоположное значение k-ой компоненты в векторе x равно

$$\Delta J(x) = -4 y_k x_k .$$

С учетом этой формулы предлагается следующий алгоритм наискорейшего спуска для минимизации величины $J(x)$.

1. Некоторым образом задаем начальное приближение x^0. Для него вычисляем вектор y^0 и величину $J^0 = J(x^0) = (x^0, y^0)$. Полагаем $t = 0$.
2. Находим $k = \arg\max_{j=1,\ldots,N} x_j^t y_j^t$. Если $x_k^t y_k^t \le 0$, то конец алгоритма (достигнут локальный минимум).
3. Полагаем $x_k^{t+1} = -x_k^t$, $y_i^{t+1} = y_i^t - 2a_{ik} x_k$ $(i = 1,\ldots,N)$, $J^{t+1} = J^t - 4 y_k^t x_k^t$.
4. Полагаем $t = t + 1$ и переходим ко второму шагу.

Отметим, что предложенный метод является линейным по N (за исключением вычисления вектора y^0 на первом шаге), и поэтому достаточно быстрым. Кроме того, возможно совмещение операции вычисления вектора y^{t+1} (шаг 3 алгоритма) и поиска максимума величины $y_j^t x_j^t$ на следующей итерации (шаг 2) в одном цикле. Хотя

предложенный метод является методом безусловного спуска, в силу чего находит всего лишь локальный минимум близкий (в некотором смысле) к начальному приближению, он может использоваться как часть более сложных алгоритмов, например, генетических, или как завершающая процедура для разных эвристических методов (гарантирующая попадание в локальный минимум). В настоящей же работе этот метод применяется также для численного тестирования подхода, описанного в предыдущем пункте.

Результаты численных расчетов

Предлагается следующий подход к бисекции ориентированных графов. Для заданного ориентированного графа Γ, с помощью алгоритма описанного выше, строится модифицированный граф $\tilde{\Gamma}$ (реально происходит только перерасчет матрицы Лапласа, так как вершины графа Γ при описанной модификации не меняются). Затем производится бисекция модифицированного (неориентированного) графа $\tilde{\Gamma}$. Для этого используется программная система Chaco. На последнем этапе, используя бисекцию, сгенерированную Chaco в качестве начального приближения, с помощью вышеописанного метода наискорейшего спуска ищется окончательное решение (уже для ориентированного случая).

Для тестирования предложенного подхода использовались несколько наборов стандартных графов, такие как случайные графы, геометрические графы, двоичные деревья графы некоторых вычислительных алгоритмов, в частности графы алгоритма Гаусса-Жордана решения систем линейных уравнений, и другие.

N	J_{ave}	J_{best}	J_{min}
40	18	19.6	18
80	38	42.6	38
120	58	69.2	58
160	78	92.4	78
200	98	113.8	98
240	118	128.6	118
280	138	167.6	138
320	158	184.2	158
360	178	211.2	178
400	198	239.5	204

Таблица 1. Результаты расчетов для графов A_L, $N = 4L$

В таблице 1 приведены результаты численных расчетов для графов A_L (см. рисунок 1, L – высота графа). Для каждого из графов

размера $N = 4L$ (число вершин) производилось по 10 расчетов по предложенной выше схеме, на основании чего вычислялось среднее J_{ave} достигнутых значений функционала $J(x)$ и наилучшее значение J_{best}. Кроме того, в таблице приведены точные значения $J_{min} = 2L - 2$ минимумов $J(x)$, так как для рассматриваемого типа графов оптимальная бисекция строится тривиальным образом.

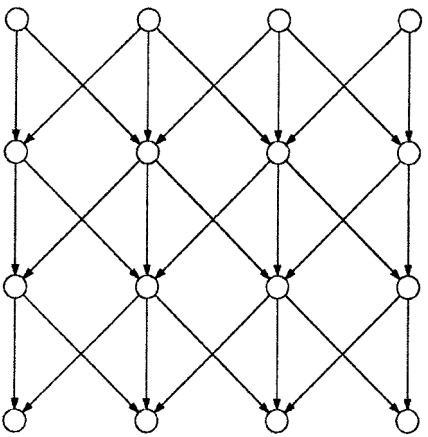


Рисунок 1. Граф A_4

В таблице 2 представлены результаты расчетов для так называемых геометрических графов. Геометрический граф $G_{d,N}$ строится следующим образом. В единичном квадрате на координатной плоскости располагаются случайным образом N вершин. Если расстояние между двумя вершинами меньше или равно d, то эти вершины связываются дугой, направленной в вершину с меньшей ординатой. Представленные в таблице результаты были получены для графов, в которых параметр d определялся как величина обратная $\sqrt{N}$.

N	J_{loc}	J_{ave}	J_{best}
50	14	17.0 (18.3)	13 (13)
100	13	14.5 (15.0)	12 (12)
150	27	22.8 (24.1)	16 (17)
200	40	28.3 (30.1)	23 (23)
250	47	30.8 (32.1)	23 (25)
300	66	56.3 (58.3)	43 (44)
350	88	54.8 (58.3)	43 (50)
400	131	54.4 (58.5)	41 (43)

450	151	68.1 (70.5)	48 (54)
500	137	75.9 (78.2)	62 (63)

Таблица 2. Результаты расчетов для геометрических графов

Так как для рассматриваемых графов точное решение задачи бисекции неизвестно, то для тестирования предложенного подхода применялся следующий алгоритм. Проводилось 100 расчетов, в каждом из которых составлялось случайное начальное разрезание, к которому применялся метод наискорейшего спуска. Из полученных таким образом результатов выбиралось наименьшее значение J_{loc}. Кроме того, в последних двух столбцах в скобках приведены найденные значения $J(x)$ до применения завершающей процедуры поиска ближайшего локального минимума (методом наискорейшего спуска). Видно, что применение такой завершающей процедуры оправдано, так как на это улучшение решения (для приведенных расчетов) тратится на порядок меньше времени, чем на первые два шага, т.е. на модификацию графа и построение бисекции модифицированного графа.

Заключение

В настоящей статье предложен новый подход к решению задачи бисекции ориентированных графов, основанный на специальной модификации заданного графа в неориентированный граф и последующего решения задачи бисекции неориентированного графа с использованием программной системы Chaco. Кроме того предложен метод наискорейшего спуска для решения задачи минимизации квадратичной формы на множестве векторов, состоящих из ± 1, к которой сводится, в частности, и задача бисекции ориентированных графов. Этот метод используется в данной работе как завершающая процедура в предложенном подходе к бисекции орграфов, так и для тестирования этого подхода.

Приведены результаты численных расчетов, показывающие эффективность предложенной схемы. Для тех задач, в которых известно точное решение, предложенный подход дает оптимальные (или близкие к оптимальным) решения. Для задач же с неизвестным точным решением, предложенный подход дает решения лучшие, чем простой перебор локальных минимумов, полученных методом наискорейшего спуска.

Литература

1. Pothen A. Graph partitioning algorithms with applications to scientific computing //Keyes D.E., Sameh A.H., and Venkatakrishnan V., eds., Parallel Numerical Algorithms. Kluwer Academic Press, 1995

2. Ershov N.M., Popov A.M. Evolutionary Stochastic Model for Optimization of Parallel Computations //Proc.of the 1st International Conference EvCA-96, Moscow 1996, c.222 -231

3. Simon H., Sohn A., Biswas R. HARP: A fast spectral partitioner //The Ninth ACM Symposium on Parallel Algorithms and Architectures, Newport, Rhode Island, June 1997

4. Karypis G., Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs //SIAM Journal on Scientific Computing, 20(1): 359-392, 1999

5. Karypis G., Kumar V. METIS 4.0: Unstructured graph partitioning and sparse matrix ordering system //Technical report, Department of Computer Science, University of Minnesota, 1998.

6. Hendrickson B., Leland R. The Chaco User's Guide Version 2 //Sandia National Laboratories, Albuquerque NM, 1995.

Трошин С. В.

Консолидация и предобработка информации из журналов регистрации ВС для систем обнаружения вторжений [1]

В статье рассматриваются вопросы сбора исходных данных из журналов регистрации системами обнаружения вторжений: основные подходы к определению набора журналов регистрации (лог файлов) и интересующих систему параметров. Рассмотрены методы промежуточного представления и передачи данных. Предложены основные концепции архитектуры системы консолидации записей системных журналов регистрации. Проведен анализ существующих технологий реализации основных компонентов системы консолидации логов с целью выбора оптимальных для реализации системы.

Вступление

Уязвимость (vulnerability) информационной системы – это любая характеристика информационной системы, использование которой нарушителем может привести к угрозе. **Угроза (threat)** – потенциально возможное событие, действие, процесс или явление, которое может вызвать нанесение ущерба ресурсам системы. Событие – минимальная единица, которой оперируют средства защиты. События в узле могут быть представлены с помощью двух составляющих – действия и адресата. **Действие** – это шаги, предпринимаемые субъектом системы для достижения некоего результата. **Адресат** – это логический или физический компонент системы.

Атакой (attack) [1]на информационную систему называется действие или последовательность связанных между собой действий нарушителя, которые приводят к реализации угрозы путем использования уязвимостей этой информационной системы.

IDS. Принципы работы.

Системы выявления атак (IDS) решают задачу мониторинга информационной системы на сетевом, системном и прикладном уровне с целью выявления нарушений безопасности и оперативного

[1] работа поддерживается грантами РФФИ 05-01-00744а и РФФИ 03-01-00745.

реагирования на них. Сетевые IDS используют в качестве источника данных для анализа сетевые пакеты, IDS системного уровня (хостовые — host based) анализируют записи журналов аудита безопасности ОС и приложений. При этом методы анализа (выявления атак) остаются общими для всех классов IDS.

Было предложено [2, 3] немало различных подходов к решению задачи выявления атак (в общем случае речь идет о злоумышленной активности, включающей в себя помимо атак, также действия, выполняемые в рамках предоставленных полномочий, но нарушающие установленные правила политики безопасности). Однако все существующие IDS можно разделить на два основных класса: системы, использующие статистический анализ и системы, использующие сигнатурный анализ.

Статистические методы основаны на предположении о том, что злоумышленная активность всегда сопровождается какими-то аномалиями, изменением профиля поведения пользователей, программ и аппаратуры.

Но основным методом выявления атак, используемом в большинстве современных коммерческих продуктов, является сигнатурный анализ. Относительная простота данного метода позволяет с успехом использовать его на практике. IDS, применяющие сигнатурный анализ, обычно ничего не знают о правилах политики безопасности, реализуемых МЭ, (поэтому в данном случае речь идет не о злоумышленной активности, а только об атаках). Основной принцип их функционирования — сравнение происходящих в системе/сети событий с сигнатурами известных атак — тот же принцип, который используется и в антивирусном ПО.

Анализ журналов регистрации

Ранние системы обнаружения вторжений работали по принципу периодического просмотра собранных системой лог файлов аудита и приложений. Большинство современных систем используют эту же технику анализа логов на хосте для получения необработанных данных о событиях. Хостовые логи, а именно лог аудита, системный лог и лог приложений обычно обеспечивают легкий доступ к информации о поведении системы, не подверженной вторжениям.

К тому же, логи, сгенерированные более высокоуровневыми сущностями часто объединяют в себе множество низкоуровневых событий (так, например, одна запись лога HTTP приложения отражает множество системных вызовов).

Общие особенности подхода

Тем не менее, некоторые факты затрудняют использование таких логов. Главным недостатком является то, что достоверность логов на атакованном хосте может быть под сомнением, особенно сгенерированных после момента попытки вторжения. Во-вторых, восстановление контекста разнесенных частей события может быть довольно трудоемким, особенно в случае взаимодействия нескольких распределенных процессов. И, наконец, качество информации, хранимой в этих логах обычно недостаточно: записи не включают критические данные, при этом включая большие объемы бессмысленных деталей.

Однако работы в этом направлении активно ведутся. Одним из немаловажных направлений развития техники выявления вторжений путем анализа журналов является корреляция [4] данных из журналов между собой, а так же корреляция журналов с данными, полученными от наблюдения за сетевой активностью.

Корреляция (event correlation) - это поиск взаимосвязей между разнородными данными. Модуль корреляции в IDS не только автоматизирует процесс сопоставления разнородных данных, но и проводит анализ воздействия вторжений на ресурсы. Корреляция способствует более эффективному обнаружению, а так же позволяет получать информацию, необходимую для последующих предотвращений и реакций. Но основной ее задачей является снижение процента ложных срабатываний. Для достижения поставленных целей большинство решений собирают логи с разных точек на одном хосте, где далее, используя механизмы редукции и корреляции, выявляют возможные вторжения.

Подход сверху - вниз(Top – down)

Во-первых, требуется определить логи, в которых могут содержаться следы атак. Для этого проще всего проанализировать известные атаки. Так же не сложно теоретически определить логи, которые содержат информацию обо всем классе атак. Подход прост, так как не требуст детального анализа огромных лог файлов, для большинства атак этот анализ может быть проведен на теоретическом уровне без учета деталей реализации, построения моделей и, тем более, без реализации атак. В Таблица 2 представлены логи, отвечающие (согласно [4]) за соответствующие классы атак.

Атака	Лог								
	Sys log	Firewall	Netflow	TCP	DNS	Auth	Web	Mail	FTP
Dictionary	+	+	+	+		+	+	+	+
FTP-Write	+			+		+			+
Imap	+	+	+	+				+	
Named	+		+		+				
Phf	+			+			+		
Sendmail	+	+	+	+	+	+		+	
Xsnoop	+		+						
Apache2	+	+	+	+			+		
Back	+			+			+		
Mailbomb	+	+	+	+				+	
SYN Flood	+	+	+	+	+				
Ping of Death		+	+	+					
Process Table		+	+	+				+	
Smurf			+	+					
Udpstorm			+	+	+				

Таблица 2

Например, атакующий, использующий атаку Winnuke (DoS атака против системы DNS), для определения, уязвима ли система, может пойти по следующему пути: Сначала использовать "nslookup" для определения DNS сервера. Затем пропинговать сервер, чтобы убедиться в его работоспособности. И, наконец, просканировать порт 139 (NetBios) для определения, активна ли Windows система. Для определения этой атаки нам следует рассмотреть следующие логи: DNS, NetFlow и syslog.

Подход снизу – вверх (Bottom-up)

В рамках данного подхода информация, необходимая для определения наличия конкретных атак, собирается из разных логов. Атаки выявляются путем последовательного анализа, начиная с простых событий, таких как неудачные попытки авторизации или множественные записи о пинговании. Как только в одном из логов обнаруживается аномалия, следующим шагом является анализ остальных логов за этот же временной период. Для определения следующего анализируемого лога используются деревья решений. Естественно, главным преимуществом подхода является возможность обнаруживать новые атаки. В рамках этого подхода так же применяется механизмы агрегации для определения и исключения записей нормального поведения из логов, таким образом, для корреляции остается только подозрительная активность.

Подход сверху – вниз для корреляции логов

Существует много независимых логов, хранящих информацию для их собственных нужд. Бывает, они плохо структурированы, и между

логами встречаются пропуски информации. Можно выбрать, например, следующие логи: Netflow, Firewall, Syslog, TCPdump, DNS, authentication log, web log, mail log и FTP лог, и рассмотреть, как влияют на них известные атаки.

Вместо детального изучения уязвимостей системы, каждая из которых может быть реализована несколькими разными путями, можно проанализировать конкретные атаки и их влияние на различные логи. Атаки можно разделить на две группы: ограничивающие доступ остальным (путем сильной загрузки или неправильными запросами) и получающие доступ для себя (обычно для злонамеренного использования).

Атак встречается довольно много, по край мере одна на каждую уязвимость. Таблица 2 отражает, на какие логи влияют атаки. Рассматривая более детально, как атаки влияют на конкретные логи, можно выделить два важных лога: syslog и NetFlow, именно их следует просматривать в первую очередь. Следует заметить, что анализ журнала syslog не такая уж тривиальная задача, так как он содержит информацию из нескольких других системных логов и обычно размеры этой информации довольно внушающие. Здесь можно использовать методы Data Mining для выделения важной информации.

Так же следует обратить внимание и на остальные логи, как было сказано ранее, корреляция с данными из этих логов может повысить вероятность обнаружения атаки.

Система консолидации логов

Реализуя на практике предложенный выше подход, а именно следующие механизмы управления записями о событиях:

1. Нормализация - устранение избыточной информации. Другими словами, в процессе нормализации система удаляются повторяющиеся данные (события), которые зафиксированы различными источниками
2. Агрегирование (event aggregation) - группирование однотипных событий вместе. Агрегирование облегчает отображение данных и их анализ. Агрегирование не спасает от появления сообщений об атаках, которые не несут в себе никакой угрозы. Нужна более интеллектуальная обработка событий, которую дает механизм корреляции.
3. Корреляция (event correlation) - поиск взаимосвязей между разнородными данными. Например, модуль корреляции в IDS не только автоматизирует процесс сопоставления разнородных данных, но и сам проводит анализ воздействия вторжений на наши ресурсы.

Следует начать с объединения всех интересующих нас записей лог файлов в едином месте и с единым интерфейсом доступа, не

зависящим от конкретного лог файла и его вида. Поставленную задачу решает подсистема консолидации логов – система сбора и предобработки информации из логов для дальнейшего анализа. Предлагается использовать систему консолидации логов со следующей архитектурой, представленной на Рисунок 1. Схема системы консолидации логов

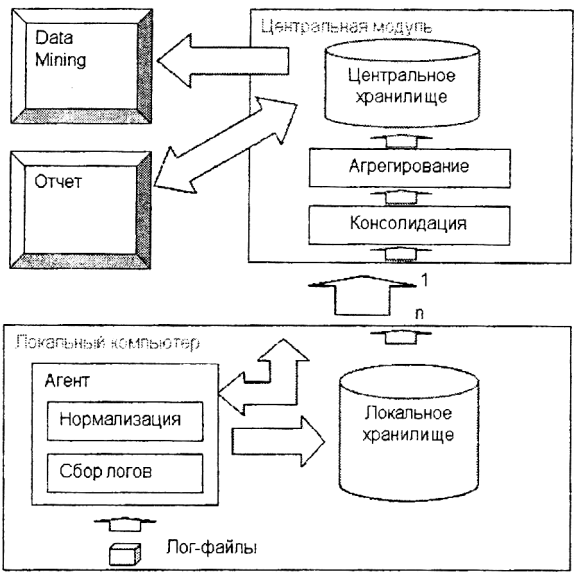


Рисунок 1. Схема системы консолидации логов

Система состоит из 2ух частей: первая часть, работающая на центральном компьютере – серверная часть (сервер или центральный модуль), вторая часть - агент, работающий на компьютере, с которого собираются записи лог файлов. При этом серверная часть представлена в системе в единичном экземпляре, а агентов может быть произвольное количество, причем агенты могут работать на разных платформах и собирать информацию из разных лог файлов (предполагается под каждую ОС создание отдельного агента, собирающего данные из логов этой ОС).

Собранную информации агенты сохраняют в своем локальном хранилище и периодически (или по запросу сервера, например) передают ее в центральный модуль, где она сохраняется в едином центральном хранилище.

В момент сбора информации агентами может быть проведена первичная нормализация по заранее известным критериям или настройкам.

Центральный модуль отвечает за консолидацию и агрегирование полученных данных.

Более формально архитектура мультиагентной системы консолидации журналов регистрации имеет следующие компоненты:

Агенты

- Агент автоматически собирает информацию о событиях из различных журналов регистрации на целевом компьютере (например, из логов DNS log, IIS log,DB log System log и т.д.), в универсальный формат данных и затем нормализует их.
- Нормализованные данные из логов (журналов регистрации) сохраняются в локальном хранилище. Имеется возможность передать нормализованные данные прямо в центральное хранилище в зависимости от запроса центрального модуля, при этом эти логи не будут сохраняться в локальном хранилище после передачи.
- Имеется возможность включать в систему новые агенты, работающие на других платформах и для новых видов логов.
- Имеется возможность модифицировать агенты отдельно от системы с целью добавления компонентов для чтения новых логов в рамках платформы, где используется агент или удалить ненужные компоненты.

Центральный модуль (сервер)

- Опрашивает агентов с целью получения собранных ими записей из журналов регистрации, находящихся в локальных хранилищах агентов, и перемещает записи в центральное хранилище.
- Группирует события по группам и категориям и переводит в базу данных – центральное хранилище.
- Содержит набор фильтров для представления событий по требованию анализа или для построения отчетов.
- Обеспечивает защищенный доступ к системе.

При реализации предложенной системы требуется иметь четкое представление по следующим пунктам:

1. Необходимо определить интерфейс взаимодействия центрального модуля и агентов – выбрать механизм передачи данных и выработать единый формат представления передаваемых записей лог фалов – событий.
2. Необходимо выбрать правильный тип базы данных, отвечающий всем требованиям поставленной задачи: производительности, удобству написания запросов для работы с описанной структурой представления лог файлов, безопасностью.

Механизм передачи данных

Под механизмом передачи данных здесь понимается механизм межпроцессорного взаимодействия (взаимодействия между центральным модулем и агентами), а так же метод представления записей журналов регистрации, передаваемых по сети. Решая задачу выбора метода межпроцессорного взаимодействия, выбор автоматически падает на механизм сокетов.

При этом, занимаясь проблемой планирования опроса агентов, нельзя не коснуться вопроса выбора объема данных, которые буду передаваться из локального хранилища агента в центральный модуль единовременно. Здесь могут быть несколько вариантов:

1. Агенты накапливают фиксированный объем информации или фиксированное количество записей журналов и затем «выгружают» их на центральный модуль.
2. Агенты через определенные промежутки времени (которые могут устанавливаться с сервера) или по запросу центрального модуля выгружают все имеющиеся у них в локальном хранилище данные, не зависимо от их объема.
3. Агент может передавать на центральный модуль данные о каждой новой записи в лог файле, не помещая ее в свое локальное хранилище.

Рассмотрим каждый из них в отдельности.

Минус первого подхода заключается в том, что при небольшой активности на клиенте, необходимое количество записей лог файлов может быть достигнуто за время, неудовлетворительное для последующих систем анализа собранных лог файлов, например для IDS. Среди собранных записей могут находиться и крайне важные для определения наличия атаки в системе, но они не будут вовремя получены модулем выявления вторжений, так как агент будет ожидать, пока не наберется достаточный объем данных. Однако этот подход к опросу агентов при всех остальных равных требует меньше всего ресурсов на суммарную полную передачу собранных данных.

Второй подход может показаться самый логичным, но минусом его являются лишние накладные расходы на передачу в случае слишком маленького промежутка времени или слишком незначительной активности системы.

Третий подход самый расточительный, но в случае использования системы консолидации логов, как источника данных для систем обнаружения вторжений с жесткими временными требованиями, может оказаться единственно пригодным.

Учитывая сравнительную простоту реализации всех трех вариантов, предлагается создать систему, агенты которой могут

настраиваться центральным модулем на работу по любому из перечисленных вариантов.

Формат представления данных

В предложенной архитектуре системы консолидации логов только агенты обладают информацией о формате конкретных лог файлов, с которыми они работают. Поэтому только агенты могут представить эту информацию в неком универсальном для всех логов формате. И именно в таком формате она будет передана по сети центральному модулю. Учитывая то, что центральный модуль будет представлять записи лог файлов в формате XML и помещать в итоге в XML базу данных, логично переводить записи в XML представление сразу на агенте, что избавит систему от лишнего преобразования из одного формата в другой.

Изменяющаяся структура лог файлов

Реляционные базы данных чрезвычайно хороши для хранения высоко структурированной информации и не слишком хороши для управления полуструктурированными данными[5].

Полуструктурированные данные - это данные, которые обладают некоторой структурой, но не являются жестко структурированными. Примером полуструктурированных могут служить как раз записи разных лог файлов, каждый лог файл содержит свой набор полей, и соответственно свой набор записей, состоящих из набора значений, типы которых удовлетворяют типам полей лог файла. Полуструктурированные данные очень легко хранить как XML.

Но полуструктурированные данные сложно хранить в реляционной базе данных, поскольку в этом случае возникает либо много различных таблиц (что означает многочисленные соединения и продолжительное время поиска), либо единственная таблица с множеством пустых колонок. Дело в том, что существует некоторое несоответствие между тем как может быть представлено отношение и элемент XML, поскольку элемент XML отличается большей гибкостью по сравнению с отношением. Реляционное отношение (строка в таблице) состоит из постоянного числа *характеристик* (иными словами, *столбцов*) и каждая характеристика содержит один фрагмент данных, тогда как элемент XML может содержать произвольное число характеристик (в качестве *субэлементов* или *атрибутов*), а каждый субэлемент может содержать несколько фрагментов данных.

Реляционное отношение можно легко представить в коде XML, но загрузка данных из произвольного документа XML в реляционную базу данных может оказаться затруднительной[6]. В связи с этим может возникнуть необходимость модифицировать реляционную базу данных, чтобы она допускала применение более разнообразных структур

данных, чем при обычных обстоятельствах. Например, чтобы представить разное число вхождений субэлементов в элемент одного и того же типа, может потребоваться создать отдельное отношение для представления этого набора данных и связать его с первоначальным отношением. Это может привести к значительному усложнению реляционной базы. Хотя полуструктурированную информацию можно преобразовать в реляционную модель, возможны накладные расходы на исполнение, особенно при выполнении запросов. NXD (native XML database) или XED (XML-enabled database) с возможностью индексации XML-данных великолепно подходят для такого типа информации. Этот факт заставляет задуматься над тем, что настало время перейти к использованию СУБД XML.

СУБД XML обеспечивает непосредственный доступ к документам и фрагментам документов XML и обладает способностью выполнять запросы к информации, представленной в этих документах и фрагментах.

Однако на этом очевидном пути возникает довольно много проблем. Прежде всего, неясно даже, что следует понимать под термином «XML-база данных» Это могут быть только базы данных, которые сохраняют XML в его родном формате или же еще и базы данных, просто сохраняющие и получающие XML-данные, независимо от того, каким образом данные сохраняются в ней. Если XML-документ не сохранен внутри базы данных именно как XML-документ, назовем это «база данных с поддержкой XML» (XML-enabled database). Если же XML-документ фактически сохранен внутри как XML, такую базу данных мы будем называть «чистой XML-базой данных» (native XML database).

Имеется ряд причин использовать существующие базы данных и сохранять XML-данные, даже не в родной XML-форме. Во-первых, обычные реляционные и объектно-ориентированные базы данных известны, в то время как чистые XML-базы данных в новинку. Во-вторых, в результате знакомства с реляционными и объектно-ориентированными базами данных известно их поведение, особенно в отношении производительности. По общему мнению, реляционные и объектно-ориентированные базы данных - это гарантированно безопасный выбор. Базы данных с поддержкой XML прекрасно работают с XML-данными. Эти базы данных обычно разделяют XML-документ на части, которые соответствуют схеме базы данных, и сохраняют эти части как обычные данные, а при запросе соединяют их.

Но есть и отрицательные стороны использования реляционных и объектно-ориентированных баз данных для хранения XML-документов[7]. Например, одна из привлекательных особенностей XML - это иерархическая организация, которую разрушает табличная организация обычных баз данных. Реляционные базы данных должны

отобразить XML в реляционных таблицах и поэтому сглаживают XML-структуры в простые строки и столбцы, но все это очень трудно поддерживать. Кроме того, преобразование XML для хранения в обычной базе данных требует значительной мощности, особенно для больших или сложных XML-документов. Причем проблема производительности может обостриться в самый неподходящий момент, когда используется одна из сильных возможностей XML - динамическое создание, например, Web-страниц или отчетов из XML-документов.

Приверженцы чистых XML-баз данных уверены, что для преодоления этих проблем необходимо сохранять в базе данных чистые XML-документы. Странно, но одна из основных проблем чистых XML-баз данных - все та же производительность. Некоторые аналитики прогнозируют проблемы с поиском информации, которая может находиться в конце большого документа. Без дополнительных механизмов чистая XML-база данных должна была бы пройти через целый документ, чтобы завершить поиск. (Реляционные и объектно-ориентированные базы данных избегают этой трудности, так как разделяют документ на меньшие части, каждая из которых доступна для поиска.) Однако эта трудность преодолима. Дополнительные возможности индексации компенсируют проблему поиска информации, которая может находиться в конце большого документа. Производительность - это один из основных вопросов при создании системы консолидации логов. Хранение чистого XML устраняет ненужные операции преобразования. Актуальной остается только задача выбора конкретной XML БД.

Заключение

В статье были рассмотрены различные вопросы, связанные с построением систем консолидации журналов регистрации, используемых как часть систем обнаружения вторжений, а так же определения важности некоторых лог файлов для системы обнаружения атак. Однако в настоящее время еще многое предстоит сделать.

Дальнейшие планы подразумевают уточнение деталей архитектуры системы, ее моделирование, с целью определения узких мест, ее корректировку и реализацию.

Планируется определить максимально возможное число одновременно работающих агентов, исходя из этого, определить средние и пиковые нагрузки на базу данных и сеть.

Исходя из этих данных, планируется выбрать реализацию и построить схему базы данных, удовлетворяющую поставленной задаче.

Определить формат представления записей журналов регистрации для передачи по сети.

Провести серию экспериментов по моделированию работы центрального модуля с базой данных с требуемыми нагрузками. Провести моделирование работы центрального модуля, отражающее разные стратегии опроса агентов с целью определения наиболее оптимальной с точки зрения производительности. Выбрать наиболее оптимальную стратегию опроса, отвечающую требуемым параметрам производительности и задержкам между возникновением записи о событии и его поступлением на центральный модуль.

Планируется продумать защищенность самого центрального модуля и агентов от атак. Например, если злоумышленник узнает параметры его работы с агентами, то простая DoS атака, направленная на него, может вызвать массовую потерю или задержку поступления записей различных журналов, пересылаемых агентами, что в свою очередь может повлечь за собой необнаружение других атак.

Литература

1. А.Лукацкий. Обнаружение атак. СПб.: БХВ-Петербург, 200. 624с.

2. Theuns Verwoerd, Ray Hunt. Intrusion Detection Techniques and Approaches // Department of Computer Science University of Canterbury, New Zealand, 2002, с. 2-14.

3. Kathleen A. Jackson. Intrusion detection system (ids) product survey // Distributed Knowledge Systems Team Computer Research and Applications Group Computing, Information, and Communications Division Los Alamos National Laboratory Los Alamos, New Mexico USA, 1999: с. 6-22.

4. Cristina Abadyz, Jed Taylory, Cigdem Senguly, William Yurcik. Log Correlation for Intrusion Detection: A Proof of Concept // Department of Computer Science, University of Illinois at Urbana-Champaign, 2003: с. 3-6.

5. Ли Доддз. XML и базы данных? Доверьтесь своей интуиции. [HTML] (http://www.iso.ru/journal/articles/206.html)

6. Марк Грейвс. Проектирование баз данных на основе XML. М.: Вильямс, 2002. 640с.

7. Даниил Фертф. Где хранить надежду электронной коммерции? // Сетевой. 2001. №1. [HTML] (http://www.setevoi.ru/cgi-bin/text.pl/magazines/2001/1/58)

Розинкин А.Н.

Модели представления электронных писем в обучаемых системах классификации электронной почты[1]

Введение

Один из актуальных вопросов компьютерной безопасности сегодня – активное развитие несанкционированных массовых рассылок электронной почты. По различным данным объем спама достигает 90% от общего количества всех электронных сообщений. Это приводит к излишним нагрузкам на сети передачи данных и почтовые сервера, трате времени пользователей на сортировку и удаление писем. Часто спам используется в качестве канала для распространения вирусов.

Проблема борьбы с несанкционированными рассылками стоит достаточно давно, поэтому к настоящему времени созданы и функционируют различные решения для предотвращения получения спама. В последнее время активно развиваются адаптивные обучаемые системы классификации электронной почты, основанные на методах статистики и искусственного интеллекта.

Формально, задача классификации электронной почты – это классическая задача информационного поиска. По заданному обучающему набору документов, для каждого из которых известен его класс, построить алгоритм определения класса у нового документа аналогичного происхождения.

В основе алгоритма классификации лежит определенная модель документа. Под моделью понимается набор признаков документа, которые учитываются алгоритмом при его обработке. В данной статье рассматривается развитие модели представления электронных писем для алгоритма классификации.

Опираясь на результаты предыдущих исследований[1, 2] среди методов классификации, применяемых для задачи классификации электронной почты, в качестве перспективного можно выделить метод опорных векторов, основанный на теории потенциальных функций[3, 4]. Этот метод был выбран в качестве базового алгоритма классификации, используемого в экспериментах при разработке модели представления электронных писем.

---

[1] Работа поддерживается грантом РФФИ # 05-01-00744а

Выбор модели представления

При выборе модели представления электронных писем рассматриваются два противоречивых аспекта, от которых зависит эффективность алгоритма классификации. Во-первых это точность классификации, и во-вторых – используемые ресурсы вычислительной системы. Чем более сложное представление электронных писем используется, тем выше качество их классификации, но в тоже время это приводит к использованию больших вычислительных ресурсов.

Для представления писем была выбрана модифицированный вариант наиболее распространенной векторной модели[5]. В терминах этой модели набор писем представляется в виде матрицы: $D = a_{ik}$, в которой a_{ik} - это весовой коэффициент *i-ого* признака, входящего в письмо *k*. Размерность этой матрицы определяется количеством писем в наборе и количеством выбранных признаков письма.

Для определения признаков, характеризующих письмо используется несколько подходов. Можно выделить два типа признаков: текстовые признаки и признаки, не связанные с текстовым содержимым. Текстовые признаки – это признаки, которые характеризуют наличие или отсутствие определенных слов в письме и возможно их взаимное положение. К нетекстовым признакам, учитывая специфику предметной области, можно отнести информацию из заголовка письма (кто, когда, откуда отправил письмо), кодировку, структуру и форматирование письма, наличие, размер и состав прикрепленных файлов, некоторые статистические данные (размер, среднее количество слов, знаки препинания, использование регистра и т.д.) [6].

Методы определения весовых коэффициентов признаков основываются на двух очевидных утверждениях: во-первых чем чаще признак встречается в письме, тем он более значимый (релевантный) для той категории, которой принадлежит письмо, и во-вторых чем чаще признак встречается во всех письмах набора, тем меньше информации он несет об отличии различных писем набора.

Были рассмотрены несколько распространенных методов определения весовых коэффициентов признаков, формирующих модель представления писем. В связи с особенностями реализации системы классификации почты, для обеспечения возможности дообучения алгоритма необходимо использовать такой метод определения весовых коэффициентов, который использовал бы только данные о конкретном письме. Т.е. данные о всей коллекции, такие как например количество объектов в коллекции, встречаемость конкретного признака в коллекции, недоступны.

Была проведена серия экспериментов с несколькими методами определения весовых коэффициентов. В качестве алгоритма классификации использовался модифицированный метод опорных векторов. Эксперименты проводились на эталонном наборе электронных писем[7]. В данных условиях результат классификации зависел незначительно от выбора метода определения весовых коэффициентов. Таким образом, была выбрана достаточно простая норма, обеспечивающая необходимую точность классификации:

$$a_{ij} = f_{ij} \Big/ \sqrt[2]{\sum_{i=1}^{m} f_{ij}^{\,2}}$$

где a_{ij} - это весовой коэффициент *i-ого* признака, входящего в письмо j, f_{ij} – частота встречаемости *i-го* признака в письме j, m – количество признаков в данном письме.

Оптимизация модели представления

Основная проблема при разработке статистических методов классификации – большая размерность пространства признаков.

В данном случае был выбран следующий подход для первоначального формирования набора признаков. Все признаки делятся на две группы: текстовые и нетекстовые признаки. Нетекстовые признаки определялись эмпирически на основании экспертных оценок. Их количество невелико и фиксировано. В частности, возможные нетекстовые признаки это:

- наличие, тип, размер прикрепленных файлов;
- кодировка, использование форматирования;
- статистика, полученная на основании анализа текста письма.

При определении модели представления электронных документов основная проблема заключается в формировании оптимального набора текстовых признаков.

Для сокращения пространства признаков обычно применяются два типа методов:

- методы выбора оптимальных признаков на основе оценки их информационной значимости;
- репараметризация, или отображение исходного пространства признаков в некоторое другое, меньшей размерности.

Были рассмотрены три наиболее эффективных[8] метода выбора набора оптимальных признаков на основе оценок их информационной значимости: Information Gain, хи-квадрат статистика и выбор признаков по частоте их встречаемости.

Information Gain – характеристика каждого признака, показывающая насколько присутствие или отсутствие данного признака в документе обучающего набора влияет на принадлежность его к тому или иному классу:

$$IG(x) = -\sum P(y_j)\log P(y_j) + P(x)\sum P(y_j \mid x)\log P(y_j \mid x) +$$

$$P(\overline{x})\sum P(y_j \mid \overline{x})\log P(y_j \mid \overline{x})$$

Хи-квадрат статистика оценивает степень зависимости данного признака и класса. Для нашей задачи с двумя классами для обоих классов эта оценка оказывается симметричной:

$$\chi^2(w, y_j) = \frac{(a+b+c+d)(ad-bc)^2}{(a+b)(c+d)(b+d)(a+c)};\ w \in x, x \in X$$

где a – количество документов из класса y_j, которые содержат признак w, b – количество документов которые содержат признак w, но не принадлежат классу y_j, c – количество документов из класса y_j, которые не содержат признак w, d – количество документов, которые не принадлежат классу y_j и не содержат признак w.

Самый простой метод отбора признаков – отбор по частоте встречаемости, основывается на предположении, что чем в меньшем количестве документов признак встретился, тем он менее информативен для определения категории документов. В экспериментах с этим методом мы также удаляли из рассмотрения слишком часто встречаемые признаки используя для этого словарь наиболее встречаемых слов.

Для каждого из этих методов определяется граница, по которой отбирается некоторое число наиболее информационно-значимых признаков.

Было показано[8], что методы выбора оптимальных признаков имеют схожие вычислительную сложность и эффективность. Наши эксперименты показывают, что эти методы недостаточно точны для существенного сокращения признаков, то тем не менее весьма эффективны для первоначального грубого отбора значимых признаков. Из рассмотренных методов наилучших результатов удалось достичь для метода выбора признаков по частоте встречаемости.

В методах репараметризации новое пространство признаков строится как трансформация оригинального пространства. В экспериментах был использован метод скрытого семантического индексирования (Latent Semantic Indexing), который основан на предположении, что существуют скрытые зависимости в использовании слов или конструкций из слов в документах и статистические методы позволяют выявить эти структуры[9]. Для построения нового набора признаков используется сингулярное разложение матрицы документов тренировочного набора[10].

С помощью сингулярного разложения матрицы документов, находится заданное количество сингулярных векторов, с помощью которых для каждого документа строится вектор представляющий его. Таким образом размерность представления можно выбирать, используя только то число сингулярных значений, какова требуемая размерность. Сингулярное разложение строится на тренировочном наборе, затем находится матрица, преобразующая вектор произвольного документа в псевдо-документ. Затем полученная матрица используется для тренировки классификатора. С помощью той же преобразующей матрицы строится представление тестового вектора в новом пространстве.

Ни один из этих методов в отдельности не подходит рассматриваемой задачи. Методы выбора оптимальных признаков недостаточно точны при значительном сокращении количества признаков, а методы репараметризации обладают слишком высокой вычислительной сложностью и при размерности пространства признаков в несколько тысяч элементов практически неприменимы.

Для решения этой задачи был предложен комбинированный подход, суть которого заключается в суперпозиции двух методов. Сначала набор признаков сокращается одним из методов выбора пространства признаков. Затем к существенно уменьшенному набору применяется алгоритм репараметризации. Такой подход позволил сократить пространство признаков на два порядка при этом практически не ухудшив точность классификации.

Эксперимент

Была проведена серия экспериментов, в которых использовались представленные методы с различными параметрами. В качестве тренировочного и тестового набора документов использовался общедоступный архив электронных писем с сайта одного из производителей антиспамовского ПО SpamAssassin[7]. Весь набор писем был разбит на несколько частей, каждая из которых последовательно использовалась в качестве тестовой, а все оставшиеся – в качестве обучающих. Затем результаты экспериментов для каждой из частей объединялись. К обучающему набору применялись

исследуемые методы уменьшения пространства признаков с различными параметрами, затем для каждого эксперимента использовался один и тот же алгоритм классификации – модифицированный метод опорных векторов. Для сравнения результатов различных экспериментов применялась стандартная процедура оценки алгоритмов, имеющих ошибки первого и второго рода с использованием ROC-кривых.

На Рис.1 самый верхний график – результат работы метода опорных векторов на всем наборе признаков, без удаления. Его можно рассматривать как эталонный. Два других графика представляют два метода сокращения признаков, для которых была выбрана одинаковая граница количества отобранных признаков.

Самый нижний серый график – выбор признаков с помощью метода оценки информационной значимости. Как видно, результаты классификации значительно ухудшились. Второй график – результат предлагаемого решения. Как видно, результаты классификации отличаются незначительно от результатов с использованием всего пространства признаков. Таким образом, удалось сократить пространство признаков на два порядка, практически не ухудшив качество классификации.

Выводы

Основная проблема при разработке статистических методов классификации – большая размерность пространства признаков.

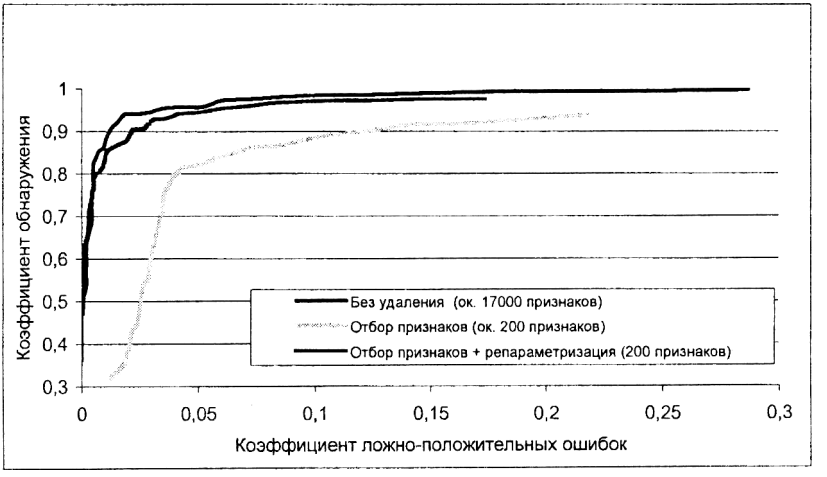


Рис.1. Сравнительная оценка алгоритмов сокращения пространства признаков

Существующие методы сокращения количества признаков плохо подходят для задачи классификации электронной почты. Методы выбора оптимальных признаков недостаточно точны при значительном сокращении количества признаков, а методы репараметризации обладают слишком высокой вычислительной сложностью и при размерности пространства признаков в несколько тысяч элементов практически неприменимы. Для решения этой проблемы был предложен комбинированный подход, суть которого заключается в суперпозиции двух методов. Сначала набор признаков сокращается одним из методов выбора пространства признаков. Затем к существенно уменьшенному набору применяется алгоритм репараметризации. Такой подход позволил сократить пространство признаков на два порядка при этом практически не ухудшив точность классификации.

Литература

1. T. Joachims, Text categorization with support vector machines: learning with many relevant features, *Proceedings of the 10th European Conference on Machine Learning (ECML 1998), April 21–23, 1998*, Chemnitz, Germany. LNCS vol. 1398, pp. 137–142.

2. H. Drucker, with D. Wu and V. Vapnik. Support Vector Machines for Spam Categorization, *IEEE Trans. on Neural Networks , vol 10, number 5*, pp. 1048-1054. 1999

3. V. Vapnik, Statistical Learning Theory. Wiley, 1998.

4. B. Scholkopf, A. Smola, Learning with kernels: Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge, 2000.

5. Salton, G. & McGill, J. (1983). An introduction to modern information retrieval. New York: McGraw-Hill

6. O. de Vel, Anderson, A., Corney, M. & Mohay, GM, Mining email content for author identification forensics. SIGMOD Record, 30(4), pp. 55-64, 2001

7. Apache Software Foundation (2004b) *The Apache SpamAssassin Public Corpus* http://spamassassin.apache.org/publiccorpus/

8. Y. Yang and J. O. Pedersen, A comparative study of feature selection in text categorization, *Proceedings of the Fourteenth International Conference on Machine Learning*, Morgan Kaufman Publishers, 1997, pp.412-420.

9. Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas and Richard A. Harshman, Indexing by Latent Semantic

Analysis, *Journal of the American Society of Information Science, Vol. 41, No. 6*, pp. 391-407, 1990

10. Wall, Michael E., Andreas Rechtsteiner, Luis M. Rocha."Singular value decomposition and principal component analysis". in *A Practical Approach to Microarray Data Analysis*. D.P. Berrar, W. Dubitzky, M. Granzow, eds. pp. 91-109, Kluwer: Norwell, MA (2003). LANL LA-UR-02-4001.

Розинкин А.Н.

Оптимизация тренировочного набора для системы классификации электронной почты на основе алгоритма опорных векторов[1]

Введение

Данная работа посвящена исследованиям, связанным с развитием методов построения фильтрации спама на основе методов интеллектуального анализа данных.

Полученные результаты экспериментов[1, 2] показывают, что среди методов классификации, применяемых для задачи классификации электронной почты, в качестве перспективного можно выделить метод опорных векторов, основанный на теории потенциальных функций[3, 4].

Пусть $D = \{(x, y) : x \in R^m, y \in \{-1;1\}\}$, где m – размерность вектора. Тогда обучающий набор – конечное подмножество $D_{train} \in D, |D_{train}| = N$ и задача классификации состоит в определении класса $y \in \{-1;1\}$ для любого нового $x \in R^m$.

Как было уже сказано, наилучшие результаты в применении к задаче классификации спама показывает метод опорных векторов, суть которого заключается в следующем.

В случае, если два множества объектов, принадлежащих различным классам, линейно разделимы в пространстве признаков, то

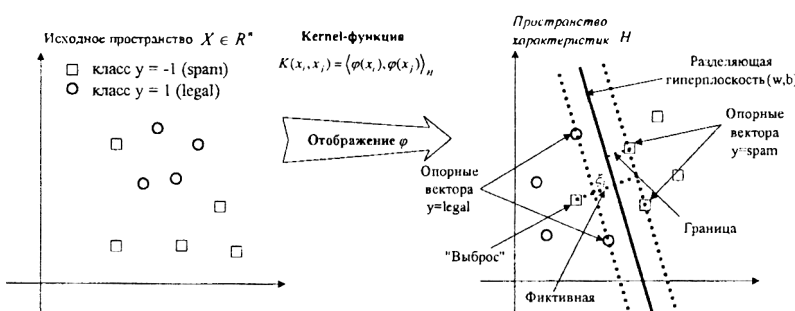


Рис.1. Метод опорных векторов

[1] Работа поддерживается грантом РФФИ # 05-01-00744а

решение состоит в отыскании гиперплоскости (см. Рис.1):

$(w \cdot x) + b = 0;\ w \in R^m, b \in R$ такой что $\frac{1}{2}\|w\|^2 \to \min$ при условии

$y_i \cdot ((w \cdot x_i) + b) \geq 1,\ i = 1,...N$

В соответствии с теорией статистического обучения Вапника во-первых, среди всех таких гиперплоскостей существует единственная, с максимальной границей между классами во-вторых, чем больше эта граница, тем функция риска для тестовых примеров будет принимать меньшие значения[3]. Построение этой гиперплоскости сводится к решению такой задачи оптимизации:

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \to \max$$

при условии $\sum_{i=1}^{N} \alpha_i y_i = 0,\ \alpha_i \geq 0, i = 1..N$ и решающая функция для тестовых примеров принимает такой вид:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i \cdot (x \cdot x_i) + b \right)$$

Здесь коэффициенты α_i отличны от нуля только для части векторов y_i, которые и называются опорными векторами. Таким образом, для классификации новых объектов необходимо хранить только часть векторов тренировочного набора, и соответствующие им коэффициенты, полученные при решении задачи оптимизации.

В случае если классы документов линейно неразделимы, задача заменяется на:

$\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \xi_i \to \min$, при условии $y_i \cdot ((w \cdot x_i) + b) \geq 1 - \xi,\ \xi \geq 0$

В этом случае решается та же задача максимизации $W(\alpha) \to \max$, но в условиях таких ограничений: $0 \leq \alpha_i \leq C$.

Таким образом, результат работы алгоритма – модель классификации, которая используется для классификации новых объектов – это набор векторов тренировочного набора, которые формируют разделяющую гиперплоскость (опорные вектора) и соответствующие им весовые коэффициенты.

Постановка задачи

Несмотря на то, что метод опорных векторов для задачи классификации электронной почты показывает хорошие результаты по сравнению с другими алгоритмами классификации, основная проблема при его использовании связана с размером и консистентностью тренировочного набора.

В работе[5] была рассмотрена проблема определения модели представления электронных писем для обучаемой системы классификации почты. В результате было выбрано векторное представление, где координаты вектора – это весовые коэффициенты набора признаков письма. Там же были предложен метод выбора оптимального набора признаков.

Вычислительная сложность алгоритма классификации на основе метода опорных векторов на этапе обучения оценивается как $O(m \cdot N2)$, где m – размерность пространства признаков, N – количество примеров для обучения. Задача выбора пространства признаков рассматривается в работе[5]. Задача выбора примеров для обучения и формирование оптимального тренировочного набора – предмет рассмотрения этой статьи.

При разработке алгоритма классификации электронных сообщений возникли следующие проблемы, на которые влияет состав тренировочного набора. Скорость обучения алгоритма квадратично зависит от количества примеров в тренировочном наборе. В реальной эксплуатации размер тренировочного набора часто бывает достаточно велик, что отрицательно сказывается на производительности системы классификации почты. Другая проблема, связанная с размером тренировочного набора – это размер модели классификации, которая является результатом работы алгоритма обучения и используется для классификации новой почты. Чем больше эта модель, тем во-первых необходимо больше места для ее хранения, и во-вторых, тем большее время необходимо для классификации новых сообщений. В таких условиях задача оптимизации размера тренировочного набора становится весьма важной.

В настоящее время в основном применяются эвристические методы для уменьшения количества примеров для обучения. Самый простой и распространенный способ – сокращение тренировочного набора путем случайного отбора части его элементов. Но такой способ не подходит для применяемого алгоритма классификации на основе метода опорных векторов, так как при случайном удалении объектов из тренировочного набора можно также удалить и объекты, которые при обучении сформировали бы опорные вектора.

Предлагаемое решение

Если посмотреть на решение в методе опорных векторов то понятно что те объекты в тренировочном наборе, которые лежат не на плоскостях, поддерживающих разделяющую гиперплоскость – они не нужны ни для обучения (построения гиперплоскости), ни для классификации. Но проблема заключается в том, что о том, какие объекты лежат вне разделяющей гиперплоскости можно узнать только после ее построения (Рис.2).

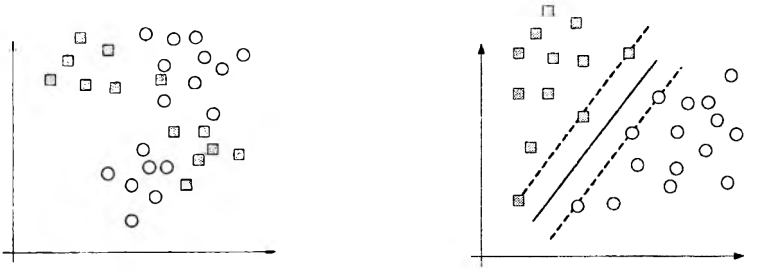


Рис.2. Построение разделяющей гиперплоскости

Если бы было возможно определить те объекты, которые будут лежать далеко от гиперплоскости еще до запуска алгоритма обучения, то их можно было бы исключить из процесса построения классификатора.

Решение основывается на следующем предположении. Если в исходном пространстве построить кластеры специального вида, такие что:

- кластер имеет максимальный радиус, такой, что кластеру принадлежат объекты только одного класса;
- кластеры не пересекаются.

Тогда предполагается что элементы, находящиеся около границы кластера с большей вероятностью будут находиться ближе к разделяющей гиперплоскости. Таким образом, удалив некоторую часть элементов из центров построенных так кластеров, мы получим сокращенный тренировочный набор, который будет меньше исходного, но при этом классификаторы, построенные на основе двух тренировочных наборах будут эквивалентны[6] (Рис.3).

Для данного метода было необходимо предложить «быстрый» алгоритм кластеризации, имеющий сложность, не большую чем сам метод опорных векторов.

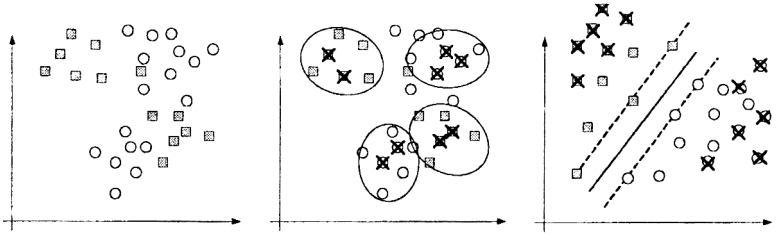


Рис.3. Кластеризация тренировочного набора

Эксперименты

Были рассмотрены три метода кластеризации: k-mean, метод последовательной кластеризации и метод случайного выбора центров кластеров и проведена серия экспериментов с использованием этих методов. В качестве тренировочного и тестового набора документов использовался эталонный общедоступный архив электронных писем SpamAssassin[7]. Весь набор писем был разбит на несколько частей, каждая из которых последовательно использовалась в качестве тестовой, а все оставшиеся – в качестве обучающих. Затем результаты экспериментов для каждой из частей объединялись. Тренировочный набор уменьшался с помощью одного из исследуемых методов, затем на получившимся тренировочном наборе строился классификатор с помощью модифицированного метода опорных векторов. Далее для каждого эксперимента для построенного классификатора оценивалось качество классификации на соответствующем тестовом наборе.

Результаты и выводы

Результаты эксперимента представлены в данной таблице:

Набор	Метод удаления	Удаленные вектора, %	Точность классификации, %			Построение модели, сек
			Общая	Спам.	Норм.	
Набор #1	Без удаления	0	97,0	87,0	99,4	2,79
	Random	36	97,9	88,8	99,3	1.56
	Series clustering	31	97,8	88,5	99,3	1,92
Набор #2	Без удаления	0	98,2	88,2	99,8	2,51
	Random	37	98,2	88,2	99,7	1,46
	Series clustering	41	98,2	88,3	99,8	1,34
Набор #3	Без удаления	0	97,8	86,5	99,6	2,91
	Random	36	97,9	88,5	99,3	1,53
	Series clustering	36	98,0	88,0	99,6	1,51
Набор #4	Без удаления	0	97,2	85,0	99,2	2,56
	Random	39	97,4	87,2	99,0	1,45
	Series clustering	38	97,2	86,8	98,8	1,40
Набор #5	Без удаления	0	97,5	86,0	99,3	3,13
	Random	28	97,8	87,3	99,4	1,84
	Series clustering	31	97,7	88,5	99,1	1,65
Весь набор	Без удаления	0	100	100	100	30,95
	Random	41	99,9	100	99,9	18,37
	Series clustering	35	99,9	99,9	99,9	15,95

Таблица 1. Результаты эксперимента

Как видно, наилучшие результаты были достигнуты с помощью алгоритма последовательной кластеризации. Для тренировочного набора, сокращенного в два раза точность классификации не ухудшается, а часто увеличивается. Это объясняется тем, что из тренировочного набора удаляется шум, или примеры, «нехарактерные» для того класса к которому они относятся. При этом на этапе построения классификатора было достигнуто в среднем двукратное увеличение производительности.

Литература

1. T. Joachims, Text categorization with support vector machines: learning with many relevant features, Proceedings of the 10th European Conference on Machine Learning (ECML 1998), April 21–23, 1998, Chemnitz, Germany. LNCS vol. 1398, pp. 137–142.

2. H. Drucker, with D. Wu and V. Vapnik. Support Vector Machines for Spam Categorization, IEEE Trans. on Neural Networks , vol 10, number 5, pp. 1048-1054. 1999

3. V. Vapnik, Statistical Learning Theory. Wiley, 1998.

4. B. Scholkopf, A. Smola, Learning with kernels: Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge, 2000.

5. Разработка и оптимизация модели представления электронных писем в обучаемых системах классификации электронной почты / Розинкин А.Н.; МГУ им. Ломоносова, ф-т ВМиК – Москва, 2005.

6. Ravindra Koggalage and Saman Halgamuge "Reducing the Number of Training Samples for Fast Support Vector Machine Classification" Neural Information Processing, Vol. 2, No. 3, March 2004, pp.57-65

7. Apache Software Foundation (2004b) The Apache SpamAssassin Public Corpus http://spamassassin.apache.org/publiccorpus/

Раздел II

Методы обработки экспериментальных данных

Певцов С.Е., Попов А.М.

Разработка алгоритма восстановления структуры пептидов по масс-спектру

Введение

Тандемная масс-спектрометрия является на сегодняшний день самым надежным инструментом для идентификации протеинов [1]. Сейчас существует несколько типов масс-спектрометров, которые предоставляют MS/MS данные с высокой точностью для определения пептидных последовательностей в протеинах. Однако определение последовательности по необработанному спектру вручную происходит очень медленно. В связи с этим наиболее популярным способом идентификации последовательности является поиск по базам данных уже известных протеинов, когда необработанный спектр сравнивается со спектрами в базе данных. Описано несколько реализаций этого подхода, самыми известными среди которых являются Sequest и Mascot [2, 3]. Это эффективные методы, но зачастую идентификация происходит неверно. Поиск в базах данных по массе и по частичной последовательности (sequence tag) предоставляет более надежные результаты [4]. Однако для неизвестных пептидов необходимо производить секвенирование (de novo sequencing) для получения (возможно, частичной) последовательности. Определение последовательности аминокислот в пептиде сильно зависит от качества данных и осложняется тем, что фрагментация может быть неполной, а массы могут быть измерены неточно из-за влияния многих параметров на результат эксперимента. Для облегчения процесса интерпретации спектра применяются некоторые приемы, способствующие стабильному образованию определенных "*b*" и "*y*" ионов [5]. Также применяется маркировка изотопов [6] для идентификации "*y*" ионов в спектре. Для определения последовательности по масс-спектру было разработано

множество алгоритмов [7-14]. Некоторые фирмы-производители оборудования пытались создать свои собственные программы для идентификации пептидов, однако, в большинстве своем, не очень успешно. Пристальное внимание привлекла независимо разработанная программа Lutefisk [13]. Как и в большинстве программ, в Lutefisk спектр представляется в виде "спектрального графа", когда вершинам в графе соответствуют пики в спектре, а ребро между вершинами проводится, если разность масс между соответствующими пиками в спектре совпадает с массой одной из аминокислот. Далее задача сводится к нахождению пути от N-терминали к C-терминали. Кроме того, часто используется динамическое программирование. Одной из наиболее известных программ с таким подходом является PEAKS [14], где ищется такая последовательность, спектр которой даст совпадение как можно большего числа пиков с наибольшими интенсивностями.

Данная работа основывается на представлении спектра пептида в виде спектрального графа. Алгоритм идентификации последовательности основывается на поиске пути от N-терминали к C-терминали. Найденные пути (последовательности аминокислот) оцениваются при помощи скоринг-функции, учитывающий разные факторы при сравнении теоретически рассчитанного спектра с экспериментальным.

Постановка задачи

Пусть $P = \{a_1, a_2, ..., a_N\}$ - пептид, состоящий из N аминокислот, a_i - одна из 20 аминокислот с массой $m_{a_i}, i = 1...N$. Масса пептида $m_P = \sum_{i=1}^{N} m_{a_i}$. Частичным N-концевым пептидом называют последовательности вида $\{a_1, a_2, ..., a_q\}, q = 1...N$ и обозначают P_i. Частичным C-концевым пептидом называют соответственно последовательности вида $\{a_1, a_2, ..., a_r\}, r = 1...N$ и обозначают $\overline{P_i}$. Экспериментальный спектр $S = \{(m,h)_k, k = 1...K\}$ это множество пар вида *(масса, интенсивность)*, полученных при эксперименте на масс-спектрометре, причем в набор входят как массы ионов, так и массы ионов химического шума с их интенсивностями. Теоретически рассчитанный спектр $T(P)$ последовательности $P = \{a_1, a_2, ..., a_N\}$ может быть получен при рассмотрении

всевозможных частичных пептидов. При идеальном случае фрагментации пептида в спектре будут присутствовать пики, соответствующие всем частичным пептидам, а также частичным пептидам с отколовшимися от них водой H_2O и/или аммиаком NH_3. В общем случае фрагментация пептида может быть охарактеризована множеством чисел $\Delta = \{\delta_1, \delta_2, ..., \delta_L\}$, характеризующих различные типы ионов, соответствующих химическим группам, которые могут отколоться от частичного пептида. δ-ион *N*-концевого частичного пептида P_i - это модификация P_i, имеющая массу $m_{P_i} - \delta$, соответствующая потере (обычно небольшой) химической группы массы δ при фрагментации P. Чаще всего образуются *N*-концевые ионы, называемые "*b*"-ионами (ион b_i соответствует частичному пептиду P_i с $\delta = -1.00782503\ 2 = -\mathrm{m}(\mathrm{H}^+)$), и *C*-концевые ионы, называемые "*y*"-ионами (y_i соответствует $\overline{P_i}$ с $\delta = -19.0183897\ 18 = -\mathrm{m}(\mathrm{H}_2\mathrm{O} + \mathrm{H}^+)$).

В этих обозначениях задача секвенирования пептидов (de novo sequencing) формулируется следующим образом: по заданному экспериментальному спектру S, множеству возможных типов ионов Δ и массе пептида M найти пептид P такой же массы, чей теоретический спектр $T(P)$ имеет лучшее совпадение с экспериментальным в смысле выбранной скоринг-функции $F(S,T)$.

Скоринг-функция $F(S,T)$ представляется в виде взвешенной суммы двух функций f_1 и f_2, значение первой из которых соответствует совпадению масс последовательности-кандидата и пептида; вторая является мерой совпадения теоретически-рассчитанного спектра с экспериментальным.

Для определения первой функции (см. (1)) вводится точность совпадения масс ε. Если масса последовательности-кандидата M_Z лежит в ε-окрестности массы пептида, f_1 полагается равной 1. Вне 2ε-окрестности f_1 полагается равной 0.0001. В интервалах $[M_P - 2\varepsilon, M_P - \varepsilon]$ и $[M_P + \varepsilon, M_P + 2\varepsilon]$ f_1 линейна.

$$f_1(z) = \begin{cases} 1, & |M_P - M_Z| \le \varepsilon \\ 1 - \dfrac{M_P - M_Z}{2\varepsilon}, & M_P - 2\varepsilon \le M_Z \le M_P - \varepsilon \\ 1 - \dfrac{M_Z - M_P}{2\varepsilon}, & M_P + \varepsilon \le M_Z \le M_P + 2\varepsilon \\ 0.0001, & |M_P - M_Z| > 2\varepsilon \end{cases} \quad (1)$$

За основу скоринг-функции, оценивающей совпадение спектров и используемой в данном алгоритме, была взята широко известная функция из программы SEQUEST [2]. Приведем ее описание.

Если аминокислотная последовательность совпадает по массе с родительским пептидом с заданной допустимой точностью, она подвергается оценке по разным параметрам. Во-первых, по числу предсказанных пиков, совпавших с экспериментальными с заданной точностью, n_i, вычисляется сумма интенсивностей этих пиков i_m. Во-вторых, продолжительность совпадения пиков для каждого совпавшего фрагмента β. В-третьих, увеличивается параметр ρ, отвечающий за присутствие *immonium ion* аминокислот *His, Tyr, Trp, Met* и *Phe*. Если таких ионов нет, то параметр уменьшается. Обычно для β и ρ используют значения *0.075* и *0.15* по умолчанию соответственно. Наконец, вычисляется общее число ионов предсказанного спектра n_t. Итак, скоринг вычисляется по следующей формуле:

$$S_P = (\sum_m i_m) n_i (1 + \beta)(1 + \rho) / n_t, \quad (2)$$

где суммирование интенсивностей ведется по всем совпавшим пикам.

В скоринг-функцию было внесено несколько усовершенствований.

Следует отметить, что в данной работе рассматривается только один тип пост-трансляционных модификаций аминокислот – карбамидометилирование.

Описание алгоритма

Предлагается следующий алгоритм определения последовательности аминокислот по масс-спектру пептида.

1. Из файла считывается спектр, информация о заряде и массе пептида.
2. Спектр подвергается предварительной обработке: обрабатываются ионы с двойным зарядом и изотопы. Производится

обработка многозарядных ионов и изотопов. Начиная с этого шага, все ионы в спектре считаются однозарядными. Считается также, что в спектре нет изотопов.

3. Проводится обработка спектра на предмет увеличения интенсивности "*b*"-ионов:

(a) Если масса иона соответствует потере родительским ионом воды H_2O или аммиака NH_3 и заряд родительского иона *2* или *3*, интенсивность пика уменьшается в *10* раз.

(b) Если есть подозрение, что пик получен от иона-изотопа, его интенсивность уменьшается в *25* раз.

(c) Если пик получен из другого при потере воды, аммиака или *CO*, его интенсивность уменьшается в *5* раз.

(d) В спектр добавляются комплементарные ионы.

(e) Если для пика в спектре уже присутствует комплементарный, то к их интенсивностям добавляется среднее геометрическое двух интенсивностей. Поскольку для масс-спектрометров с ионной ловушкой и малой энергией при столкновительной диссоциации характерно образование преимущественно "*a*", "*b* и "*y*" ионов и ионов, полученных при потере воды, аммиака или *CO*, то эта процедура увеличивает интенсивности "*b*" и "*y*" ионов.

(f) Интенсивности пиков с $m/z=1$ и $m/z=M_P-17$ (то есть нулевого и отвечающего родительскому иону) полагаются равными максимальной интенсивности в спектре.

(g) Создается спектр, состоящий только из "*b*"-ионов ("*b*"-спектр). Для каждого пика: если существует комплементарный ему ион и ион, соответствующий потере *CO* ("*a*"-ион), то ион считается ионом "*b*"-типа и используется для создания "*b*"-спектра. Интенсивности рассчитываются как среднее геометрическое комплементарных пиков. Затем "*b*"-спектр складывается с исходным спектром, что увеличивает интенсивности "*b*"-ионов.

4. В спектр добавляются комплементарные пики согласно алгоритму, представленному в [24]. Рассчитываются координаты вершин $cord(x), x \in V$.

5. Совпавшие вершины уничтожаются.

6. Спектр представляется в виде спектрального графа *G=(V,E)*, где *V* - множество вершин, *E* - матрица связности.

7. Рассчитывается матрица разностей $K_{ij}=V_i-V_j,\quad i,j=1...N$, где $V_i,\quad i=1...N$ - вершины спектрального графа.

8. Между вершинами *x* и *y* из *V* проводится ребро *E(x,y)=1*, если выполняются три условия:

(a) *x* и *y* получены не от одного пика,

(b) $x<y$, т.е. $cord(x)<cord(y)$,

(c) $cord(y)-cord(x)$ совпадает с массой какой-либо аминокислоты с заданной точностью.

Таким же образом ищутся пары аминокислот (например, *AG*, *GT* и т.п.) с массой, не превышающей заданного порога (был взят порог *200Da*).

9. Находятся комплементарные пики в исходном экспериментальном спектре и заносятся в отдельный список для дальнейшего использования при подсчете скоринга.

10. Начинается поиск в матрице ребер E с нулевой вершины и в обратном порядке, от родительской вершины. Поиск от нулевой вершины осуществляется рекурсивно следующим образом:

(a) Выбирается вершина.

(b) Анализируются вершины с большей координатой.

(c) Если находится ребро, связывающее текущую вершину с вершиной с большей координатой, осуществляется переход по ребру в следующую вершину, и метка ребра добавляется в строку-результат.

(d) Если добавлением метки (т.е. аминокислоты или пары аминокислот) превышается масса родительского пептида, данная ветвь рекурсии обрывается. При добавлении меток создается маска последовательности вида ----**-**--. Знаки ** означают, что была добавлена пара аминокислот из расширенного списка.

(e) Для следующей вершины совершается переход на шаг (a) пока не достигнем родительской вершины.

Для поиска в обратном порядке алгоритм аналогичен.

11. В процессе поиска запоминаются все префиксы и постфиксы длиной не более трех аминокислот. Дело в том, что, начиная поиск от нулевой вершины, в идеальном случае используются только "*b*"-ионы. Так же, начиная поиск от родительской вершины, мы используем "*y*"-ионы. Но понятно, что в какой-то момент мы можем перескочить на ионы другого типа, что, конечно, не может привести к правильному ответу, но, тем не менее, возможно из-за отсутствия априорной информации о типе ионов. В идеальном случае мы можем восстановить последовательность двумя способами, основанными на одном принципе: 1) начиная поиск от нулевой вершины и 2) начиная поиск от родительской вершины.

Кроме того, очевидно, что если мы начнем поиск с вершины, соответствующей иону воды H_2O^+, или с вершины, соответствующей родительскому иону с потерей воды M_P - H_2O^+, то получим инвертированные последовательности.

Таким образом, проводятся четыре итерации поиска начальных частичных последовательностей длиной не более 3 аминокислот (две в сторону возрастания *m/z*: от нулевой вершины и от вершины H_2O^+; и

две в сторону убывания *m/z*: от родительской вершины и от вершины $M_P - H_2O^+$). Далее соответствующие пары анализируются на предмет совпадения. В случае, если мы получили последовательность $\{a_{i_1}, a_{i_2}, a_{i_3}\}$, осуществляя поиск от вершины H_2O и такую же последовательность, идя от родительской вершины в обратном порядке, это свидетельствует о большой вероятности того, что искомая аминокислотная последовательность заканчивается на $\{a_{i_3}, a_{i_2}, a_{i_1}\}$. Итак, создается специальный массив, в котором хранятся префиксы и постфиксы последовательности, использующийся в дальнейшем для подсчета скоринга.

12. Для каждой последовательности-кандидата из списка подсчитывается скоринг в соответствии с введенной скоринг-функцией, которая будет описана ниже.

13. Список последовательностей сортируется в соответствии с рассчитанным скорингом. Объявляется решение.

Результаты

В работе использовалось 107 спектров известных пептидов, полученных на QTOF масс-спектрометре, любезно предоставленных Ричардом Джонсоном из компании "Amgen". Примеры спектра и спектрального графа изображены на Рис. 1 и 2. Спектральный граф получен после предварительной обработки спектра, поэтому число вершин в нем невелико. Однако и на этом примере видно, что возможно несколько вариантов получения последовательности.

В Табл. 1 представлены результаты определения последовательностей. Маска последовательности маркирует места, где для построения последовательности использовались двойные аминокислоты. Эта информация необходима для дальнейшей идентификации протеинов по базам данных. В левом столбце отображена искомая последовательность. Во втором и третьем столбцах представлены последовательности-кандидаты с наилучшим скорингом и соответствующие маски. В четвертом столбце указан ранк в соответствии со скорингом. Представленный алгоритм в 12% случаев выдаёт искомую последовательность на первом месте с точностью до перестановок и эквивалентных замен. В 37% случаев искомая последовательность находится в первой десятке списка. Исследования показали, что важнейшую роль в определении правильной последовательности аминокислот играет способ задания скоринг-

функции. Алгоритм получения последовательностей-кандидатов из спектрального графа в большинстве случаев выдает искомую последовательность, которая в дальнейшем неверно оценивается на этапе подсчета скоринга. Очевидно, недостаточно учитывать только совпадение позиций пиков, поскольку при фрагментации образуется большое число ионов разных типов, что может привести к ошибочному результату. Восстановление последовательности производится по "*b*"- или "*y*"-ионам, и если ион ошибочно относится к ионам этого типа, происходит ошибка. Для правильной оценки последовательности необходимо использовать информацию об интенсивностях пиков в спектре, чтобы производить сравнение не только по массам ионов, но и по распределению интенсивностей. Наиболее перспективным здесь видится моделирование фрагментации пептида в ловушке масс-спектрометра с использованием кинетической модели [15, 16], чему и будут посвящены дальнейшие исследования. Статистические способы расчета интенсивностей обладают неполнотой данных в связи с огромным разнообразием пептидных последовательностей.

Работа выполнена как часть исследований по гранту РФФИ 05-07-90238.

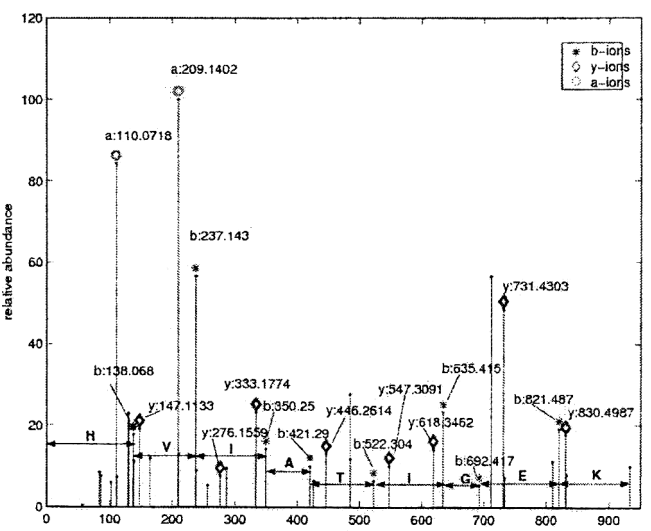


Рис. 1. MS/MS-спектр пептида HVLATLGEK

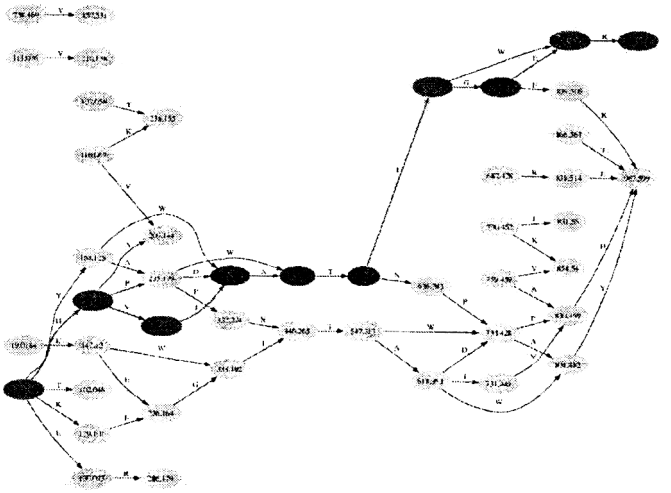


Рис. 2. Спектральный граф пептида HVLATLGEK

Последовательность	Кандидат	Маска	Ранк
AITIFQER	AITIFFEH	**.....**	3
	PSTIFRTR	**...**.	
	AITIFRTR	**...**.	
AEFVEVTK	EAFKAKKT	.**.....	2
	EAFVEVKT	.....**	
ALPMHIR	AIPHFPR	...**...	3
	AIPMSYR	....**.	
	AIPMHIR	....**.	
DGFIDKNDLR	ICCIDKNDAPT	.**.....**.	3
	IGMMDKNDAPT	..**.....**.	
	DGFIDKNDAPT	.**.....**.	
DLGEEHFK	EVWEHFK	**.....	5
	MPWEHFK	**.....	
	DIWEHFK	**.....	
HVLATLGEK	HVIATIGEK	**.....	1
EAFLLFDR	TVEIMFGIT	.....**.	4
	ISEIMFGIT	**.....	
	CPEIMFGIT	**.....	
	ISFIIFDR	**.....	
IDPNAWVER	IDPNIWVSR	.**.....	5
	IDPNIEGVSR	.....**...	
LISWYDNEFGYSNR	IISWYDNEFGYSRN	**.....**	1
	EPSWYDNEFGYSRN	**.....**	
	VVDWYDNEFGYSRN	**.....**	
LIVTQTMK	IIVNDTMK	...**...	2
	IIVTKTMK	....**..	
NEEIDEMIK	KDEIDEMIK	....	4
	KDEIIMMIK	....**...	
	KDKNDFEPAG	.....**	
NLAENISR	ARAENISR	**..**..	3

	KVAEKVSR	**--**--	
	NIAENISR	**--**--	
QTALVELLK	KTAIVEIIK	-----**--	1
	AGTAIVEIIK	**--------	
QASEGPLK	KASSVPIK	--**----	3
	KADTGPIK	--**----	
	KAESGPIK	--**----	
QQQDEFK	KKKDEFK	----**-	1
TKIPAVFK	AGTIPAVFK	---------	3
	AGTIAPVFK	----**---	
	KTIPAVFK	**------	
VFADYEEYVK	VFADYEEYVK	----------	1
	VFWYAYDVVT	----**----	
VVDLMVHMASK	VVDIMVFIFF	--------**	5
	VVDIMVFIMY	--------**	
	VVDIMDMIFF	-----**-**	

Таблица 1: Результаты работы программы.

Литература

1. Govorun V.M., Archakov A.I., Proteomic technologies in moder biomedical science.// Biochemistry (Moscow), 2002, v.67, No. 10, pp.1109-1123.

2. Eng J.K., McCormack A.L. and Yates J.R., An approach to correlate tandem mass spectral data pf peptides with amino acid sequences in a protein database.// J. Am. Soc. Mass Spectrom., v.5, pp.976-989, 1994.

3. http://www.matrixscience.com/home.html

4. Mann M. and Wilm M., Error-tolerant identification of peptides in sequence databases by peptide sequence tags.// Anal. Chem., v.66, pp.4390-4399, 1994.

5. Munchbach, M., Quadroni, M., Miotto, G., James, P., Quantitation and facilitated de novo sequencing of proteins by isotopic, N-terminal labeling of peptides with a fragmentation-directing moiety. //Anal. Chem. 2000. 72(17), 4047-4057.

6. Uttenweiler-Joseph S., Neubauer G., Christoforidis S., Zerial M., and Wilm M., Automated de novo sequencing of proteins using the differential scanning technique.//Proteomics, v.1, pp.668-682., 2001.

7. Fenyo D., Qin J. and Chait B.T., Protein identification using mass spectrometric information.//Electrophoresis, v.19, pp.998-1005, 1998.

8. Pevzner P.A., Dancik V. and Tang C.L., Mutation-tolerant protein identification by mass-spectrometry.//J. Comput. Biol., v.7, pp.777-787, 2000.

9. Pevzner P.A., Mulyukov Z., Dancik V. and Tang C.L., Efficiency of database search for identification of mutated and modified proteins via mass spectrometry.//Genome Res., v.11, pp.290-299, 2001.

10. Bartels C., Fast algorithm for peptide sequencing by mass spectrometry.//Biomed. Environ. Mass Spectrom., v.19, pp.363-368, 1990.

11. Dancik V., Addona T.A., Clauser K.R., Vath J.E. and Pevzner P.A., De novo peptide sequencing via tandem mass spectrometry.// J. Comput. Biol., v.6, pp.327-342, 1999.

12. Taylor J.A. and Johnson R.S., Sequence database searches via de novo peptide sequencing by tandem mass spectrometry.//Rapid Commun. Mass Spectrom., v.11, pp.1067-1075, 1997.

13. J. A. Taylor and R. S. Johnson, Implementation and uses of automated de novo peptide sequencing by tandem mass spectrometry. 2000. Anal. Chem. 73:2594-2604.

14. Ma B., Zhang K., and Liang C., An effective algorithm for peptide de novo sequencing from MS/MS spectrum.

15. Zhang Z., Prediction of low-energy collision-induced dissociation spectra of peptides.//Anal. Chem., v.76, pp.3908-3922., 2004.

16. Wisocki V.H., Tsaprailis G., Smith L., Breci L.,// Mass Spectrom., v.35, pp.1399-1406, 2000.

Попова Е.А.

Локализация нейронных источников электрической активности мозга с помощью метода Random Forest.

1. Введение.

Работа посвящена методам автоматизированного анализа электрической активности мозга. В последнее время одним из актуальных направлений в информатике является разработка так называемого интерфейса мозг- компьютер (Brain-Computer Interface). К настоящему времени уже достигнуто немало значительных результатов в этой области исследований [1]-[4]. Необходимо понять, каким же образом создается связь между мозгом человека и компьютером, какие алгоритмы следует использовать их для автоматизированного анализа. Судить об электрической активности мозга мы можем с помощью электроэнцэфалограммы. Электроэнцефалограммой [5] называется запись слабых (порядка 5-100 □V) электрических потенциалов, генерируемых мозгом. ЭЭГ сигнал представляет собой разность потенциалов между электродами, размещенными на поверхности головы.

С тех пор, как были разработаны методы компьютерной обработки ЭЭГ сигнала, появилась возможность создания непосредственного канала связи между мозгом и реальным миром. Среди первых исследований в области BCI были проекты под руководством доктора Дж. Видаля (J. Vidal), директора лаборатории интерфейса "мозг-компьютер" в Калифорнийском Университете (Brain-Computer Interface Laboratory at UCLA), которые демонстрировали первые успешные попытки создания BCI [5]. Одним из ключевых моментов в развитии явился семинар по технологии BCI, прошедший в Олбани в 1999 г., где 22 исследовательские группы представили свои работы[2]. Регистрируя электрическую активность, можно судить об источнике активности и области нарушения нейронной системы. ЭЭГ пациентов, страдающих эпилепсией, может иметь характерные волновые формы: острые волны, пики (спайки). Известно, что за генерацию эпилептического припадка отвечают локальные группы нервных клеток (т.н. очаги). Часто эпилепсию лечат медикаментами, но существуют группы пациентов, которым лекарства не помогают. В этом случае возможным решением является хирургическое удаление эпилептогенного очага. Во время подготовки к операции проводятся многочисленные исследования для локализации этой зоны. Одним из исследований является длительный (в течение 3-5 дней) мониторинг ЭЭГ. На основе ЭЭГ данных, а также поведения пациента во время приступа, неврологи могут определить

локализацию эпилептогенного очага. Но помимо клинических приложений использования BCI следует отметить еще ряд важных областей и задач его использования. Управление автомобилем, самолетом, мультимедийные коммуникации, передача чувств, запаха, эмоций.

Рассмотрим модель нейронной активности, которая будет в дальнейшем использоваться в этой работе. Модель основана на предположениях о нейронных клетках как источниках (генераторах) электрического тока. Мозг моделируется как некий объемный проводник с неоднородной электропроводностью с имплантированными в него диполями. Прямая задача состоит в вычислении трехмерного электрического поля, создаваемого источниками-диполями [7],[8]. Полученное решение - потенциал электрического поля на поверхности головы - моделирует измеряемый в эксперименте с помощью электродов потенциал электрического поля. Делая различные предположения о распределении и количестве диполей, можно добиваться совпадения экспериментального потенциала с модельным. Обратная задача ЭЭГ состоит в определении положений источников по измеренным потенциалам на поверхности головы [9]-[11]. Для решения обратной задачи требуется решение большого числа прямых задач[14].
При использовании одного диполя в качестве источника предполагается, что только одна локальная область с параллельно ориентированными пирамидальными клетками является активной. Неизвестных в задаче в этом случае шесть: три пространственные координаты расположения диполя и три компоненты вектора тока, характеризующего направление и величину момента диполя. Наиболее сложной проблемой является выделение такой информации из пространственно-временных данных, которая помогает найти объемную структуру нескольких источников, активных одновременно. Существует два подхода к локализации источников. Первый подход основан на фиксированном расположении диполей, а задача сводится к отысканию их силы тока (задача компьютерной томографии) [11]- [12].

Таким образом, определяются наиболее активные фиксированные источники. В другом случае, задача сводится к регуляризации функционала ошибки по Тихонову [13]. Модели анализа пространственно-временного сигнала ЭЭГ можно найти в [14]-[16].
В модели \'изучения отклонений \' (The deviation scan) предполагается подгонка одиночного диполя, повторенная для большого числа его расположений. Модель многократной классификации сигнала MUSIC (Multiple Signal Classification)[14] определяет алгоритм поиска

источников, являющихся независимыми друг от друга во времени. Активность, которая не может быть приписана к источнику в фиксированном расположении рассматривается как шум.
В модели "распределенных источников" предполагается, что источники распределены по некоторой заранее известной поверхности. Часть поверхности коры головного мозга, на которой распределены источники, определяется априори.
В работе [15] рассматривалась модель подгонки нескольких дипольных источников. При решении обратной ЭЭГ задачи ищутся как дипольные моменты (токи) так и их расположения. Обратная задача является нелинейной в этом случае. Основная идея работы состоит в использовании Генетического алгоритма для ЭЭГ сигнала при пространственной реконструкции источников.
В настоящей работе анализ сигналов ЭЭГ проводится на основе метода Random Forest [16]. Алгоритм Random Forest один из наиболее многосторонних алгоритмов классификации данных известный в области Data Mining. Дипольная функция источника позволяет моделировать электрический потенциал, измеренный в 21 точке на поверхности головы через параметры дипольного источника - его координаты (пространственную локализацию) и вектор момента. Параметры источника, его локализация могут быть восстановлены по экспериментальным данным в одном срезе по времени. Дипольная модель не связывает измеренные сигналы по времени, однако учет пространственно-временного характера данных позволяет решить задачу о локализации нейронных источников более точно и устойчиво.
В работе предлагается алгоритм локализации, основанный на аппарате деревьев решений. Для каждого момента времени случайным образом несколько диполей располагаются внутри области, и строится свое дерево решений классифицирующее набор параметров дипольных источников по уровню допустимой ошибки приближения ЭЭГ потенциалов. Каждое дерево из временного ансамбля оценивает область локализации дипольных источников, вычисляя порог классификации. К первому классу относятся диполи дающие ошибку в восстановлении потенциала меньше задаваемого порога.

Классификация проходит без усечения ветвей на случайно выбранном для каждого момента наборе признаков - параметров диполя. Построенный таким образом ансамбль деревьев классификации представляет случайный лес (Random Forest) [17]. Решение о зоне локализации принимается, если оно подтверждено большинством деревьев из леса.

2. Дипольная модель электрической активности мозга.

В дипольной модели мозг рассматривается как объемный, трехмерный проводник. Обозначим V_{ol} его объем и S_{tot} - поверхность, ограничивающую объем. Источниками электрической активности являются электролитические токи внутри нервных клеток коры головного мозга. Закон Ома в рассматриваемом проводнике запишется в следующем виде:

$$\mathbf{j} = \mathbf{j}_{in} + \sigma \mathbf{E} \qquad (1)$$

где $\mathbf{j}_{in}$ - плотность стороннего (ионного, электролитического по происхождению) тока создаваемого нервной клеткой. Источники тока помещены в проводящую среду с неоднородной электронной проводимостью $\sigma(\mathbf{r})$. Ионные токи в объеме мозга порождают электрическое $\mathbf{E}(\mathbf{r}, t)$ и магнитное $\mathbf{E}(\mathbf{r}, t)$ поля которые описываются уравнениями Максвелла.

Показано, что в данной задаче временная и пространственная составляющие уравнений могут быть разделены, и пространственная часть в каждый момент времени удовлетворяет стационарному уравнению.

$$(\nabla \bullet j) = 0 \qquad (2)$$

Применяя оператор дивергенции к обеим частям (1), используя (2), и введя потенциал электрического поля, $U(\mathbf{r}, t)$:

$$\mathrm{E} = -\nabla U, \quad [\nabla \times \mathrm{E}] = \mathbf{0} \qquad (3)$$

получаем уравнение Пуассона :

$$(\nabla \bullet \sigma_0 \nabla U((r, \tau))) = -(\nabla \bullet j_{in}) \qquad (4)$$

Предполагая, что все источники тока заключены в объем V_{ol}, а в остальном пространстве V_{rest} источников нет, можно записать частное решение уравнения Пуассона в виде объемного потенциала в области с постоянной проводимостью σ_0.

Задача состоит в вычислении потенциала U как решения уравнения 4. Для этого потенциал U представляется в виде суммы:

$$U(\mathbf{r}, t) = W(\mathbf{r}, t) + V(\mathbf{r}, t) \tag{5}$$

где W - потенциал, создаваемый диполями:

$$W(\mathbf{r}_M, t) = \sum_{i=1}^{N_i} \left(\frac{\upsilon_i(\mathbf{r}_P^i)}{\sigma_0} \cdot \nabla \frac{1}{R_{MP_i}}\right) \tag{6}$$

здесь N_i - число диполей и $V(\mathbf{r}, t)$ - неизвестный потенциал индуцированного поля, который создается из-за наличия границ и неоднородной электропроводности. Если известны положения диполей и их моменты (прямая задача), тогда потенциал $W(\mathbf{r}_M, t)$, создаваемый диполями, известен и требуется найти $V(\mathbf{r}, t)$. В областях однородности проводимости для функции $V(\mathbf{r}, t)$ имеем задачу Неймана для однородного уравнения Лапласа

$$\Delta V(\mathbf{r}, t) = 0 \tag{7}$$

с неоднородным граничным условием на скальпе (граница с непроводящей средой):

$$\frac{\partial V}{\partial n} = -\frac{\partial W}{\partial n} \tag{8}$$

и условиями сшивки на каждой поверхности S_k разрыва проводимости.

$$V_k = V_{k+1}|_{Sk} \tag{9}$$

Заметим, что W непрерывно вместе со своими производными во всем пространстве за исключением точек расположения источников.

Нормальные производные потенциала V терпят разрыв:

$$\sigma_k \frac{\partial V_k}{\partial n} = \sigma_{k+1} \cdot \frac{\partial V_{k+1}}{\partial n} + (\sigma_{k+1} - \sigma_k) \cdot \frac{\partial W}{\partial n}\Big|_{S_k} \tag{10}$$

Здесь индекс k обозначает номер поверхности, ограничивающей слой проводимости.

При учете проводимости области вне головы (проводимость геля

электродов, учета других проводящих слоев вне скальпа) задача ставится в неограниченной области.

Метод решения прямой задачи основан на использовании общего решения уравнения Лапласа в виде сферических гармоник [13]. Коэффициенты общего решения находятся методом наименьших квадратов для удовлетворения граничного условия Неймана на произвольной (не сферической) границе и условий сшивки на границах смены электропроводности. В общем случае будем считать, что k=1, 2, 3, ...,K поверхностей, заданных уравнениями $r^{(k)}=r^{(k)}(\vartheta, \varphi)$ в сферических координатах, разделяют области с различными проводимостями. Здесь k=1 обозначает самую внутреннюю поверхность, содержащую начало координат, и k=K - поверхность скальпа (последняя). Пусть $\Omega = \{(\vartheta_i,\varphi_j) \mid i=1,...,N_\vartheta;\ j=1,...,N_\varphi\}$ - сетка по ϑ и φ. Обозначим $r_{ij}^{(k)} = r^{(k)}(\vartheta_i, \varphi_j)$.

Ищем решение в k-й области в виде разложения по сферическим гармоникам:

$$V^k(r,\vartheta,\varphi) = \sum_{n=0}\Big\{r^n \sum_{m=0}^{n}(A_{nm}^k \cos m\varphi + B_{nm}^k \sin m\varphi)P_n^{(m)}(\cos\vartheta) + \qquad (11)$$

$$r^{-(n+1)}\sum_{m=0}^{n}(C_{mn}^k \cos m\varphi + D_{nm}^k \sin m\varphi)P_n^{(m)}(\cos\vartheta)\ \Big\}$$

Для любых коэффициентов A,B,C и D функции V^k удовлетворяют уравнению Лапласа (7). Коэффициенты A_{nm}^k, B_{nm}^k, C_{nm}^k, D_{nm}^k, k = 1, ...,K находятся из условий сшивки (9)-(10) на произвольной поверхности $r_{ij}^{(k)}$.

Для k=1 коэффициенты $C_{nm}^1 = D_{nm}^1 = 0$ из требования ограниченности в нуле, и для k=K $A_{nm}^K = B_{nm}^K = 0$ из ограниченности на бесконечности. Остальные коэффициенты находятся из условий сшивки (9)-(10) в смысле наименьших квадратов.

3.Задача локализации нейронного источника.

Математическая постановка обратной задачи сводится к нахождению правой части неоднородного трехмерного эллиптического уравнения (4) с неоднородными коэффициентами по решению U заданному в точках поверхности S^{meas} и граничному условию Неймана на этой поверхности.

Для численного решения обратной задачи используются приближенные граничные условия в виде условий минимума функционалов ошибки:

$$\varepsilon_1^K(\nu, \mathbf{r}_p) = \|U_{exp} - W(\nu, \mathbf{r}_p, \mathbf{r}) - V_K(\mathbf{r})\|^2] \rightarrow \min \qquad (12)$$

$$\varepsilon_2^k(\nu, r_p) = \left\| \frac{\partial[W(n, rp, r) + VK(r)]}{\partial n} \right\|^2 \rightarrow \min \qquad (13)$$

Задача состоит в определении величин и координат диполей ν, $\mathbf{r}_p$ для наилучшего приближения потенциала $U_{model} = W(\nu, \mathbf{r}_p, \mathbf{r}) + V_K(\mathbf{r})$ на поверхности скальпа $S^{mcas}{}_K$ к экспериментально измеренному распределению потенциала U_{exp} путем минимизации функционала $\varepsilon = \varepsilon_1{}^K + \varepsilon_2{}^K$. Для итеративного решения обратной задачи необходимо решать прямую задачу на каждой итерации, т.е. по дипольным источникам с известными расположением и мощностью находить трехмерное распределение потенциала.

4. Алгоритм обучения системы классификации на тренировочном наборе данных с помощью деревьев решений.

На рис.1 приведена анализируемая электроэнцефалограмма - зависимость измеренного потенциала во времени на 14-ти сенсорах на поверхности головы. Для локализации диполя мы рассматриваем лишь небольшое временное окно: t=1.2s - t=1.3s. В этом окне 10 временных точек.

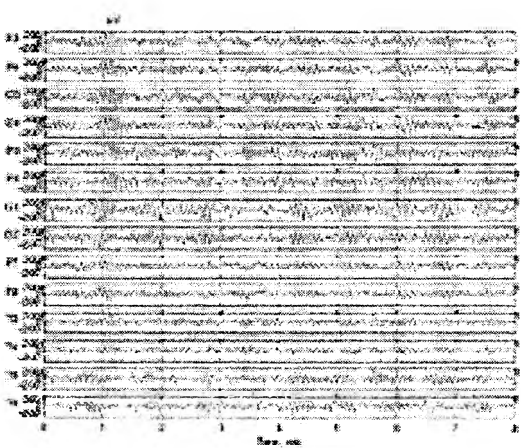


Рисунок 1: Электроэнцефалограмма - зависимость потенциала по времени на 14ти сенсорах расположенных на поверхности головы.

Покажем работу алгоритма на базе данных, которую назовем тренировочным набором данных.

Алгоритм обучения на тренировочном наборе данных соответствует известному алгоритму C4.5 [17]

Пусть нам задано некоторое обучающее множество T, содержащее образы, каждый из которых характеризуется M признаками (атрибутами, независимыми переменными). Пусть через $C_1,C_2,...C_k$ обозначены метки классов (зависимая переменная, целевая переменная). Обозначим вектор признаков через

$$\mathbf{X}^p = \{ x_1,x_2,..x_i,..x_M \mid C_k \}^p$$

где p- номер примера, к - номер класса, x_i^p i -ый признак p-го примера

Случайным образом в момент времени t^1 располагаются N_p (полное число примеров равно N_p) диполей внутри головы (полной допустимой области) и выбираются их моменты: Здесь $(\mathbf{x}^1,\mathbf{x}^2,\mathbf{x}^3,...)$ параметры диполей:

$$\mathbf{x}^1 = \{ \mathbf{x}^1_k \} = \{ \mathbf{r}_p^1, \vartheta_p^1, \varphi_p^1, \nu_r^1, \nu_\vartheta^1, \nu_\varphi^1 \}$$
$$\mathbf{x}^2 = \{ \mathbf{x}^2_k \} = \{ \mathbf{r}_p^2, \vartheta_p^2, \varphi_p^2, \nu_r^2, \nu_\vartheta^2, \nu_\varphi^2 \}$$
$$\mathbf{x}^3 = \{ \mathbf{x}^3_k \} = \{ \mathbf{r}_p^3, \vartheta_p^3, \varphi_p^3, \nu_r^3, \nu_\vartheta^3, \nu_\varphi^3 \}$$
$$...$$

Каждый из наборов параметров $(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3,...)$ принадлежит соответственно первому диполю $(\mathbf{x}^1)$, второму $(\mathbf{x}^2)$, третьему $(\mathbf{x}^3)$ и т.д. Для каждого p -ого диполя вычисляется ошибка

$$\varepsilon^p = \sum_i \sum_j [(U_{exp}(\vartheta_i, \varphi_j) - V_K (\vartheta_i, \varphi_j)) - w^{(p)} (\mathbf{x}^p \mid \vartheta_i, \varphi_j)]^2 \sin\vartheta_i \, h_{\vartheta i} h_{\varphi j} \qquad (14)$$

Задаем уровень допустимой ошибки ε_{th}, и разделяем параметры диполей на два класса - ниже порога (первый класс - C_1) $\varepsilon^p < \varepsilon_{th}$ и выше порога (второй класс - C_2). Таким образом получена тренировочный набор данных. Основой классификации является уровень допустимой ошибки

по потенциалу. Этот уровень определяет точность классификации. На рис.2(a,b,c,d) представлены линии уровня ошибки по потенциалу ε = RRE на плоскости двух параметров - координат диполя (r_p – θ_p). Линии уровня соответствуют ошибке получающейся в четыре последовательных момента времени: a) t=0,b)t=1,c)t=2,d)t=3. Здесь использовано международное обозначение ошибки по потенциалу - RRE (Residual Ralative Error).

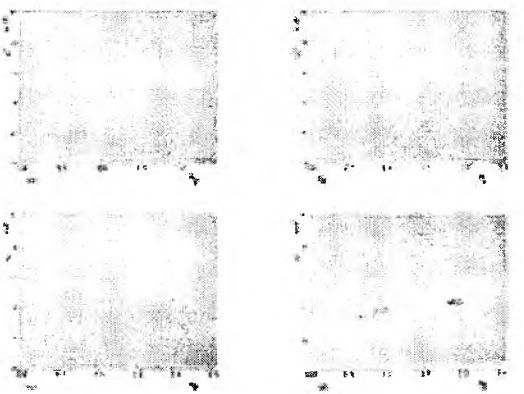


Рисунок 2: Линии уровня ошибки при аппроксимации потенциала RRE=const на плоскости координат диполя (r_p – θ_p)в четырк последовательных момента времени:a) t=0,b)t=1,c)t=2,d)t=3.

Результаты представленные на рисунке позволяют судить о связи ошибки по потенциалу с ошибкой в локализации диполя. Конечно, это суждение здесь очень приблизительно, так как показан только одна проекция в шестимерном пространстве параметров. Рисунок также показывает сложность задачи локализации - это наличие очень плоских оврагов в функционале ошибки. Другой принципиальной чертой задачи распознавания является ее некорректность по отношению к тестовым данным. Чем меньше ошибка по потенциалу, тем больше вероятность того, что области локализации по различным деревьям не перекрываются. В этом случае вероятность структурного риска повышается.

Узлом дерева является узел проверки, а листом - узел решения. Условие в узле будем называть тестом. Множество T содержит примеры, относящиеся к разным классам. Цель построения дерева - разбить множество T на некоторые подмножества T_i которые будут состоять в основном их примеров относящихся к одному классу. Нас интересует разбиение внутренности головы на подобласти по координатам, в которых примеры относятся к первому классу,т.е. такому, в котором ошибка по потенциалу наименьшая.

Вначале выбирается наиболее значимый атрибут для разбиения. Выбранный атрибут должен разбить множество так, чтобы получаемые подмножества состояли из объектов, принадлежащих к одному классу максимально приближены к этому, т.е. количество образов-примесей в каждом из этих множеств было как можно меньше.

Мы используем теоретико-информационный критерий (алгоритм С4.5). для выбора наиболее значимого атрибута. Для различных классов и различных примеров каждый признак x^i упорядочивается в порядке возрастания величины. Вычисляются средние значения соседних величин признака:

$$x_i^* = (x_i + x_{i+1})/2$$

Эти значения x_i^* являются границами разбиения на подмножества T_i по признаку x_i, i=1,...M-1. Пусть $N_{k,i}$ число примеров из T_i относящихся к одному классу C_k . Тогда вероятность того, что случайно выбранный пример из множества T_i будет принадлежать к классу C_k равна $P_{k,i}=N_{k,i}/N_i$, где N_i есть полное число образов в множестве T_i. Информация содержащаяся в выборе $x_{i,k}$ равна $I_{k,i}=-\log_2 P_{k,i}$, а средняя информация приходящаяся на такой выбор есть энтропия:

$$H(x_i,T_i) = -\sum_{k=1}^{n} \{ P_{k,i} * \log_2 P_{k,i} \} * \frac{N_i}{N} \qquad (15)$$

Обозначим через $H(x_i,T)$ энтропию полного множества T при анализе по признаку x_i. Тогда критерием выбора признака будет минимальное значение

$$G(X) = H(x_i,T) - H(x_i,T_i) \qquad (16)$$

по всем признакам. Итак, множества $T_1,T_2,...T_n$ получены при зазбиении исходного множества T по признаку x_i Выбирается атрибут, дающий максимальное значение по критерию (15)

Пусть выбран признак x^i для всех примеров Теперь разбиваем множество T на два подмножества путем вычисления порога по выбранному признаку. Для этого вычисляется пороговое значение признака $x_{i,th}$ из набора x_i^* из условия максимизации энтропии (215), т.е. $\max_{xi}{}^* H(x_i^*,T_i)$. Исходное множество примеров разбивается на два подмножества $T_{xi}{}^L$ и $T_{xi}{}^R$ по критерию:

$$x_i < x_{i,th}, \quad (17)$$

множество T_{xi}^{L} и

$$x_i > x_{i,th}, \quad (18)$$

множество T_{xi}^{R}.

Далее рассматриваем множество T_{xi}^{L}. Делаем проверку, если в T_{xi}^{L} находятся примеры одного класса, то получено решение и алгоритм останавливается. Если в T_{xi}^{L} присутствуют примеры других классов, то продолжаем разбиение каждого из полученных множеств T_{xi}^{L} и T_{xi}^{R}. Пусть анализируем дальше T_{xi}^{L} , в котором выбираем наиболее информативный признак для следующего разбиения, используя указанную выше процедуру. Причем признак x_i исключается из соревнования. Пусть выбран признак x_j для следующего разбиения. Из условия максимизации энтропии вычисляется порог $x_{j,th}$ по признаку x_j и множество T_{xi}^{L} оказывается разбитым на два : T^{LL} и T^{LR} по признаку x_j. Если полного разделения образов не получилось, то разбивается множество T_{xi}^{R} на два множества T^{RL} и T^{RR}. Таким образом, используя два признака мы разбили исходное множество на 4 подмножества T^{LL}, T^{LR}, T^{RL} и T^{RR}. Далее процедура повторяется. В данной работе мы используем следующий критерий остановки алгоритма как ограничение глубины дерева : когда проведено разделение множества примеров по всем признакам алгоритм останавливается. После остановки алгоритма принимается решение об отнесении образов к классу. Считается вероятность принадлежности примера (образа) к классу.

Классификация новых данных происходит с помощью дерева и выдается ошибка классификации. Пусть N_k число примеров из T^{LR} относящихся к одному классу C_k. Тогда вероятность того, что случайно выбранный пример из множества T^{LR} будет принадлежать к классу C_k равна $P_k=N_k/N_{LR}$, где N_{LR} есть полное число образов в множестве T^{LR}. Если $P_k > P_{th}$, где P_{th} заранее выбранное пороговое значение вероятности классификации. Например, если число признаков равно двум, то дерево содержит три узла (пороговые значения для разбиения) и ссылку на потомков. Если остановки не произошло по ходу разбиения, то дерево содержит четыре листа , как решения принятые в четырех подмножествах множества образов. Эти пороги x_{th}, указатели и листы (P_{th}) должны содержаться в памяти для классификации образов.

Основа программы обучения тестировалась на известной базе данных [31] [32] и результаты сравнивались с результатом работы C4.5. Вычисления показали, что результаты работы программы адаптации на одинаковых тестовых базах данных не уступает известным программам.

5. Локализация положения диполей методом Random Forest.

В случайном лесе выращивают много деревьев классификации. Для того, чтобы классифицировать новый образ из входного вектора, входной вектор подается на каждое дерево из леса. Каждое дерево дает классификацию, и мы можем говорить о том, что оно голосует за определенный класс. Лес выбирает классификацию имеющую наибольшее количество голосов по всем деревьям из леса.

Каждое дерево строится следующим образом:

1. Если число примеров в тренировочном наборе есть N_p , то выбираем N_p примеров случайным образом. Эта выборка будет тренировочным набором данных для выращивания дерева.

2. Если число признаков образов есть M (в нашем случае шесть), то задается число m << M такое,что на каждом узле, m признаков выбираются случайным образом из M и поддерживается постоянным в течении выращивания леса. 3. Каждое дерево растет до наибольшего возможного размера.

Покажем работу алгоритма на тренировочной базе данных.

Предлагаемый алгоритм локализации, т.е. нахождения возможных координат диполя, следующий :

1) Случайным образом в момент времени t^1 располагаются N_p диполей внутри головы (полной допустимой области) и выбираются их моменты

2) Для каждого p -ого диполя вычисляется ошибка

$$\varepsilon^p = \sum_i \sum_j [\,(U_{exp}(\vartheta_i, \varphi_j) - V_K(\vartheta_i, \varphi_j)) - w^{(p)}(xp \mid \vartheta_i, \varphi_j)]^2 \sin\vartheta_i \, h_{\vartheta i} h_{\varphi j} \qquad (19)$$

3)Задаем уровень допустимой ошибки ε_{th}, и разделяем базу данных (параметры диполей) на два класса - ниже порога (первый класс) $\varepsilon^K < \varepsilon_{th}$ и выше порога (второй класс). Таким образом получена тренировочный набор данных.

4) Для классификации этих данных применяем дерево решений, но используя лишь три (m=3) случайно выбранных признака. Так как набор признаков мал - то это дерево строится без усечения. Выходом построения первого дерева будут три параметра (r_{th}^1,ϑ_{th}^1, φ_{th}^1) - пороги

определяющие трехмерную область локализации диполя первого класса.

5) Для второго момента времени процедура повторяется и строится второе дерево из леса. Выходом построения второго дерева будут свои три параметра - пороги (r_{th}^2,ϑ_{th}^2, φ_{th}^2) определяющие трехмерную область локализации диполя первого класса.

6) Процесс построения леса повторяется и выбирается такая классификация (локализация диполя) за которую проголосовали большинство деревьев.

На рис.3 показана последовательность зон локализации источника, соответствующая четырем моментам времени.

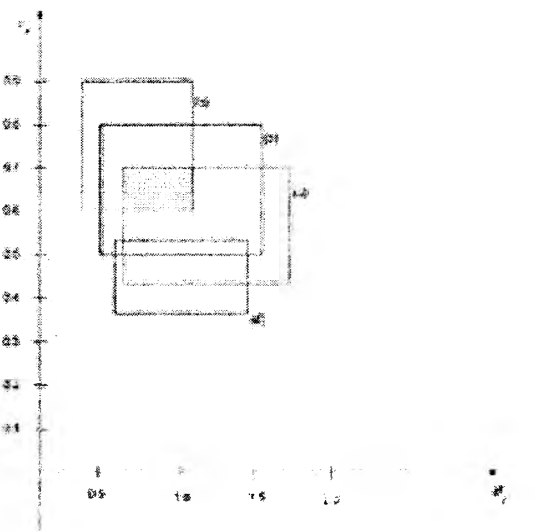


Рисунок 3: Области локализации диполя (на плоскости $r_p - \theta_p$) определенные деревьями решений в четыре последовательные моменты времени

Видно, что в процессе обучения на тренировочном наборе данных, область перекрытия зон, найденных деревьями становится уже и искомое положение диполя находится в этой зоне. Мы можем говорить о сходимости к некоторому среднему положению диполя.

Качество использованного алгоритма классификации зависит от задаваемой точности восстановления потенциала. Поэтому необходимо определить зависимость точности определения локализации диполя от задаваемой погрешности по потенциалу. ε^K.

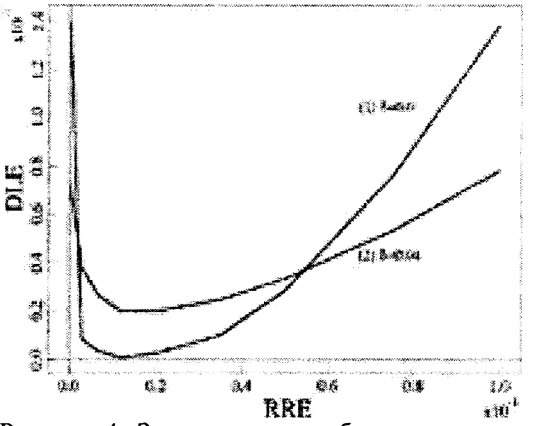


Рисунок 4: Зависимость ошибки локализации диполя DLE от ошибки аппроксимации потенциала на измерительной поверхности RRE: 1) параметр регуляризации δ = 0.01,2) параметр регуляризации δ = 0.04

Ошибка локализации диполя есть сумма ошибки на тренировочных данных и вероятной ошибки на тестовых данных. Чем меньше ошибка по потенциалу, тем уже области локализации в каждый момент времени, тем более вероятно, что зоны перекрытия у большинства деревьев не будет, и повышается вероятность ошибки на тестовых данных. Для уравновешивания этих ошибок вводится параметр регуляризации по областям локализации - δ.

$$\varepsilon = \varepsilon_{train} + \delta \cdot \varepsilon_{test} \qquad (20)$$

Чем больше δ, тем шире зона локализации и , тем меньше точность.

Кривые приведенные на рис.4 показывают, что при небольшой погрешности по потенциалу, меньше 5 % ,погрешность локализации диполя становится того же порядка. При большой погрешности по потенциалу сходимость теряется и получается случайный набор зон и окончательная зона перекрытия слишком большая, чтобы можно было говорить о локализации. Дальнейшее уменьшение задаваемой погрешности было бы некорректно.

В [27] показано, что величина ошибки зависит от двух обстоятельств:

1)-Корреляции между любыми двумя деревьями в лесу. Возрастание корреляции приводит к возрастанию ошибки леса.

2)Силы каждого индивидуального дерева в лесу. Дерево с малой величиной ошибки является сильным (хорошим

классификатором). Возрастание силы индивидуальных деревьев уменьшает ошибку леса.

Уменьшение m уменьшает как корреляцию так и силу дерева. Увеличение m приводит к увеличению обоих этих факторов. Где-то между ними находится оптимальная область значений m - обычно это достаточно широкая область. Используя известную оценку ошибки значение m в области может быть быстро найдено. Это единственный подгоночный параметр по отношению к которому случайные леса чувствительны.

Мы првели численное исследование с целью определения оптимального числа случайно выбираемых признаков из полного набора 6-ти признаков.

Расчеты показали, что 3-4 признака в задаче являются определяющими. В расчетах число m было выбрано равным трем. При этом лучшая достигнутая точность локализации составила 2.1 %.

6. Заключение.

В настоящей работе предложен алгоритм анализ сигналов ЭЭГ на основе метода Random Forest [16]. В рамках дипольной модели источников электрический потенциал измеренный по нескольким каналам на поверхности головы выражен через параметры дипольного источника - его координаты (пространственную локализацию) и вектор момента, которые составляют пространство признаков задачи. Зона локализация источника является одним из выходов деревьев классификации в каждом срезе по времени. Ансамбль деревьев решений ищет перекрытие зон локализации , полученных каждым деревом в отдельные моменты времени. Показано, что использование выборки признаков равной трем в каждом дереве, наиболее эффективно и устойчиво решает проблему локализации, при этом величина структурного риска при классификации на тестовых данных минимальна. Относительная погрешность определения зоны локализации того же порядка как и априори задаваемая погрешность в восстановлении потенциала. Предложенный алгоритм требует одного решения прямой задачи ЭЭГ в каждый момент времени. Вычисления могут быть ускорены за счет параллелизма самого алгоритма Random Forest.

Приводятся экспериментальные расчеты с модельной базой данных , допускающей аналитическое рассмотрение, показывающие эффективность предложенного подхода. Алгоритм хранит данные

классификации в наиболее сжатом виде, что позволяет классифицировать данные ЭЭГ в реальном времени.

Работа выполнена при поддержке гранта РФФИ 05-07-90238

Список литературы

1. K e i r n Z. A. and A u n o n J. I. A new mode of communication between man and his surroundings. // IEEE Trans. Biomed. Eng., vol. 37, pp. 1209-1214, Dec. 1990.

2. Wolpaw J. R., Birb a u in e r N., M c F aria n d D. J.. P f u r t s c h e l l e r G., and V a u g h a n T. M. Brain-computer interfaces for communication and control. // Clin. Neurophysiol., vol. 113, pp. 767-791, 2002.

3. Wolpaw J. R., Birb a u in e r N., M c F aria n d D. J.. P f u r t s c h e l l e r G., and V a u g h a n T. M. Brain-computer interfaces for communication and control. // Clin. Neurophysiol., vol. 113, pp. 767-791, 2002.

4. V i d a l J.I. Real-time detection of brain events in EEC /, Proc. IEEE, vol. 65, pp. G33-664. May 1977.

5. Гнездицкий В.В. "Обратная задача ЭЭГ и клиническая электроэнцефалография" .*Таганрог:Издательство ТРТУ* .-2000.-640 с.

6. J. Tripp, "Physical concepts and mathematical models", *Biomagnetism: An Interdisciplinary approach*, S.J. Williamson, G.L. Romani, L.Kaufman, and I.Modena, Eds. New York: Plenum, 1983, pp. 101-139.

7. К. Хофманн "Численное решение прямых математических задач электроэнцефалографии", *Siemens AG-МГУ*, 2003

8. Захаров Е.В., Коптелов Ю.М. "О некоторых математических проблемах в решении обратной задачи электроэнцефалографии" .- *ДАН* ,1987, т.292, с.578-581.

9. J. Sarvas, "Basic mathematical and electromagnetic concepts of the bio-magnetic inverse problem", *Phys. Med. Biol.*, vol. 32, pp. 11-22, 1987.

10. К. Хоффманн, А.М.Попов, И.А.Федулова, С.Е.Певцов.Численной решение обратных математических задач электроэнцэфалографии, Вестник МГУ, 2004

11. T.R.Knosche.*Solutions of the neuroelectromagnetic inverse problem.* PhD thesis, Univ. of Twente, The Netherlands, 1997.

12. А.Н.Тихонов,А.А.Самарский "Уравнения математической физики" .*Наука,М.* .-1977.-736 с.

13. U.Schmitt, et al. Efficient algorithms for the regularization of dynamic inverse problems-part II: Applications.*Inverse Problems* ,2002. in press.

14. К. Хоффманн, А.М.Попов, И.А.Федулова, С.Е.Певцов.Численной решение пространственно-временные данные обратных математических задач электроэнцэфалографии, Вестник МГУ, 2004

15. L.Breiman Random Forests .- *Machine Learning* ,45,5-32,2001.

16. J.Ross Quinlan C4.5 : Programs for Machine learning *Kaufmann Publishers*, 1993

Федулова И.А.

Идентификация протеинов по аминокислотной последовательности, восстановленной из масс-спектра[1]

Введение

Идентификация протеинов является одной из основных задач протеомики. В последние годы тандемная масс-спектрометрия стала стандартным инструментом для идентификации протеинов [1, 2]. При идентификации протеины сначала разделяются на короткие фрагменты – пептиды с помощью специальных энзимов. Энзимы разрывают связи между определенными аминокислотами. В масс-спектрометре отдельные пептиды изолируются и подвергаются столкновительной диссоциации, разбиваясь на более мелкие фрагменты. Масс-спектр представляет собой совокупность значений масс результирующих фрагментов и их относительных содержаний. Каждый спектр является результатом фрагментации большого числа копий одного и того же пептида [3].

В процессе столкновительной диссоциации чаще всего разрушаются связи между аминокислотами, поэтому в экспериментальном спектре присутствуют пики, соответствующие массам фрагментов пептида, состоящим из целых аминокислот [3]. Это свойство делает возможным предсказание теоретического спектра пептида по его аминокислотной последовательности. Зная последовательность, можно предсказать, появление каких масс (пиков) ожидается в спектре. Кроме того, это свойство можно использовать также для восстановления аминокислотной последовательности пептида по его масс-спектру. Изучая разницы между массами фрагментов (расстояние между пиками в спектре) можно определить аминокислотный состав пептида, а также последовательность, в которой располагались аминокислоты.

Аминокислотные последовательности кодируются буквенными строками, где каждая буква кодирует определенную аминокислоту. В настоящее время существует большое число баз данных, хранящих аминокислотные последовательности различных протеинов. Например, популярная база SwissProt [4] содержит последовательности около 200 000 протеинов.

[1] Работа выполнена при поддержке гранта РФФИ 05-07-90238

Существует два подхода к идентификации протеинов на основе данных масс-спектрометрии. Первый способ основан на том, что в базе данных разыскиваются такие пептиды, для которых теоретически предсказанный спектр схож с экспериментальным. Второй подход основан на восстановлении аминокислотной последовательности пептида по его масс-спектру, с последующим поиском ее, как символьной строки, в базе данных. Таким образом, в первом подходе сравниваются спектры, а во втором – последовательности (строки).

Восстановление аминокислотной последовательности пептида по его масс-спектру называется de novo секвенированием. В последние годы было разработано несколько алгоритмов de novo секвенирования [1-3, 5-7]. Однако существует несколько проблем, которые затрудняют точное восстановление последовательности. Процесс фрагментации пептида во время столкновительной диссоциации зависит от многих параметров, и является почти стохастическим [3]. Связи между некоторыми аминокислотами оказываются слишком сильными, чтобы разрушиться при столкновительной диссоциации. В результате фрагментация получается неполной, и в спектре отсутствуют соответствующие пики. Некоторые пики в спектре могут быть, наоборот, результатом разрушения других (не пептидных) связей, или присутствия химического шума. Существуют две пары аминокислот с очень близкими массами, которые невозможно различить на большинстве использующихся спектрометров. Также, некоторые аминокислоты имеют массу, равную сумме масс двух других аминокислот. Кроме того, дополнительным источником ошибок являются модельные упрощения, которые необходимы при построении алгоритмов de novo секвенирования. Все это приводит к тому, что последовательности, восстановленные алгоритмами de novo секвенирования, содержат ошибки [8]. При поиске de novo секвенированной последовательности в базе данных необходимо учитывать возможные ошибки секвенирования. Таким образом, необходимы алгоритмы приближенного поиска строк (approximate string matching), учитывающие специфику типичных ошибок секвенирования.

В настоящее время существует несколько программ, осуществляющих идентификацию протеинов по de novo последовательности, восстановленной какой-либо из программ de novo секвенирования: CIDentify [5], OpenSea [9] и SPIDER [10]. Все эти программы по-разному учитывают специфику типичных ошибок de novo секвенирования.

Программа CIDentify основана на свободно распространяемом коде FASTA [11], изначально предназначенном для поиска гомологичных последовательностей. Авторами CIDentify код FASTA был изменен таким образом, что замена одной или двух аминокислоты на одну или две другие аминокислоты с такой же суммарной массой

учитывались, как ошибки de novo секвенирования. Авторы OpenSea предложили подход, основанный на выравнивании последовательностей масс, а не букв. OpenSea допускает замены фрагментов произвольной длины, массы которых равны. Программа SPIDER также допускает замены фрагментов произвольной длины, но основана на выравнивании буквенных последовательностей и специальной функции расстояния между строками. Отметим, что программы OpenSea и SPIDER допускают замены фрагментов с эквивалентными массами произвольной длины (предполагаемые ошибки de novo секвенирования). Однако, ясно, что с ростом длины число всевозможных последовательностей, имеющих заданную массу, растет. Таким образом, увеличивается вероятность случайных совпадений и ложных идентификаций.

Целью данной работы является создание алгоритма идентификации протеинов по de novo последовательности, который будет устойчив к некоторым типичным ошибкам de novo секвенирования. Алгоритм реализован на C++. Статья организована следующим образом: в §1 приводится определение специальной функции расстояния между строками и постановка задачи. Вопрос об эффективности алгоритма и возможности применения более эффективных алгоритмов приближенного поиска строк обсуждается в §2. В §3 описывается система оценки результатов поиска. §4 посвящен сравнению производительности предлагаемого алгоритма с двумя известными программами SPIDER и CIDentify на реальных масс-спектрометрических данных.

1. Определение функции расстояния и постановка задачи

Задаче приближенного поиска строк было уделено большое внимание в информатике благодаря широкому кругу приложений, однако существует не так много работ, направленных адаптацию универсальных алгоритмов к специфике задачи идентификации протеинов.

Аминокислотные последовательности пептидов и протеинов могут быть представлены строками, в которых каждая аминокислота закодирована буквой из взвешенного алфавита Σ. Для измерения степени сходства между двумя аминокислотными последовательностями необходимо определить функцию расстояния. В дальнейшем будем обозначать буквами S, A, B строки над алфавитом Σ. Длину строки обозначим $|S|$. Для $i \in 1..|S|$ обозначим $S[i]$ i-й

символ строки S. Обозначим $S[i..j]$ подстроку S, начинающуюся с i-го символа и заканчивающуюся j-м символом. Верхним индексом R обозначим реверсированную строку: например, если $S =$ **'abc'**, то $S^R =$ **'cba'**.

Расстояние редактирования $d(S_1, S_2)$ между строками S_1 и S_2 определяется как минимальная суммарная стоимость операций редактирования, необходимых для трансформации строки S_1 в S_2. Типичными операциями редактирования являются: замена одного символа другим, удаление и вставка символов. Расстояние над этими тремя операциями называется расстоянием Левенштейна [12]. Благодаря удалениям и вставкам, расстояние редактирования позволяет сравнивать строки разной длины. Обозначим C_{sub}, C_{ins}, C_{del} стоимости замены, вставки и удаления соответственно. Наиболее распространенным выбором стоимости является единичная стоимость всех операций. Например, при единичной стоимости всех операций расстояние между строками **'abc'** и **'ad'** равно d(**'abc'**,**'ad'**) = 2.

Расширение расстояния редактирования, предложенное Дамерау [13] добавляет еще одну операцию: перестановка двух соседних символов (т.е. подстановка вида **'ab'**$\rightarrow$**'ba'**). Обозначим стоимость операции перестановки C_{swap}. Подстрочным индексом D будем обозначать расстояние Дамерау: d_D.

Первый и наиболее гибкий алгоритм для вычисления расстояния редактирования основан на динамическом программировании [14]. Он позволяет использовать произвольные стоимости операций редактирования. Значение расстояния $d(A,B)$ между строками A и B, длины которых $m = |A|$ и $n = |B|$ может быть вычислено заполнением $(m+1)\times(n+1)$ матрицы D, в которой элемент $D[i,j]$ содержит значение расстояния $d(A[1..i], B[1..j])$. Трудоемкость заполнения таблицы оценивается временем $O(mn)$.

Приведенный ниже Алгоритм 1 описывает заполнение матрицы динамического программирования D для расстояния Левенштейна.

Алгоритм 1.

1. $D[0,0] = 0$
2. **for** $i = 1$ **to** m
3. $\quad D[i,0] = D[i-1,0] + C_{del}$
4. **for** $j = 1$ **to** n
5. $\quad D[0,j] = D[0,j-1] + C_{ins}$
6. **for** $i = 1$ **to** m

7. **for** $j = 1$ **to** n

8. **if** ($A[i] = B[j]$)

9. $Sub = 0$

10. **else** $Sub = C_{sub}$

11. $$D[i,j] = \min \begin{cases} D[i-1, j-1] + Sub \\ D[i-1, j] + C_{del} \\ D[i, j-1] + C_{ins} \end{cases}$$

Алгоритм для вычисления расстояния Дамерау выглядит аналогично, и отличается только учетом добавления еще одной операции редактирования.

Последовательность операций редактирования, трансформирующих S_1 в S_2, может быть записана в форме выравнивания (alignment). Выравнивание – это набор из двух строк $\tilde{S}_1$ и $\tilde{S}_2$ одинаковой длины над алфавитом $\tilde{\Sigma} = \Sigma + '-'$, где `'-'` обозначает пустой символ. Строки $\tilde{S}_1$ и $\tilde{S}_2$ образованы из S_1 и S_2 вставкой пустых символов `'-'`. Выравнивание обычно записывается как матрица из двух строк, где $\tilde{S}_1$ записана в первой, а $\tilde{S}_2$ – во второй строке. В таком представлении каждый столбец обозначает операцию редактирования: замену символа $S_1[i]$ символом $S_2[j]$ $\begin{pmatrix} S_1[i] \\ S_2[j] \end{pmatrix}$ (если они одинаковы, то стоимость операции равна 0), удаление i-го символа S_1 $\begin{pmatrix} S_1[i] \\ '-' \end{pmatrix}$ или вставку j-го символа S_2 $\begin{pmatrix} '-' \\ S_2[j] \end{pmatrix}$. Будем считать, что ни один столбец не содержит пустых символов в обеих строках, поэтому выравнивание может состоять из максимум $|S_1| + |S_2|$ столбцов. Стоимость выравнивания определяется как сумма стоимостей операций редактирования во всех столбцах. Оптимальное выравнивание соответствует минимальному расстоянию между строками, то есть расстоянию редактирования. Оптимальное выравнивание не является единственным, хотя расстояние редактирование для любых двух строк единственно. Например, два различных оптимальных выравнивания для строк **`'abc'`** и **`'ad'`**:

$$A_1 = \begin{pmatrix} \texttt{'abc'} \\ \texttt{'ad-'} \end{pmatrix} \qquad A_2 = \begin{pmatrix} \texttt{'abc'} \\ \texttt{'a-d'} \end{pmatrix}$$

Оптимальное выравнивание может быть получено обратным проходом по матрице динамического программирования, начиная с

ячейки $D[m,n]$. В каждой ячейке $D[i,j]$ проверяются соседние $D[i-1,j-1]$, $D[i-1,j]$ и $D[i,j-1]$, и определяется из какой ячейки алгоритм мог прийти в $D[i,j]$. В некоторых ячейках выбор может быть неоднозначным, и окончательное выравнивание зависит от нашего выбора. Трудоемкость построения выравнивания оценивается также временем $O(mn)$.

Наиболее типичными ошибками de novo секвенирования являются: перестановка двух соседних аминокислот, замена одной аминокислоты на две более легкие, с такой же суммарной массой, или замена пары аминокислот на одну более тяжелую с такой же массой. Возможны также замены более длинных сегментов последовательности на другие сегменты с такой же суммарной массой [8].

Для обеспечения робастности функции расстояния к перечисленным типичным ошибкам алгоритмов de novo секвенирования в настоящей работе предлагается модификация расстояния Дамерау. Для поиска последовательности пептида в базе данных необходима такая функция расстояния между строками, которая не наказывает типичные ошибки. В идеале, расстояние между ошибочной de novo последовательностью и истинной последовательностью из базы данных должно быть равно нулю. Введенное расстояние будем обозначать d_P.

За основу берется расстояние Дамерау, так как оно позволяет естественным образом учесть перестановки двух соседних символов. Для этого достаточно положить стоимость операции перестановки двух соседних символов равной нулю.

Обозначим h максимальную длину ошибочно определенного сегмента. Алгоритм 2 описывает вычисление модифицированного расстояния. Сначала строится оптимальное выравнивание $\begin{pmatrix} \tilde{S}_1 \\ \tilde{S}_2 \end{pmatrix}$. При построении выравнивания, в каждом случае неоднозначности выбора предпочтение отдается операции замены, а не удаления или вставки, так как в данной задаче ожидается нахождение строки такой же или похожей длины. Далее, в обеих строках разыскиваются соответствующие сегменты выравнивания $\tilde{S}_1[i..(i+h)]$ и $\tilde{S}_2[i..(i+h)]$ длины h, в которых имеется несоответствие (ненулевое расстояние Дамерау d_D между сегментами), а массы сегментов близки с точностью до порога θ. Если такой сегмент найден, то он считается ошибкой de novo алгоритма, и его вклад в расстояние редактирования вычитается из общего расстояния между строками.

Алгоритм 2.

1. $k = d_D(S_1, S_2)$

2. Построение оптимального выравнивания $\begin{pmatrix} \tilde{S}_1 \\ \tilde{S}_2 \end{pmatrix}$

3. $L = |\tilde{S}_1| \quad (= |\tilde{S}_2|)$

4. $k_P = k$

5. **for** $i = 1$ **to** $(L - h)$

6. **if** $(d_D(\tilde{S}_1[i..(i+h)], \tilde{S}_2[i..(i+h)]) > 0$ **and**
$|\text{mass}(\tilde{S}_1[i..(i+h)]) - \text{mass}(\tilde{S}_2[i..(i+h)])| < \theta$

7. $k_P = k_P - d_D(\tilde{S}_1[i..(i+h)], \tilde{S}_2[i..(i+h)])$

8. $d_P(S_1, S_2) = k_P$

Важным свойством модифицированного расстояния является его немонотонное поведение по расстоянию Дамерау, на котором оно основано. В некоторых случаях большое расстояние Дамерау может быть снижено до нуля из-за обнаружения сегментов, содержащих de novo ошибки. В других случаях, когда de novo ошибки не были обнаружены, модифицированное расстояние равно расстоянию Дамерау.

При сравнении строк различной длины более удобно использовать относительное расстояние редактирования $\alpha(S_1, S_2)$, которое определяется как

$$\alpha(S_1, S_2) = \frac{d(S_1, S_2)}{\max\{|S_1|, |S_2|\}}$$

где d – расстояние Левенштейна, Дамерау или модифицированное Дамерау. Относительное расстояние редактирования удовлетворяет неравенству $0 \le \alpha \le 1$.

После того, как определена функция расстояния, можно сформулировать постановку задачи.

Задача 1. По базе данных протеинов $\{T^q, q = 1..Q\}$, de novo последовательности P и порогу ε найти все такие индексы q, i^q, j^q, что выполняется неравенство

$$\alpha_P(T^q[i^q..j^q], P) \le \varepsilon$$

То есть: найти индексы протеинов q, и подстроки в протеинах (пептиды) $T^q[i^q..j^q]$, похожие с точностью до ε в смысле введенного модифицированного расстояния Дамерау d_P на предъявленную de novo последовательность P.

2. Эффективность вычисления расстояния

Алгоритм 1 описывает вычисление расстояния редактирования между двумя строками. Для приближенного поиска короткой строки длины m (пептида) в тексте длины n (протеине) существует простая модификация метода динамического программирования. Единственное изменение заключается в том, что любая позиция в тексте может быть потенциальным началом искомой подстроки. Для этого необходимо положить $D[0, j] = 0$ для всех $j = 1..n$ (см. строку 5 в Алгоритме 1).

В таком подходе заполняется одна большая матрица динамического программирования, нижняя строка которой содержит значения расстояний редактирования между предъявленной строкой и подстроками текста. Существует большое число алгоритмов для вычисления расстояния Левенштейна и Дамерау гораздо более эффективных, чем $O(mn)$ [15-17]. Подробный обзор современных алгоритмов может быть найден в [18]. Большинство из них основывается либо на заполнении только необходимых частей матрицы динамического программирования [15], либо на использовании параллелизма выполнения компьютером битовых операций [16-17]. Однако оба подхода основаны на свойствах монотонности матрицы динамического программирования благодаря специальному выбору стоимости операций редактирования – единичным стоимостям для всех операций.

Вычисление предложенного модифицированного расстояния Дамерау требует построения выравнивания и его последующую обработку, прежде чем окончательное значение расстояния будет получено. Даже если расстояния Дамерау между предъявленной строкой и подстроками текста будут вычислены эффективно, невозможно определить порог для отбора потенциальных кандидатов для вычисления модифицированного расстояния из-за его немонотонного поведения. В таком случае, придется проверять все подстроки текста, выстраивая выравнивания и осуществляя поиск de novo ошибок в каждом из них. Кроме того, при исследовании биологических мутаций в аминокислотной последовательности различных протеинов зачастую требуется, чтобы стоимости различных операций редактирования были различны. Эти стоимости определяются из известной экспериментально статистики, из которой следует, что вероятности замен одних аминокислот на другие аминокислоты различны. Все вышесказанное делает невозможным применение существующих эффективных алгоритмов приближенного поиска строк в данной задаче.

Число всех возможных подстрок в базе данных протеинов слишком велико, для того, чтобы проверять модифицированное расстояние Дамерау до предъявленной de novo последовательности P

для каждой подстроки. Необходимо сделать некоторые предположения, для того, чтобы сузить пространство поиска. Во-первых, можно ограничить длину подстроки-кандидата из базы данных, ограничивая число разрешенных удалений и вставок. Удаления и вставки аминокислот могут являться не только следствием биологических мутаций, но также результатом de novo ошибок, когда одна аминокислота подменяется двумя, или наоборот. Также предлагается использовать технику предварительной фильтрации, когда потенциальная строка-кандидат из базы данных отбирается на основе менее трудоемкой операции сравнения, и только затем вычисляется модифицированное расстояние Дамерау. Более конкретно, предлагается требовать точного совпадения между любой подстрокой длины l de novo последовательности P, и любой подстрокой потенциального кандидата из базы данных, где $l < |P|$ (т.н. l-мер фильтрация). Эффективный поиск общей подстроки длины l между de novo последовательностью P и последовательностью протеина T^q может быть реализован через построение дерева всех l-меров строки P, и затем пропускания всех l-меров текста T^q через это дерево. Если достигнут терминальный лист дерева, то это означает, что общая подстрока длины l найдена. В этом случае определяются позиции i^q и j^q начала и конца потенциального кандидата, и вычисляется модифицированное расстояние Дамерау. Однако, так как разрешены удаления и вставки, то необходимо проверять также соседние позиции $i^q \pm 1$, $i^q \pm 2$, ..., $i^q \pm b$, где b - максимальное число удалений или вставок (см. рис. 1). Аналогично и для позиций конца подстроки-кандидата.

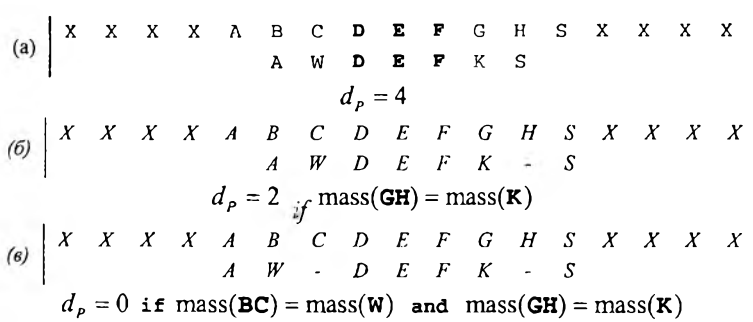


Рис. 1. Пример выравнивания строки **'AWDEFKS'** к подстроке **'ABCDEFGHS'** в тексте. После того, как найдено точное совпадения 3-меров **'DEF'**, разыскиваются наилучшие позиции

начала и конца подстроки в тексте. Так как разрешены удаления и вставки, необходимо проверять несколько соседних позиций.

3. Система оценки результатов поиска

Число подстрок, найденных в базе данных, удовлетворяющих условию Задачи 1, зависит от величины порога ε. Однако, из-за величины базы данных даже для небольших значений порога может быть найдено множество подстрок, среди которых необходимо выбрать правильный ответ. Необходима дополнительная оценка результатов поиска в Задаче 1, чтобы отсеять ошибочные идентификации. В биоинформатике подобная оценка называется скорингом (scoring). После вычисления скоринга все найденные последовательности сортируются в порядке убывания скоринга. В идеальном случае правильный ответ должен иметь максимальный скоринг. На практике ожидается, что правильный ответ должен находиться в первой пятерке последовательностей.

Существующие в настоящее время алгоритмы оценки [5, 9, 10] основаны на сравнении de novo последовательности со последовательностями пептидов-кандидатов. Для каждой последовательности-кандидата оценивается вероятность получения ее из de novo последовательности на основе известной экспериментально статистики биологических мутаций. Однако эта статистика не оценивает вероятность появления de novo ошибок. Информация о масс-спектре используется этими алгоритмами косвенно, только в виде de novo последовательности, предоставленной программами секвенирования.

В данной работе предлагается следующая оценка, основанная на непосредственном использовании масс-спектра идентифицируемого пептида. Каждая найденная в результате решения Задачи 1 последовательность $T^q[i^q..j^q]$ оценивается как взвешенная сумма трех слагаемых:

$$\mathrm{F}(T^q[i^q..j^q]) = w_P f_P + w_M f_M + w_S f_S$$

где w_P, w_M и w_S - веса, удовлетворяющие уравнению $w_P + w_M + w_S = 1$;

f_P - *строковая компонента*, отражающая модифицированное расстояние Дамерау d_P между de novo последовательностью P и последовательностью $T^q[i^q..j^q]$;

f_M - *массовая компонента*, отражающая разность между массами P и $T^q[i^q..j^q]$;

f_S - *спектральная компонента*, отражающая степень сходства между экспериментальным масс-спектром идентифицируемого пептида (по которому была восстановлена последовательность P) и теоретически предсказанным спектром последовательности $T^q[i^q..j^q]$.

Параметры f_P, f_M и f_S удовлетворяют неравенствам $0 \le f_{P,M,S} \le 1$, следовательно скоринг также нормирован: $0 \le F \le 1$ и бóльшие значения скоринга соответствуют лучшему качеству сходства.

Строковая компонента скоринга основана на модифицированном расстоянии Дамерау α_P. Для снижения числа случайных совпадений, в этой компоненте также учитывается длина максимального сегмента в выравнивании, который дает нулевой вклад в расстояние (точное совпадение, с точностью до ошибок de novo). Вероятность случайного возникновения длинного точного совпадения мала. Например, рассмотрим схемы двух выравниваний, соответствующих одинаковому расстоянию: $A_1 = (*...*...*..*)$ и $A_2 = (.........****)$. Здесь точка обозначает совпадение в выравнивании, а звездочка – несовпадение (операцию редактирования). При этом ошибки de novo, найденные Алгоритмом 2, считаются совпадениями. Число несовпадений в обоих выравниваниях одинаково. Однако выравнивание A_2 более вероятно соответствует правильному ответу, чем A_1, если последние четыре несовпадения в A_2 соответствуют ошибке de novo, пропущенной из-за того, что ее длина больше h. Обозначим за L длину выравнивания, а за η длину максимального совпадающего сегмента. Тогда относительная длина максимального совпадающего сегмента определяется как $\mu = \eta / L$.

Определим ненормированную строковую компоненту скоринга формулой $\tilde{f}_P = 1 - \alpha_P + \mu$. Опуская доказательство, заметим, что ее максимальное значение $\tilde{f}_P = 2$. Минимальное значение зависит от порога ε из условия Задачи 1, длины выравнивания L и длины l точного совпадения l-мера, по которому отбираются последовательности кандидаты из базы данных: $\min(\tilde{f}_P) = 1 - \varepsilon + l / L$. Нормированное значение строковой компоненты скоринга, удовлетворяющее $0 \le f_P \le 1$, определяется формулой

$$f_P = \frac{\tilde{f}_P - \min(\tilde{f}_P)}{2 - \min(\tilde{f}_P)}$$

Массовая компонента скоринга учитывает разность между массой идентифицируемого пептида MW_{exp}, и массой пептида-

кандидата, найденного в базе данных MW_{cand}: $\Delta m = MW_{\text{exp}} - MW_{\text{cand}}$. Разность Δm должна находиться в пределах точности σ, обеспечиваемой масс-спектрометром:

$$f_M = \exp\left(-\frac{|\Delta m|^2}{\sigma^2}\right)$$

Спектральная компонента скоринга f_S отражает степень сходства между экспериментальным масс-спектром идентифицируемого пептида, и спектром, предсказанным теоретически для пептида-кандидата из базы данных. Чем больше теоретически предсказанных пиков присутствует в экспериментальном спектре, тем больше вероятность того, что найденный в базе пептид-кандидат соответствует идентифицируемому пептиду, то есть является правильным ответом. Сначала проводится предварительная обработка экспериментального спектра, необходимая для снижения числа шумовых пиков. Деконволюция многозарядных пиков и удаление пиков, соответствующих различным изотопам проводится по алгоритму, описанному в работе [19]. Затем для каждой теоретически предсказанной массы в экспериментальном спектре разыскивается пик с такой же массой. Чем больше теоретически предсказанных масс обнаруживается в экспериментальном спектре, тем большим считается сходство между теоретическим и экспериментальным спектрами. Конкретный метод вычисления сходства между спектрами, используемый в данной работе, взят из работы [2]. Полученная методом PEAKS мера сходства затем нормируется таким образом, чтобы $0 \le f_S \le 1$, и значения, близкие к 1 соответствуют большой степени сходства.

4. Результаты тестов

Для тестирования предлагаемого алгоритма был использован набор данных, состоящий из 144 масс-спектров. Для каждого спектра известна истинная аминокислотная последовательность пептида. Для de novo секвенирования была использована популярная программа PEAKS 3.0 [2]. В качестве базы данных была использована популярная база SwissProt Release 46.5 (12-Apr-2005), содержащая 178 940 протеинов.

Длина l для точно совпадающего l-мера, по которому отбираются последовательности-кандидаты, была взята равной 3. Большинство (119 из 144, 83%) de novo последовательностей, восстановленных PEAKS, имели точно совпадающий 3-мер с истинной последовательностью. Остальные 25 последовательностей не

удовлетворяли этому условию, и поэтому не могли быть правильно идентифицированы алгоритмом из-за редукции модели. Таким образом, верхняя граница для процента успешно идентифицированных последовательностей ограничена 83%.

Оптимальное значение для порога ε было получено экспериментально, путем изучения тестового набора последовательностей. При определении порога необходимо найти баланс между чувствительностью и специфичностью поиска. Для этого было измерено модифицированное расстояние Дамерау между de novo последовательностями, и соответствующими им правильными последовательностями. Гистограмма распределения расстояний изображена на рис. 2. При $\varepsilon = 0.7$ 128 последовательностей удовлетворяют условию $\alpha_P \le \varepsilon$, причем 14 из оставшихся 16 последовательностей не удовлетворяют *l*-мер условию.

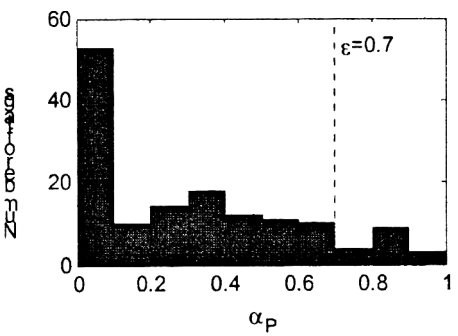


Рис. 2. Гистограмма распределения расстояний α_P для определения оптимального значения порога ε.

Длину h сегмента, содержащего типичную de novo ошибку, было принято равным 2. Большие значения очень сильно замедлили бы поиск. При этом большое значения порога ε позволяют отбирать для последующего скоринга пептиды-кандидаты из базы данных, содержащие более длинные ошибки de novo. В то же время система скоринга позволяет отбросить большинство неправильных ответов.

Значения весов для скоринга были взяты следующими: $w_P = 0.5$, $w_M = 0.25$, $w_S = 0.25$. Эти значения были подобраны экспериментально, чтобы максимизировать число правильно идентифицированных последовательностей. Максимальное число удалений и вставок было ограничено двумя.

Для сравнения производительности предлагаемого алгоритма с существующими программами те же самые de novo последовательности были идентифицированы двумя популярными программами

идентификации протеинов по de novo последовательностям: SPIDER и CIDentify. Результаты работы всех трех программ представлены в Таблице 1. Из таблицы видно, что предлагаемый алгоритм позволяет правильно идентифицировать больше последовательностей, чем две другие программы.

Алгоритм	Top 1	Top 5
Предлагаемый алгоритм	**88 (61%)**	**96 (67%)**
SPIDER	76 (53%)	81 (56%)
CIDentify	70 (49%)	77 (53%)

Таблица 1. Сравнительная производительность трех программ идентификации протеинов на 144 тестовых de novo последовательностях. Столбец Top 1 содержит число правильно идентифицированных пептидов среди последовательностей с максимальным скорингом. Столбец Top 5 содержит число правильно идентифицированных пептидов среди первой пятерки последовательностей.

Цель системы скоринга состоит в том, чтобы по ней можно было судить о вероятности правильности идентификации. Число правильно идентифицированных последовательностей, имеющих максимальный скоринг, отражает качество системы скоринга. Соотношение между последовательностями, имеющими высокий скоринг ($F > 0.7$), и правильно идентифицированными последовательностями, имеющими максимальный скоринг, показано на рис. 3. Общее число последовательностей, имеющих $F > 0.7$, равно 96. Из них 77 последовательностей было правильно идентифицировано, и имело максимальный скоринг. С другой стороны, 88 последовательностей, имеющих максимальный скоринг, было идентифицировано правильно. Из них 77 имело $F > 0.7$, а среднее значение скоринга оставшихся 11 последовательностей равно 0.65. Следовательно, высокие значения скоринга свидетельствуют о правильности идентификации с большой вероятностью.

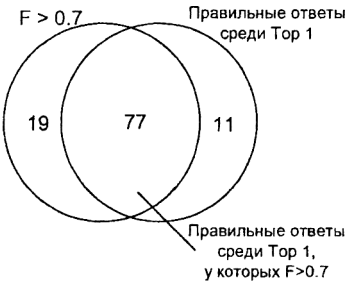


Рис. 3. Соотношение между последовательностями, имеющими высокий скоринг ($F > 0.7$), и правильно идентифицированными последовательностями, имеющими максимальный скоринг («Правильные ответы среди Top 1»).

Заключение

В работе предложена функция расстояния между строками, робастная по отношению к некоторым типичным ошибкам de novo секвенирования. Предложен также новый метод оценки результатов приближенного поиска последовательностей по базе данных, основанный на непосредственном использовании масс-спектра идентифицируемого пептида. Разработанный метод был протестирован на 144 реальных масс-спектрах и правильно идентифицировал больше последовательностей, чем популярные программы SPIDER и CIDentify.

Литература

1. Frank, A.; Pevzner, P. Anal. Chem. 2005, 77, 964-973.

2. Ma, B.; Zhang, K.; Hendrie, C; Liang, C; Li, M.; Doherty-Kirby, A.; Lajoie, G. Rapid Commun. Mass Spectrom. 2003, 17, 2337-2342.

3. Bafna, V.; Edwards, N. Bioinformatics 2001, 17 Suppl 1, S13-21.

4. http://us.expasy.org/sprot/

5. Taylor, J. A.; Johnson, R. S. Rapid Commun. Mass Spectrom. 1997, 11, 1067-1075.

6. Taylor, J. A.; Johnson, R. S. Anal. Chem. 2001, 73, 2594-2604.

7. Dancik, V.; Addona, T. A.; Clauser, K. R.; Vath, J. E.; Pevzner, P. A. J. Comput. Biol. 1999, 6, 327-342.

8. Johnson, R.S.; Davis, M.T.; Taylor, J.A.; Patterson, S.D. Methods 2005, 35(3), 223-236.

9. Searle, B. C; Dasari, S.; Turner, M.; Reddy, A. P.; Choi, D.; Wilmarth, P. A.; McCormack, A. L.; David, L. L.; Nagalla, S. R. Anal. Chem. 2004, 76, 2220-2230.

10.Han, Y.; Ma, B.; Zhang, K. J. Bioinform. Comput. Biol. 2005, 3, 697-716.

11.Pearson, W. R.; Lipman, D. J. Proc. Natl. Acad. Sci. USA 1988, 85, 2444-2448.

12.Levenshtein, V. Soviet Physics Doclady 1966, 10, 707-710.

13.Damerau, F. Comm. of the ACM 1964, 7, 171-176.

14.Wagner, R.; Fisher, M. Journal of the ACM 1974, 21, 168-178.

15.Ukkonen, E. J. Algor. 1985, 6, 132-137.

16.Myers, G. J. ACM 1999, 46, 395-415.

17.Proc. of the Prague Stringology Conference (PCS 2002) A bit-vector algorithm for computing Levenshtein and Damerau edit distances; 2002.

18.Navarro, G. ACM Computing Surveys 2001, 33, 31-88.

19.Wehofsky, M.; Hoffmann, R. Eur. J. Mass Spectrom. 2001, 7, 39-46.

Раздел III

Имитационное моделирование

Савенков К.О.

Использование зависимостей при масштабировании имитационных моделей[1]

Введение

При работе с моделями сложных нелинейных систем часто возникает необходимость их анализа и модификации. Для её решения требуется подобрать такое представление системы, которое бы позволило в задачах анализа глубже понять её, а при модификации – сохранить проверяемые свойства исходной модели. Одна из таких задач – масштабирование имитационных моделей [7], при котором требуется абстрагироваться от излишне детального поведения модели, не влияющего на выполнимость её проверяемых свойств.

Под имитационной моделью мы понимаем совокупность параллельно выполняющихся последовательных процессов, взаимодействующих при помощи механизма передачи сообщений [1]. У каждого процесса есть набор входных и выходных буферов. Сообщение, посланное одним процессом другому при помощи оператора передачи сообщения, помещается во входной буфер последнего, и может быть считано оттуда оператором приёма сообщения [1].

Также в состав модели входит набор исполнителей, определяющих скорость выполнения процессов, привязанных к ним. Каждый процесс обладает своей временной шкалой (модельным временем). При выполнении процессом того или иного оператора генерируется событие, включающее в себя описание выполненного оператора и его операндов, а также временную метку. Эти события фиксируются наблюдателем при выполнении модели с конкретными входными

[1] Работа выполнена при частичной поддержке РФФИ, грант № 04-01-00556

данными и заносятся в трассу имитационного эксперимента. Далее собранная трасса используется для проверки требуемых свойств модели.

Масштабирование имитационных моделей

В случае, если для проверки свойств достаточно наблюдать поведение имитационной модели на высоком уровне абстракции (например, достаточно информации о выполнении операций посылки сообщений), а в трассу заносится более детальная информация (например, содержание пересылаемых сообщений), может оказаться полезным повысить уровень абстракции описания имитационной модели так, чтобы при выполнении полученной модели машинное время не расходовалось на моделирование поведения, не существенного для выполнения проверяемых свойств. Эта задача решается при помощи масштабирования имитационной модели [7].

В имитационном моделировании все проверяемые свойства исходной системы формулируются в терминах наблюдаемых событий имитационной модели. При масштабировании имитационной модели нам дан набор событий, которые должны попасть в поле зрения наблюдателя при проведении имитационного эксперимента. Множество наблюдаемых событий может быть описано как совокупность операторов, информация о выполнении которых нас интересует, и набора операндов, информация о значении которых также должна включаться в событие.

Необходимо, с одной стороны, убрать из описания модели как можно больше операторов, не попавших в поле зрения наблюдателя, а с другой стороны – сохранить поведение модели неизменным с точки зрения наблюдателя. При этом необходимо позаботиться о сохранении временных свойств модели, поэтому часть таких операторов мы можем просто удалить, другую часть необходимо заменить на операторы продвижения модельного времени, остальные нужно оставить на своём месте.

Механизм зависимостей

Таким образом, нам требуется аппарат, ограничивающий нас в удалении ненаблюдаемых операторов из описания модели. Этот аппарат должен помочь нам определить, какие операторы могут быть удалены из описания модели, а какие – влияют на события, генерируемые при

выполнении наблюдаемых операторов, и должны быть оставлены на своём месте.

В качестве подобного аппарата мы предлагаем использовать механизм зависимостей, аналогично тому, как зависимости используются при сечении последовательных программ [5]. Все зависимости мы разделяем на два основных типа: зависимости между операторами одного последовательного процесса (внутрипроцессные), и зависимости, связывающие операторы различных процессов (межпроцессные).

Тогда масштабирование имитационной модели можно провести по аналогии с сечением последовательных программ [5], используя межпроцессные зависимости для транзитивного распространения внутрипроцессных зависимостей за рамки одного последовательного процесса. В этом случае масштабирование имитационной модели может быть выполнено по следующей схеме [7]:

1. Построение множества внутри- и межпроцессных зависимостей между операторами имитационной модели.
2. Построение на основе проверяемых свойств «окна наблюдателя» – множества наблюдаемых операторов модели.
3. Построение остаточных моделей последовательных процессов при помощи транзитивного замыкания множества внутрипроцессных зависимостей по множеству операторов из «окна наблюдателя».
4. Расширение «окна наблюдателя» на основе множества межпроцессных зависимостей.
5. Выполнение шагов 3-4, пока остаточная модель не сойдётся.
6. Удаление или замена на операторы продвижения модельного времени операторов, не попавших в остаточную модель.

Для последовательных процессов имитационной модели мы будем использовать известные зависимости по данным и по управлению [3,4], добавив к ним зависимость по времени выполнения. Для того, чтобы учесть зависимости, возникающие между процессами при их взаимодействии, мы распространим действие зависимостей по данным, управлению и времени выполнения за рамки одного процесса, добавив к межпроцессным зависимостям по данным и управлению зависимости по синхронизации и по исполнителю.

Внутрипроцессные зависимости

Определения основных зависимостей в последовательных программах заимствованы нами не из классических статей Ферранте и Харрольда [3,4], а из статьи Подгурски и Кларка [6], где они изложены в

форме, подходящей не только для последовательных программ, но и для реактивных систем.

Отношение зависимости между операторами последовательного процесса может быть выявлено по тексту описания процесса и используется для предсказания возможного хода его выполнения. Существует два основных типа зависимостей в последовательной программе: зависимости по управлению, которые определяются управляющими конструкциями программы, и зависимости по данным, которые определяются переменными, используемыми в программе.

```
1. input (X,Y);
2. if X > Y then
3.   Max := X;
     else
4.   Max := Y;
     endif;
5. output (Max);
```

Рис 1. Фрагмент процесса, вычисляющего значение максимума.

Оператор s *зависит по управлению* от предиката c, который содержится в операторе условного ветвления, если, согласно структуре потока управления программы, от выбора пути выполнения, на который потенциально влияет c, зависит, будет ли выполнен оператор s. Например, в программе на рис. 1 оператор 3 зависит по управлению от условия ветвления в строке 2.

Оператор s *зависит по данным* от оператора s', если данные, определяемые в s', и используетмые в s, потенциально могут достичь s через последовательность присваиваний переменных. Например, в программе на рис. 1 оператор 5 зависит по данным от опратора 1, поскольку значение Max в операторе 5 потенциально зависит от значений X и Y, определяемых в операторе 1.

Оператор s *синтаксически зависит* от оператора s', если s транзитивно зависит от s' через цепочку зависимостей по данным и управлению. Оператор s *семантически зависит* от оператора s', если существует такая интрепретация модели, что функция, вычисляемая в s, зависит от функции, вычисляемой в s'.

Потенциально, синтаксическая зависимость одного оператора программы от другого может также означать семантическую зависимость [6]. Можно сформулировать утверждение, что s семантически зависит от s', если s синтаксически зависит от s', и s' достижим из s.

Оператор s зависит от оператора s' *по времени выполнения*, если оператор s' может быть выполнен до оператора s', и, тем самым, временная метка выполнения оператора s будет зависеть от временной

метки выполнения оператора s'. Так, в приведённом примере оператор 5 зависит по времени выполнения от оператора 4.

Использование внутрипроцессных зависимостей

Итак, мы рассматриваем зависимости как фактор, ограничивающий возможность удаления операторов из модели. Если событие, отмечающие факт выполнения оператора s, входит в наблюдаемое поведение модели, то множество операторов S_c, от которых s зависит по управлению, также войдут в остаточную модель. Также туда войдут все операторы, от которых операторы из множества S_c зависят синтаксически.

Аналогичным образом, если значения некоторых операндов оператора s отражаются в наблюдаемом поведении модели, то множество операторов S_d, от которых зависят значения данных операндов в точке выполнения s, также войдут в остаточную модель вместе со всеми операторами, зависящими от них синтаксически.

Зависимость по времени выполнения используется на этапе замены части описания модели, не попавшей в остаточную модель, на операторы продвижения модельного времени. Мы можем объединить в одном операторе продвижения модельного времени задержки для ряда операторов, которые в исходной модели связаны отношением непосредственного доминирования [6]. Те операторы, от которых зависят по управлению операторы, заменяемые на задержки модельного времени, должны быть добавлены в остаточную модель.

Межпроцессные зависимости

Существующие работы, посвящённые зависимостям в распределённых программах, рассматривают программы над общей памятью, лишь вскользь упоминая зависимости, возникающие при передаче сообщений [2].

Для того, чтобы учесть влияние взаимодействующих процессов друг на друга, мы вводим ряд зависимостей между операторами посылки и приёма сообщений различных процессов. Это зависимости по данным и управлению, определяемых очевидным образом, а также *зависимость по синхронизации*, которая связывает два оператора различных процессов, один из которых должен дождаться выполнения другого. Данный вид зависимости используется для того, чтобы

распространить зависимость по времени выполнения на процессы, с которыми взаимодействует данный процесс.

Коснёмся ещё одного вида зависимостей – *зависимости по исполнителю*. Если два последовательных процесса привязаны к одному исполнителю, то это накладывает ограничения на частичный порядок выполнения их действий. Их параллельное исполнение будет организовано по принципу чередования, что приведёт к дополнительным зависимостям по времени выполнения между действиями данных процессов.

Заключение

В данной работе мы адаптировали известные зависимости, используемые для анализа последовательных программ, для анализа дискретно-событийных имитационных моделей, разработанных для среды DYANA. Также мы предложили две новых зависимости, по времени выполнения и исполнителю, которые позволяют анализировать временные свойства модели.

Описанные подобным образом зависимости можно использовать и для масштабирования систем взаимодействующих процессов, описанных в других парадигмах имитационного моделирования.

Подгурски и Кларк [6] доказали, что синтаксическая зависимость между двумя операторами является необходимым, не достаточным условием существования между ними семантической зависимости. Этот результат означает, что при масштабировании имитационной модели ряд операторов, оставленных в ней согласно построенному множеству синтаксических зависимостей, могут оказаться несущественными для выполнения проверяемых свойств модели.

Наиболее перспективным способом уточнения синтаксической зависимости нам представляется использование анализа выполнимости путей в имитационной модели, в частности, статический анализ потока данных между процессами.

Литература

1. A. Bakhmurov, A. Kapitonova, R. Smeliansky, DYANA: An Environment for Embedded System Design and Analysis, *Proc. of 32-nd Annual Simulation Symposium*, San Diego, California, USA, April 11-15, 1999.

2. Jingde Cheng. Dependence Analysis of Parallel and Distributed Programs and Its Applications. *APDC 1997*: 370-377.

3. Jeanne Ferrante , Karl J. Ottenstein , Joe D. Warren, The program Dependence Graph and its Use in Optimization, *Proceedings of the 6th Colloquium on International Symposium on Programming*, p.125-132, April 17-19, 1984

4. Harrold, M. J., Malloy, B., and Rothermel, G. 1993. Efficient construction of program dependence graphs. In *Proceedings of the 1993 ACM SIGSOFT ISSTA* '93. ACM Press, New York, NY, 160-170.

5. Horwitz, S., Reps, T., and Binkley, D. 1990. Interprocedural slicing using dependence graphs. *ACM Trans. Program. Lang. Syst.* 12, 1 (Jan. 1990), 26-60.

6. Podgurski, A. and Clarke, L. A. 1990. A Formal Model of Program Dependences and its Implications for Software Testing, Debugging, and Maintenance. *IEEE Trans. Softw. Eng.* 16, 9 (Sep. 1990), 965-979

7. Савенков К.О., Смелянский Р.Л., Автоматическое масштабирование дискретно-событийных имитационных моделей, готовится к публикации.

Прус В. В.

Метод оценки наихудшего времени выполнения для процессоров с конвейерной архитектурой

Введение

Для систем реального времени значительную актуальность представляет задача оценки наихудшего времени выполнения (WCET) программ. Общая схема решения задачи состоит из:

1. Нахождения пути с максимальным временем выполнения в графе программы.
2. Проверки пути на выполнимость с точки зрения семантики языка программирования.
3. В случае невыполнимости найденного пути – перебора следующих по времени выполнения путей.

В случае, если время выполнения всех инструкций является константой, первая подзадача решается классическим графовым алгоритмом [2, 3]. Для проверки выполнимости путей также имеется несколько методов [1, 5]. Наконец, в работе [10] нами предложен метод эффективного перебора путей в порядке уменьшения длины.

Однако, при конвейерной архитектуре процессора время выполнения последовательности инструкций не равно сумме времен выполнения отдельных инструкций. То есть, при поиске пути с максимальным временем выполнения необходимо учитывать логику работы конвейера и классические алгоритмы поиска самого длинного пути непосредственно не применимы. С другой стороны, известные методы учета конвейерного выполнения при оценке WCET являются относительно неэффективными по времени.

Построение эффективного по времени и при этом точного алгоритма поиска пути с максимальным временем выполнения в графе программы для процессоров с конвейерной архитектурой и является целью данной статьи.

Дальнейшая часть работы построена следующим образом. В разделе 2 кратко кратко описывается используемый метод моделирования процессора. В разделе 3 строится алгоритм, который на основе конкретного способа описания процессора достигает потактовой точности оценки. В разделе 4 приводятся теоретическая и практическая оценки построенного алгоритма. Раздел 5 подводит итоги работы и формулирует направления дальнейшего исследования.

Метод описания процессора

В нашей работе мы будет использовать метод моделирования процессора, предложенный в работе [8]. Этот метод достаточно прост и эффективен, однако его применимость к оценке WCET ранее не исследовалась.

Все методы описания временного поведения процессора могут быть представлены как сочетание метода описания состояния процессора в данный момент, и правил, которые по состоянию процессора в конкретный момент времени и набору очередных инструкций определяют состояние процессора в следующий момент времени. В работе [8] предполагается, что процессор содержит набор *ресурсов*, каждый из которых в конкретный момент времени либо захвачен, либо свободен. *Состояние процессора* определим как набор моментов времени, в которых каждый конкретный ресурс будет, или был освобожден. Формально, состояние процессора это вектор $\{r_1,\dots,r_R\}, r_i \in N$, где R – количество ресурсов. Один из ресурсов r_f выделен – его время освобождения считается оценкой времени выполнения. Состояние процессора рассматриваются только в моменты времени после запуска на выполнение очередной инструкции. Для того, чтобы имея состояние процессора и очередную инструкцию, получить состояние процессора после запуска этой инструкции, каждой инструкции приписан *набор формул*. Формулы выражают времена освобождения ресурсов после запуска инструкции через времена освобождения ресурсов в текущем состоянии процессора. Все формулы имеют следующий вид:

$$r_i = max_j\left(r'_j + c'_{ij}\right)$$

где r_i – новое время освобождения ресурса i, r'_j – текущее время освобождения ресурса j, а все c'_{ij} – константы. В случае, если r_i не зависит от r'_j, будем считать, что c'_{ij} равна $-\infty$. В случае, если для одного из ресурсов не задана формула, его состояние не изменяется. Оценка времени выполнения конкретного пути производится путем последовательного применения формул для всех инструкций к состоянию процессора. При этом будет получено состояние процессора после выполнения всех инструкций. Числовая оценка времени выполнения может быть получена по финальному состоянию процессора как значение времени освобождения ресурса r_f.

Алгоритм оценки

Будем предполагать, что граф программы, подлежащей оценке, является ациклическим. При анализе WCET количество итераций каждого цикла априорно ограничено, и поэтому для произвольной программы можно развернуть все циклы, и получить программу с ациклическим графом.

Каким образом, при заданном методе описания процессора, можно найти путь с максимальным временем выполнения в ациклическом графе? Возможно по очереди оценить все пути в графе, и найти максимальное время выполнения. Но поскольку количество путей может быть экспоненциальным, сложность такого подхода неприемлема.

Нам требуется какой-то метод ``объединения" состояния процессора в промежуточных вершинах графа. Приведем пример. Пусть имеются два пути из начальной вершины в конечную, и эти пути имеют общий финальный подпуть (см. рис 1). Если оба пути оцениваются независимо, то учет времени выполнения инструкций $T_1,\dots,T_N$ выполняется дважды. Чтобы избежать этого, необходимо каким-либо образом ``объединить" состояния процессора в точке T_1, в которой пути сходятся. То есть, если F^1 и F^2 – состояния в вершине T_1 на двух разных путях, желательно заменить двукратную оценки пути $T_1 \to T_N$ с начальными состояниями F^1 и F^2 однократной оценкой с некоторым состоянием F и получить ту же оценку. Оказывается, что для выбранного способа описания процессора такое ``объединение" состояний возможно.

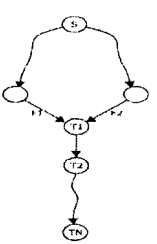


Рис 1: Пример графа программы

Напомним, что состояние процессора определяется как вектор из R натуральных чисел и одно из этих чисел используется как оценка времени. Тем самым, получив поэлементный максимум состояний процессора по всем путям в конечной вершине графа, возможно получить и наихудшее время выполнения.

Пусть имеется ориентированный граф $G(V,E)$ с начальной вершиной S. Пусть из вершины S в вершину T_1 ведут два пути, и в зависимости от выбранного пути имеются два возможных состояния процессора перед выполнением вершины T_1 – F^1 и F^2. Определим $F \circ P$ как состояние процессора, получаемое после выполнения пути P при начальном состоянии F. Определим $max(F^1,F^2)$ как поэлементный максимум состояний F^1 и F^2.

Теорема. Если T_1 и T_N – вершины графа, F^1 и F^2 – два возможных состояния процессора перед выполнением вершины T_1, а P – путь от T_1 до T_N, то имеет место следующее равенство:

$$max(F^1 \circ P, F^2 \circ P) = max(F^1, F^2) \circ P \qquad (1)$$

Доказательство. Рассмотрим i-ю вершину на пути $T_1 \to T_N$ – T_i. Пусть F_i^1 и F_i^2 – состояния процессора перед выполнением вершины T_i, а F_{i+1}^1 и F_{i+1}^2 – после выполнения этой вершины. Для j-го элемента в состоянии процессора имеем

$$max(F_{i+1}^1, F_{i+1}^2)_j = max\left(max_k\left(F_{i,k}^1 + c_{j,k}^{i+1}\right), max_k\left(F_{i,k}^2 + c_{j,k}^{i+1}\right)\right)$$
$$= max_k\left(max\left(F_{i,k}^1, F_{i,k}^2\right) + c_{j,k}^{i+1}\right)$$

из чего получаем:

$$max(F_{i+1}^1, F_{i+1}^2) = max(F_i^1, F_i^2) \circ (T_i \to T_{i+1})$$

и для вершины T_N получим:

$$max(F_N^1, F_N^2) = max(F_{1,}^1 F_1^2) \circ (T_1 \to T_N)$$

Теорема доказана.

На основании доказанной теоремы, поиск наихудшего пути может быть выполнен по следующему алгоритму:

1. Сопоставить начальной вершине графа начальное состояние процессора

2. Произвести топологический обход графа.
3. Для каждой вершины в порядке обхода, найти поэлементный максимум состояний процессора, сопоставленных всем вершинам-предшественникам. Применить формулу для вершины к полученному состоянию, и сопоставить результат текущей вершине.

Оценим сложность построенного алгоритма. Прежде всего отметим, что для графов программ количество исходящих дуг ограничено двумя, если не рассматривать передачи управления по вычисленным адресам. То есть, количество дуг в графе не превосходит удвоенного количества вершин.

Сложность топологического обхода составляет $O(V+E)=O(V)$. Для каждой вершины необходимо найти поэлементный максимум всех входящих состояний, что требует $O(ER)$ операций, где R – количество ресурсов процессора. Наконец, необходимо применение формул в каждой вершине. Сложность этого – $O(F)$, где F – суммарное количество элементов формул, не равных $-\infty$. Таким образом, итоговая сложность алгоритма – $O(VR+F)$.

Сравнение с существующими алгоритмами

Реализация описанного алгоритма не представляет практических сложностей и была выполнена нами на языке C++ с использованием библиотеки Boost [11]. В качестве моделируемого процессора использовался процессор NM6403 [7].

Из существующих методов оценки WCET наиболее эффективным является [6]. В этой работе для каждой пары последовательных инструкций вычисляется количество *сэкономленных тактов*. Это позволяет свести задачу оценку WCET к поиску самого длинного пути в графе. Однако, при этом возможна переоценка максимального времени. Так, в работе [9] обнаружен сценарий, на котором переоценка для процессора NM6403 может достигать 50 процентов.

Для вычисления количества сэкономленных тактов необходимо вычислить время выполнения каждой пары инструкций. Оценим сложность этой операции. Для каждой пары вершин мы последовательно применяем формулы, соответствующие им, к начальному состоянию процессора. Количество пар линейно зависит от V, и для каждой пары вычисляется R элементов, что дает $O(VR)$. Суммарное количество операций при вычислении самих формул

составляет $O(F)$, и в итоге получаем $O(VR+F)$. Сложность последующего поиска самого длинного пути $O(V+E)=O(V)$, таким образом, итоговая сложность составляет $O(VR+F)$, что совпадает со сложность разработанного нами алгоритма.

При равной сложности, преимуществом нашего алгоритма является потактовая точность

Практическое сравнение

Помимо теоретической оценки сложности, целесообразно произвести также практическое сравнение. Работа алгоритма зависит от двух факторов – структуры графа программы и конкретных инструкций в вершинах графа. Для максимально полного тестирования было решено использовать набор автоматически сгенерированных тестовых программ. Алгоритм построения тестовой программы таков:

1. Задается количество линейных участков в программе.
2. Для каждого линейного участка случайным образом определяется количество инструкций (от 1 до заданного максимума)
3. В качестве последней инструкции каждого линейного участка выбирается инструкция перехода.
4. Все остальные инструкции выбираются случайным образом
5. Линейный участок, на который производится переход, также выбирается случайным образом, с тем ограничением, все переходы в программе направлены в сторону последнего участка. Это позволяет гарантировать ацикличность получаемого графа.

Параметрами алгоритма являются максимальный размер линейного участка и максимальное расстояние перехода. Инструкции выбирались случайным образом из набора инструкций, взятого из реальных программ. Оказалось, что результаты сравнения практически не зависят от параметров алгоритма генерации программ. Для одного из набора параметров, графики зависимости времени работы алгоритма от количества линейных участков, для первой версии предложенного алгоритма и алгоритма [6] приведено на рис. 2. В среднем, разработанный нами алгоритм в два раза быстрее.

Рис 2: Результаты практического сравнения

Сравнение с другими работами

Работа [6] является единственной, которая пытается решить задачу оценки наихудшего времени для конвейеризованных процессоров с помощью непосредственного поиска пути по графу. Все другие работы пытаются использовать сведение задачи к задаче целочисленного линейного программирования, что требует значительных вычислительных ресурсов.

Можно также отметить связь нашей работы с работой [4], в которой каждой инструкции сопоставляется набор вершин (каждая из которых соответствует ступени конвейера), а дуги между вершинами соответствуют зависимостям между ступенями конвейера. Однако, эта модель используется лишь для доказательства различных временных свойств. Представляется возможным использование этой модели для поиска наихудшего времени выполнения, однако этот вопрос в работе [2] не рассматривается. Кроме этого, применимость указанной модели к реальным процессорам также не рассмотрена.

Заключение и направления дальнейшей работы

В работе был разработан алгоритм поиска пути с наихудшим временем выполнения для процессоров с конвейерной архитектурой. Главным отличием алгоритма от существующего аналога является потактовая точность оценки. Про этом, по теоретической сложности алгоритм не хуже, а практической производительности несколько превосходит существующий алгоритм.

Главным ограничением является требование ациклического графа программы. То есть, для практического применения все циклы в программе должны быть развернуты. Хотя такое решение является традиционным для задачи WCET, из соображений производительности желательно избежать полного развертывания. Эта тема заслуживает дальнейшего исследования.

Литература

1. P. Altenbernd. On the false path problem in hard realtime programs//Proceedings of the 8th Euromicro Workshop on RealTime Systems : pages 102–107, June 1996.

2. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. Introduction to Algorithms: MIT Press/McGrawHill, 1990.

3. E. W. Dijkstra. A note on two problems in connection with graphs// Numerische Math , 1:269–271, 1959.

4. J. Engblom and B. Jonsson. Processor pipelines and their properties for static wcet analysis, 2002.

5. Torsten Robschink and Gregor Snelting. Efficient path conditions in dependence graphs//Proceedings of the 24th International Conference on Software Engineering: ACM Press, 2002, pages 478–488..

6. Friedhelm Stappert, Andreas Ermedahl, and Jakob Engblom. Efficient longest executable path search for programs with complex flows and pipeline effects//CASES , pages 132–140, 2001.

7. АО "Модуль". Л1879ВМ1 (NM6403): Руководство пользователя, 2000.

8. Савенков К.О. и Ющенко Н.В. Методика описания поведения процессора для оценки времени выполнения программы//Труды Всерссийской неучной конференции 'Методы и средства обработки информации', стр. 486-591, 2003

9. В.В. Слайковский. Учет архитектурных особенностей при анализе наихудшего времени выполнения программы. Курсовая работа// Московский Государственный Университет, факультет ВМиК, 2003

10.Прус В.В. Эффективный алгоритм перебора кратчайших путей в графе//*Труды Всерссийской неучной конференции 'Методы и средства обработки информации'*, стр. 474-479, 2003

11.C++ boost libraries. http://boost.org

Беззубцев С.О., Ющенко Н.В.

Использование адаптера Arinc429-3 в стенде полунатурного моделирования

Введение

Краткое описание стенда моделирования

Стенд полунатурного моделирования [1] предназначен для поддержки процесса интеграции отдельных компонентов авиационной бортового оборудования в единую систему. Стенд позволяет проводить эксперименты, в ходе которых некоторые компоненты бортовой системы представлены реальными приборами, а другие заменены программными моделями. Взаимодействие с реальными приборами осуществляется по каналам бортовых интерфейсов MIL-STD 1553B (ГОСТ 52070-2003) и Arinc429 (ГОСТ 18977-79).

Аппаратные компоненты стенда включают:

- инструментальные машины частных моделей (ИМ ЧМ) – машины, на которых выполняются программные модели компонентов бортовой системы;
- макет бортовой центральной вычислительной машины (БЦВМ) – вычислительная машина, выполняющая в процессе моделирования функции бортовой вычислительной машины;
- автоматизированные рабочие места (АРМ) – вычислительные машины, предназначенные для работы инженеров – разработки моделей, подготовки и запуска экспериментов, оперативного контроля за ходом экспериментов и т.д.;
- каналы бортовых интерфейсов.

Программное обеспечение ИМ ЧМ включает средства, обеспечивающие выполнение моделей в реальном времени, взаимодействие моделей между собой и с реальными приборами, сбор трассы событий и т.п. Имеется также библиотека моделей, в т.ч. программные модели каналов MIL-STD и Arinc. ИМ ЧМ работают под управлением ОС Linux с расширениями, обеспечивающими работу в жёстком реальном времени (RTMS – RealTime Modelling Support).

Краткое описание стандарта Arinc429

Arinc429 [2] - стандарт передачи данных по линиям последовательного интерфейса. Стандарт определяет принципы взаимодействия устройств по каналам связи, форматы данных, характеристики физической линии передачи и т.д.

Устройства соединяются каналами по принципу точка-точка. Данные передаются в виде 32-разрядных слов. Возможные форматы слов определяются стандартом; в большинстве форматов выделяется 8-битовое поле адреса, указывающее назначение данных, поле данных, бит чётности.

Стандарт определяет несколько частот работы каналов: 12.5 КГц, 50 КГц, 100 КГц. На частоте 100 КГц пропускная способность канала составляет 2778 слов в секунду.

Кроме каналов данных, стандартом определены сигнальные линии. Сигнальная линия – дополнительное соединение двух устройств. В каждый момент времени на сигнальной линии возможно состояние с высоким потенциалом, что соответствует логическому нулю, или с низким потенциалом, что соответствует логической единице.

Сигнальные линии используются для организации обменов "по запросу" и "по готовности". При обмене "по запросу" сначала приёмник устанавливает высокий (или низкий) уровень на сигнальной линии, а в ответ на это передатчик начинает передачу слова или нескольких слов. При обмене "по готовности" передатчик сначала уведомляет приёмник о готовности данных путём выставления высокого (или низкого) уровня на сигнальной линии, а через некоторое время начинает передачу. Существует также "асинхронный" режим, в котором сигнальные линии не используются, а передача ведётся непрерывно.

Краткое описание адаптера Arinc429-3

Адаптер семейства Arinc429-3 [3] предназначен для сопряжения вычислительной машины семейства x86 с приборами и устройствами по линиям последовательного интерфейса Arinc429. Адаптеры Arinc429-3 выполнены в виде платы, встраиваемой в 16-ти разрядный слот шины ISA. В зависимости от конкретной модели, адаптеры имеют 4 или 8 входящих каналов, 2, 4 или 8 исходящих каналов, 4 входящих и 4 исходящих сигнальных линии. Адаптер функционирует под управлением контроллера ПЛИС EPM 7000S, работающего на частоте 32 МГц. В состав адаптера входит внутреннее ОЗУ ёмкостью в 32768 16-разрядных слов, доступ к которому

осуществляется через порты шины ISA. Время записи или чтения слова составляет от 0.5 до 1 микросекунды.

Все каналы могут работать параллельно и независимо.

Для каждого выходного канала в ОЗУ адаптера заводится расписание обмена, каждый элемент которого содержит адрес, откуда должно быть взято слово для передачи. После запуска канала расписание может быть выполнено один раз, или же обмен может выполняться циклически, согласно настройке канала.

Входящие каналы могу работать в двух режимах – приём по адресу слова и приём по счётчику слов. При приёме по адресу, в ОЗУ заводится буфер из 256 адресов, и ячейка для сохранения каждого слова определяется значением поля адреса этого слова. При приёме по счётчику слов, все принятые слова записываются в последовательные ячейки ОЗУ.

Организация работы адаптера

Обычный режим использования адаптера Arinc429-3 предполагает предварительную запись значительного объёма информации в память адаптера, после чего адаптер работает в автономно, а обмен между компьютером и адаптером ограничивается модификацией отдельных слов передаваемой циклограммы и чтения принятых слов. Кроме того, обычно для задачи, обменивающейся данными по каналу Arinc с помощью данного адаптера, выделяется отдельный компьютер; это позволяет оптимизировать работу компьютера именно под эту задачу.

В случае применения адаптера в стенде требования к режиму использования совсем другие:

- может быть определено несколько циклограмм, между которыми требуется переключение за время, сравнимое со временем передачи слова;
- принятые слова надо передавать моделям в реальном времени приёма;
- необходимо фиксировать время приёма отдельных слов;
- во избежание рассогласования модельного времени с физическим, недопустимо использовать для обмена с адаптером непрерывные интервалы времени, превышающие несколько десятков микросекунд.

Для обеспечения выполнения этих требований, работа с адаптером была организована в соответствии со следующими принципами.

- Принцип фоновой обработки.

Действия по поддержке работы адаптера разбиваются на небольшие подзадачи, каждая из которых может быть выполнена не более чем за 30-40 микросекунд. Для выполнения этих подзадач работа ИМ ЧМ прерывается не чаще чем раз в 150-200 микросекунд. Такой режим позволяет обслуживать адаптер, не занимая процессор надолго и тем самым не нарушая синхронизацию модельного времени с физическим в процессе моделирования. Техническую возможность такого режима обеспечивает среда RTMS.

- Принцип ранней записи статической информации. Всё, что можно записать в ОЗУ адаптера до начала моделирования, записывается до начала моделирования. В частности, заранее создаются циклограммы обмена.
- Принцип поздней записи данных. Запись в ОЗУ адаптера слова для передачи откладывается на последний момент, когда её можно сделать. Это позволяет уменьшить количество обменов в случае, когда модификация слова циклограммы осуществляется чаще, чем его передача. Конкретно, запись n-го слова осуществляется в интервал времени (продолжительностью 360 микросекунд), когда адаптер передаёт (n-2)-е слово. Техническую возможность получить управление в точно указанный момент времени предоставляет среда RTMS.
- Принцип раннего чтения. Делается попытка узнать о данных, принятых адаптером, как можно раньше. При этом нельзя обеспечить прерывания по приходу каждого слова, т.к. в наихудшем случае одновременной работы 8 каналов на максимальной частоте это приведёт к 8 прерываниям за 360 микросекунд, т.е. в среднем к прерыванию каждые 45 микросекунд. Подобный режим современный компьютер не выдержит.

Основной мерой допустимости количества операций, выполняемых за одно "фоновое пробуждение" драйвера адаптера, является количество операций обмена с адаптером. В наихудшем случае (который на практике случается очень часто), один обмен занимает 1 микросекунду, при этом процессор фактически простаивает[1].

[1] Точнее, в течение этой микросекунды процессор выполняет единственную машинную операцию обмена. Для сравнения, при

Соответственно, для того, чтобы не занимать время процессора больше чем на 30-40 микросекунд подряд, необходимо так спланировать работу, чтобы общее количество обменов не превышало 20-25. Оставшиеся микросекунды требуются на получение прерывания по таймеру, запуск обработчика и выполнение его кода.

Большинство операций по управлению адаптером сводятся к чтению или записи слов данных в ОЗУ адаптера. При этом аппаратный интерфейс таков, что для чтения или записи n последовательных слов требуется (n+1) операция обмена – сначала необходимо занести необходимый адрес в регистр адреса, а потом осуществлять обмены через регистр данных, при этом каждый обмен приводит к автоинкременту адреса.

В частности, для чтения регистра состояния одного канала (в частности, сколько слов принято или передано), требуется 2 обмена. Для чтения или записи одного слова Arinc429, в котором 32 разряда, требуется 3 обмена.

Фоновая работа организована следующим образом.

Выделяются "слоты" - моменты, когда возможно получение управления фоновым обработчиком.

Выделяются действия, которые может потребоваться сделать с каналом:

- запись данных на исходящий канал,
- проверка состояния входящего канала,
- чтение данных входящего канала.

Выполнение действий планируется на слоты согласно следующим правилам:

- запись слова данных на исходящий канал планируется в соответствии с принципом поздней записи;
- чтение данных входящего канала планируется в соответствии с результатами проверки состояния, и так, чтобы количество запланированных на слот обменов не превысило допустимый предел;
- проверка состояния планируется, если состояние достаточно давно не проверялось, и так, чтобы количество запланированных на слот обменов не превысило допустимый предел.

тактовой частоте в 1 ГГц время выполнения "обычной" машинной команды имеет порядок 1 наносекунды, т.е. в 1000 раз меньше.

В случае, если на некоторый слот никаких действий не запланировано, работа ИМ ЧМ в соответствующий момент времени не прерывается.

Такие действия как запуск и останов каналов, смена циклограмм, получение от работающих моделей данных для передачи и т.п., осуществляются вне слотов. Для этого драйвер предоставляет соответствующий интерфейс, используемый моделями непосредственно.

Синхронизация часов

Используемая организация управления адаптером требует высокой точности в выборе момента времени, в который выполняются действия на компьютере, по отношению к работе каналов адаптера.

Для высокоточного измерения времени на компьютере можно использовать команду rdtsc процессора, считывающую аппаратный счётчик тактов. Разрешение такого "таймера" определяется тактовой частотой процессора; при тактовой частоте в 1 ГГц оно составляет одну наносекунду – что на несколько порядков точнее всех остальных доступных методов.

Среда RTMS предоставляет возможность запланировать вызов обработчика на указанный такт процессора. Технически это реализуется при помощи программирования таймера APIC; точность срабатывания которого составляет порядка сотен тактов процессора. Однако, из-за возможности запрета прерываний на некоторые интервалы времени обработчик может получить управление с запаздыванием, определяемым максимально возможной продолжительностью интервала запрета прерываний. На практике, с учётом параллельной работы драйверов адаптеров Ethernet и MIL-STD, удаётся добиться гарантированной точности порядка 30-40 микросекунд, при этом большинство вызовов осуществляются с точностью в пределах единиц микросекунд.

Однако, точного вызова по отношению к тактам процессора недостаточно. Каналы на адаптере управляются тактовым генератором адаптера; частоты как тактового генератора процессора, так и тактового генератора адаптера выдерживаются с конечной точностью, и постепенно "часы адаптера" и "часы компьютера" будут расходиться. Если погрешность часов составляет 0.001% (величина, близкая к реальности), то расхождение в 180 микросекунд, которое вызовет некорректное управление адаптером, будет накапливаться за 18 секунд. В то время как продолжительность экспериментов на стенде может составлять многие минуты и даже часы.

Это означает, что для корректного управления адаптером требуется принимать меры по синхронизации "часов адаптера" с "часами компьютера". Точнее, расхождение часов необходимо учитывать при определении моментов времени, когда будут осуществляться обмены с адаптером.

Чтобы обеспечить это, необходим некоторый механизм получения информации о "ходе времени" на адаптере. В качестве такого механизма можно использовать прерывания, генерируемые адаптером после передачи n-го слова. Согласно документации адаптера, эти прерывания генерируются не позже, чем через 80 микросекунд после передачи слова. С учётом возможной задержки вызова обработчика прерывания, этот механизм позволяет получить информацию о времени адаптера с точностью порядка 120 микросекунд. Этого оказывается достаточно для поддержания устойчивого управления адаптером.

Если ни один исходящий канал на адаптере не работает, то этот метод применить нельзя. Однако, в этом случае не требуется и столь высокая точность синхронизации – так как операции по обслуживанию входящих каналов не требуется выполнять в моменты, синхронизированные с работой каналов.

Организация циклограмм

Для обеспечения работы адаптера в стенде пришлось также преодолеть ряд ограничений.

Так, хотя адаптер и может обеспечить циклическую передачу, когда выполнение циклограммы после завершения начинается заново, такая работа требует располагать циклограмму начиная с нулевого смещения в буфере канала. В результате для канала может быть определено не более двух таких циклограмм (по числу буферов). Это ограничение для задач стенда оказывается существенным.

Однако возможность выдавать команды на адаптер с высокой точностью позволила преодолеть это ограничение. А именно, если в момент передачи последнего слова циклограммы перезаписать соответствующий регистр адаптера, то адаптер без остановки продолжит передачу с начала циклограммы.

Эксперименты показали, что такая схема работает надёжно. В результате количество циклограмм ограниченно только размером буферов для их размещения; на практике этого оказалось достаточно.

Заключение

В данной статье была рассмотрена организация работы адаптера Arinc429-3 фирмы "Элкус" в составе стенда полунатурного моделирования. Несмотря на плохие аппаратные характеристики адаптера, особенно в части времени обменов с компьютером, удалось добиться надёжной работы в реальном времени с высокой точностью.

Таким образом была показана практическая осуществимость управления устройством с обычного персонального компьютера для случая, когда управляющие воздействия требуется выдавать с точностью порядка десятков микросекунд, и компьютер при этом загружен в т.ч. и другими задачами.

Однако, разработка необходимого драйвера оказалась достаточно трудоёмкой, а отладка – сложной, как в техническом, так и в человеческом плане (оказалось, что человеку психологически тяжело работать со столь малыми интервалами времени). Это означает, что несмотря на возможность высокоточного программного управления, реализация необходимой функциональности на аппаратном уровне во многих случаях оправдана.

Хочется отдельно подчеркнуть, что одним из решающих факторов, обеспечивших успех этой работы, является открытость всего используемого программного обеспечения. Благодаря открытости и возможности вставить отладочный код в любое место, вплоть до точки, куда аппаратно передаётся управление при возникновении прерывания, всегда оказывалось возможным понять происходящие в системе низкоуровневые процессы, в частности ответить на вопрос "куда деваются микросекунды".

Литература

1. Смелянский Р.Л., Бахмуров А.Г. Применение метода полунатурного моделирования для отработки комплекса бортового оборудования летательного аппарата. Изд-во ТРТУ , 2002.

2. Руководящий технический материал авиационной техники РТМ 1495-75. Обмен информацией двуполярным кодом в оборудовании летательных аппаратов.

3. Техническое описание адаптера Arinc429-3. http://www.elcus.ru

4. IA-32 Intel Architecture Software Developers Manual http://developer.intel.com

Захаров В.А., Коннов И.В.

Об одном подходе к верификации асинхронных параметризованных систем

Введение

Одной из задач верификации программ является задача проверки свойств на моделях, параметризованных по числу однотипных процессов. Известно, что в общем случае задача неразрешима [2].

В работе [4] был предложен метод верификации параметризованных систем, описываемых сетевыми грамматиками, с использованием абстрактных инвариантов. Данный метод использует отношение предпорядка на размеченных системах переходов, обладающее свойством монотонности.

Отношения предпорядка, используемые для сравнения асинхронных систем, как правило, не обладают свойством монотонности. Как сказано в работе [3], на 2002 год способы применения метода [4] к асинхронным системам неизвестны.

В докладе предлагается отношение симуляции, обладающее свойствами монотонности и сокрытия применительно к операции асинхронной параллельной композиции. Данный вид симуляции позволяет применить методы верификации параметризованных моделей синхронных распределённых программ для асинхронного случая.

2. Основные определения

Размеченной системой переходов (LTS) назовём шестёрку $M = (S, S_0, R, \Sigma, L, A)$, в которой: S – конечное множество состояний модели, $S_0 \subseteq S$ – множество начальных состояний, $R \subseteq S \times A \times S$ – отношение перехода, Σ – конечное множество пропозициональных переменных модели программы, $L : S \to 2^{\Sigma}$ - функция оценки состояний переменными, E – множество действий. Значение переменной $p \in \Sigma$ истинно в состоянии $s \in S$ тогда и только тогда, когда $p \in L(s)$. Множество A содержит специальное действие τ, соответствующее неименованному внутреннему действию.

Для LTS можно определить операцию параллельной композиции. Пусть даны две такие LTS $M'=(S', S_0', R', \Sigma', L', A')$ и $M''=(S'', S_0'', R'', \Sigma'', L'', A'')$, что $A' \cap A'' = \{\tau\}$. Пусть также задана такая пара $\Gamma = (\Delta, co)$, что $\Delta \in A' \setminus \{\tau\}$ и $co: \Delta \to A'' \setminus \{\tau\}$, $co(\Delta) = \{a \in A'' \mid \exists b \in \Delta \cdot co(b) = a\}$. Пара

Γ называется синхронизатором. Множество Δ является множеством действий модели M', участвующих во взаимодействии с моделью M''. На каждое действие $a \in \Delta$ модель M'' осуществляет действие $co(a)$.

Размеченная система переходов $M=(S, S_0, R, \Sigma, L, A)$ является асинхронной параллельной композицией M' и M'' относительно Γ (обозначается $M = M' \|_\Gamma M''$), если выполняются следующие условия:

- $S = S' \times S''$.
- $S_0 = S_0' \times S_0''$.
- $\Sigma = \Sigma' \cup \Sigma''$.
- $L((s', s'')) = L'(s') \cup L''(s'')$.
- $A = A' \cup A'' \setminus (\Delta \cup co(\Delta))$.
- $((s', s''), a, (t', t'')) \in R$ для (s', s''), $(t', t'') \in S$, $a \in A$, если выполняется одно из условий:
  - $a \in A' \wedge a \notin \Delta \wedge (s', a, t') \in R' \wedge s'' = t''$.
  - $a \in A'' \wedge a \notin co(\Delta) \wedge (s'', a, t'') \in R'' \wedge s' = t'$.
  - $a = \tau \wedge \exists b \in \Delta \cdot ((s', b, t') \in R' \wedge (s'', co(b), t'') \in R'')$.

Пусть дана LTS $M=(S, S_0, R, \Sigma, L, A)$. LTS можно рассматривать как машину Мура, выделив явно множества O действий вывода и I действий ввода, положив при этом $A = \{\tau\} \cup 2^{I \cup O}$. Размеченная система переходов $M=(S, S_0, R, \Sigma, L, I, O)$ называется синхронной параллельной композицией $M'=(S', S_0', R', \Sigma', L', I', O')$ и $M''=(S'', S_0'', R'', \Sigma'', L'', I'', O'')$ (обозначается $M = M' \|_S M''$), если выполняются следующие условия (определение аналогично [4]):

- $S = S' \times S''$.
- $S_0 = S_0' \times S_0''$.
- $\Sigma = \Sigma' \cup \Sigma''$.
- $L((s', s'')) = L'(s') \cup L''(s'')$.
- $O = O' \cup O''$.
- $I = (I' \cup I'') \setminus (O' \cup O'')$.
- $((s', s''), a, (t', t'')) \in R$ для (s', s''), $(t', t'') \in S$, $a \in E \cup I$, если верно $(s', a', t') \in R'$, $(s'', a'', t'') \in R''$ для таких a', a'', что либо $a' = \tau$ и $a = a''$, либо $a'' = \tau$ и $a = a'$, либо $a' \cap (I'' \cup O'') = a'' \cap (I' \cup O')$ и $a = (a' \cup a'') \cap (I \cup O)$.

Для LTS $M=(S, S_0, R, \Sigma, L, A)$ назовём путём последовательность $\pi = s_1 \xrightarrow{a} s_2 \xrightarrow{b} \ldots \xrightarrow{z} s_n$, где $s_i \in S$, $(s_i, l, s_{i+1}) \in R$, $l \in \{a, b, \ldots, z\} \subseteq A$.

3. Отношение симуляции и его расширения

3.1. Строгая симуляция

Для LTS широко применяется отношение строгой симуляции [7]. Пусть даны две LTS $M'=(S', S_0', R', \Sigma', L', A')$ и $M''=(S'', S_0'', R'', \Sigma'', L'', A'')$. Отношение $H \subseteq S' \times S''$ называется отношением строгой симуляции, если для любой пары состояний $(s', s'') \in H$ выполняются условия:

$L'(s') = L''(s'')$

Для любых таких состояния $t' \in S'$ и действия $a \in A$, что $(s', a, t') \in R'$, найдётся такое состояние $t'' \in S''$, что $(s'', a, t'') \in R''$.

LTS M' и M'' находятся в отношении строгой симуляции ($M' \leq M''$), если найдётся отношение строгой симуляции H, что для любого состояния $s' \in S_0'$ найдётся такое состояние $s'' \in S_0''$, что $(s', s'') \in H$. Операции синхронной и асинхронной параллельной композиции обладают свойством монотонности относительно строгой симуляции:

Теорема 1. Для любых таких LTS M_1, M_2, M_1', M_2', что $M_1 \leq M_2$ и $M_1' \leq M_2'$, верно $M_1 \|_S M_1' \leq M_2 \|_S M_2'$ и $M_1 \|_\Gamma M_1' \leq M_2 \|_\Gamma M_2'$ (для любого Γ, удовлетворяющего одновременно M_1, M_1' и M_2, M_2').

На практике отношение симуляции оказывается слишком строгим для сравнения LTS, являющихся результатом асинхронной параллельной композиции нескольких других LTS. Например, при сравнении LTS $M_1 \|_{\Gamma 1} M_2 \|_{\Gamma 2} M_3$ и $M_1 \|_{\Gamma 1} M_2$ первая LTS содержит дополнительные внутренние переходы LTS M_3. В случае синхронной композиции для LTS $M_1 \|_S M_2 \|_S M_3$ и $M_1 \|_S M_2$ найдётся отношение симуляции при сокрытии переменных модели M_3.

3.2. Блочная симуляция

В работе [5] введено отношение блочной бисимуляции. В блочной бисимуляции, в отличие от строгой, рассматриваются не одиночные переходы, а последовательности внутренних действий, оканчивающиеся наблюдаемым действием.

В работе [5] введено отношение блочной бисимуляции, которое позволяет совершать одной LTS несколько ненаблюдаемых действий и одно наблюдаемое на несколько ненаблюдаемых (возможно другое количество) действий и одно наблюдаемое действие другой LTS.

Отношение бисимуляции является отношением эквивалентности на LTS. Более слабым отношением является

отношение блочной симуляции [1], определённое аналогично блочной бисимуляции.

Пусть V – множество значимых переменных, $MAXF(s)$ – множество таких конечных путей вида $\sigma = s \xrightarrow{\tau} s_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} s_n \xrightarrow{a} v$, что для любого i: $1 \le i \le n$ верно $s \cap V = s_i \cap V$ и либо $a \neq \tau$, либо $s \cap V \neq v \cap V$. В дальнейшем блок σ указанного вида будем обозначать $\sigma = s \to \bullet \xrightarrow{a} v$. $MAXI(s)$ – множество таких бесконечных путей σ из состояния s, что для любых $s, t \in \sigma$ выполняется $s \cap V = t \cap V$

Пусть даны две LTS $M'=(S', S_0', R', \Sigma', L', A')$ и $M''=(S'', S_0'', R'', \Sigma'', L'', A'')$. Отношением блочной симуляции относительно переменных V называется такое $H \subseteq S' \times S''$, что для $(s', s'') \in H$ выполняется:

1. $s' \cap V = s'' \cap V$.
2. Для любого пути $\sigma = s' \to \bullet \xrightarrow{a} v \in MAXF(s')$ найдётся такой путь $\delta = s'' \to \bullet \xrightarrow{a} w \in MAXF(s'')$, для которого справедливы условия:
   2.1. $(v, w) \in H$
   2.2. $(s, t) \in H$ для любых $t' \in \sigma$, $t'' \in \delta$ (монотонность).
3. Для любого $\sigma \in MAXI(s')$ найдется $\delta \in MAXI(s'')$.

Модель M' блочно симулирует модель M'' ($M' \le_V M''$), если существует такое отношение блочной симуляции H для любого начального состояния $s_0' \in S_0'$ найдется начальное состояние $s_0'' \in S_0''$, что $(s_0', s_0'') \in H$.

Отношение блочной симуляции между моделями означает, что поведение, наблюдаемое в M'', есть и в M'.

Теорема 2. Пусть $M' \le_V M''$, а s_0' и s_0'' – начальные состояния M' и M'' такие, что $(s_0', s_0'') \in H$. Тогда для любой формулы φ логики **ACTL*** без оператора O (Next_time), содержащей пропозициональные символы из V, выполняется $M'', s_0'' \models \varphi \Rightarrow M', s_0' \models \varphi$.

В некоторых случаях использование отношения блочной симуляции позволяет свести задачу проверки моделей с любым числом однотипных процессов к задаче проверки нескольких конечных моделей [1]. Однако отношение блочной симуляции не обладает свойствами монотонности и сокрытия действий, используемыми, например, в методах [4] и [6]. Для того, чтобы распространить эти методы на асинхронные системы, мы вводим новый тип симуляции – полублочную симуляцию.

3.3. Полублочная симуляция

Введём на отношении переходов LTS $M = (S, S_0, R, \Sigma, L, A)$ дополнительную функцию разбиения блоков p_V относительно множества переменных V. Функция разбиения $p_V : R \to \{ \mathbf{F}, \mathbf{T} \}$ должна обладать следующим свойством: каков бы ни был переход $(s, a, t) \in R$ если $L(s) \cap V \neq L(t) \cap V$ или $a \neq \tau$, то верно $p_V(s, a, t) = \mathbf{T}$. Это означает, что p_V обязательно разбивает переходы по внешним действиям и переходы, изменяющие наблюдаемые переменные. Другие переходы могут не разбиваться.

Рассмотрим блок $\sigma = s_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} s_n \xrightarrow{a} v \in MAXF(s_1)$. Подблоком пути σ назовём всякий такой путь $\delta = s_i \xrightarrow{\tau} s_{i+1} \xrightarrow{\tau} \ldots \xrightarrow{\tau} s_{i+m-1} \xrightarrow{a} s_{i+m}$, что δ является подпутём σ и для любого перехода $s_j \xrightarrow{\tau} s_{j+1} \in \delta$, где $i \le j \le i+m-2$, выполняется $p_V(s_j, \tau, s_{j+1}) = \mathbf{F}$, а для перехода $s_{j+m-1} \xrightarrow{a} s_{j+m} \in \delta$ выполняется $p_V(s_{j+m-1}, a, s_{i+m}) = \mathbf{T}$.

Через SB(σ) обозначим разбиение пути σ на подблоки $SB(\sigma) = B_1 B_2 \ldots B_k$. При этом предполагается, что последнее состояние каждого подблока совпадает с первым состоянием следующего блока.

Пусть даны две LTS $M' = (S', S_0', R', \Sigma', L', A')$ и $M'' = (S'', S_0'', R'', \Sigma'', L'', A'')$, множество наблюдаемых переменных V, разбиения p_V' для M', разбиения p_V'' для M''. Отношением полублочной симуляции относительно переменных V называется такое $H \subseteq S' \times S''$, что для любой пары $(s, t) \in H$ выполняется:

$L(s) \cap V = L(t) \cap V$.

Для любого блока $\sigma = s' \to \bullet \xrightarrow{a} v \in MAXF(s')$ найдётся такой путь $\delta = s'' \to \bullet \xrightarrow{a} w \in MAXF(s'')$, для которого справедливо соотношение:

$(v, w) \in H$.

Разбиения на подблоки путей $\sigma = SB(\sigma) = B_1 B_2 \ldots B_k$, $\delta = SB(\delta) = C_1 C_2 \ldots C_k$ таковы, что для любого i: $1 \le i \le k$, любого $s' \in B_i$, $t' \in C_i$ выполняется $(s', t') \in H$ (полублочная монотонность).

Для любого $\sigma \in MAXI(s)$ найдётся $MAXI(t)$.

Отношение полублочной симуляции обладает свойством монотонности для асинхронной параллельной композиции:

Теорема 3. Пусть заданы такие модели $M^i = (S^i, S^i_0, R^i, \Sigma^i, L^i, A^i)$, $i = \overline{1,4}$, что $\Sigma^1 \cap \Sigma^3 = \varnothing$, $\Sigma^2 \cap \Sigma^4 = \varnothing$, $A^1 = A^2$, $A^3 = A^4$. Пусть для некоторых множеств $\Sigma' \subseteq \Sigma^1 \cap \Sigma^2$, $\Sigma'' \subseteq \Sigma^3 \cap \Sigma^4$, разбиений $p^1_{\Sigma'}$, $p^2_{\Sigma'}$, $p^3_{\Sigma''}$, $p^4_{\Sigma''}$ верно $M_1 \le [p^1_{\Sigma''}, p^2_{\Sigma''}]M_2$, $M_3 \le [p^3_{\Sigma''}, p^4_{\Sigma''}]M_4$. Пусть задан такой синхронизатор $\Gamma = (\Delta, co)$, что $\Delta \subseteq A_1$, $co : \Delta \to A_3$.

Тогда $M_1 \|_\Gamma M_3 \le M_2 \|_\Gamma M_4$ для некоторых разбиений $p^{1,3}_{\Sigma' \cup \Sigma''}$, $p^{2,4}_{\Sigma' \cup \Sigma''}$.

4. Применение метода абстрактных инвариантов

В работе [4] предложен метод абстрактных инвариантов. Идея его такова. Семейство параметризованных моделей задаётся с помощью сетевой грамматики. Терминальным символам грамматики соответствуют прототипы процессов. Модели параметризованного семейства выводятся из начального символа сетевой грамматики.

Для каждого процесса-терминала T сетевой грамматики и заданного автомата-спецификации строится абстрактный процесс $h(T)$, причём $T \le h(T)$. Для каждого нетерминала A сетевой грамматики строится абстрактная система переходов $rep(A)$, получающаяся параллельной композицией абстракций терминалов и нетерминалов, стоящих в правой части правила вывода для A. Если для каждого правила вывода $A \to B \parallel C$ выполняется $A \ge B \parallel C$, то для каждой сети (модели) a, порождённой данной сетевой грамматикой выполняется $A \ge B \parallel C$. В этом случае достаточно проверить спецификацию на абстрактной сети $rep(A)$.

Например, для алгоритма обедающих философов параметризованное семейство можно описать следующей грамматикой:

$S \to A \|_2 r$

$A \to l \|_1 r \mid A \|_1 r$

Терминальные символы l и r описывают философа-левшу и философа-правшу соответственно. Левша сначала берёт левую вилку, потом - правую. Правша, наоборот, берёт сначала правую вилку, потом – левую.

В качестве $rep(A)$ по методу [4] нужно взять $h(l) \|_1 h(r)$. При этом должно выполняться $h(l) \|_1 h(r) \|_1 h(r) \le h(l) \|_1 h(r)$. Последнее отношение не выполняется, но если развернуть нетерминалы один раз, то выполняется $h(l) \|_1 h(r) \|_1 h(r) \|_1 h(r) \le h(l) \|_1 h(r) \|_1 h(r)$. Поэтому в качестве $rep(A)$ достаточно взять $h(l) \|_1 h(r) \|_1 h(r)$. Метод [4] успешно

применяется к задаче верификации параметризованного семейства обедающих философов, объединяемых асинхронной параллельной композицией.

В работе [4] описан пример применения метода к синхронному кольцевому алгоритму Дейкстры взаимного исключения. Для синхронной параллельной композиции в качестве отношения $\leq$ достаточно взять отношение симуляции. В случае асинхронной параллельной композиции отношения симуляции недостаточно. Однако использование отношения полублочной симуляции позволяет применить метод к асинхронному варианту.

5. Заключение

В работе предложено отношение симуляции, которое обладает свойствами, необходимыми для применения метода [4]. Метод удалось применить к асинхронному варианту кольцевого алгоритма Дейкстры и алгоритму обедающих философов. В дальнейшем мы планируем применить метод к более сложным моделям и исследовать вопросы эффективного вычисления полублочной симуляции.

6. Литература

1. И.В. Коннов, В.А. Захаров. О верификации параметризованных симметричных распределённых программ // Труды первой Всероссийской научной конференции "Методы и средства обработки информации" (1-3 октября 2003 г., Москва). –М.: Издательский отдел факультета ВМиК МГУ им. М.В. Ломоносова, 2003, с. 395-400.

2. K. Apt, D. Kozen. Limits for automatic verification of finite-state concurrent systems // Information Processing Letter, volume 15, pp. 307-309.

3. M. Calder, A. Miller. Five ways to use induction and symmetry in the verification of networks of processes by model-checking // In Proceedings of the Second Workshop on Automated Verification of Critical Systems, 2002.

4. E.M. Clarke, O. Grumberg, and S. Jha. Verifying parameterized networks using abstraction and regular languages // 6th International Conference on Concurrency Theory, 1995.

5. E. Emerson, K. Namjoshi. Reasoning about Rings // In Proceedings of the 22th ACM Symposium on Principles of Programming Languages. ACM Press, 1995, pp. 85-94.

6. O. Grumberg, D. Long. Model checking and modular verification // In CONCUR □91, LNCS 527. Springer Verlag, 1991.

7. R. Milner. An algebraic definition of simulation between programs // In Proc. of the 2nd International Joint Conference on Artificial Intelligence, pp. 481- 489, 1971.

Глазкова В. В.

Обзор современных методов моделирования и визуализации облаков в реальном времени [1]

1. Введение

Реалистичная визуализация облаков в реальном времени имеет важные приложения в компьютерных играх, мультипликации и интерактивном моделировании полета в тренажерах для пилотов самолетов и других летательных аппаратов. Под интерактивностью в данном случае подразумевается то, что траектория перемещения наблюдателя определяется действиями пользователя в процессе взаимодействия с системой. В решении задачи реалистичной визуализации облаков в реальном времени можно выделить две основные подзадачи: трехмерное *моделирование* облаков и *визуализация* полученных моделей в реальном времени.

Существует два подхода к *моделированию* облаков. Первый подход основан на моделирование физических процессов динамики жидкостей и газов. Методы, основанные на данном подходе, обеспечивают высокую точность моделирования, но являются дорогостоящими в вычислительном отношении. Для моделирования облаков *в реальном времени* используются эвристические методы, частично упрощающие физическое моделирование. Основные из этих методов - вокселное моделирование и моделирование на основе системы частиц.

Имеющиеся на настоящий момент методы визуализации облаков, моделируемых системой частиц, являются быстрыми (обеспечивают работу в реальном времени) и универсальными с точки зрения возможности адаптации на похожие погодные явления (дождь, снег, дымка, туман). В аспекте реалистичности основным недостатком таких методов является то, что они не рассматривают рассеяние света, обусловленное атмосферными частицами, а также тени на Землю. Методы визуализации облаков на основе вокселных моделей уступают выше упомянутым методам по скорости, но превосходят их по реалистичности - отображаются как тени на землю, так и лучи света через облака (благодаря учёту атмосферного рассеяния). В связи с этим в данном обзоре рассматриваются методы визуализации облаков,

[1] Работа поддерживается грантами РФФИ 03-01-00745, РФФИ 06-01-00691.

основанные как на вокселном моделировании (Miyazaki, Dobashi, Nishita [1]), так и на моделировании системой частиц (Harris[2,3] и Umenhoffer, Szirmay-Kalos [4]). В настоящей статье приводится анализ однопроходного метода [1] визуализации динамических облаков. Этот метод обладает лучшей производительностью в случае вокселного моделирования и позволяет визуализировать тени на землю и лучи света во всём пространстве моделирования (ввиду однопроходности). Вычисление цвета облаков принимает во внимание однократное рассеяние света, а многократное рассеяние учитывается с помощью постоянного по объёму облака коэффициента.

2. Два основных метода моделирования погодных явлений в реальном времени

2.1. Вокселное моделирование

Пространство моделирования разбивается на набор *одинаковых* трёхмерных ячеек – вокселей, заполняющих *всё* пространство моделирования (рисунок 1). Таким образом, области моделирования можно сопоставить трёхмерный массив. Каждый элемент массива в простейшем случае имеет значение 1, если соответствующая ячейка представляет собой область, занятую неким объектом, и 0 – в противном случае. Такой трёхмерный массив задаёт приближение объектов в области моделирования с точностью, определяемой размером воксела.

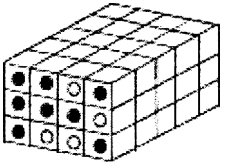


Рисунок 1. Вокселная модель

Типичные методы вокселной визуализации обрабатывают трёхмерный массив и формируют проекцию *каждого* его элемента на плоскость изображения.

Достоинства вокселной модели

- Возможность достаточно точно описывать объекты произвольной формы.
- Эволюция объекта моделируется путём задания некоторых простых правил перехода, в соответствие с которыми на каждом временном шаге изменяются значения логических переменных, закреплённых за каждым вокселом.

Недостаток вокселной модели - при повышении точности моделирования резко увеличиваются затраты памяти (например, объём

256*256*256 имеет небольшую разрешающую способность, но требует свыше 16 миллионов вокселей).

2.2. Моделирование на основе системы частиц

Будем рассматривать *систему частиц* как набор, состоящий из значительного числа простых примитивов, каждый из которых может быть задан единственной точкой в трёхмерном пространстве и небольшим количеством атрибутов: радиусом, цветом и текстурой. Параметры частиц изменяются динамически. Если параметры совокупности частиц скоординированы, то с помощью неё можно представить объект.

Одним из часто встречающихся представлений частиц при работе с системами частиц являются билборды (billboard) - прямоугольные пластины с наложенной текстурой, всегда расположенные параллельно плоскости изображения. Такие объекты обычно задаются при помощи положения своего центра и размера (центр билборда помещается в центр частицы).

2.3. Сравнение вокселного моделирования и моделирования на основе системы частиц

Воксслы заполняют всё пространство моделирования, тогда как частицы заполняют только объём объектов, причём объём объектов можно заполнить частицами разного размера. По этой причине с помощью системы частиц наиболее удобно моделировать явления, представленные множеством объектов, сильно разрозненных в пространстве (дождь, снег, туман), поскольку для этих явлений не требуется хранить информацию о состоянии пространства моделирования между объектами. Для моделирования облаков также удобно использовать систсму частиц, ввиду небольшого объёма необходимой для моделирования памяти. На вокселных же моделях удобно задавать правила перехода к следующему временному шагу для описания динамики процессов, происходящих в облаках, так как массив – упорядоченная структура данных, и все правила перехода могут быть выражены простыми логическими операциями, исходя из состояний вокселей вокруг данного.

3. Моделирование облаков и их визуализация в реальном времени

3.1. Метод визуализации статических облаков в реальном времени (Harris [2,3])

Поскольку облака состоят из частиц – капелек воды, Харрис (Harris) моделирует облака системой частиц. В случае представления облаков в виде набора частиц с текстурами прозрачности, контур и изображение облаков определяется перекрытием полупрозрачных частиц, что соответствует физической структуре настоящего облака. Такое представление облаков позволяет управлять освещением на уровне каждой частицы, которое можно рассчитывать на основе моделей различной сложности.

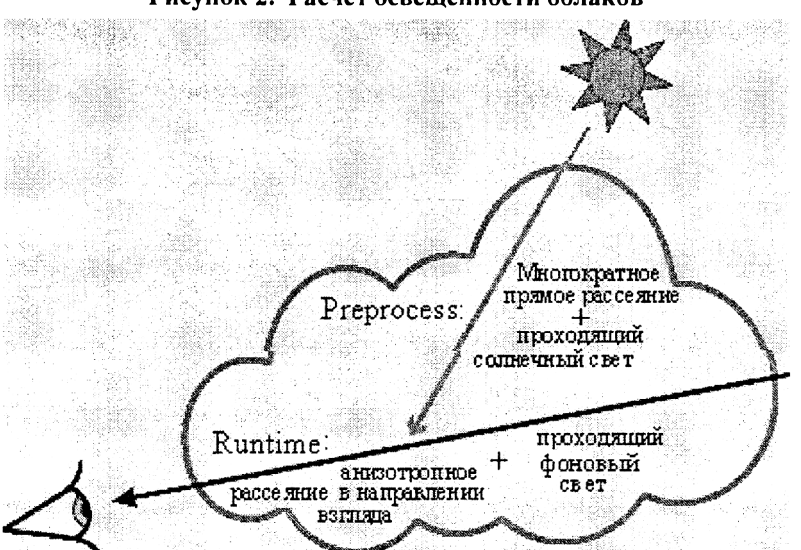
Особенности метода.

1. Облака моделируются системой частиц.
2. Предположение о статичности облаков позволяет:
   - Формировать облака заранее;
   - Производить расчёт освещённости частиц облаков один раз для всей сцены на этапе предобработки (это занимает в среднем несколько секунд);
   - В реальном времени визуализировать очень реалистичные сцены, содержащие облака.

Характеристика метода.

Алгоритм расчёта освещённости облаков имеет два этапа (рисунок 2).

Рисунок 2. Расчёт освещённости облаков



Preprocess: вычисляется интенсивность света, падающего на каждую частицу, учитывая многократное рассеяние в направлении света.
Runtime: учитывая анизотропное однократное рассеяние, вычисляется, какая часть падающего на частицу света рассеивается к точке наблюдения.
Для ускорения визуализации облаков применяется техника динамически генерируемых импостеров (imposters).

Освещённость облаков.

Цвет (или интенсивность света) каждой частицы в облаке зависит от степени ослабления света от внешних источников освещения и количества света, рассеянного к частице от других частиц. В этом приближении предполагается, что частицы рассеивают свет, главным образом, в направлении движения волны. Следовательно, в затенение, производимое любой отдельно взятой частицей, будут вносить вклад частицы, находящиеся между источником света и рассматриваемой частицей. Эта особенность называется многократным прямым рассеянием.

1 этап: preprocess - вычисление интенсивности света, падающего на каждую частицу, учитывая многократное рассеяние

Свет, падающий на частицу P по направлению *l*, равен сумме прямого света по направлению *l*, который не поглощён промежуточными частицами (проходящий свет), и света, рассеянного к частице P другими частицами:

$$I(P,\omega) = I_0 \cdot e^{-\int_0^{D_P} \tau(t)dt} + \int_0^{D_P} g(s,\omega) e^{-\int_s^{D_P} \tau(t)dt} ds,$$

I_0 – интенсивность света, падающего на границу облака

D_P – глубина частицы P в облаке по направлению света

g(s,ω) – интенсивность света, рассеянного в направлении ω в точке s

τ(t) – оптическая глубина (мера похождения света через полупрозрачную среду)

Приближаем интегралы суммами Римана

$$I_P = I_0 \cdot \prod_{j=1}^{N} e^{-\tau_j} + \sum_{j=1}^{N} g_j \prod_{k=j+1}^{N} e^{-\tau_k}.$$

N – число частиц облака, расположенных на луче из частицы P к источнику света.

$g_k = a_k \cdot \tau_k \cdot p(l,-l) \cdot I_k / 4\pi$ – интенсивность света, рассеянного k-ой частицей. l – направление света.

a_k – коэффициент рассеяния (процент затухания света из-за рассеяния k частицами вдоль пути света).

τ_k – оптическая глубина. $p(\theta)$ –фазовая функция.

Преобразуем формулу для вычисления интенсивности в виде рекуррентного соотношения, которое можно эффективно реализовать на аппаратном обеспечении машинной графики

$$I_k = \begin{cases} g_{k-1} + T_{k-1} I_{k-1}, & 2 \le k \le N \\ I_0, & k = 1 \end{cases} \qquad (1)$$

$T_k = e^{-\tau_k}$ – прозрачность частицы P_k.

Это соотношение означает, что интенсивность света, падающего на любую частицу P_k. равна сумме интенсивности света, рассеянного к P_k из P_{k-1}, и интенсивности света, проходящего через P_{k-1} (определяемой прозрачностью T_{k-1}).

Реализация с использованием функции смешивания OpenGL.

Имеем соотношение для функции смешивания OpenGl:

$$c_{result} = f_{src} \cdot c_{src} + f_{dest} \cdot c_{dest} \qquad (2)$$

Если положить $C_{result} = I_k$, $f_{src} = 1$, $C_{src} = g_{k-1}$, $f_{dest} = T_{k-1}$, $C_{dest} = I_{k-1}$, то соотношения (1) и (2) будут эквивалентны, если содержимое буфера кадра перед смешиванием есть I_0. Функция смешивания устанавливается так, что $f_{src}=1$ и $f_{dest}=T_{k-1}$, т.е. *glBlendFunc(GL_ONE, GL_SRS_ONE_MINUS_ALPHA)*, так как непрозрачность (равная 1-прозрачность) сохранена в α-компоненте цвета. Такой режим смешения «заставляет» каждую частицу затенять охваченные ею пиксели в буфере кадра пропорционально ослаблению света частицей и освещать их пропорционально рассеянию света данной частицей.

Итак, используя рекуррентное соотношение (1), можно рассчитать цвет частицы. Затем, используя этот вычисленный цвет, частица отображается в буфер кадра в виде отпечатка с Гауссовой текстурой (это количество света, который частица рассеивает на другие частицы в пределах некоторого пространственного угла). Используя команду смешения OpenGL и считывание только что записанной информации из буфера кадра, в последнем, в результате, сохраняется количество света, достигающего частиц на некоторой глубине в пределах облака.

2 этап: визуализация - вычисление интенсивности света, достигающего точки наблюдения

Как только цвета всех частиц рассчитаны и сохранены, буфер кадра (то есть позиция камеры) устанавливается в точку наблюдения. В этой точке можно определить, какая часть света, падающего на каждую частицу, достигает точки наблюдения. Это второе приближение - облака проявляют анизотропное рассеяние, которое можно наблюдать в виде серебристых переливов (silver lining) и изменяющихся оттенков цвета.
Рекуррентное соотношение, приближающее однократное анизотропное рассеяние света в направлении точки наблюдения, имеет вид

$$E_k = S_k + T_k \cdot E_{k-1},\ 1 \le k \le N$$

Интенсивность E_k света, выходящего из любой частицы P_k, равна сумме интенсивности света, падающего на неё и не поглощённого ($T_k \cdot E_{k-1}$), и интенсивности S_k рассеянного света. Интенсивность света, рассеянного частицей P_k в направлении точки наблюдения, рассчитывается по формуле $S_k = a_k \cdot \tau_k \cdot p(\omega,-l) \cdot I_k / 4\pi$,

ω – направление взгляда.
I_k - интенсивность света, падающего на частицу P_K, вычисленная на первом этапе.
$p(\omega,-l)$ - фазовая функция, определяющая, какой процент света частицы достигает точки наблюдения (это функция угла между направлением взгляда и направлением света).
Реализация с использованием функции смешивания OpenGL:
glBlendFunc (GL_ONE, GL_SRS_ONE_MINUS_ALPHA).
На этом шаге в позицию каждой частицы опять отображается Гауссова текстура с определённым цветом, и это изображение, наконец, визуализируется на экран.

Эффективная визуализация облаков при помощи импостеров

Чтобы добиться высоких скоростей передачи кадров для сложных облачных сцен, можно либо уменьшить количество визуализируемых пикселей, либо понизить стоимость визуализации облака с помощью многократных кадров. Динамически генерируемые импостеры позволяют сделать и то, и другое. Как показано на рисунке 3, импостер заменяет объект в сцене полупрозрачным многоугольником с отображенной на нём текстурой с изображением объекта.

Изображение импостера – это визуализация объектов из точки наблюдения V; эта визуализация допустима (в пределах некоторого допуска ошибки) для точек наблюдения, близких к V. Импостеры, используемые для подходящих точек наблюдения, дают очень близкое приближение непосредственной визуализации объектов. Импостер допустим (без ошибки) для *точки* наблюдения, из которой его изображение было сгенерировано, независимо от изменений в

направлении рассмотрения. Импостеры могут быть предвычислены для объектов из множества точек наблюдения, требуя значительного объема памяти, или они могут быть сгенерированы только в те моменты, когда этонеобходимо. Последняя методика называется динамически генерируемые импостеры [7]. Использование импостеров позволяет увеличить скорость визуализации в 3-5 раз.

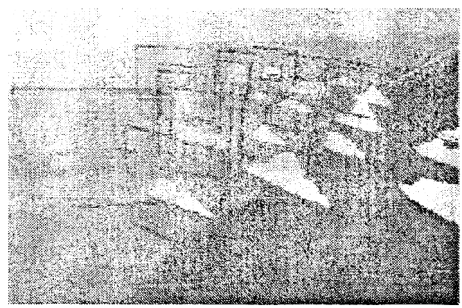


Рисунок 3. Использование импостеров [2]

Объекты в облаках

Для реалистичной визуализации облачных сцен необходимо предусмотреть возможность прохождения объектов через облака. В этом случае использование импостеров вызывает проблемы, потому что они двумерны. Объекты, которые проходят через импостеры, выглядят так, как будто они проходят через изображения, плавающие в пространстве, а не через пушистые объемные облака. Чтобы решить эту проблему нужно обнаружить, когда объекты проходят в пределах объема облака, и разбить импостер, представляющий это облако, на несколько слоёв. Как только один объект оказывается в некотором облаке, то это облако визуализируется как два слоя: один для части облака, которая находится приблизительно позади объекта относительно точки наблюдения, и один для части, которая находится приблизительно перед объектом. Если два объекта лежат в пределах облака, необходимо использовать три слоя и так далее. В результате получается реалистичное изображение облака, содержащего объекты. Разделение импостера обеспечивает дополнительное преимущество над прямой визуализацией облаков, содержащих объекты. Когда частицы облака визуализируются непосредственно в буфер кадра, многоугольники, используемые для их визуализации, могут пересечь геометрию близлежащих объектов. Эти пересечения могут быть

причиной артефактов. Разделение импостера избавляет от подобных артефактов.

Достоинства метода Харриса.

Метод Харриса является наиболее общим и универсальным методом представления облаков и легко модифицируется для любых требований, таких как полёт через облака, возможность рассматривать облака под любым углом, полёт над облачным слоем. Для моделирования и визуализации облаков метод Харриса представляется наиболее перспективным по следующим причинам:

1. Реальное время - благодаря технике динамически генерируемых импостеров;
2. Реалистичность - учитывается как многократное рассеяние в направлении света (без учёта многократного рассеяния облака выглядят нереалистично тёмными), так и анизотропное однократное рассеяние в направлении взгляда;
3. Метод не требует хранения информации о состоянии пространства моделирования между объектами, как следствие, универсальность распространения на другие погодные явления (дождь, снег, туман).

3.2. Метод визуализации динамических облаков в реальном времени, используя импостеры глубины (Umenhoffer, Szirmay-Kalos)

Для сокращения вычислительной нагрузки при визуализации, в данном методе облака строятся из более мелких похожих блоков частиц. Это приближение допустимо, так как большинство природных явлений проявляет самоподобность. Блоки визуализируются один раз для текущего направления взгляда и направления света. Один блок включает частицы, которые близки друг к другу в пространстве. Далее роль отдельных частиц принимается этими блоками. Чтобы исключить считывание из цветового буфера, во время светового прохода вычисляется ослабление в дискретизованных выборках глубины. Это позволяет быстро вычислять самозатенение и многократное прямое рассеяние.

Перед визуализацией для данного направления взгляда, изображение частиц блока определяется из этого направления. Затем это изображение, называемое импостером глубины, используется вместо отдельных частиц. Пиксел импостера глубины сохраняет информацию о частицах, которые проецируются на этот пиксел (в частности, их полную непрозрачность, их минимальную (переднюю) и максимальную (заднюю) глубины). Во время визуализации пиксел

импостера действует как так называемая супер-частица, которая представляет все те частицы блока, которые на нее проецируются. Полная непрозрачность используется в передаче освещенности, в то время как глубины принимаются во внимание, чтобы устранить артефакты, вызванные объектами, находящимися в объёме облака.

В случае замены частиц блоками, вычислительная нагрузка значительно сокращается. Так для системы из N частиц, достаточно построить блок из b частиц и повторить его N/b раз. Иерархический подход требует только b+N/b вычислений вместо N вычислений неиерархического метода.

3.2.1. Использование импостеров глубины во время светового прохода

Визуализация облаков состоит из светового прохода, учитывающего многократное рассеяние света в облаке, и прохода результирующего сбора.

Во время светового прохода блоки частиц классифицируются в группы по их расстояниям от источника света и промежуточное изображение сохраняется в текстурах при данных выборочных расстояниях. Эти текстуры называются слайсами (slices). Первая текстура будет отображать совокупную непрозрачность первой группы блоков частиц, вторая – будет показывать непрозрачность первой и второй групп блоков частиц и так далее. Требуемое число выборок глубины зависит от числа частиц и формы облака. Для приближению сферической формы облака и относительно небольшого числа частиц (где перекрытие не преобладает) даже четыре глубины может быть достаточно. Рисунок 4 показывает эту методику с пятью слайсами глубины.

3.2.2. Использование импостеров глубины во время результирующего прохода

Во время результирующей визуализации импостеры глубины блоков, полученные для направления взгляда, сортируются и визуализируются один за другим от заднего к переднему. Член от внутреннего рассеяния получается из выборочных текстур слайсов, между которыми содержится пиксел блока (рисунок 4).

Накопленная непрозрачность слайсов может быть использована, чтобы определить освещенность пикселов этих слайсов и, наконец, исходящую освещенность частицы между слайсами. Зная положение частицы, определяется, какие два слайса её ограничивают. Линейной интерполяцией между значениями, читаемыми из этих двух текстур, можно аппроксимировать ослабление цвета источника света. Чтобы получить лучшую аппроксимацию многократного рассеяния, принимается во внимание освещенность тех пикселов обоих

ограничивающих слайсов, для которых фазовая функция больше некоторого порогового значения.

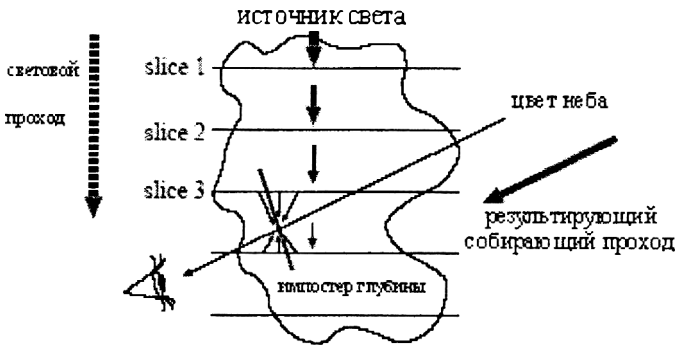


Рисунок 4. Результирующий проход для блока [4].

3.2.3. Объекты в облаках

Главная проблема с системами частиц билбордного типа состоит в том, что билборды являются плоскостями, и, таким образом, они не имеют протяженности вдоль одного измерения. Это может вызвать артефакты, когда билборды пересекают объекты, т.е. пересечение плоскости билборда и объекта становится ясно заметно (рисунок 5). Суть проблемы состоит в том, что билборд обесцвечивает те объекты, которые расположены позади него в соответствии с его прозрачностью, как если бы объекты были полностью позади сферы частицы. Однако те объекты, которые находятся впереди плоскости билборда, не обесцвечиваются вообще, так что прозрачность меняется резко в местах пересечения объекта с билбордом. Эта проблема решается с использование расширения блока частиц, т.е. интервала блока в направлении глубины, которая сохраняется в элементах текстуры импостера.

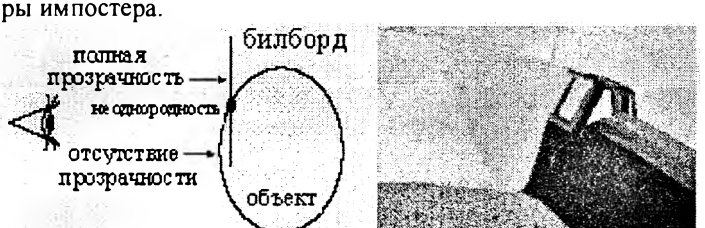


Рисунок 5. Артефакты, вызванные объектами в объёме, визуализируемом билбордами. В тех местах, где плоскость билборда пересекает объект, видна неоднородность объета [4].

Чтобы решить эту проблему, сначала визуализируются все объекты сцены и буфер глубины сохраняется в текстуре. Затем блоки частиц визуализируются один за другим, включая тест глубины, но

отключая запись глубины. При визуализации блока частиц вычисляется интервал величины перемещений луча внутри блока, добавляя значение глубины центра блока к переднему и заднему значениям глубины импостера глубины. Этот интервал сравнивается со значением, сохраняющим глубину видимого объекта. Если значение глубины объекта находится вне интервала, то объект или полностью виден или полностью закрыт сферой частицы. Это достигается, благодаря использованию аппаратных средств z-буфера кадра и α-смешивания. Однако. когда интервал блока ограничивает глубину объекта, только часть объемного блока закрывает объект. В этом случае непрозрачность блока частиц масштабируется в соответствии с интервалом глубины блока частиц и с относительным расстоянием между передней глубиной блока частиц и объектом. Это масштабирование соответствует предположению, что плотность однородна в пределах блока.

3.3. Однопроходный метод Miyazaki, Dobashi, Nishita для визуализации динамических облаков

Как видно из вышеизложенного рассмотрения, имеющиеся на настоящий момент методы визуализации облаков, моделируемых системой частиц, не рассматривают рассеяние света, обусловленное атмосферными частицами, а также тени на Землю. Методы визуализации облаков на основе вокселных моделей превосходят методы на основе системы частиц по реалистичности - отображаются как тени на землю, так и лучи света через облака (благодаря учёту атмосферного рассеяния). Однако, так как визуализация облаков и лучей света через облака во всех двупроходных методах – различные процессы, двупроходные методы вокселного моделирования для визуализации лучей света через облака (метод сферических оболочек [5] и метод виртуальных плоскостей [8]) позволяют визуализировать лучи света только ниже основания облаков. Чтобы устранить этот недостаток двупроходных методов и увеличить скорость визуализации при вокселном моделировании, R. Miyazaki, Y. Dobashi, T. Nishita был предложен однопроходный метод визуализации динамических облаков с учётом атмосферного рассеяния [1], позволяющий визуализировать лучи света через облака во всём пространстве моделирования. Метод также вычисляет цвет облаков и тени на землю. При вычислении цвета облаков принимается во внимание однократное рассеяние света, а многократное рассеяние учитывается с помощью постоянного по объёму облака коэффициента. Реализация метода [1] предполагает максимальное использование аппаратных возможностей машинной графики, поскольку бóльшая часть процесса визуализации должна производиться операциями с текстурами. Этот метод обладает лучшей производительностью в случае вокселного моделирования.

В методе [1] для визуализации облаков необходимо рассматривать ослабление света в двух направлениях: в направлении солнца (от солнца к облаку) и в направлении взгляда (от облака к точке наблюдения). Для вычисления этих ослаблений в одном проходе пространство моделирования разбивается на слои параллельными плоскостями. Эти параллельные плоскости будем называть "виртуальными плоскостями" (virtual planes). При двухпроходном методе виртуальные плоскости устанавливались перпендикулярно лучу рассмотрения, так как к началу второго прохода ослабления в направлении солнца были уже вычислены. При однопроходном методе виртуальные плоскости нельзя размещать перпендикулярно лучу рассмотрения, так как в случае, когда солнечные лучи также перпендикулярны лучу рассмотрения, они будут «скользить» вдоль этих плоскостей, не позволяя учитывать ослабления солнечного света в направлении солнца. При однопроходном методе виртуальные плоскости задаются своей нормалью, которая занимает промежуточное положение между вектором направления солнечных лучей и вектором направления взгляда. За счёт выбора направления виртуальных плоскостей удаётся уменьшить различия в интервалах выборки и фазовых углах для различных лучей, исходящих из точки наблюдения. Это уменьшение различий позволяет сократить вычислительные расходы. Однопроходный метод [1] не нуждается во временных объёмных данных для накопительной функции ослабления в направлении солнца и может визуализировать изображения быстрее, чем двупроходный алгоритм. Этот метод использует объемную текстуру для облака и двумерную текстуру для атмосферы, поскольку плотность атмосферы зависит от высоты над землёй. Интенсивность рассеянного света эффективно вычисляется, используя обе текстуры.

Метод моделирования, используемый в методе [1] для получения объёмной плотности облаков, учитывает физические процессы динамики жидкостей и газов, происходящие в облаках [6].

3.4. Сравнение методов визуализации облаков

Авторы	Метод	Универсальность	Реалистичность				Производительность
			Рассеяние		Тени на земле	Лучи света	
			Однократное	Многократное			
Miyazaki Dobashi Nishita	Вокселы	нет	есть	Задаётся постоянным по	есть	есть	Близкая к real-time

				объёму коэф-фици-ентом			
Harris	Час-тицы	есть (дождь снег туман)	есть (run-time)	есть (prepro-cess)	нет	нет	real-time
Umen-hoffer	Час-тицы	есть	есть	есть	нет	нет	real-time

Таблица 3. Сравнение методов визуализации облаков

4. Заключение

Для моделирования облаков *в реальном времени* используются эвристические методы, частично упрощающие физическое моделирование. Основные из этих методов - вокселное моделирование и моделирование на основе системы частиц. Достоинства методов моделирования на основе системы частиц, следующие: быстрота (обеспечивают работу в реальном времени), универсальность (с точки зрения возможности адаптации на похожие погодные явления - дождь, снег, дымка, туман), реалистичность (учитывается многократное рассеяние). Вокселные же методы хорошо подходят для моделирования динамики процессов, происходящих в облаках. Методы визуализации облаков на основе вокселных моделей уступают методам на основе системы частиц по скорости, но превосходят их по реалистичности - отображаются как тени на землю, так и лучи света через облака (благодаря учёту атмосферного рассеяния).

Имеющиеся на настоящий момент методы визуализации облаков, моделируемых системой частиц, являются *только двупроходными* и *недостаточно реалистичными* (не рассматривают рассеяние света, обусловленное атмосферными частицами, а также тени на Землю) [2-4]. В качестве продолжения работы по *повышению реалистичности* метода Харриса для визуализации облаков планируется провести адаптацию, программную реализацию и апробацию *однопроходного* метода [1] для визуализации динамических облаков, моделируемых *системой частиц*. При этом для получения объёмной плотности облаков, планируется использовать метод Харриса моделирования динамики облаков на основе системы частиц [3].

Литература

1. R. Miyazaki, Y.Dobashi, T. Nishita. "A Fast Rendering Method of Clouds using Shadow-View Slices", *Proc. CGIM 2004*, 93-98, 2004

2. M. J. Harris, A. Lastra. "Real-Time Cloud Rendering", *Computer Graphics Forum (Eurographics 2001 Proceedings)*, 20(3):76-84, 2001

3. M. J. Harris. "Real-Time Cloud Simulation and Rendering", University of North Carolina, *Ph.D. Dissertation*, 2003

4. T. Umenhoffer, L. Szirmay-Kalos. "Real-Time Rendering of Cloudy Natural Phenomena with Hierarchical Depth Impostors", *EUROGRAPHICS Short Papers*, 2005

5. Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, T. Nishita. "A Simple, Efficient Method for Realistic Animation of Clouds", *Proc. SIGGRAPH2000*, 19-28, 2000

6. R. Miyazaki, Y. Dobashi, T. Nishita. "Simulation of Cumuliform Clouds Based on Computational Fluid Dynamics", *Proc. EUROGRAPHICS 2002*, 405-410, 2002

7. G. Schaufler. "Dynamically Generated Impostors", *Proceedings of GI Workshop "Modeling – Virtual Worlds - Distributed Graphics"*, infix Verlag, 129-135, 1995

8. Y. Dobashi, Y. Yamamot, T. Nishita. "Interactive Rendering Method for Displaying Shafts of Light", *Proc. Pacific Graphics 2000*, 31-37, 2000

Брусенцов Н.П., Владимирова Ю.С.,
Рамиль Альварес Х.

Троичный логико-алгебраический и арифметический процессор

Существенные достоинства троичного арифметического процессора были выявлены К.Шенноном в статье о симметричных кодах чисел [1], на которую почему-то практически отсутствуют ссылки. Затем, двадцать лет спустя, Д.Кнут охарактеризовал уравновешенную троичную систему счисления как "быть может, самую изящную", предрекая ей серьезные применения, когда "флип-флоп" заменится на "флип-флэп-флоп" [2, с.218]. Надо сказать, что за десять лет до этого прогноза уже существовала и была выпущена небольшой серией наша машина "Сетунь", на практике подтвердившая преимущества симметричной троичности. А теперь установлено, что еще в 1840 г. Томас Фаулер сконструировал и построил работающий в симметричном троичном коде механический вычислитель [3]. Странно, что все еще сохраняется представление, будто главное достоинство троичного кода в том, что он самый экономный, поскольку 3 - ближайшее к основанию натуральных логарифмов целое число. Ведь экономия относительно двоичного кода теоретически менее 5% и едва ли реализуема технически. Вместе с тем то, что троичный код является простейшим, позволяющим осуществить симметричную (уравновешенную, сбалансированную) систему счисления, оказывается исключительно важным как в принципе, так и практически. Двоичная система счисления, не обеспечивающая непосредственного представления числа со знаком, удовлетворительного округления и варьирования длины операндов, по сравнению с симметричной троичной явно неполноценна. Пример "Сетуни" наглядно показал, что троично-симметричный арифметический процессор естественней, проще, дешевле и эффективней двоичных.

Но в несравнимо большей степени неадекватность двоичности проявилась при исследовании логико-алгебраических основ информатики [4]. Установлено, что так называемые парадоксы материальной импликации, устранить которые безуспешно пытались виднейшие логики, обусловлены противоестественным "законом исключенного третьего" и составляют неотъемлемую особенность двухзначности. Полноценное содержательное следование, представленное в аристотелевой силлогистике общеутвердительной посылкой "Всякое x есть y" ("Из x с необходимостью следует y"), в двухзначной логике непредставимо и вырождено в утратившую

смысловую взаимосвязь терминов импликацию, что и обусловило бессодержательность этой логики. Основоположники современной математической логики Д.Гильберт и В.Аккерман отклонились от Аристотеля в истолковании общеутвердительной посылки, оправдывая это "потребностями математических применений логики, где класть в основу аристотелево понимание было бы нецелесообразно" [5, с.79]. На самом деле их отклонение было предопределено двухзначностью, введенной в логику античными стоиками, тогда как подлинная силлогистика трехзначна и содержательна. В условиях двухзначности, наряду с парадоксами материальной импликации, частные посылки не подчинены общим, т.е. отвергается силлогизм подчинения: "Всякое x есть y" не влечет "Некоторые x есть y", и поэтому вопреки здравому смыслу отклонены вполне очевидные модусы в 3-й и 4-й фигурах. Понятно, почему двухзначная "классическая" (а в сущности ущербная) логика не используется при решении реальных проблем и не прижилась в школе, хотя предмета, призванного систематически развивать умы, там явно не хватает. Впрочем, теперь пробел устранен - логика покоряет школу, обретя высокотехнологичный облик компьютерной информатики. Но компьютеры двоичные, логика все та же двухзначная, последствия компьютеризации пагубны [6], причем не только в системе образования.

Реальная возможность выхода из сложившейся катастрофической ситуации - внедрить адекватный троичный компьютер с благоразумной содержательной логикой и безупречной уравновешенной арифметикой. Теоретически все еще далеко не всеми осознаваемые достоинства симметричной троичности, будучи овеществленными в "Сетуни", с завидной легкостью осваивались и успешно применялись каждым, кому довелось работать с этой машиной. По-своему "правильно" поняли суть дела даже бюрократы, добившиеся прекращения ее выпуска и дальнейших разработок троичной цифровой техники. Сегодня троичный компьютер, укомплектованный полноценным логико-алгебраическим процессором, станет действенным средством воссоздания аристотелевой содержательной логики, первоосновы троичной диалектической информатики.

Ключевой элемент воссоздаваемой адекватной логики - отношение содержательного, необходимого следования. Именно к нему стремились устранители парадоксов материальной импликации и изобретатели трехзначных импликаций, также не достигшие цели, поскольку действовали формально, не вникая в содержание того, к чему стремятся. А материальная импликация естественно и просто трансформируется в содержательное следование, если принять во внимание, что в ее характеристической функции

$$\rightarrow(x, y) = xy \vee x'y \vee x'y'$$

статус члена $x'y$ не равносилен статусу членов xy и $x'y'$ [7, 8]. Класс вещей, удовлетворяющих отношению $x \Rightarrow y$ ("из x следует y", "y содержится в x"), необходимо включаст xy-вещи и $x'y'$-вещи, не может включать xy'-вещей, а включенность $x'y$-вещей несущественна, привходяща - могут быть, могут не быть. Обозначив значение истинности привходящего буквой σ, $0<\sigma<1$, можно охарактеризовать необходимое содержательное следование $x \Rightarrow y$ и соответствующий нечеткий класс трехзначной функцией

$$\Rightarrow(x, y) = xy \vee \sigma x'y \vee x'y'$$

В троичном компьютере эта функция и охарактеризованное ею отношение кодируются значением упорядоченной четверки тритов (ДК-шкалой) +-0+, тогда как материальной импликации $x \rightarrow y$ соответствует +-++. Произвольное n-терминное отношение кодируется 2^n-тритной шкалой, позволяющей отобразить и всякое отношение меньшей арности. Но содержательной логике принадлежат только трехзначные отношения, удовлетворяющие диалектическому принципу сосуществования противоположностей [9]. Над ДК-шкалами определены потритные операции конъюнкции, дизъюнкции и инверсии, а также отображения данной шкалы в шкалу большей арности и извлечения следующих из нее шкал меньшей арности. Этот набор операций, представляющих собой обобщение булевой и теоретико-множественной алгебр, оказался достаточным для исчерпывающей алгебраизации и компьютеризации силлогистики [10].

Арифметический сопроцессор наследует двухстековую (стек операндов и стек процедур) архитектуру "Сетуни 70", модифицированную соответственно требованиям структурированного программирования Э.Дейкстры и пополненную командой exit выхода из подпрограммы. Посредством этой команды обеспечена структурируемость всех неструктурирующихся традиционными средствами алгоритмов [11, с.82-86].

Литература

1. Shannon C.E. A symmetrical notation for numbers. // The American Mathematical Monthly, v. 57, n 2, Feb. 1950, pp. 90 - 93

2. Кнут Д. Искусство программирования для ЭВМ. Получисленные алгоритмы. Т.2. м.: Мир, 1977.

3. The ternary calculating machine of Thomas Fowler. / M.Glusker, D.M.Hogan, P.Vass. // IEEE Annals, v. 27, n. 3, July-Sept. 2005, pp. 4-22.

4. Брусенцов Н.П. Неадекватность двоичной информатики. // Современные информационные технологии и ИТ-образование». Сб. докладов. – М.: МАКС Пресс, 2005. С. 501-503.

5. Гильберт Д., Аккерман В. Основы теоретической логики. – М: ИЛ, 1947.

6. Компьютеры и обучение. / Н.П.Брусенцов, Ю.С.Владимирова, Х.Рамиль Альварес // Вестник Моск. ун-та. Сер. Педагогическое образование. 2005, №1. С. 103 - 105.

7. Брусенцов Н.П. О сложности и запутанности проблемы логического следования. // Современная логика: проблемы теории, истории и применения в науке. Материалы VIII общероссийской научной конференции. 24-26 июня 2004 СПб, 2004. С. 231-233.

8. Брусенцов Н.П., Владимирова Ю.С. Булевы уравнения и логический вывод. // Программные системы и инструменты № 5. Под ред. Л.Н. Королева.– М.: Изд. отд. ф-та ВМиК МГУ, 2005. С. 10-12.

9. Брусенцов Н.П. Интеллект и диалектическая триада. // Искусственный интеллект, 2'2002. – Донецк, 2002. С. 53-57.

10. Брусенцов Н.П. Реанимация аристотелевой силлогистики. // Реставрация логики. – М.: Фонд «Новое тысячелетие» – 2005. С. 140-145.

11. Брусенцов Н.П. Микрокомпьютеры. – М.: Наука 1985.

Раздел IV.

Сетевая обработка

Гурьев Д.Е., Демьянов П.Ю., Лызлов В.Е., Миронов Н.Ю., Харин В.А., Чихичин Д.А.

Устройств сопряжения мультиплексного канала обмена (MIL-STD-1553) и их программное обеспечение.

В этой работе рассматриваются цели, задачи, достигнутые результаты и ближайшие перспективы проекта по разработке аппаратного и программного обеспечения устройств сопряжения мультиплексного канала обмена бортовых сетей.

Протокол мультиплексного канала обмена и его особенности. Протокол «Интерфейс магистральный последовательный системы электронных модулей», известный также как ГОСТ Р 52070-2003 [1], MIL-STD-1553B [2], «манчестерский канал», применяется для организации бортовых сетей летательных аппаратов различного назначения, а также для бортовых сетей кораблей и судов и сетей управления технологическими процессами.

Опыт применения протокола насчитывает более 20 лет. По современным меркам, учитывающим такие требования, как передачу данных мультимедиа в реальном времени, протокол считается медленным (скорость передачи – 1Mbps), и обсуждается замена этого протокола более скоростными [3]. Однако для решения традиционных задач управления протокол обеспечивает требуемую степень надежности и отказ от него в обозримом будущем не прогнозируется.

Протокол MIL-STD-1553B характеризуется несколькими существенными особенностями:

- повышенная помехоустойчивость, что обеспечивается специальными конструкциями кабелей и гальванических развязок и сравнительно невысокой скоростью передачи,
- повышенная надежность, что обеспечивается как минимум одной резервной линией передачи данных и автоматическим переключением между линиями;
- предсказуемость в соответствии с требованиями «жесткого» реального времени, что обеспечивается наличием специального

абонента – «контроллера шины» (КШ), управляющего передачей данных между всем абонентами в соответствии с заранее заданным планом («кадром»);
- наличие еще 2х типов абонентов: «оконечное устройство» (ОУ) – обычный управляемый абонент, – и «монитор шины» (МТ), «подслушивающий» пересылки в целях диагностики и отладки).

Аппаратное обеспечение. Устройства сопряжения MIL-STD-1553B давно производятся несколькими компаниями США и других зарубежных стран. В России к моменту начала этого проекта не выпускалось ни одного устройства сопряжения, полностью соответствующего стандарту.

Наш коллектив поставил себе цель создать универсальное многофункциональное устройство сопряжения, которое полностью соответствовало бы требованиям стандарта и обладало бы следующими функциями и характеристиками:
- поддержка всех типов абонентов (КШ, ОУ, МТ) в одном устройстве,
- возможность работы как в составе вычислительной машины (включающей программируемый управляющий вычислитель (УВ)), так и в составе устройства без УВ и программного управления,
- возможность взаимодействия с целевым оборудованием при помощи шин разных типов (8- и 16-битные, с квитированием и без)
- возможность применения как в целевых системах, так и в макетных стендах и наземных комплексах отладки.

За образец был взята хорошо зарекомендовавшая себя серия микросхем компании ILC DDC (в частности, БИС BU-61580), и была поставлена задача независимо создать функциональный аналог.

Процесс разработки следовал традиционной для таких проектов схеме:
- разработка и отладка функциональной логики (ФЛ) при помощи логического моделирования,
- отладка ФЛ в составе устройства с перепрограммируемой логической БИС,
- проектирование и выпуск собственного кристалла.

Для отладки ФЛ была построена «симуляционая лаборатория», включающая модели аппаратного окружения ФЛ и позволяющая создавать и отслеживать события в модели манчестерского канала, создавать и анализировать образы сообщений в памяти ФЛ, проверять функционирование ФЛ в условиях всевозможных сюжетов взаимодействия как со стороны манчестерского канала, так и со стороны системной магистрали УВ.

В процессе отладки ФЛ были проведены исследования особенностей работы с различными типами отечественных и зарубежных приемопередатчиков.

В процессе реализации проекта было создано несколько устройств сопряжения, в том числе:

- ячейка МВ11.01 на основе БИС BU-61580; разработка этого устройства позволила накопить опыт работы с протоколом, создать системное программное обеспечение, а само устройство служило «эталоном» при последующей разработке и тестировании,
- ячейки МВ11.04 и МВ26.04 со сменными приемопередатчиками, на которых было выполнена отладка ФЛ,
- тестер протокола на основе ячейки МВ11.04,
- БИС 1879ВА1Т
- ячейка МВ26.14 на основе БИС 1879ВА1Т,
- ячейка МВ26.17 – многоабонентское устройство для имитации до 31 абонента сети в наземных комплексах отладки,
- и другие.

Все эти устройства, за исключением ячейки МВ11.01 и тестера протокола построены на основе ядра ФЛ, которое мы называем МСП (машина связная последовательной шины данных), и представляют собой разные версии или специальные модификации этого ядра ФЛ.

Ядро ФЛ МСП состоит из следующих основных компонентов:

- шифратор (кодировщик) информации последовательной шины данных (ПШД);
- сдвоенный дешифратор информации ПШД;
- полную многопротокольную логику, обеспечивающую режимы работы «контроллера шины» (КШ), «оконечного устройства» (ОУ), «монитора» (МТ);
- логику управления доступом к разделяемому с УВ ОЗУ;
- логику прерывания управляющего вычислителя;
- логику обеспечения передачи информации между управляющим вычислителем и МСП по шинам адреса и данных (логика «интерфейса с УВ»);
- защелки адреса и двунаправленные шинные формирователи для непосредственного подключения связной машины к шине УВ.

Логика интерфейса с УВ допускает множество различных конфигураций, необходимое для сведения к абсолютному минимуму количество вспомогательной внешней логики требуемой для сопряжения МСП с шинами 8-ми, 16-ти и 32-х разрядных процессоров. В дополнение имеется свойство, облегчающее сопряжение с процессорами, которые могут обмениваться информацией по системной магистрали исключительно без ожидания готовности от внешнего

устройства – режим обмена без ожидания. Наконец, МСП поддерживает надежный интерфейс с внешним ОЗУ. Это уменьшает долю производительности УВ, которая требуется для обеспечения доступа к внутреннему ОЗУ устройства. Существует возможность взаимодействия с УВ в режиме прямого доступа к памяти.

Элементами высокоуровневого интерфейса с УВ являются сущности канального уровня: сообщения, кадры, журналы приема-передачи («стеки команд»), представленные объектами в памяти внутреннего или внешнего ОЗУ устройства сопряжения.

Тестирование на соответствие протоколу MIL-STD-1553B проводилось при помощи устройства «тестер протокола», которое обеспечивает пословное управление передачей данных по ПШД с возможностью внесения различных ошибок, в том числе ошибок типа «инверсия бита», «неверный временной интервал», «ошибка бифазного кодирования», а также пословной записи ответов тестируемого устройства для последующего анализа. Для проведения тестирования были разработаны программы, реализующие стандартизованные тест-планы в соответствии с [4-6]. ФЛ МСП успешно прошла тестирования и аттестацию.

БИС 1879ВА1Т прошла полный цикл испытаний на устойчивость к внешним воздействиям. Детальная информация представлена в утвержденных технических условиях на БИС.

Программное и тестовое обеспечение. Параллельно с разработкой и отладкой собственно аппаратуры и для обеспечения этой разработки был создан комплекс программного и тестового [7] обеспечения, позволяющий проводить

- тестирование БИС и ячеек,
- разработку и отладку прикладных программ,
- стендовое макетирование бортовых сетей.

Программное обеспечение, поставляемое с устройствами, состоит из комплекта драйверов, библиотеки времени выполнения, обеспечивающей программное управление устройствами, и прикладного ПО с графическим пользовательским интерфейсом, позволяющим интерактивно конфигурировать устройства и тестировать передачу данных в бортовой сети или ее макете.

Библиотека времени выполнения. Основой системного ПО является библиотека времени выполнения («RTL2»), которая обеспечивает:

- доступ к управляющим регистрам и встроенному ОЗУ устройства,

- конфигурирование устройства для работы в каждом из режимов КШ, ОУ или МТ, включая конструирование кадров, а также управление подрежимами и дополнительными функциями,
- задание пользовательской подпрограммы обработки прерываний от устройства,
- предоставление единого процедурного интерфейса (API) ко всем устройствам линейки, скрывая их технические различия,
- сокрытие значительного числа других технических деталей,
- дополнительный уровень абстракции, приближенный к терминам протокола, в том числе – API для работы с объектами «сообщение», «кадр», «стек», размещенными в ОЗУ устройства,
- автоматизированное распределение памяти в ОЗУ устройства,
- контроль параллельного доступа к разделяемым объектам в ОЗУ устройства со стороны протокольной логики устройства и управляющего вычислителя,
- согласование темпа передачи данных из и в устройство (путем искусственного введения холостых циклов шины УВ) и другие действия, обусловленные аппаратными особенностями устройств,
- одновременную работу с любым числом устройств,
- и многое другое.

В состав библиотеки также входит специальный модуль для управления устройством «тестер протокола» для исполнения тестов соответствия протоколу MIL-STD-1553B.

При разработке библиотеки предъявлялись и были выполнены следующие технологические требования:

- возможность переноса в различные ОС и среды исполнения, в том числе такие, в которых отсутствует полноценная ОС и стандартные библиотеки;
- поддержка всех устройств линейки и постепенное накопление «знаний» об особенностях их работы и применения в едином комплекте исходных кодов;
- функциональная расширяемость и единый стиль построения API при функциональных расширениях.

Язык реализации – C. Предусмотрены специальные «прослоечные» модули для обеспечения независимости от ОС, компилятора и стандартной библиотеки C. При проектировании API произведен тщательный выбор независимых и свободно сочетаемых примитивных операций. Доступ к объектам в ОЗУ устройства осуществляется при помощи дескрипторов, между типами которых установлено отношение наследования; имеются полиморфные функции API. Библиотека в эксплуатации с 2000 года.

Драйверы. Сложность устройства и управления им, а также возможное отсутствие ОС (следовательно, и драйверов) на перспективных целевых платформах, привели к решению сосредоточить основную функциональную нагрузку в библиотеке времени выполнения и максимально разгрузить драйвер устройства. В текущих реализациях функции драйвера ограничены:

- обеспечением и контролем монопольноого доступа к устройству со стороны приложений;
- картированием управляющих регистров и встроенного ОЗУ устройства в адресное пространство приложения;
- при возникновении прерываний от устройства – при необходимости – передачей управления подпрограмме обработки прерываний, определенной приложением (и исполняющейся в адресном пространстве приложения).

Разработаны драйверы для ОС MS Windows 98, ОС MS Windows 2000 и ОС MS Windows 2000 с расширением реального времени RTX [8]. В ближайших планах – разработка драйверов для ОС Linux: для стандартного ядра и ядра с расширением реального времени RTAI [9].

Прикладное программное обеспечение. Прикладное программное обеспечение, поставляемое вместе с устройствами, по природе вещей является «нецелевым». Его задача – помочь разработчику целевой системы убедиться в работоспособности отдельных устройств-абонентов и сети (или ее макета) в целом. В настоящее время с устройством поставляются 3 интерактивные графические программы, обеспечивающие управление устройством в режимах КШ, ОУ и МТ соответственно, и выполняющиеся на платформе MS Windows. Каждое из приложений позволяет сконфигурировать устройство, задать тестовые данные для передачи, получать в реальном времени информацию об ошибках передачи данных, приложение для режима МТ – сохранять трассу передачи данных для последующего анализа. По историческим причинам эти приложения разрабатывались в разное время независимыми разработчиками.

На основе опыта использования этих программ было осознано, что прикладное ПО, поставляемое с устройствами, должно эффективнее способствовать разработки целевых систем, быть точнее нацеленным на задачи, решаемые разработчикам целевых систем, быть теснее интегрированным как внешне, с точки зрения пользователя (единообразные интерфейсные решения для управления аналогичными сущностями, меньшее число отдельных приложений), так и внутренне (повторное использование функционального кода, форматов данных и

самих данных). Кроме этого, необходимы версии интерактивных программ и для других ОС, в первую очередь – для ОС Linux.

Для конкретизации этих пожеланий был проведен анализ типичных областей применения интерактивных программ, сформирован перечень типовых задач и разработана типовая архитектура таких приложений.

Областями применения интерактивных программ являются стенды разработчиков целевых систем и наземные комплексы отладки. Для целей разработки важно иметь возможность быстро смакетировать будущую бортовую сеть и проверить ее работоспособность, убедиться в работоспособности своего устройства в ней. К сожалению, после этого макет сети приходится реализовывать заново в своем оборудовании или ПО уже без помощи интерактивных программ. Более эффективным был бы процесс разработки, при котором части макета можно было бы постепенно переносить в собственное оборудование или ПО.

На более глубоких стадиях разработки необходимо создавать для целевого устройства среду, имитирующую работу других абонентов сети, в том числе и содержание передаваемых сообщений.

Для диагностики и отладки сетевых отказов необходимы развитые средства сбора и анализа трасс сетевых взаимодействий.

Резюмируя сказанное, задачи интерактивного ПО должны быть следующими:

- быстрое и эффективное макетирование бортовых сетей:
  - графическое проектирование структуры сети,
  - интерактивное управление конфигурациями нескольких устройств из одного приложения,
  - возможность сохранения конфигураций с управлением версиями,
  - поддержка устройства МВ26.17, специально предназначенного для имитации нескольких абонентов сети;
- возможность статического задания тестовых передаваемых данных,
- возможность задания расчетного правила для тестовых передаваемых данных, с учетом полученных данных (фактически – модели поведения абонента),
- возможность экспорта в том или ином виде конфигурации устройств для последующего встраивания в целевое аппаратное или программное обеспечение,
- возможность сбора и анализа трасс передачи данных, включая отбор по сложным условиям при записи и в постобработке,
- возможность воспроизведения записанных трасс.

В основу типовой архитектуры прикладного ПО положен принцип деления на уровни:

- уровень взаимодействия с устройствами представлен библиотекой времени выполнения,
- уровень данных обеспечивает сохранение, чтение, объединение/разбиение и другие операции над данными; данных в этих приложениях достаточно много (конфигурации, трассы, тестовые данные для передачи и др.) и они достаточно сложно организованы; в качестве основного формата данных предложен XML,
- функциональный уровень, к которому относится код, реализующий назначение приложения,
- уровень пользовательского интерфейса.

При этом уровни реализуются по возможности независимо друг от друга, а компоненты и решения каждого уровня должны быть унифицированы и использоваться повторно в разных приложениях.

Отдельного внимания заслуживает проблема создания приложений для разных платформ. Идеальным было бы решение, позволяющее скомпилировать один и тот же комплект исходных кодов под разные платформы. Функциональный уровень практически не зависит от платформы, уровень данных также можно реализовать как независимый при аккуратном выборе вспомогательных библиотек (в частности, для разбора и генерации данных в формате XML), в библиотеке времени выполнения проблемы мобильности решены. Наиболее существенную проблему создает уровень пользовательского интерфейса, но и здесь возможны приемлемые решения. В настоящее время производится изучение нескольких платформно-независимых библиотек графических компонентов с целью выбора в качестве внутрипроектного стандарта разработки, в частности рассматриваются Trolltech QT [10], Borland CLX [11], GTK+ [12] и FLTK [13].

Изложенные цели и принципы будут положены в основу разработки новых и модернизации имеющихся интерактивных прикладных программ управления устройствами сопряжения мультиплексного канала обмена.

Направления дальнейшего развития проекта. В ближайшей перспективе планируются разработки по следующим направлениям:

- специализированные варианты ФЛ МСП для целевых систем,
- системы-на-кристалле с использованием ФЛ МСП в качестве встраиваемого функционального блока,
- специализированные ячейки МСП и их ПО для применения в наземных комплексах отладки,
- тестово-аттестационный стенд для проверки соответствия протоколу ГОСТ Р 52070-2003 устройств сопряжения мультиплексного канала обмена других разработчиков (см. подробнее в [7]).

Заключение. Рассмотрена реализация функциональной логики устройств сопряжения мультиплексного канала обмена и линейка устройств на основе этой функциональной логики, в том числе БИС 1879ВА1Т, вкратце рассмотрена технология разработки, отмечены основные результаты и направления дальнейшего развития проекта. Рассмотрены основные компоненты программного обеспечения устройств сопряжения, функции, технологические особенности и принципы разработки этих программных компонентов. Предложена архитектура для интерактивных прикладных программ управления устройствами сопряжения.

Литература

1. ГОСТ Р 52070-2003. Интерфейс магистральный последовательный системы электронных модулей. Общие требования.
2. MIL-STD-1553B NOTICE 4. DIGITAL TIME DIVISION COMMAND/RESPONSE MULTIPLEX DATE BUS. DOD, 1996.
3. Уилф Салливан (Wilf Sullivan). Что заменит MIL-STD-1553 в роли сетевой магистрали военных систем следующего поколения?//М.: Мир компьютерной автоматизации, № 4, 1999.
4. ГОСТ Р 51739-2001. Интерфейс магистральный последовательный системы электронных модулей. Тестирование опытных образцов интерфейсного модуля в режиме контроллера шины. Общие требования к методам контроля
5. ГОСТ Р 51765-2001. Интерфейс магистральный последовательный системы электронных модулей. Тестирование опытных образцов интерфейсного модуля в режиме оконечного устройства. Общие требования к методам контроля
6. ГОСТ Р 52073-2003. Интерфейс магистральный последовательный системы электронных модулей. Тестирование интерфейсных модулей, функционирующих в режиме монитора шины. Общие требования к методам контроля
7. Гурьев Д.Е., Миронов Н.Ю., Харин В.А., Чихичин Д.Л. Тестовое обеспечение связных машин мультиплексного канала обмена. Наст. сборник.
8. Ardence RTX Is The Real-Time Performance Solution For Microsoft Windows. http://www.ardence.com/embedded/products.aspx?ID=70
9. RTAI. http://www.rtai.org/
10. Qt Overview. http://www.trolltech.com/products/qt/index.html
11. Why Develop Applications with Borland Kylix 3? Borland Software Corporation, 2002. http://www.borland.com/resources/en/pdf/white_papers/why_develop_applications_with_borland_kylix3.pdf

12.GTK+ - The GIMP Toolkit. http://www.gtk.org/
13.Fast Light Toolkit. http://www.fltk.org/

Гурьев Д.Е., Миронов Н.Ю., Харин В.А., Чихичин Д.А.

Тестовое обеспечение связных машин мультиплексного канала обмена.

Эта работа посвящена проблемам тестирования универсальных связных машин (МСУ) мультиплексного канала обмена (МКО) [1], выполненного согласно требованиям ГОСТ Р 52070-2003 [2] (MIL-STD-1553B [3]), а также различных устройств сопряжения с МКО (адаптеров), построенных на основе МСУ и используемых в наземных комплексах отладки бортовых систем управления. Представлены основные результаты, а также обсуждаются некоторые планы на ближайшую перспективу.

Тестовое обеспечение проекта использовалось для решения следующих задач:

- функциональной отладки и проверки работоспособности разрабатываемой функциональной логики МСУ;
- в качестве контрольных тестов при изготовлении микросхем;
- для проверки соответствия протоколу.

По исполнению и способу использования тестовое обеспечение делится на функциональные тесты и тесты протокола. Функциональные тесты предназначены для отладки функциональной логики и производственного контроля. Тесты протокола предназначены для проверки и аттестации на соответствие протоколу ГОСТ Р 52070-2003, а для отладки и производственного контроля используются лишь выборочно. Для отладки функциональной логики была создана «симуляционная лаборатория», которая явилась средой исполнения функциональных тестов. Для исполнения тестов протокола было разработано специальное устройство «тестер протокола» и управляющее системное и прикладное программное обеспечение.

«Симуляционная лаборатория». Для функциональной отладки и проверки работоспособности разрабатывасмой функциональной логики МСУ манчестерского канала (впоследствии – кристалла МСУ МКО) в целом и его отдельных узлов на этапе проектирования применялась методика функционально-логического тестирования как ее виртуальной (идеальной) модели, так, впоследствии, для разработки кристалла, и модели, полученной путем синтеза с учетом библиотек компонентов, поставляемых фирмой-

изготовителем. Эта методика подразумевает создание так называемой «симуляционной лаборатории» с использованием средств языка Verilog. Данная лаборатория включает в себя сам тестируемый модуль, а также ряд дополнительных вспомогательных средств, обеспечивающих все необходимые входные воздействия на исследуемый объект, а также автоматический контроль над его поведением. Вспомогательные средства включают в себя следующие функциональные узлы:

- Кодер манчестерского кода с возможностью впрыскивания ошибки, обеспечивающий симуляцию информационных сигналов на линии передачи данных и позволяющий создать любые, в том числе и ошибочные, последовательности битов и сообщений.
- Аппаратный интерфейс, обеспечивающий программирование устройства со стороны системной магистрали.
- Вспомогательную логику, обеспечивающую возможность объединения выходных сигналов устройства и сигналов кодера на линии передачи данных
- Агент инициализации, предназначенный для автоматического программирования устройства и задания программы работы кодера.
- Агент памяти, предназначенный для автоматического контроля адреса и данных на внутренних шинах ОЗУ на соответствие их данным файлов шаблонов.
- Агент прерываний, предназначенный для отслеживания любых прерываний, выставляемых устройством на протяжении всей трассы и автоматического определения соответствия их ожидаемым значениям битов регистра состояния прерываний.
- Набор специальных функций, повышающих эффективность и скорость разработки тестов, читабельность их кода, а также удобство их использования и быстроту отладки.
- Формирователь базы векторов, предназначенной для дальнейшей оценки работоспособности устройства при различных условиях (задержках) с помощью эмулятора тестера на основании SDF-файла, а также оценки идентичности идеальной и синтезированной моделей устройства.

Функциональные тесты. Функционально-логическое моделирование предполагает имитацию по возможности всех ситуаций реальной жизни. Качество выполнения данного этапа работы должно гарантировать максимально возможное покрытие проверки всех алгоритмических ветвей исследуемого устройства. Должно быть гарантировано соответствие требованиям стандарта MIL-STD-1553B, а

также спецификации устройства. С целью достижения максимума покрытия разработан документ «согласованный план верификации модели», в котором перечислены все функции устройства для различных режимов работы. Все эти функции были проверены специальными тестами. В то же время разработка полного функционального теста – практически не разрешимая задача, и функциональные тесты ограничиваются кругом наиболее часто встречающихся сюжетов, а также основных конфликтных ситуаций. Контроль полноты тестирования выполнялся автоматически средствами САПР, в качестве меры полноты использовался процент покрытия «константных неисправностей». Показатели полноты составили 90% в среднем по устройству или от 45% до 95% по различным его блокам и были признаны приемлемыми. В состав функциональных тестов были также включены избранные тесты на соответствие протоколу.

Функциональные тесты использовались для отладки функциональной логики МСУ (в среде «симуляционной лаборатории»), а также для контроля годности при производстве БИС 1879ВА1Т.

Тесты протокола. Тестирование на соответствие протоколу MIL-STD-1553 должно проводиться отдельно для каждого типа (режима) устройства: контроллера шины, оконечного устройства и монитора шины в соответствии со стандартизованными тест-планами:

- Контроллер шины – ГОСТ Р 51739-2001 «Тестирование опытных образцов интерфейсного модуля в режиме контроллера шины. Общие требования к методам контроля» [4].
- Оконечное устройство – ГОСТ Р 51765-2001 «Тестирование опытных образцов интерфейсного модуля в режиме оконечного устройства. Общие требования к методам контроля» [5].
- Монитор шины – ГОСТ Р 52073-2003 «Тестирование опытных образцов интерфейсного модуля в режиме монитора шины. Общие требования к методам контроля» [6].

Тест-планы контроллера шины и оконечного устройства делится на две почти независимые друг от друга части: это электрические тесты и тесты логики протокольного автомата. Те и другие тесты требуют разной сложности аппаратуры, и программного обеспечения (ПО) для них. Если в электрических тестах требуется в основном решение оператора, то для протокольных тестов необходим программный код обработки полученных данных и принятия решения о корректности реакции. Для электрических тестов требуется тестер с поддержкой некоторых специальных требований к аппаратуре и наличие дополнительного оборудования (осциллографы, вольтметры и т.п.). Для

логических тестов тестер должен обладать способностью формировать сообщения с ошибками разных типов.

Тест-планы для разных типов устройств качественно различаются по необходимости доступа к внутреннему состоянию устройства в процессе тестирования. Тестировать устройство стороннего производителя предусматривается только для оконечного устройства, для верификации которого не требуется иметь информацию о его состоянии, а заключение о его соответствии требованиям стандарта можно сделать на основе его интерактивной реакции по сети МКО.

В отличие от оконечного устройства, специфика работы контроллера и монитора шины требует знание внутренних параметров (таких как состояние регистров, памяти, возникновение прерываний и т.п.) для сопоставления этих данных с сущностями (состояниями реакции) описанными в тест-планах.

Для тестирования соответствия МСУ (и адаптеров на их основе) требованиям стандарта было разработано устройство «тестер протокола» и программное обеспечение, реализующее сценарии тестирования, специфицированные в указанных выше документах.

Тестер протокола. В ЗАО НТЦ «Модуль» разработан и успешно прошел испытания тестер на базе ячейки МВ11.04 [1] для проверки на соответствие протоколу. Тестер позволяет воспроизводить любую последовательность сообщений, работать одновременно за контроллер шины или за передающий и за принимающий оконечные устройства, а также за несколько разных устройств одновременно. В нем также предусмотрен гибкий мониторинг шины MIL-STD-1553B. Тестер позволяет вносить все виды протокольных ошибок требуемых по тест-планам, таких как ошибка бифазного кодирования, уменьшение или увеличение длины слова (кол-ва разрядов), ошибка четности и др. Функционально тестер делится на две части: генератор словных последовательностей (ГСП) и словный монитор. ГСП позволяет описывать пословно выдаваемые в канал сообщения, с внесением всех видов протокольных ошибок (или без оных), а так же управлять включением словного монитора, формировать паузы, «зацикливать» кадры и т.п. Тестер может автоматически переключаться из режима ГСП в режим словного монитора и обратно в соответствии с заданным сценарием тестрования. Для аттестации тестера было создано специализированное ПО, реализующее выборочные тесты из всех трех тест-планов.

Для управления тестером библиотека времени выполнения RTL2 [1] была дополнена специальным модулем gsw.c, позволившим скрыть аппаратные особенности реализации тестера, предоставить программисту высокоуровневый интерфейс для управления устройством в терминах тест-планов. Интерфейс gsw.c обеспечивает

программирование тестов протокола на уровне взаимодействующих абонентов, с определением роли (или ролей) исполняемой(ых) тестером, а проверка корректности ответной реакции абонентов автоматизирована. При этом соблюдена преемственность стиля интерфейса библиотеки RTL2.

Программная реализация тестов протокола. Прикладное программное обеспечение представляет собой пошаговую реализацию стандартных тест-планов. Основная задача решаемая ПО – проверка соответствия устройств разработанных для МКО в ЗАО НТЦ «Модуль» параметрам и критериям предъявляемым в тест-планах. Особенности задачи, а также необходимость непосредственной отладки устройств при прохождении теста, обязало ПО строго привязаться к конкретным устройствам (модулям).

Были полностью реализованы требования по тестированию во всех трех режимах работы – контроллер шины, оконечное устройство, монитор шины. Кроме того, был реализован электрический тест 5.2.2 (стабильность формы входного сигнала при переходе через нулевой уровень) [5], который возможно было реализовать без использования дополнительного оборудования.

Дополнительные тестовые программы. В целях полноты тестирования локальных протоколов модуля реализующего функциональную логику МСУ МКО, было разработано дополнительное тестовое ПО:

- Тестирование коллизий совместного доступа к разделяемому ОЗУ.
- Тестирование системы прерываний.
- Выборочные тесты конфигурирования неразрешенными параметрами.

С помощью реализованного тестового обеспечения удалось отладить логику универсального высокоинтеллектуального устройства, являющуюся основой для серии модулей МВ11.XX и МВ26.XX, которая в дальнейшем была воплощена в кристалле (БИС 1879ВА1Т), выпущенном летом 2005 года ЗАО НТЦ «Модуль». Данное тестовое обеспечение используется также для проверки и отладки новых функциональных плат разрабатываемых на том же предприятии, а так же для контроля годности при изготовлении БИС 1879ВА1Т.

Дальнейшее развитие тестового обеспечения. В процессе использования ПО тестов протокола, а также из возникающих новых задач, была осмысленна необходимость дальнейшего

совершенствования тестирующих программ в сторону создания независимых аттестационных комплексов, для полного или частичного тестирования устройств MIL-STD-1553B сторонних разработчиков. Для этих целей необходима доработка ПО тест-плана оконечного устройства – отвязкой его от конкретных интерфейсов, конкретных устройств. С реализацией тест-планов контроллера и монитора канала для независимых устройств дело обстоит намного сложнее, необходима разработка механизмов и интерфейсов сопряжения для получения с их помощью информации о внутреннем состоянии устройства. Получение это информации затрудняется неоднозначностью соответствия состояния в понятиях стандарта (к примеру - недостоверный ответный сегмент и т.п.) и состояния реального физического устройства. Затрудненность такого рода, происходит в основном из-за примитивности существующих и разрабатываемых устройств. К числу отрицательных моментов против реализации независимого тестирования контроллера и монитора шины, как наиболее важный момент относится отсутствие данных о потребности таких тестов от сторонних разработчиков.

Исходя из этого, можно с уверенностью сказать - перспективно развитие только ПО тестирования оконечного устройства. В настоящее время в ЗАО НТЦ «Модуль» планируется создать автономный тестово-аттестационный стенд, при этом сделать программу тестирования ОУ расширяемой, т.е. с возможностью введения пользователем собственного сценария теста в дополнение к стандратному. Для обеспечения легкого переноса на другие операционные системы, планируется воспользоваться стандартным методом построения многоплатформных систем – разделение ПО на части платформно-зависимые и платформно-независимые. Проектируемое ПО будет состоять из следующих частей:

- Тестирование коллизий совместного доступа к разделяемому ОЗУ.
- Драйвер ядра ОС (платформно-зависимый компонент).
- Библиотека времени выполнения RTL, написанная на стандартном языке «С» без использования спецефических для ОС библиотек (платформно-независимый компонент).
- Логика управления, на стандартном языке C++ (платформно-независимый компонент).
- Графический интерфейс пользователя с использованием многоплатформных графических библиотек типа GTK+ [7], Qt [8] или FLTK [9] (платформно-независимый компонент).

Заключение. Рассмотрены основные компоненты тестового обеспечения универсальной связной машины [1], основные принципы их разработки и применения для отладки, тестирования и аттестации

устройств. С применением этого тестового обеспечения запущена в производство БИС 1879ВА1Т и ряд других устройств. В дальнейшем планируется разработка тестово-аттестационного стенда для проверки устройств сопряжения других производителей.

Литература

1. Гурьев Д.Е., Демьянов П.Ю., Лызлов В.Е., Миронов Н.Ю., Харин В.А., Чихичин Д.А. Устройства сопряжения мультиплексного канала обмена (MIL-STD-1553) и их программное обеспечение. Наст. сборник.
2. ГОСТ Р 52070-2003. Интерфейс магистральный последовательный системы электронных модулей. Общие требования.
3. MIL-STD-1553B NOTICE 4. DIGITAL TIME DIVISION COMMAND/RESPONSE MULTIPLEX DATE BUS. DOD, 1996.
4. ГОСТ Р 51739-2001. Интерфейс магистральный последовательный системы электронных модулей. Тестирование опытных образцов интерфейсного модуля в режиме контроллера шины. Общие требования к методам контроля
5. ГОСТ Р 51765-2001. Интерфейс магистральный последовательный системы электронных модулей. Тестирование опытных образцов интерфейсного модуля в режиме оконечного устройства. Общие требования к методам контроля
6. ГОСТ Р 52073-2003. Интерфейс магистральный последовательный системы электронных модулей. Тестирование интерфейсных модулей, функционирующих в режиме монитора шины. Общие требования к методам контроля
7. GTK+ - The GIMP Toolkit. http://www.gtk.org/
8. Qt Overview. http://www.trolltech.com/products/qt/index.html
9. Fast Light Toolkit. http://www.fltk.org/

Афанасьев М.К.

Алгоритм динамического распределения данных на основании анализа запросов узлов

В современных глобальных децентрализованных хранилищах данных информация оказывается распределенной среди достаточно большого числа узлов сети. Примерами являются Интернет, система CFS для безопасного совместного использования файлов [1], децентрализованное хранилище данных FARSITE [2], peer-to-peer системы обмена файлами [3] и т.д. В процессе функционирования подобного рода хранилищ данных возникает проблема неоптимальности расположения данных по узлам сети. Т.е. блоки данных, к которым узлы обращаются наиболее часто, могут оказываться значительно удаленными от них, что приводит к неоправданному увеличению задержек при передачи данных между узлами сети. В связи с этим возникает необходимость разработки подходов для оптимизации расположения информации по узлам гетерогенных распределенных систем с большим количеством отдельных хранилищ данных..

Так актуальность данной проблемы для сетей Интернет привела к созданию в 1998 году компании Akamai (Cambridge, MIT), которая посредством 14000 промежуточных серверов расположенных в 65 странах, позволяет значительно сократить время передачи данных. Идея разработанного Akamai алгоритма – кэширование данных на серверах Akamai (промежуточных серверах) с целью минимизации задержек при передаче информации конечным пользователям. Например, рассмотрим Интернет-сайт содержание которого формируется на основании запросов к базе данных. При обращении пользователя к сайту происходит его переадресация к ближайшему серверу Akamai, который пытается построит страницу на основании информации, которая хранится у него "в кэш". В случае, если информации в кэш не достаточно, сервер Akamai запрашивает информацию у Интернет-сайта и пополняет ею свой кэш. Таким образом, сервера Akamai играют роль "региональных кэшей". Тем не менее, алгоритм Akamai подвержен проблемам: 1) в связи с частым обновлением данных консистентность между базой данных и информацией в кэш на сервере Akamai быстро нарушается. Для решения этой проблемы, Akamai хранит в кэш только те данные, доступ к которым критичен для скорости передачи. 2) после большого числа запросов информация из базы данных целиком оказывается в кэшах сервера Akamai (а поскольку сервер Akamai не имеет доступа к структуре базы данных, информация в кэш хранится в виде записей, т.е. хранится заведомо менее эффективно чем на исходном сервере) 3) часто информация в кэш соседних серверов

Akamai дублируется, что приводит к неэффективному использованию файлового пространства.

Таким образом, возникает задача разработки интеллектуальных сетевых агентов, способных, координируя деятельность между собой, автоматически перемещать блоки данных в сети ближе к тем узлам, которые наиболее часто обращаются или будут обращаться к этим данным. Открывающиеся при использовании таких алгоритмов возможности позволяют значительно ускорить процесс обмена информацией, увеличить безопасность и надежность хранения данных в сети, сократить необоснованное потребление ресурсов и т.д. Кроме того, заметно разгружаются каналы связи, решается проблема фрагментации сетевого пространства и т.д.

Заметим, что одним из традиционных способов ускорения процесса обмена информацией является использование "локальной кэш" (в отличии от "региональной кэш" в Akamai). В этом случае полученные данные в течение некоторого времени хранятся локально. При последующих запросах сначала осуществляется поиск данных локально, и лишь затем на удаленной машине. Иногда осуществляется сохранение локально не только блока данных, к которому происходит обращение, но находящихся рядом с ним. Такой подход используется в сетевых приложениях, ориентированных на обмен относительно небольшими объемами информации, например, в браузерах. Но при работе с большими объемами информации, например, медиа-ресурсами и распределенными базами данных, этот подход теряет свои преимущества [2]. Это связано с тем, что поступающие объемы данных относительно быстро изменяются, сопоставимы или превосходят возможный размер кэш, отсутствует "интеллектуальная" составляющая алгоритма и т.д.

Для примера рассмотрим два соседних узла, C и D, которые в цикле обращаются к одним и тем же блокам данных B_1 и B_2. Блоки B_1 и B_2 лежат на узле B, который находится на значительном удалении от C и D, и каждый из блоков имеет размеры приблизительно равные объему "кэш" у C и D. В таком случае "кэш" каждого из узлов будет постоянно обновляться: то в ней B_1, то B_2. Как результат, B_1 и B_2 будут постоянно "гоняться" по сети между узлом C и B, а также между D и B, что приведет и необоснованной перегрузке сетевого трафика, и к замедлению выполнения задач на узлах C и D. В подобном случае ускорения обмена информацией можно добиться лишь за счет координированного распределения данных по узлам сети и динамической корректировки этого распределения в процессе работы системы. В данном примере оптимум достигается в случае, если копия одного из блоков хранится на C, а копия другого – на D, и эти узлы обращаются за данными друг к другу. Таким образом, идеальный

алгоритм должен учитывать то, какие блоки данных уже находятся на узлах соседних с теми, на которые данные планируется перекладывать.

Нахождение и корректировка такого оптимального распределения данных представляет собой задачу переборного типа. Так в [6] рассматриваются способы оптимального распределения файлов в полносвязном графе при ограничении, что число копий каждого файла фиксировано. В [7] исследуется проблема распределения копий файлов по узлам линейного графа с целью минимизации трафика передачи данных вдоль ребер графа. В [8] автор предложил модель сетевого распределения файлов, в которой все обращения к файлам разделяются на два типа – транзакции на запись (update transactions) и транзакции на чтение (query transactions); модель использовалась для минимизации совокупных издержек[1] распределения копий файлов по узлам в межуниверситетских хранилищах данных в США. Двойственная к данной проблема – распределение емкости (capacity) по узлам при заданной матрице обмена информацией (flow matrix) – исследовалась в [9]. В общей сложности к середине 80-х было предложено 12 базовых моделей оптимизации распределения данных по сети, которые систематически перечисленны в [10].

Дальнейшее развитие математики и теории алгоритмов позволило на базе этих 12 моделей построить более продвинутые системы. Так в [11,12] предложено решение задачи распределения[2] на основе теории полуопределенного (semidefinite) программирования. Пусть имеется беспроводная (wireless) сеть с m узлами $a_k \in R^2$ использующими n блоков данных $x_j \in R^2$. Пусть d_{ij} – мера расстояния между блоками данных i и j, d_{kj} – между узлом a_k и блоком данных x_j. Тогда оптимизационная задача принимает вид:

$$\min_{x}\left(\sum_{j=1}^{n}\sum_{i=1}^{j-1}\alpha_{ij}+\sum_{k=1}^{m}\sum_{j=1}^{n}\alpha_{kj}\right)$$

при условии

$$\| x_i - x_j \|^2 = d_{ij}^{\ 2} + \alpha_{ij} \quad \forall i, j, i < j$$

$$\| a_k - x_j \|^2 = d_{kj}^{\ 2} + \alpha_{kj} \quad \forall k = 1...m, j = 1...n$$

$$\| x_i - x_j \|^2 \geq R^2 \quad i < j$$

$$\| a_k - x_j \|^2 \geq R^2 \quad k = 1...m, j = 1...n$$

---

[1] Т.е. издержек хранения + издержек передачи

[2] на примере задачи распределения сенсоров

$\alpha_{kj} \geq 0 \quad \alpha_{ij} \geq 0$

К сожалению, алгоритм предложенный авторами применим только к беспроводным "статичным" сетям.

Еще один подход, уже на базе самокорректирующихся кодов (error-correcting codes), предложен в [13]. Основная идея – перекодирование и распределение информации таким образом, что для ее полного восстановления надо опрашивать лишь часть узлов сети.

Формальная постановка проблемы в (Jiang, 2004) имеет вид: Пусть имеется направленный граф G=(V,E) и информация объемом N символов. Каждое ребро e графа имеет "длину" l(e) – время передачи 1 бит по данному ребру. Тогда можно определить d(u,v) – расстояние между узлами u и v - как кратчайший путь между узлами u и v. Каждому узлу сопоставлено множество $R(v)=\{(r_i(v),k_i(v)): 1\leq i\leq n_v$ $r_i(v)\in R^+, k_i(v)\in N\}$ называемое множеством требований (requirements set). R(v) задает требования на то, как далеко блок данных $k_i(v)$ может находиться от узла v[1].Также для каждого узла u задано ограничение на объем хранимой на нем информации $W_{min}(u)\leq w(u)\leq W_{max}(u)$. Цель – при заданных для каждого узла требованиях R(v), $W_{min}(v)$ и $W_{max}(v)$ - минимизировать требуемый совокупный объем хранимой информации $\sum_{u\in V} w(u)$. Автором предложен подход, позволяющий эффективно решить данную задачу. Все файлы кодируются с помощью специального алгоритма на основе самокорректирующихся кодов и распределяются определенным образом по сети. Когда узлу требуется некий файл – он опрашивает соседние узлы и получает от них закодированные части файла и уже на их основе восстанавливает исходный файл. Основное достоинство алгоритма – высокая робастность. Модификации данного алгоритма сейчас активно используются в peer-to-peer сетях.

Недостатком всех вышеупомянутых исследований является предположение о статичности сети [13]. Ведь любая современная сеть, а Интернет особенно, постоянно изменяется! Таким образом, задача осложняется тем, что загруженность сети постоянно меняется, и алгоритм, занимающийся оптимизацией распределения данных по сети, должен в динамике учитывать изменения в загруженности каналов и востребованности тех или иных данных. Как результат, стандартные подходы, где экзогенные параметры должны оставаться неизменными в процессе работы алгоритма, оказываются неприемлемыми. В связи с этим целесообразно использовать адаптивные алгоритмы оптимизации,

---

[1] расстояние от v до ближайшего узла с k(v) не должно превышать v

где целевая функция может изменяться в процессе функционирования самого алгоритма. В данной работе используется один из таких адаптивных подходов – генетические алгоритмы, в которых параметры кодирования хромосом и функции качества могут экзогенно меняться в процессе работы алгоритма [4]. Это позволяется естественным образом объединить процессы нахождения и корректировки оптимального распределения данных.

Предлагаемый генетический алгоритм получает на вход следующие параметры:

- число узлов n и блоков данных m в сети[1]
- размер каждого блока данных w_s и объем дискового пространства каждого узла S_k
- загруженность каналов сети между узлами k и m - α_{km}. Значение α_{km} строится на основании статистических данных, которые могут меняться в процессе работы алгоритма. В простейшем случае в качестве оценки α_{km} может выступать медианное наблюдение, в более сложном – прогнозируемое посредством теории временных рядов значение α_{km} (это позволяет учесть влияние времени суток, дня недели, сезонность и т.д.)
- частота обращения каждого узла k к каждому блоку данных s - β_{ks}. Как и значения α_{km}, β_{ks} также оцениваются посредством анализа статистических данных.
- текущее распределение данных – многозначная функция $\phi(s)$, которая каждому блоку данных s ставит в соответствие узлы, на которых он находится.

Параметры α_{km} и β_{ks} могут менять в процессе работы генетического алгоритма. Эти изменения должны "мгновенно подхватываться" и учитываться алгоритмом. В случае богатого набора статистических данных (т.е. алгоритм работает уже достаточно долго) новое наблюдение не несет кардинально большого количества новой информации, поэтому изменение оценок происходит достаточно плавно. В случае бедного набора данных следует "искусственно уменьшать" количество информации несомое новым наблюдением.

Минимизируемая в процессе работы алгоритма функция качества представляет собой среднюю сглаженную загруженность сети при ограничении на вместимость каждого из узлов:

---

[1] Для лучшего понимания узлы удобно представлять как сервера с данными, а блоки данных – информация хранящаяся в базах данных серверов.

$$C = \sum_{s=0}^{m-1} \sum_{k=0}^{n-1} \alpha_{k,\sigma(s)} * \beta_{k,s} * w_s$$

$$\sum_{\sigma(s)=k} w_s \le S_k$$

Оптимизация осуществляется по расположению данных на узлах сети, которое задается многозначной функцией σ(s).

При практической реализации алгоритма должно учитываться то, что перераспределение данных не является "бесплатной операцией", т.е. решение σ(s) сильно отличающееся от текущего распределения ϕ(s) должно "штрафоваться". Также алгоритм должен по возможности избегать решений при которых "данные нигде не хранятся", т.е. теряются. В случае дублирования данных, возникает необходимость поддержки консистентности между ними, поэтому алгоритм должен отдавать предпочтение тем решениям σ(s), где дублирование меньше. Для кодирования задачи в терминах генетического алгоритма используется метод многомерного куба [5].

Пусть ϕ(s) – текущее распределение данных по серверам. Назовем

- штрафом за перемещение данных функционал $T_1(\phi,\sigma) = f_1 * \sum_{s=0}^{n-1} I_{\{\phi(s)\neq\sigma(s)\}} * w_s$ где f_1 – величина штрафа, $I_{\{\phi(s)\neq\sigma(s)\}} = \begin{cases} 1 & \phi(s) \neq \sigma(s) \\ 0 & \phi(s) = \sigma(s) \end{cases}$ – индикаторная функция, w_s - размер каждого s-ого блока данных
- штрафом за утрату данных функционал $T_2(\sigma) = f_2 * \sum_{s=0}^{n-1} \left(I_{\{\#\sigma(s)=0\}} \sum_{k=0}^{n-1} \beta_{k,s} \right)$ где f_2 – величина штрафа, $I_{\{\#\sigma(s)=0\}} = \begin{cases} 1 & \#\sigma(s) = 0 \\ 0 & \#\sigma(s) > 0 \end{cases}$ – индикаторная функция, $\sum_{k=0}^{n-1} \beta_{k,s}$ - ценность (количество обращений в единицу времени) s-ого блока данных, $\#\sigma(s)$ - количество узлов с блоком данных s.
- штрафом за дублирование данных функционал $T_3(\sigma) = f_3 * \sum_{s=0}^{n-1} \#\sigma(s)$ где f_3 – величина штрафа, $\#\sigma(s)$ - количество узлов с блоком данных s.

Хромосома схематически имеет вид:

узел 1		узел 2		узел 3		узел 4		узел 5		узел 6	
B_{11}	B_4	B_5	B_{20}	B_1	B_{21}	B_{18}	B_{14}	B_{19}	B_{16}	B_{17}	B_{23}
B_7	B_8	B_{22}	B_6		B_{12}	B_{13}	B_2				B_9
B_3	B_{15}		B_{10}								

Каждая такая хромосома описывает одно допустимое решение (схему хранения данных) $\sigma(s)$ и состоит из нескольких ген. Каждый ген характеризует один узел-сервер и значение гена – номера блоков данных которые "хранятся" на этом узле. Например, на узле 3 (эквивалентно, гене 3) "хранятся" блоки данных B_1, B_{21} и B_{12}. При этом допустимыми решениями являются те и только те, на которых выполняется ограничение на вместимость по каждому узлу, т.е.

$$\sum_{\sigma(s)=k} w_s \le S_k$$

Функция качества имеет вид

$$C(\phi,\sigma) + T_1(\phi,\sigma) + T_2(\sigma) + T_3(\sigma) \to \min_{\sigma(s)}$$

Основными операторами генетического алгоритма являются мутация и скрещивание. Существует огромное количество способов задания этих двух операторов (Афанасьев, Дмитриев, 2000). Для примера рассмотрим по-одному способу задания мутации и скрещивания.

Мутация. Схематически может быть представлена в виде:

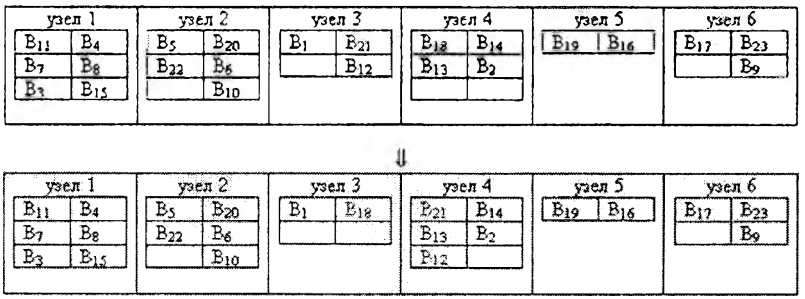
узел 1		узел 2		узел 3		узел 4		узел 5		узел 6	
B_{11}	B_4	B_5	B_{20}	B_1	B_{21}	B_{18}	B_{14}	B_{19}	B_{16}	B_{17}	B_{23}
B_7	B_8	B_{22}	B_6		B_{12}	B_{13}	B_2				B_9
B_3	B_{15}		B_{10}								

⇓

узел 1		узел 2		узел 3		узел 4		узел 5		узел 6	
B_{11}	B_4	B_5	B_{20}	B_1	B_{18}	B_{21}	B_{14}	B_{19}	B_{16}	B_{17}	B_{23}
B_7	B_8	B_{22}	B_6		B_{12}	B_{13}	B_2				B_9
B_3	B_{15}		B_{10}								

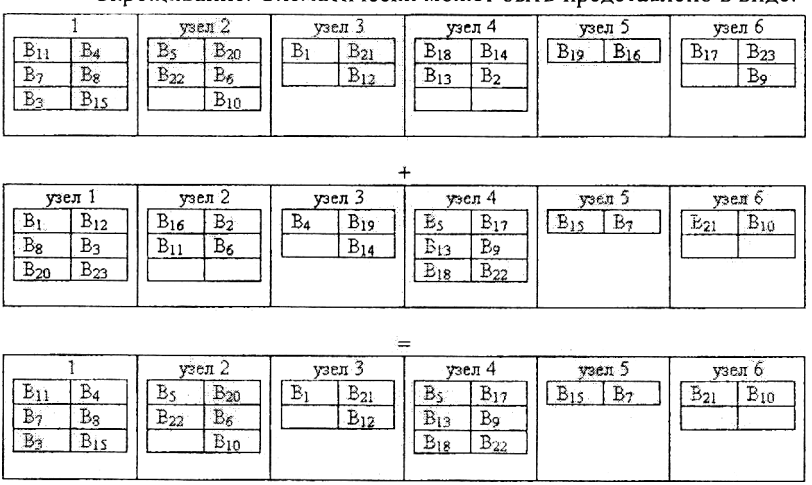
При мутации случайным образом выбираются два узла – в данном примере это узлы 3 и 4 – которые будут мутированы. На каждом из узлов выбирается по одному или несколько блоков данных. Так, на узле 3 был выбран блок данных B_{21}, а на узле 4 – B_{18}; затем B_{21} "переместился" на узел 4, а B_{18} – на узел 3. Возможна ситуация, когда полученное решение оказывается недопустимым. Например, блок B_{18}

настолько большой, что его невозможно поместить на узел 3. Для избежания подобной ситуации обмен осуществляется блоками данных или наборами блоков данных сопоставимого размера.



Если не удается подобрать наборы блоков данных сопоставимого размера, мутация считается неудачной и происходит откат.

Скрещивание. Схематически может быть представлено в виде:



При скрещивании случайным образом выбирается "точка разреза" и все гены правее точки "приходят" от первого родителя, а левее – от второго. В примере "точка разреза" находится между узлами 3 и 4, и потомок получает значения ген (узлов) 1-3 от первого родителя, а ген 4-6 – от второго.

В результате скрещивания часть данных может дублицироваться, а часть – теряется. В примере B_7 находится одновременно на узлах 1 и 5, а B_2 – теряется. Для "улучшения" функции качества потомка, в полученной хромосомс происходит, если возможно,

замена дублируемых блоков данных на отсутствующие. Если невозможно – появляется потомок с дублируемыми и/или отсутствующими данными. "Дефектность" такой хромосомы будет позднее учтена при вычислении ее функции качества.

Для упрощения понимания алгоритм удобно представлять централизованным, т.е. имеются специализированный сервер, который хранит наиболее "свежую" информацию по оценкам α_{km} и β_{ks}, и несколько вычислительных серверов, на которых с помощью островной модели генетического алгоритма осуществляется поиск оптимального решения. При этом копии генетического алгоритма выполняются на разных компьютерах, а синхронизация между ними осуществляется за счет механизма миграции. В более сложном децентрализованном случае нет специализированного сервера для хранения оценок α_{km} и β_{ks}, т.е. сервера-вычислители потенциально могут имеет разные оценки α_{km} и β_{ks}, которые необходимо периодически согласовывать.

Следует отметить, что алгоритм также позволяет оценить критичность увеличения дискового пространства за счет использования двойственных оценок [14]. На основании полученного в результате работы алгоритма оптимального решения строятся двойственные переменные и с их помощью оценивается, насколько каждое из ограничений на объем физической памяти сервера является критическим и каков будет выигрыш по скорости от увеличения дискового пространства отдельно взятого сервера.

Недостатком алгоритма является то, что алгоритм реагирует только на глобальные (долгосрочные) изменения в состоянии сети, т.е. изменения состояния окружающей среды сглажены за счет того, что оценки α_{km} и β_{ks} меняются не очень быстро. Данный недостаток может решаться за счет использования серверов Akamai в качестве "верхнего уровня", т.е. алгоритм находит наилучшее местоположение для хранилища данных, доступ же пользователя к хранилищу данных осуществляется через промежуточные сервера Akamai. Как показала практика, сервера Akamai особенно хорошо улавливают краткосрочные изменения; таким образом, предлагаемый подход и подход Akamai дополняют друг друга

Таким образом, предложенный в работе генетический алгоритм оптимального планирования распределения данных позволяет минимизировать ожидаемые суммарные затраты по передаче данных между узлами учитывая ограниченность размеров дискового пространства для каждого узла и затраты на сам процесс оптимизации распределения данных за счет добавления специальных штрафов в функцию качества.

Литература

1. F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica "Wide-area cooperative storage with CFS" in Proc.of the 18th ACM Symposium on Operating Systems Principles (SOSP'01), Chateau Lake Louise, Banff, Canada, Oct. 2001

2. Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin Theimer, Roger P. Wattenhofer FARSITE: Federated, Available, and Reliable storage for Incompletely Trusted Environment, Proc. of the 5th Symposium on Operating Systems Design and Implementation, 2002

3. Michel Barreau. KazaA, iMesh, eDonkey at tous les autres, CampusPress, France, 2003

4. М.К. Афанасьев, П.А. Дмитриев. Подходы к исследованию поведения генетических алгоритмов, Труды 4-го МС "Интеллектуальные системы (ИНТЕЛС-2000)",Москва, 2000

5. М.К. Афанасьев, Конструктор генетических алгоритмов и способы кодирования хромосом. Москва, Вестник МГУ, сер. 15, Вычисл. Матем. и Киберн.,№3, 2001

6. Chu W.W.(1969) "Optimal file allocation in a multiple computer system" IEEE Trans. On Computers, C-18, 10 (Oct 1969), pp 885-889

7. Whitney (1970) "A study of optimal file assignment and communication network configuration in remote-access computer message processing and communications systems" SEL Tech Rep No 48, U. of Michigan, Ann Arborm Mich., Sept. 1970

8. Casey, R.G. (1972) "Allocation of copies of files in an information network" Proc. AFIPS 1972 SJCC, Vol. 40, AFIPS Press, Montvale, NJ, pp 617-625

9. Gerla M. (1972) "The design of store and forward networks for computer communications" Tech. Rep. UCLA-ENG-F319, Comput. Sci. Dep, U.of California, Los Angeles, Calif, Jan 1972

10.L.W. Dowdy, D.V. Foster (1982) "Comparative Models of the File Assignment Problem", Computing Surveys, vol. 14, No 2, pp 287-313, 1982

11.Biswas, Ye Yinyu (2003) A distributed method for solving semidefinite programs arising from Ad Hoc Wireless Sensor Network Localization, Oct 2003 http://www.stanford.edu/~yyye/

12.So, Ye Yinyu (2004) "Theory of Semidefinite Programming for Sensor Network Localization", April 2004 http://www.stanford.edu/~yyye/

13.Jiang Anxiao (Andrew), Bruck Jehoshua (2005) "Network File Storage with Graceful Performance Degradation" ACM Transactions on Storage, March 2005 http://www.paradise.caltech.edu/papers/etr061.pdf

14.Robert J. Vanderbei, Linear Programming: Foundations and Extensions, Princeton University, Second Edition, 2001, rvdb@princeton.edu

Раздел V

Инструментальные средства и сообщения

Мещеряков Д.К.

Исследование свойств случайных чисел, получаемых с использованием примитивов ОС.[1]

Введение

Случайные числа и их последовательности востребованы во многих применениях, в т.ч. защите данных и ограничении доступа, игровых приложениях, имитационном моделировании, включая генетические алгоритмы. Каждое применение предъявляет свои требования к качеству используемых случайных чисел.

Работа [1] описывает способ получения последовательностей чисел со случайной природой и результаты исследования статистических свойств выборок среднего объема. В связи с тем, что данный способ дает, вообще говоря, апериодическую последовательность чисел, представляет интерес исследование свойств последовательности чисел на выборке большого объема. Настоящая работа продолжает исследование статистических свойств на случай выборки большого объема. Предлагается архитектурное решение, позволяющее использовать исследуемый подход в любых программах, требующих для работы случайные числа, в рамках многопоточной модели выполнения.

Предварительные замечания

Исследуемый способ получения случайных чисел предполагает использование двух компонентов вычислительной системы, связь между которыми сложна и труднопредсказуема. В качестве таковых

[1] Работа выполнена при поддержке гранта РФФИ № 05-07-90238.

выступают счетчик тактов центрального процессора (присутствует в моделях Intel Pentium, AMD K5 и выше и ряде других) и планировщик операционной системы.

Счетчик тактов представляет собой целочисленный регистр большой разрядности (64 двоичных разряда у упомянутых моделей ЦП), который аппаратно увеличивается на единицу с каждым тактом работы ЦП (см. [2]). Планировщик ОС решает задачи распределения времени ЦП между выполняемыми в системе потоками команд. Планировщики современных ОС ([3]) исключительно сложны, поэтому точное предсказание границ интервалов времени, в которые будет выполняться тот или иной поток, затруднительно или не представляется возможным.

Одной из сильных сторон исследуемого подхода является простота реализации. Считывание счетчика тактов осуществляется любой программой (не обязательно работающей в привилегированном режиме) за несколько тактов процессора следующей процедурой (здесь и далее используется псевдокод, имитирующий язык Си и ассемблер Intel 80x86 ([4])):

```
unsigned char getrndbyte()
{
    unsigned char rndbyte;
    asm
    {
        /* считать счетчик тактов в регистры eax:edx*/
        rdtsc
        /*считать младший байт в переменную*/
        move byte ptr rndbyte, al
    }
    /*вернуть младший байт*/
    return rndbyte;
}
```

Цикл для получения последовательности чисел построить также несложно:

```
while(1)
{
    getrndbyte();          //получить следующее число
    Sleep(1);              //приостановить выполнение потока
}
```

Использование функции Sleep() из Win32 API как раз и позволяет создать сложную зависимость между работой потока в режиме выполнения и работой планировщика ОС. Эта функция приостанавливает поток на число миллисекунд, не меньшее заданного. Поскольку младшие разряды (до 15-го включительно) счетчика тактов

можно считать изменяющимися циклически и с высокой скоростью, а точное время возврата управления функцией Sleep() сложно предсказуемо, получаемые процедурой числа следует считать случайными.

В случае, если необходимо осуществить получение одного числа, можно выполнять вызов функции Sleep() перед получением очередного значения из счетчика тактов.

Следует отдельно заметить, что потребность описанного алгоритма во времени ЦП является крайне низкой, что позволяет получать последовательность в отдельном потоке команд параллельно с работой основного потока программы и без существенного снижения скорости выполнения последнего. Одновременно нетрудно заметить, что скорость генерации чисел существенно ограничена возможностями планировщика и тактовой частотой ЦП. Очевидно, для существенного увеличения выхода случайных чисел в единицу времени следует осуществлять генерацию нескольких последовательностей в нескольких параллельных потоках.

В [1] приводятся результаты исследования статистических свойств получаемой таким образом последовательности, они показывают высокое качество последовательности, но проведены на небольших выборках и в течение коротких промежутков времени. Представляет большой интерес исследование статистических свойств на выборках большего объема (соответственно, полученных в течение более длительных промежутков времени) и с анализом влияния различных аспектов функционирования вычислительной системы на свойства последовательности.

Архитектура программного средства

Для исследования была разработана и реализована специальная программа (исторически ей было присвоено название Dropper), осуществляющая генерацию нескольких последовательностей параллельно. Она реализована в виде многопоточного Win32 приложения и включает в себя потоки двух видов:

- 40 однотипных низкоприоритетных потоков («генерирующих»), в каждом из которых осуществляется генерация своей последовательности. Число потоков было выбрано как компромиссное. С одной стороны, увеличение числа потоков увеличивает выход случайных чисел в единицу времени, с другой, - после превышения некоторого числа потоков операционная система начинает расходовать на планирование практически все процессорное время, в результате ЭВМ становится непригодной для использования под какие-либо другие цели.

- Один поток среднего приоритета («пользовательский»), выполняющий обработку пользовательского ввода, выдачу результатов, сохранение результатов и другие вспомогательные функции.

Выбранная архитектура позволяет проводить испытания на ЭВМ, используемой как рабочая станция, без ущерба для возможностей использования данной ЭВМ в обычном режиме.

Программа Dropper определяет равномерность распределения 7-го (старшего в младшем байте) разряда в получаемых случайных числах. Каждый генерирующий поток организован в виде двух вложенных циклов.

Внутренний цикл выполняет с помощью вышеописанной процедуры генерацию 1600 случайных байт и определяет число тех из них, которые содержат единицу в 7-м разряде. По окончании цикла выполняется обновление двух глобальных счетчиков (переменных, разделяемых всеми потоками). Один счетчик содержит общее число испытаний, другой – число испытаний, в которых 7-й разряд полученного байта содержал единицу. Для обеспечения атомарности изменения группы счетчиков используются примитивы синхронизации.

Внешний (объемлющий) цикл выполнен бесконечным. Он запускается примитивом CreateThread() (создание потока) из Win32 API при запуске программы и разрывается при выполнении пользовательским потоком примитива TerminateThread() из Win32 API во время завершения работы программы.

Пользовательский поток выполняет следующие функции:

- Отображает общее число испытаний и число единичных 7-х разрядов (показывает значения глобальных счетчиков) и долю единичных 7-х разрядов в общем числе испытаний.
- Периодически (каждые 51.2 с) сохраняет значения глобальных счетчиков в системном реестре (энергонезависимой памяти). Такое решение позволяет практически полностью оградить эксперимент от происходящих время от времени сбоев ЭВМ.
- При запуске программы считывает значения глобальных счетчиков из реестра, а при завершении работы программы – сохраняет их в реестре. Если чтение осуществить не удается, то, очевидно, эксперимент на данной ЭВМ ранее не проводился, а потому счетчикам присваиваются нулевые значения. Данное решение позволяет вести эксперимент на обычной рабочей станции, которая не работает непрерывно, а включается на некоторые промежутки времени.
- При запуске программы запускает генерирующие потоки, а при завершении ее работы – останавливает их.

Пользовательский поток также выполняет чтения и обновления счетчиков атомарно.

Для синхронизации доступа потоков к глобальным счетчикам используется общая критическая секция (реализация исключающей переменной в Win32 API).

Представляется целесообразным использование элементов архитектуры программы Dropper при разработке любых программ, использующих случайные числа в своей работе. Один или несколько генерирующих потоков могут сохранять полученные числа в разделяемой области памяти, откуда числа могут изыматься для использования основным потоком. Такое решение позволит основному потоку избежать задержек, связанных с необходимостью ожидания генерации чисел, при условии, что скорость генерации не ниже скорости потребления и генерирующим потокам дается время в начале работы программы для генерации некоторого относительно небольшого запаса чисел. Неизбежной технической сложностью остается необходимость использования примитивов синхронизации для потоков программы. Использование примитивов синхронизации связано также и с некоторыми накладными расходами, которые, однако, могут быть сделаны мало влияющими на скорость выполнения программы при их грамотном использовании. Именно необходимо, чтобы обращения потоков к разделяемой памяти (и соответствующие обращения к примитивам синхронизации) производились относительно редко, а в остальное время потоки работали только с принадлежащей им памятью (доступ к ней не связан с необходимостью использования примитивов синхронизации) подобно тому, как это реализовано в программе Dropper.

В дополнение к программе Dropper было использовано вспомогательное средство (программа LoggerA), позволяющее с достаточной точностью оценивать общее астрономическое время эксперимента. Программа LoggerA, используя механизмы оповещения, предлагаемые платформой Win32, отмечает моменты запуска и останова ОС ЭВМ и сохраняет в заранее выбранный файл журнала записи о времени означенных событий. Поскольку программа Dropper запускается и завершается приблизительно в те же моменты, то на основе анализа записей о времени запуска и останова ОС ЭВМ впоследствии возможна оценка полного астрономического времени эксперимента с относительной погрешностью около 1%.

Эксперимент и оценка результатов

С использованием описанной выше программы был проведен эксперимент по получению выборки большого объема. Под эксперимент была задействована ЭВМ, используемая в качестве рабочей станции и эксплуатируемая в обычном режиме в течение всего

времени эксперимента. Для нее были характерны следующие варианты вычислительной нагрузки:

- Почти полное бездействие. Выполняются ядро ОС, служебные программы и программы интерфейса ОС. Доля свободного времени ЦП порядка 97%.
- Воспроизведение музыки в формате MP3 одновременно с предыдущим. Доля свободного времени ЦП порядка 80%.
- Одновременно с одним из предыдущих - выполнение программ, дающих короткую (порядка секунды) или продолжительную (до нескольких часов) полную загрузку ЦП. К первой группе следует отнести программы типа текстовых редакторов и терминальных клиентов, ко второй – разнообразные программы имитационного моделирования, сжатия данных, воспроизведения потокового видео. В ряде случаев программы из обеих групп выполнялись одновременно.

Одновременно выполнялась программа Dropper. Она запускалась сразу после запуска ОС ЭВМ и завершалась непосредственно перед остановом ОС ЭВМ. Следует особо отметить, что работа программы Dropper не сказывалась на возможности использования ЭВМ в обычном режиме, что прямо указывает на возможность использования многих потоков генерации чисел в составе многопоточных программ без заметного снижения как скорости выполнения последних, так и производительности ЭВМ в целом.

Общее астрономическое время эксперимента оценивается в 175 суток. За это время было выполнено около 152 млрд. испытаний.

Основной интерес представляют два аспекта. Прежде всего, - это равномерность распределения 7-го разряда. Были получены следующие результаты:

Всего испытаний	152199888000
Единичных 7-х разрядов	76100035507

Таблица 1. Результаты эксперимента.

Итого доля единичных 7-х разрядов составляет 0.50000 (дальнейшие разряды незначительно менялись на протяжении всего эксперимента). Для проверки гипотезы о равномерности распределения 7-го разряда используется критерий согласия Пирсона – вычисляется величина χ^2 и по таблице χ^2-распределения с одной степенью свободы ([5]) определяется доверительная вероятность.

Из таблицы для ДВ = 0.7	0.15
Из таблицы для ДВ = 0.5	0.45
Вычислено	0.22

Таблица 2. Тест на равномерность для 7-го разряда.

В соответствии с критерием согласия Пирсона, значения 7-го разряда можно считать равномерно распределенными с доверительной вероятностью не менее 0.5.

Кроме того, интересно влияние изменения вычислительной нагрузки на ход эксперимента. Было замечено, что скорость получения чисел и доля тех или иных исходов практически не зависела от вычислительной нагрузки на ЭВМ. Это может представляться подозрительным, поскольку генерирующие потоки имеют минимальный возможный приоритет, в то время как одновременно выполняемые вычислительно интенсивные потоки из других приложений – более высокий средний и предположительно должны потреблять время ЦП в ущерб выполнению генерирующих потоков. Полученный результат следует отнести на счет т.н. динамического повышения приоритета, т.е. временного повышения приоритета потока, который выходит из состояния ожидания (ф-ия Sleep() возвращает управление потоку). В результате генерирующие потоки получают достаточно времени ЦП, чтобы выполнять свои функции.

В ряде применений может быть важна скорость генерации чисел. В данном эксперименте она составила порядка 250 чисел в секунду на каждый поток или, соответственно, порядка 10 тыс. чисел в секунду на все генерирующие потоки.

Заключительные замечания

В рамках исследования статистических свойств случайных чисел, получаемых по методике из [1], предложена архитектура программ, позволяющая получать случайные числа одновременно с решением программами их основных задач. Установлено, что такое решение позволяет получать последовательности одноразрядных случайных чисел со скоростью и качеством, которые могут быть достаточными для ряда применений.

Литература

1. Мещеряков Д.К. Простой способ генерации случайных чисел с использованием примитивов ОС. //Программные системы и инструменты. Тематический сборник № 5, М.: Изд-во факультета ВМиК МГУ, 2004.

2. Гук М. Процессоры Pentium II, Pentium Pro и просто Pentium. – СПб.: Питер, 1999.

3. Соломон Д., Руссинович М. Внутреннее устройство Microsoft Windows 2000. Мастер-класс. /Пер. с англ. – СПб.: Питер; М.: Издательско-торговый дом «Русская редакция», 2001.

4. Юров В. Assembler: специальный справочник. – СПб.: Питер, 2001.

5. Ивашев-Мусатов О.С. Теория вероятностей и математическая статистика. – М.: ФИМА, 2003.

Сидорова Ю. А.

Распознавание эмоций по акустическому сигналу: к вопросу оптимизации человеко-машинного интерфейса

Несколько десятилетий назад акустики обратили внимание на существование корреляции между эмоциональным состоянием говорящего и акустическими характеристиками речи, появилось понятие «акустического профиля» эмоции [1]. Множество эмоций для распознавания зависит от конкретной задачи. Альтернативы – стандарт MPEG-4 (отвращение, удивление, радость, страх, гнев, грусть и нейтральное для сравнения) или другой, более подходящий поставленной задаче набор. Преимущество множества из стандарта MPEG-4 – возможность сравнить полученные результаты с существующими распознавателями [3], [4], что необходимо при тестировании нового алгоритмического подхода или работе с языком, для которого распознающий речевые эмоции агент конструируется впервые. На данный момент не существует базы данных эмоциональной речи в открытом доступе, разработчику, вероятнее всего, придется собирать свою.

Эмоциональное состояние можно определить на основе: параметров, связанных с частотой основного тона, формантными значениями, длительностью, с качеством голоса, артикуляцией. Перед обучением классификатора акустические параметры необходимо нормализовать. Основная дилемма - между статистическими и снятыми во временном окне (20 -100 мс) параметрами, вопрос достаточно подробно обсуждается в [4]. Большинство исследователей используют статистические параметры, было показано [5], что такой выбор приводит к более высокой точности распознавания эмоционального состояния. Для снятия этих параметров существуют ПО в свободном доступе, например Praat [6]. Для отбора оптимального множества параметров низкого уровня целесообразно использовать алгоритмы выбора параметров и их редукции. Далее параметры одной части базы данных используются для обучения классификатора, а другой для его тестирования.

Основные и наиболее подходящие для данной задачи классификаторы: k-Means, kNN, GMM, MLP, SMV, ML-SMV [2]. В таблице 1 приведены данные по [2] касательно их способности к классификации в дикторозавимых и дикторонезависимых условиях, видно, что классификатор, сохраняющий структуру объекта более устойчив при переходе от дикторозависимого контекста к

дикторонезависимому. Альтернативные классификаторы и акустическое моделирование – актуальная задача, решение которой поможет решить некоторые, до сих пор нерешенные задачи области: например, классификация эмоционального состояния по речи, независимо от входящего языка, что теоретически возможно [7].

Классификатор	Д/Н. распознавание % ошибки	Д/Н.распознавание % ошибки
kMeans	57.05	27.38
kNN	30.41	17.39
GMM	25.17	10.88
MLP	26.85	9.36
SVM	23.88	7.05
ML-SVM	18.71	9.05

Таблица 1: сравнение классификаторов в дикторонезависимых и дикторозависимых условиях

Грамматический вывод на древесных языках и деревья принятия решений

В данном параграфе предложено еще одно алгоритмическое решение классификатора [8].

Шаг 1: конвертирование предложений в выражение на древесном языке. Произнесенные предложения конвертируются в древесные высказывания с постоянным скелетом (алгеброй) [9], представляющем структуру предложения с лингвистической точки зрения. Грамматика, определяющая скелет, задана ниже:

{

Акустическая_информация_высказывания > (Просодия) (Сегментная_информация);

Просодия > (Частота_основного_тона) (Интенсивность) (Временные_параметры);

Сегментная_информация > (Энергия) (Форманты);

Частота_основного_тона > (Математическое_ожидание_F0) (Медиана_ F0) (Максимум_ F0) (Минимум_ F0) (Диапазон_ F0) (Вариация_ F0) (Средний_подъем_ F0) (Минимальный_подъем_ F0) (Максимальный_подъем_ F0) (Среднее_падение_ F0) (Минимальное_паденис_ F0) (Максимальное_падение_ F0) (Средняя_крутизна_подъема_ F0) (Средняя_крутизна_падения_ F0);

Интенсивность > (Дисперсия интенсивности) (Медиана дисперсии_интерсивности) (Максимальная_дисперсия_интесивности) (Минимальная дисперсия интесивности) (Диапазон_интенсивности) (Вариация_интенсивности);

Временные_параметры > (Средняя_длительность_гласных) (Средняя_длительность_согласных) (Отношение_речь/пауза);

Энергия > (Относительная_энергия_ниже_500_Hz);

Форманты > (Среднее_F1) (Среднее_ F2) (Среднее_ F3) (Среднее_ F4) (Средняя_ширина_ F1) (Средняя_ширина_ F2) (Средняя_ширина_ F3) (Средняя_ширина_ F4);

}

Листья (прямые наследники переменных, которые не появляются слева от знака «>») древесных выражений (константы) – это численные значения параметров низкого уровня, извлеченные из каждого конкретного высказывания. Были получены 5 множеств древесных выражений: каждое множество содержит в себе древесные выражения предложений одного эмоционального типа (то есть гнева, иронии и т.д).

Шаг 2: методом грамматического вывода на древесных языках получить древесные грамматики для 5 автоматов, принимающих один и только один тип эмоциональных высказываний. Грамматический вывод [10] – это процесс выучивания грамматики по примерам. В нашем случае использует грамматический вывод на древесных языках, поскольку именно древесные выражения используются для представления высказывания. Более того, задача сводится к более частной, а именно выучить возможные интервалы значений, которые могут принимать переменные, не появляющиеся слева от знака «>» в грамматике скелета. Результат данного шага – 5 древесных автоматов, каждый из которых распознает древесные выражения одного из 5 множеств, соответствующих одному из пяти эмоциональному типу.

Шаг 3: Подсчет расстояний редактирования между древесными автоматами и древесными высказываниями из тренировочного множества; полученные расстояния поместить в таблицу. На данном этапе считаются расстояния редактирования между каждым древесным автоматом, выученным на шаге 2, и каждым древесным выражением, представляющем высказывание из тренировочного множества. Расстояния редактирования для древесных языков – это формально подсчитанное различие между двумя древесными структурами: сначала строятся биективные соответствия между каждым узлом сравниваемых древесных структур, затем считается минимальное число операций удаления, добавления и замены, необходимых для того, чтобы древесные структуры стали изоморфными. Подсчитанные расстояния редактирования помещены в

таблицу объемом 162 на 5 (мощность тренировочного множества помноженная на 5 выученных автоматов).

Шаг 4: Исполнить алгоритм C4.5 на таблице. C4.5 [11] выполняется на таблице, полученной на шаге 3. Результатом работы алгоритма является дерево принятия решения, классифицирующее каждое древесное высказывание на основе 5 значений расстояний редактирования как одно из 5 типов.

Тестирование предложенного решения

Использовалась речь четырех актеров из телефильма. Высказывания были перетасованы случайным образом и затем представлены на прослушивание 14 специально нетренированным людям, в задачи которых входило соотнести каждое слышимое ими высказывание с эмоцией из предложенного списка. Высказывания, в отношении эмоциональной окраски которых был получен одинаковый ответ, по крайней мере, от 10 людей, послужили материалом для эксперимента.

Первая часть базы данных была использована для целей машинного обучения классификатора (для получения дерева принятия решений на расстояниях редактирования между древесным представлением данного высказывания и пятью автоматами). Тестирование, результаты которого отражены в таблице 1, классификатора производилось на второй части базы данных. Процент правильного распознавания - 70%. Результат показывает, что алгоритмическое решение, будучи адаптировано для аудио сигнала, является оправданным для задачи распознавания речевых эмоций.

	Гнев	Ирон.	Удивл.	Страх	Неуд.	Радос	Стыд	Нейтр.
Гнев	**88%**	0%	5%	0%	0%	2%	0%	5%
Ирония	0%	**75%**	3%	0%	1%	1%	0%	10%
Удивл.	3%	0%	**85%**	3%	4%	3%	0%	2%
Страх	0%	0%	21%	**60%**	3%	1%	0%	15%
Неудов.	0%	0%	0%	0%	**88%**	0%	0%	12%
Радос.	13%	0%	16%	4%	0%	**67%**	0%	0%
Стыд	0%	0%	0%	4%	6%	0%	**80%**	10%
Нейтрал.	1%	1%	10%	1%	8%	1%	1%	**77%**

Таблица 2: ошибки распознавания

Результаты и дальнейшая работа

В данной статье обсуждаются основные вопросы, встающие перед разработчиком, решившим сконструировать агента интерфейса, чувствительного в речевым эмоциям. Предложено альтернативное алгоритмическое решение классификатора и его тестирование. На основе акустической информации, извлекаемой из речи пользователя, агент делает вывод об эмоциональном состоянии пользователя, и реагирует заданным образом. Были использованы принципы, ранее для подобных целей не использовавшиеся. Достигнутый процент правильного дикторозависимого распознавания – около 70 %, что указывает, во-первых, на адекватность алгоритмического решения целям распознавания речевой эмоции, во-вторых, на правильный выбор типа акустической информации, принимаемой во внимание, в-третьих, на возможность построения чувствительных к голосовой эмоции агентов для русского языка. В рамках дальнейшей работы планируется поиск и тестирование менее «плоского» древесного представления лингвистического объекта высказывание. Это помогло бы более эффективно применять принципы распознавания объектов со сложной структурой. Также необходима база данных большего объема для дальнейшего тестирования агента и выявления возможных слабых сторон предложенного решения.

Литература

1. [Banse, Scherer, 1996] Banse R. and Scherer K. R. "Acoustic Profiles in Vocal Emotion Expression", Journal of Personality and Social Psychology, 70/3, 614-636, 1996.

2. Augmented Multiparty Interaction. Integrated Project. D 4.1. Report on Implementation of Audio, Video, and Multimodal Algorithms. 2004.

3. Nogueiras, A., Moreno, A, Bonafonte, A, Marino, J. B. Speech emotion recognition usinf hidden Markov models. Eurospeech, 2001.

4. Hozjan V., Zdravko K. "Improved Emotion recognition with Large Set of Statistical Features", Eurospeech 2003, 2003.

5. "Recognizing Emotions in speech Using Short-term and Long-term Features", Proceedings of the International Conference on Speech and Langiuage Processing. 2255-2558. 1998

6. www.praat.org

7. Scherer, K. R., Banse, R., & Wallbott, H. G. Emotion inferences from vocal expression correlate across languages and cultures. Journal of Cross-Cultural Psychology, 32(1), 76-92. (2001).

8. Sidorova, J. A. "Speech emotion recognition: error-correcting tree language inference & decision trees". In Proc.1st Annual International Workshop on emotions in voice, Emeryville, CA, USA, March 2005.

9. Thatcher J.W. Tree Automata: an informal survey. In A.V. Aho, editor, Currents in the Theory of Computing, pages 143-178. Prentice Hall, 1973.

10.Sakakibara Y., "Recent advances of grammatical inference", Theoretical Computer Science 185, pp 15-45, 1997.

11.Quinlan J. R. C4.5: programs for machine learning. Morgan Kaufmann, 1993.

Аннотации

Корухова Ю.С. Дедуктивный синтез программ с использованием волновых правил. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В работе рассматривается синтез функциональных программ с помощью метода дедуктивных таблиц. При таком подходе формальная спецификация программы рассматривается как теорема существования, из конструктивного доказательства которой извлекается требуемая программа. Применение метода дедуктивных таблиц в "чистом" виде ведет к большому перебору, что делает его практическое использование невозможным. Для сокращения перебора предлагается использовать т. н. волновые правила, применяющиеся при доказательстве по индукции, которое возникает при построении рекурсивных программ. С помощью волновых правил строится план доказательства, которое затем проводится в дедуктивной таблице одновременно с синтезом рекурсивной ветви функции. Данный подход был реализован автором в системе автоматического синтеза программ АЛИСА.

Табл.: 7

Библиогр.: 12 назв.

Попова Е. А. Разработка адаптивной системы машинного обучения на основе аппарата деревьев решений. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В работе предложен алгоритм адаптации системы к новым условиям на основе деревьев решений. Показано, что переход от прошлого поведения к будущему может быть эффективно построен на основе выбранных признаков поведения хранимых в долговременной и кратковременной памяти системы. Основное преимущество выбранного метода состоит в том, что он позволяет хранить информацию о прошдом поведении системы в сжатом виде последовательности принимаемых решений в ответ на данные сенсоров. Вероятности же переходов, хранимые в памяти, позволяют следить за частотой появления принципиально новых, плохо классифицируемых примеров и если новые примеры поступают часто ,то принимать решение о дообучении системы. В этом случае считается, что новые данные относятся к новому классу и для классификации строится новое дерево. Приведенные эксперименты показывают эффективность процедуры адаптации с помощью аппарата деревьев решений.

Малышко В. В., Морозов В. А. Об одном подходе к решению задачи сопоставления с образцом на основе алгоритма Rete. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В статье предложены способы модификации алгоритма Rete сопоставления с образцом с целью повышения скорости его работы. Предложена новая структура Rete-сети, позволяющая исключить избыточные проверки. Предложены оптимизирующие алгоритмы компиляции Rete-сети для сокращения среднего числа производимых при сопоставлении проверок.

Ил.: 5.

Библиогр.: 4 назв.

Громыко В. И., Мальковский М.Г., Кузина Л.Н., Симакин А.Г. Обучающие системы «компьютерного» образования в высшей школе. //Программные системы и инструменты. Тематический сборник №6, М.: Изд-во факультета ВМиК МГУ, 2005.

Л.Королев, обращаясь к подготовке специалистов высшей школы в области компьютерных наук, приводит доводы о недопустимости сокращения объема математической подготовки и особом значении фундаментальности в этой подготовке. Данная статья развивает тему фундаментальности, доводя исследование до определения математических тем базового обучения информатике. В этой связи обсуждается также неизбежность использования специально реализуемой обучающей системы (интеллектуальной по Д.Поспелову) и предлагается подходящий для возникших условий метод обучения.

Ил.:1

Табл.:2

Библиогр.: 18 назв.

Ершов Н.М. Бисекция ориентированных графов //Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В настоящей статье предлагается новый подход к решению задачи бисекции ориентированных графов, основанный на специальной модификации заданного графа в неориентированный граф и последующего решения задачи бисекции неориентированного графа с использованием программной системы Chaco. Также предложен метод наискорейшего спуска для решения задачи минимизации квадратичной формы на множестве векторов, состоящих из ± 1, к которой сводится, в частности, и задача бисекции ориентированных графов. Этот метод используется в данной работе, как завершающая процедура в предложенном подходе к бисекции орграфов, так и для тестирования

самого подхода. Приведены результаты численных расчетов, показывающие эффективность предложенной схемы.

Ил.: 1

Библиогр.: 6 назв.

Трошин С. В. Подсистема консолидации логов системы обнаружения вторжений. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В статье рассматриваются вопросы сбора исходных данных из журналов регистрации системами обнаружения вторжений: основные подходы к определению набора лог файлов и интересующих систему параметров. Рассмотрены методы промежуточного представления и передачи данных. Предложены основные концепции архитектуры системы консолидации записей системных журналов регистрации. Проведен анализ существующих технологий с целью выбора оптимальной для реализации системы.

Ил.: 1

Таб.: 1

Библиогр.: 7 назв.

Розинкин А.Н. Модели представления электронных писем в обучаемых системах классификации электронной почты. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

Статья посвящается разработке модели представления электронных писем в обучаемых системах классификации электронной почты. В частности, одной из основных проблем при разработке статистических методов классификации – сокращению размерности пространства признаков, характеризующих представление документа. В статье рассматриваются различные методы уменьшения пространства признаков, даются оценки применимости данных методов к рассматриваемой задаче. В статье предлагается модифицированный метод выбора оптимальных признаков, приводятся результаты экспериментов, показывающих эффективность предлагаемого решения.

Ил.: 1

Библиогр.: 10 назв.

Розинкин А.Н. Оптимизация тренировочного набора для системы классификации электронной почты на основе алгоритма опорных векторов. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В статье рассматривается задача формирования тренировочного набора для метода опорных векторов, применяемого в системе классификации электронной почты. Рассматривается оригинальный

метод уменьшения тренировочного набора на основе предварительной кластеризации набора особого вида, представляются результаты экспериментов, показывающих эффективность предлагаемого решения.

Ил.: 3

Библиогр.: 7 назв.

Певцов С.Е., Попов А.М. Разработка алгоритма восстановления структуры пептидов по масс-спектру. // Программные системы и инструменты. Тематический сборник №6, М.: Изд-во факультета ВМиК МГУ, 2005.

В настоящей статье описывается алгоритм, позволяющий определить последовательность аминокислот в пептиде путем анализа его масс-спектра. Алгоритм основан на рекурсивном поиске путей в спектральном графе. Показано, что важнейшую роль в получении искомого результата играет способ оценки правильности полученных последовательностей.

Ил.: 2

Табл.: 1

Библиогр.: 16 назв.

Попова Е.А. «Локализация нейронных источников электрической активности мозга с помощью метода Random Forest» // Программные системы и инструменты. Тематический сборник №6, М.: Изд-во факультета ВМиК МГУ, 2005.

В настоящей работе предложен алгоритм анализ сигналов ЭЭГ на основе метода Random Forest . В рамках дипольной модели источников электрический потенциал измеренный по нескольким каналам на поверхности головы выражен через параметры дипольного источника - его координаты (пространственную локализацию) и вектор момента, которые составляют пространство признаков задачи.

Для каждого момента времени случайным образом несколько диполей располагаются внутри области и строится свое дерево решений классифицирующее набор параметров дипольных источников по уровню допустимой ошибки приближения ЭЭГ потенциалов. Классификация проходит без усечения ветвей на случано выбранном для каждого момента наборе признаков - параметров диполя. Зона локализация источника является одним из выходов деревьев классификации в каждом срезе по времени. Ансамбль деревьев решений ищет перекрытие зон локализации , полученных каждым деревом в отдельные моменты времени. Показано, что использование выборки признаков равной трем в каждом дереве, наиболее эффективно и устойчиво решает проблему локализации, при этом величина структурного риска при классификации на тестовых данных минимальна. Относительная погрешность определения зоны

локализации того же порядка как и априори задаваемая погрешность в восстановлении потенциала. Приводятся экспериментальные рассчеты , показывающие эффективность предложенного подхода.

Ил.: 4

Библиогр.: 16 назв.

Федулова И.А. Идентификация протеинов по аминокислотной последовательности, восстановленной из масс-спектра. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В статье предлагается алгоритм идентификации протеинов по аминокислотной последовательности, восстановленной из масс-спектра методом de novo секвенирования. В работе предложена функция расстояния между строками, робастная по отношению к некоторым типичным ошибкам de novo секвенирования. Предложен также новый метод оценки результатов приближенного поиска последовательностей по базе данных, основанный на непосредственном использовании масс-спектра идентифицируемого пептида. Разработанный метод был протестирован на 144 реальных масс-спектрах и правильно идентифицировал больше последовательностей, чем популярные программы SPIDER и CIDentify.

Ил.: 3

Библиогр.: 19 назв.

Савенков К.О. Использование зависимостей при масштабировании имитационных моделей. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В настоящей статье показывается, как понятие зависимостей, используемое при анализе последовательных программ, можно адаптировать для поддержки масштабирования имитационных моделей – приведения описания имитационной модели к определённому уровню абстракции с сохранением заданных свойств.

Помимо известных зависимостей по данным и управлению, позволяющих сохранить при преобразованиях поток данных и управления модели, также вводятся зависимости по времени выполнения и зависимость по исполнителю, позволяющие сохранить временные свойства имитационной модели.

В работе рассматриваются дискретно-событийные имитационные модели, построенные для среды DYANA.

Ил.: 1

Библиогр.: 7 назв.

Прус В. В. Метод оценки наихудшего времени выполнения для процессоров с конвейерной архитектурой. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В настоящей статье рассматриваются способы учета архитектуры процессора при оценке наихудшего времени выполнения программ (WCET). Предложено использовать новый, относительно простой метод описания процессора, который для определенного класса архитектур дает более точные оценки, по сравнению с существующим методом. На основании предложенного метода построен алгоритм оценки WCET, обладающий потактовой точностью и не уступающий по эффективности другим алгоритмам. В работе приведено доказательство корректности алгоритма, а также теоретический и экспериментальный анализ эффективности.

Ил.: 2

Библиогр.: 11 назв.

Беззубцев С.О., Ющенко Н.В. Использование адаптера Arinc429-3 в стенде полунатурного моделирования. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В настоящей работе описывается практика применения адаптера Arinc429-3 фирмы "Элкус" в составе стенда полунатурного моделирования. Адаптеры такого класса обычно обеспечивают приём и передачу данных по заранее подготовленным циклограммам, и оптимизированы под такой режим работы. Авторам удалось обеспечить работу адаптера для приёма и передачи отдельных слов в реальном времени моделирования.

Библиогр.: 4 назв.

Захаров В.А., Коннов И.В. Об одном подходе к верификации асинхронных параметризованных систем. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В работе рассматривается задача верификации моделей распределённых асинхронных систем, параметризованных по числу изоморфных процессов.

Предлагается отношение симуляции, обладающее свойствами монотонности и сокрытия применительно к операции асинхронной параллельной композиции. Данный вид симуляции позволяет применить известные методы верификации моделей синхронных распределённых программ для асинхронного случая.

Ил.: 0

Библиогр.: 7 назв.

Глазкова В.В. Обзор современных методов моделирования и визуализации облаков в реальном времени. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В настоящей статье проведён сравнительный анализ двух методов моделирования погодных явлений в реальном времени: вокселного моделирование и моделирования на основе системы частиц. В статье приводится описание и анализ основных методов визуализации облаков в реальном времени. Для повышения реалистичности метода Харриса визуализации облаков на основе системы частиц предлагается адаптировать однопроходный метод Miyazaki, Dobashi, Nishita, обладающий лучшей производительностью в случае вокселного моделирования и учитывающий атмосферное рассеяние света.

Ил.: 5

Табл.: 1

Библиогр.: 8 назв.

Брусенцов Н.П., Владимирова Ю.С., Рамиль Альварес Х. Троичный логико-алгебраический и арифметический процессор // Программные системы и инструменты. Тематический сборник № 6, М.: Изд. отд. ф-та ВМиК МГУ, 2005.

Показана актуальность создания и практического использования троичных арифметического и логико-алгебраических процессоров ввиду неполноценности двоичной арифметики и неадекватности двухзначной логики.

Библиогр.: 11 назв.

Гурьев Д.Е., Демьянов П.Ю., Лызлов В.Е., Миронов Н.Ю., Харин В.А., Чихичин Д.А. Устройства сопряжения мультиплексного канала обмена (MIL-STD-1553) и их программное обеспечение. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

Рассматриваются цели, задачи, достигнутые результаты и ближайшие перспективы проекта по разработке аппаратного и програмного обеспечения устройств сопряжения мультиплексного канала обмена бортовых сетей.

Библиогр.: 13 назв.

Гурьев Д.Е., Миронов Н.Ю., Харин В.А., Чихичин Д.А. Тестовое обеспечение связных машин мультиплексного канала обмена. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

Настоящая статья посвящена проблемам тестирования универсальных связных машин (МСУ) мультиплексного канала обмена (МКО), выполненного согласно требованиям ГОСТ Р 52070-2003 / MIL-STD-1553B, а также различных устройств сопряжения с МКО (адаптеров), построенных на основе МСУ и используемых в наземных комплексах отладки бортовых систем управления. Представлены основные результаты и некоторые планы ближайшей перспективы.

Библиогр.: 9 назв.

Афанасьев М.К. Алгоритм динамического распределения данных на основании анализа запросов узлов // Программные системы и инструменты. Тематический сборник №6, М.: Изд-во факультета ВМиК МГУ, 2005.

В современных хранилищах данных часто возникает ситуация при которой информация оказывается распределена по узлам сети неэффективно. Как результат, при обращении к данным конечными пользователями задержки оказывается очень большими. В работе предлагается способ решения данной проблемы - алгоритм оптимального распределения данных в компьютерных сетях с использованием генетических алгоритмов.

Ил.: 4

Библиогр.: 14 назв.

Мещеряков Д.К. Исследование свойств случайных чисел, получаемых с использованием примитивов ОС. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В работе описывается эксперимент, проведенный в целях исследования статистических свойств случайных чисел, получаемых с использованием примитивов ОС. Приведены результаты эксперимента и архитектура, позволяющая в рамках многопоточной модели выполнения осуществлять генерацию случайных чисел одновременно с работой вычислительно интенсивных программ.

Табл.: 2

Библиогр.: 5 назв.

Сидорова Ю.А. Распознавание эмоций по речи: с позиций оптимизации человеко-машинного интерфейса. // Программные системы и инструменты. Тематический сборник № 6, М.: Изд-во факультета ВМиК МГУ, 2005.

В настоящей статье рассматривается конструирование агентов (программ) человеко-машинного интерфейса, чувствительных к речевой эмоции пользователя. В статье приводится мотивация конструирования подобных агентов, обзор текущего положения дел в области,

альтернативы компонентов решения и проблемы, с которыми с столкнется разработчик, решивший конструировать такого агента. Также предложен новый классификатор и данные по его тестированию, в статье рассказывается о первой попытке создания агента, чувствительного к эмоциям в русской речи.

Таб.: 2

Библиогр.: 11 назв.

ПРОГРАММНЫЕ СИСТЕМЫ И ИНСТРУМЕНТЫ

Тематический сборник

№ 6

Под общей редакцией
чл.-корр. РАН Л.Н. Королева

Издательский отдел
Факультета вычислительной математики и кибернетики
МГУ им. М.В. Ломоносова
Лицензия ИД N 05899 от 24.09.01 г.

119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,
2-й учебный корпус

Напечатано с готового оригинал-макета
в издательстве ООО "МАКС Пресс"
Лицензия ИД N 00510 от 01.12.99 г.
Подписано к печати 27.12.2005 г.
Формат 60x90 1/16. Усл.печ.л. 15,0. Тираж 100 экз. Заказ 933.

119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,
2-й учебный корпус, 627 к.
Тел. 939-3890, 939-3891. Тел./Факс 939-3891