ПРОБЛЕМЫ ИНТЕЛЛЕКТУАЛИЗАЦИИ И КАЧЕСТВА СИСТЕМ ИНФОРМАТИКИ

Серия
“КОНСТРУИРОВАНИЕ И ОПТИМИЗАЦИЯ ПРОГРАММ”

Под редакцией
доктора физ.-мат. наук, профессора, чл.-корр. РАЕН
В. Н. Касьянова

Выпуски серии:

1. Смешанные вычисления и преобразование программ (1991)
2. Конструирование и оптимизация программ (1993)
3. Интеллектуализация и качество программного обеспечения (1994)
4. Проблемы конструирования эффективных и надежных программ (1995)
5. Оптимизирующая трансляция и конструирование программ (1997)
6. Проблемы систем информатики и программирования (1999)
7. Поддержка супервычислений и Интернет-ориентированные технологии (2001)
8. Касьянов В. Н., Мирзуитова И. Л. Slicing: срезы программ и их использование (2002)
9. Современные проблемы конструирования программ (2002)
10. Новые информационные технологии в науке и образовании (2003)
11. Программные средства и математические основы информатики (2004)
12. Методы и инструменты конструирования и оптимизации программ (2005)
13. ***Проблемы интеллектуализации и качества систем информатики***

Российская академия наук
Сибирское отделение
Институт систем информатики
имени А. П. Ершова

ПРОБЛЕМЫ ИНТЕЛЛЕКТУАЛИЗАЦИИ И КАЧЕСТВА СИСТЕМ ИНФОРМАТИКИ

Под редакцией
проф. Виктора Николаевича Касьянова

Новосибирск 2006

УДК 519.68; 681.3.06
ББК З 22.183.49+ З 22.174.2

Проблемы интеллектуализации и качества систем информатики. — Новосибирск: Ин-т систем информатики имени А. П. Ершова СО РАН, 2006. — 280 с.

Является тринадцатым в серии сборников, издаваемых Институтом систем информатики имени А.П.Ершова СО РАН. Описывает проблемы интеллектуализации и качества систем информатики.

Сборник представляет интерес для системных программистов, а также студентов и аспирантов, специализирующихся в области системного и теоретического программирования.

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

PROBLEMS OF INTELLECTUALIZATION AND QUALITY OF INFORMATICS SYSTEMS

Edited by
prof. V. N. Kasyanov

Novosibirsk 2006

This volume is the thirteenth one in a series of books published in A.P. Ershov Institute of Informatics Systems. This volume is devoted to the tools and techniques of program construction and optimization.

The volume is of interest for system programmers, students and post-graduates working in the field of system and theoretical programming.

ПРЕДИСЛОВИЕ РЕДАКТОРА

Тринадцатый выпуск серии «Конструирование и оптимизация программ» посвящен решению актуальных проблем интеллектуализации и качества систем информатики.

Продолжая уже сложившиеся традиции, данный выпуск, как и предыдущие, базируется на результатах исследований, выполненных в лаборатории по конструированию и оптимизации программ Института систем информатики имени А.П. Ершова СО РАН совместно с Новосибирским государственным университетом при финансовой поддержке Российского фонда фундаментальных исследований, Российского гуманитарного научного фонда, Министерства образования и науки Российской Федерации, а также компании «Микрософт». Объединяет статьи, составившие сборник, также то, что все они подготовлены по результатам исследований, входящих в завершающийся в этом году трехгодичный проект 3.1.5. «Методы и средства трансляции и конструирования программ» программы 3.1. «Информационное и математическое моделирование в различных областях знаний, задачи поддержки принятия решений, экспертные системы, системное и теоретическое программирование» фундаментальных и ориентированных фундаментальных исследований СО РАН.

Открывает сборник статья Р. Н. Арапбаева и Р. А. Осмонова, посвященная анализу зависимостей по данным для многомерных массивов на базе модифицированного λ-теста.

Статья Т.В. Батуры и Ф.А. Мурзина исследует задачу обработки поисковых запросов на естественном языке с помощью REFAL-подобных конструкций.

Совместная статья группы авторов (А. А.Добрынин, Л. С. Мельников, Х. Вальтер, Й. Шрейер) исследует число косых полиэдральных графов с малым числом вершин.

В статье А.А. Дунаева, Т.Ф. Валеева и Е.А. Тарасова описываются результаты исследований методов организации визуальной обратной связи в аппаратно-программном комплексе «Бослаб».

Статья А.А. Дунаева представляет проект исследовательской системы для анализа текстов на естественном языке.

В статье В.Н. Касьянова анализируются некоторые новые возможности, связанные с представлением музеев в сети Интернет и с появлением в сети так называемых виртуальных музеев.

Статья Е.В. Касьяновой описывает проект адаптивной системы поддержки дистанционного обучения программированию.

Статьи А.В. Козыревой рассматривают некоторые способы калибровки видеокамеры и вопросы определения координат мобильного устройства в пространстве на основе изображений, получаемых от его видеокамеры.

Статья Л.С. Мельникова и И.В. Петренко исследует путевые разбиения в неориентированных графах.

В статье Г.П. Несговоровой рассматриваются современные информационно-коммуникационные и цифровые технологии в сохранении культурного и научного наследия и развитии музейного дела.

Статья Р.А. Осмонова и Д.Н. Штокало описывает преобразования циклов, основанные на несингулярных матрицах.

В статьях А.Л. Серебренникова проводится сравнительный анализ нейросетевых пакетов и описывается место среди них, которое отводится автором для разрабатываемой им среды Significo. Делается обзор возможностей среды Significo на примере решения прикладной задачи.

Статья А. П. Стасенко является обзором потоковых языков программирования.

В статье А. С. Тараскиной описывается нечеткая кластеризация по модифицированному методу *c*-средних.

Статья Д.В. Шкурко представляет собой библиографический обзор одной из задач отказоустойчивости распределенных систем, получившей название задачи о соглашении византийских генералов.

В статье С.В. Юрьева описывается универсальная система построения и администрирования так называемых лабораторных веб-сайтов.

Проф. В.Н. Касьянов

Р.Н. Арапбаев, Р.А. Осмонов

АНАЛИЗ ЗАВИСИМОСТЕЙ ПО ДАННЫМ ДЛЯ МНОГОМЕРНЫХ МАССИВОВ НА БАЗЕ МОДИФИЦИРОВАННОГО λ-ТЕСТА

ВВЕДЕНИЕ

Одним из основных средств перевода последовательных программ в параллельную форму являются автоматические распараллеливающие компиляторы. Основная их задача — извлечь как можно больше скрытого параллелизма из последовательной программы. Главным источником такого потенциального параллелизма, как правило, служит гнездо цикла. Извлечение скрытого параллелизма в первую очередь связано с анализом циклов и заключается в нахождении зависимости по данным между итерациями цикла.

Для решения этой проблемы компиляторы используют тесты на зависимость по данным [1]. На практике используются несколько известных алгоритмов анализа зависимости по данным. Например, НОД-тест, неравенства Банержи [2], I-тест (интервальный тест) [3, 4], Power-тест [5], Омега-тест [6], λ-тест [7] и др.

В настоящей работе предлагается новый модифицированный вариант λ-теста. В приведенном алгоритме λ-тест интегрирован с точным IR-тестом (“interval reduction”) [8], благодаря чему он показал более точные результаты при анализе зависимостей многомерных массивов.

В разд. 2 даны основные определения, в разд. 3 представлены идеи λ-теста и IR-теста. В разд. 4 подробно описывается алгоритм модифицированного λ-теста и производится сравнение его результатов с результатами наиболее известных тестов на зависимость. В разд. 5 приводится заключение о проделанной работе и список литературы по данной тематике.

Все понятия, не определяемые в этой работе, могут быть найдены в [1].

1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Рассмотрим гнездо циклов с индексами $i_1, i_2, \ldots, i_n$:

for $i_1 = L_1$ to U_1
 for $i_2 = L_2$ to U_2
 $\vdots$
 for $i_r = L_r$ to U_r
S_1: A[$f_1(i_1, i_2, \ldots, i_r)$, $f_2(i_1, i_2, \ldots, i_r)$, …, $f_d(i_1, i_2, \ldots, i_r)$] = . . .
S_2: ...=A[$f'_1(i_1, i_2, \ldots, i_r)$, $f'_2(i_1, i_2, \ldots, i_r)$, …, $f'_d(i_1, i_2, \ldots, i_r)$]
 endfor
 $\vdots$
 endfor
endfor

где **d —** размерность массива, **r —** количество вложенных циклов.

Определение 1. Между двумя операторами $\mathbf{S_1}$ и $\mathbf{S_2}$ существует зависимость, если они оба обращаются к одной и той же ячейке памяти, и, по крайней мере, одно из этих обращений есть *запись* [1].

Компиляторы применяют тесты на зависимость, чтобы определить, обращаются ли операторы $\mathbf{S_1}$ и $\mathbf{S_2}$ к одному и тому же элементу массива **A**. Для этого тест должен учитывать не только индексные выражения $f_1(), f_2(),\ldots,f_d(), f'_1(), f'_2(),\ldots, f'_d()$, но также границы индексных переменных $L_1, L_2 \ldots, L_n, U_1, U_2 \ldots, U_n$. Таким образом, тесты на зависимость должны определять, есть ли целочисленные решения $i_1, i_2 \ldots, i_n$ системы линейных диофантовых уравнений (1), удовлетворяющие ограничениям (2):

$$\begin{cases} f_1(i_1, i_2, \ldots, i_r) = f'_1(i_{r+1}, i_{r+2}, \ldots, i_n) \\ f_2(i_1, i_2, \ldots, i_r) = f'_2(i_{r+1}, i_{r+2}, \ldots, i_n) \\ \quad\vdots \qquad\quad \vdots \qquad\quad \vdots \\ f_d(i_1, i_2, \ldots, i_r) = f'_d(i_{r+1}, i_{r+2}, \ldots, i_n) \end{cases} \tag{1.1}$$

$$L_p \le i_p \le U_p, \text{ где } p=1,\ldots,n. \tag{1.2}$$

Ключевой проблемой при анализе зависимостей в цикле является работа с массивами. Если имеется m-мерный массив **A** и индексные выражения

массива линейны, то тогда система (1.1) может быть записана в следующем виде:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + c_1 = 0 \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + c_2 = 0 \\ &\qquad\qquad \ldots\ldots\ldots\ldots \\ &a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + c_m = 0 \end{aligned} \tag{1.3}$$

и

$$L_i \le x_i \le U_i \quad \text{где } i = 1,\ldots,n \tag{1.4}$$

Следовательно, проблема зависимости представляет собой задачу целочисленного программирования.

Общий подход состоит в индивидуальном тестировании уравнений (тестирование “индекс-за-индексом”) из (1.3) вместо проверки существования решения системы в целом. Но система уравнений зависимости может не иметь решения даже в том случае, когда имеются решения в каждом из отдельных уравнений.

Определение 2. Будем говорить, что индексные выражения *сцепленные* (“coupled”), если они включают в себя одинаковые индексные переменные цикла.

Если потенциальная зависимость включает *сцепленные* индексы, то для её разрушения необходимо одновременное рассмотрение индексов многомерного массива. Заметим, что среди всего множества тестов только некоторые из них пригодны для работы со *сцепленными* индексами. Например, обобщенный НОД-тест и тесты на основе линейного и целочисленного программирования: целочисленной тест, Power-тест, Омега-тест, и др. В первом даётся ответ на существование целочисленного решения, но не учитываются ограничения на область изменения переменных. Поэтому обобщенный НОД-тест не может доказать существование зависимости, но полезен для ее опровержения. Тесты, использующие дорогостоящие методы, неэффективны на практике. Экспериментальные результаты показали, что метод исключения переменных Фурье—Моцкина выполняется в 22-28 раз дольше, чем при использовании более простых методов [1].

Согласно эмпирическому исследованию в [9], сцепленные индексы часто встречаются в реальных программах. Из всех исследованных массивов 36.23% составляют ссылки двухмерных массивов и 7% — ссылки трехмерных массивов. Доля ссылок выше трехмерного массива незначительна. Более чем в четырех тысячах пар двухмерных ссылок массива приблизитель-

но 46% являются сцепленными индексными выражениями. Что касается ссылок массива большей размерности, то только 2% являются сцепленными индексными выражениями. Поэтому на практике важно иметь эффективный тест для обработки сцепленных индексов, особенно для анализа ссылок двухмерного массива. Один из таких эффективных алгоритмов, называемый λ-тестом, предложен в работе [7].

2. ИСПОЛЬЗУЕМЫЕ ТЕСТЫ

2.1. λ-тест

λ-тест исследует систему уравнений (1.3) и неравенства (1.4) и определяет, имеет ли система действительные решения [7].

Геометрически каждое линейное уравнение в (1.3) представляет собой гиперплоскость π в пространстве $\mathbf{R^n}$. Пересечение **m** гиперплоскостей ***S*** соответствует общим решениям системы (1.3). Очевидно, если ***S*** пусто, то не имеется никакой зависимости по данным. Границы циклов соответствуют ограниченному выпуклому множеству **V** в $\mathbf{R^n}$. Уравнение имеет действительное решение, удовлетворяющее границам циклов и направлениям зависимостей, тогда и только тогда, когда соответствующая уравнению гиперплоскость π пересекается с **V**. Тестирование «индекс-за-индексом» определяет, пересекается ли каждая гиперплоскость π с **V**. Необходимо определить, пересекается ли само **S** с **V**. Если из всех гиперплоскостей найдется такая гиперплоскость, которая не пересекает **V**, то очевидно **S** не может пересекаться с **V**. Однако, даже если каждая гиперплоскость из (1.3) пересекает **V**, существует вероятность, что **S** не пересечет **V**.

На рис. 1 π_1 и π_2 — гиперплоскости, представляющие два уравнения системы, каждая из которых пересекается с **V**. Предположим, что пересечение π_1 и π_2 находится вне **V**. Если можно найти новую гиперплоскость, которая содержит **S**, но не пересекает **V**, то это доказывает, что **S** и **V** не пересекаются. На рис. 1 π' является такой новой гиперплоскостью. Следующая теорема доказывает, что если **S** и **V** не имеют пересечения, то имеет место гиперплоскость в $\mathbf{R^n}$, которая содержит **S** и не пересекает **V**. Данная гиперплоскость является линейной комбинацией гиперплоскостей из (1.3). С другой стороны, если **S** и **V** пересекутся, то никакая такая линейная комбинация не существует.

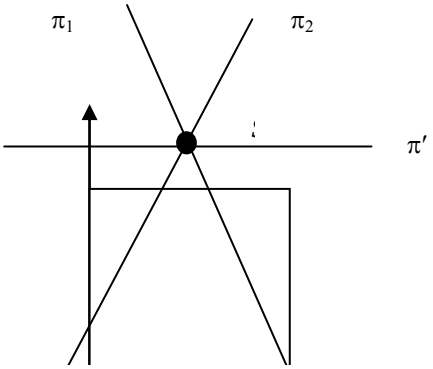


Рис. 1. Геометрическая иллюстрация λ-теста

Теорема 1. $S \cap V = \varnothing$ тогда и только тогда, когда существует гиперплоскость π, определяемая линейной комбинацией уравнений из (1.3): $\left\langle \sum_{i=1}^{m} \lambda_i \vec{a}_i, \vec{x} \right\rangle + \sum_{i=1}^{m} \lambda_i c_i = 0$ такая, что $\pi \cap V = \varnothing$, где $< \vec{a}_i, \vec{x} >$ — скалярное произведение векторов $\vec{a}_i \equiv (a_{i1}\, a_{i2}, \ldots, a_{in})$ и $\vec{x} \equiv (x_1, x_2, \ldots x_n)$ [7].

Массив (λ_1, λ_2, ..., λ_m) в Теореме 1 определяет гиперплоскость, содержащую **S**. Имеется бесконечное число таких гиперплоскостей. Задача λ-теста — исследовать по мере необходимости некоторое количество гиперплоскостей для определения пересечения **S** и **V**. Согласно Теореме 3 из [7] в общем случае, λ-тест генерирует C_n^{m-1} таких гиперплоскостей, которые являются линейной комбинацией (1.3) и называются λ-плоскостями. Чтобы определить, пересекает ли каждая λ-плоскость **V**, применяется тест Банержи—Вульфа для каждой λ-плоскости. Если хотя бы одна из λ-плоскостей не пересекает **V**, тогда нет зависимости по данным. Если каждая λ-плоскость пересекает **V**, то λ-тест принимает решение о возможном существовании зависимости.

2.2. IR-тест

IR-тест находит целочисленные решения уравнения зависимости путем сокращения интервала решений переменных с многократным проектированием. Как только эффективный интервал решений какой-нибудь переменной сжимается к пустому, то это линейное диофантово уравнение не имеет целочисленного решения [8].

Для объяснения IR-теста введем некоторые понятия.

Пусть, a — целое число, тогда положительная часть числа a:
$a^+ = \max\{a,0\}$, отрицательная часть числа a: $a^- = \max\{-a,0\}$.

Пусть L,U — числа такие, что L≤U, тогда
$\min\{ax : L \le x \le U\} = a^+ L - a^- U$,
$\max\{ax : L \le x \le U\} = a^+ U - a^- L$.

Пусть $n \ge 1$, a_j, L_j, U_j — числа и $L_j \le U_j$ для всех $j \in [1{:}n]$, и $\sum_{1 \le j \le n} a_j x_j$ – линейная функция n переменных, тогда:

$$\min\left\{\sum_{1\le j\le n} a_j x_j : (x_1,...,x_n) \in \underset{1\le j\le n}{\times} [L_j : U_j]\right\} = \sum_{1\le j\le n} (a_j^+ L_j - a_j^- U_j),$$

$$\max\left\{\sum_{1\le j\le n} a_j x_j : (x_1,...,x_n) \in \underset{1\le j\le n}{\times} [L_j : U_j]\right\} = \sum_{1\le j\le n} (a_j^+ U_j - a_j^- L_j).$$

Поскольку линейную функцию двух переменных можно представить как линию в двухмерном пространстве, то говорят, что линия $ax + by + c = 0$ содержит в себе *целую точку* тогда и только тогда, когда существует целочисленная пара (x_1, y_1) такая, что $ax_1 + by_1 + c = 0$. В связи с этим определением имеем следующую лемму.

Лемма 1. Пусть $P_1(\lfloor x_1 \rfloor, y_1)$ и $P_2(\lceil x_1 \rceil, y_2)$ будут двумя точками на линии $ax + by + c = 0$, где x_1 — произвольное вещественное число, $\lfloor x_1 \rfloor$ — нижняя целая часть числа x_1, и $\lceil x_1 \rceil$ — верхняя целая часть числа x_1. Тогда отрезок $\overline{P_1P_2}$ не содержит целую точку тогда и только тогда, когда y_1 и y_2 являются нецелыми (рис. 2).

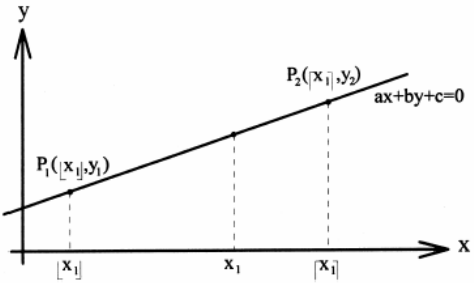


Рис. 2. Линейное уравнение в двухмерном пространстве

Следующая теорема является прямым обобщением вышеупомянутой леммы.

Теорема 2. Пусть $P_1(\lfloor x_1 \rfloor, x_2, \ldots, x_n)$ и $P_2(\lceil x_1 \rceil, x'_2, \ldots, x'_n)$ две точки на линии $\sum_{1 \le j \le n} a_j x_j + c = 0$ в n-мерном пространстве. Тогда отрезок $\overline{P_1P_2}$ не содержит целую точку тогда и только тогда, когда

(i) $\exists k \in [2{:}n]$ такое, что $x_k \notin Z$ и

(ii) $\exists m \in [2{:}n]$ такое, что $x'_m \notin Z$.

Чтобы более ясно показать идею IR-теста, сначала рассматривается случай, когда уравнение зависимости представляет собой диафонтово уравнение с двумя переменными.

Процедура теста в случае уравнения с двумя переменными приведена ниже.

$$a_1x_1 + a_2x_2 = a_0 \qquad (2.2.1)$$

при условии

$$l_1 \le x_1 \le u_1 \text{ и } l_2 \le x_2 \le u_2, \qquad (2.2.2)$$

где a_0, a_1, a_2 являются целыми константами, и x_1, x_2 — целочисленные переменные. Очевидно, что целочисленные решения будут размещаться внутри ограниченной прямоугольной области: $l_1 \le x_1 \le u_1$ и $l_2 \le x_2 \le u_2$.

Чтобы найти целочисленные решения, линия проектируется на ось x_1. Эффективным интервалом решений x_1 является пересечение интервала линии, спроектированной на ось x_1, с исходным интервалом $l_1 \le x_1 \le u_1$. Таким

образом, фактическим интервалом решений x_1 будет подмножество начального интервала ограничения.

Проектируемым интервалом линии $a_1x_1 + a_2x_2 = a_0$ на ось x_1 является:

$x_1 = -(a_2/a_1)\, x_2 + (a_0/a_1)$, где $l_2 \le x_2 \le u_2$.

Вычисляется предельное значение x_1:

$$\left(-\frac{a_2}{a_1}\right)^{+} l_2 - \left(-\frac{a_2}{a_1}\right)^{-} u_2 + \frac{a_0}{a_1} \le x_1 \le \left(-\frac{a_2}{a_1}\right)^{+} u_2 - \left(-\frac{a_2}{a_1}\right)^{-} l_2 + \frac{a_0}{a_1}.$$

Пусть

$$l_1^{(1)} = \left(-\frac{a_2}{a_1}\right)^{+} l_2 - \left(-\frac{a_2}{a_1}\right)^{-} u_2 + \frac{a_0}{a_1},$$

$$u_1^{(1)} = \left(-\frac{a_2}{a_1}\right)^{+} u_2 - \left(-\frac{a_2}{a_1}\right)^{-} l_2 + \frac{a_0}{a_1}$$

означает верхнюю и нижнюю границы x'_1 соответственно.

Фактический интервал решений x_1 :

$$\left[l_1^{(1)} : u_1^{(1)}\right] \cap \left[l_1 : u_1\right] = \left[l_1^{\prime(1)} : u_1^{\prime(1)}\right].$$

Так как $l_1^{\prime(1)}$ и $u_1^{\prime(1)}$ могут быть нецелочисленными по Лемме 1, и если уравнение (2.2.1) имеет целочисленные решения, то они должны принадлежать интервалу $[\lceil l_1^{\prime(1)} \rceil : \lfloor u_1^{\prime(1)} \rfloor]$. Если $\lceil l_1^{\prime(1)} \rceil > \lfloor u_1^{\prime(1)} \rfloor$, то интервал пуст и уравнение (2.2.1) с ограничениями (2.2.2) не имеет целочисленных решений. Иначе, так как интервал решений x_1 был сокращен (рис. 3), интервал решений x_2 изменится соответственно.

Аналогично, как описано выше, линия проектируется на ось x_2. Пусть сокращенный интервал решений x_2 будет $[\lceil l_2^{(1)} \rceil : \lfloor u_2^{(1)} \rfloor]$ (рис. 3). Если теперь $\lceil l_2^{\prime(1)} \rceil > \lfloor u_2^{\prime(1)} \rfloor$, то уравнение (2.2.1.) с ограничениями (2.2.2) не имеет целочисленных решений. Иначе линия снова проектируется на эту ось x_1 с целью сокращения интервала решений x_1.

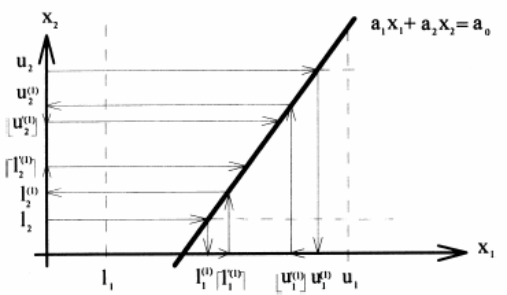


Рис. 3. Сокращение интервала решения

Повторив эту процедуру m раз, имеем:

$$x_1 \in [\lceil l_1'^{(m)} \rceil : \lfloor u_1'^{(m)} \rfloor],$$
$$x_2 \in [\lceil l_2'^{(m)} \rceil : \lfloor u_2'^{(m)} \rfloor].$$

Повторяя $(m + 1)$ раз, приходим к

$$x_1 \in [\lceil l_1'^{(m+1)} \rceil : \lfloor u_1'^{(m+1)} \rfloor],$$
$$x_2 \in [\lceil l_2'^{(m+1)} \rceil : \lfloor u_2'^{(m+1)} \rfloor].$$

В процедуре нахождения целочисленного решения уравнения (2.2.1) при условии ограничения (2.2.2), имеются следующие случаи, где IR-тест принимает соответствующие решения.

1. Если интервалы решений x_1 или x_2 были сокращены к пустым, т. е. $\lceil l_1'^{(m)} \rceil > \lfloor u_1'^{(m)} \rfloor$ или $\lceil l_2'^{(m)} \rceil > \lfloor u_2'^{(m)} \rfloor$, то уравнение (2.2.1) с ограничением (2.2.2) не имеет целочисленных решений, IR-тест показывает независимость по данным.
2. Если интервалы решений x_1 и x_2 остаются неизменными, т.е. $\lceil l_1'^{(m)} \rceil = \lceil l_1'^{(m+1)} \rceil, \lfloor u_1'^{(m)} \rfloor = \lfloor u_1'^{(m+1)} \rfloor$ и $\lceil l_2'^{(m)} \rceil = \lceil l_2'^{(m+1)} \rceil, \lfloor u_2'^{(m)} \rfloor = \lfloor u_2'^{(m+1)} \rfloor$, то уравнение (2.2.1) с ограничением (2.2.2) содержит, по крайней мере, одно целочисленное решение.

(a) Если оба интервала решений сократились к одной точке, т.е.
$\lceil l_1'^{(m)} \rceil = \lfloor u_1'^{(m)} \rfloor = \lceil l_1'^{(m+1)} \rceil = \lfloor u_1'^{(m+1)} \rfloor$ и
$\lceil l_2'^{(m)} \rceil = \lfloor u_2'^{(m)} \rfloor = \lceil l_2'^{(m+1)} \rceil = \lfloor u_2'^{(m+1)} \rfloor$,
то уравнение (2.2.1) с ограничением (2.2.2) имеет только одно целочисленное решение: $(x_1,x_2) = (\lceil l_1'^{(m)} \rceil, \lceil l_2'^{(m)} \rceil)$.

(b) Иначе, уравнение (2.2.1) с ограничением (2.2.2) содержит, по крайней мере, два целочисленных решения:
$\{(\lceil l_1'^{(m)} \rceil, \lceil l_2'^{(m)} \rceil), (\lfloor u_1'^{(m)} \rfloor, \lfloor u_2'^{(m)} \rfloor)\}$ если $(-a_1/a_2)>0$,
$\{(\lceil l_1'^{(m)} \rceil, \lfloor u_2'^{(m)} \rfloor), (\lfloor u_1'^{(m)} \rfloor, \lceil l_2'^{(m)} \rceil)\}$ если $(-a_1/a_2)<0$

Для нахождения других целочисленных решений в случае (2b), интервалы решений x_1 и x_2 уменьшаются, добавляя и вычитая единицу к нижней и верхней границам соответственно. Данные интервалы используются в качестве новых интервалов, и вышеупомянутая процедура повторяется до тех пор, пока один из интервалов не станет пустым. В конечном счете, все целочисленные решения могут быть найдены.

В случае уравнения, содержащего *n* переменных, тест решает уравнение зависимости, рекурсивно применяя для каждой переменной приведенный метод [8].

3. МОДИФИЦИРОВАННЫЙ λ–ТЕСТ

В этом разделе представлен алгоритм модифицированного λ-теста. Рассматривается проблема зависимости по данным при условии, что индексные выражения массивов линейны. Границы циклов рассматриваются как постоянные.

В модифицированном λ-тесте вместо Банержи-теста используется более мощный IR-тест. Поясним алгоритм при помощи геометрической иллюстрации. Геометрически каждое линейное уравнение в (1.3) представляет собой гиперплоскость π в пространстве $\mathbf{R^n}$. Пересечение **m** гиперплоскостей — ***S*** соответствует общим решениям всех уравнений (1.3). Очевидно, если ***S*** пусто, то не существует никакой зависимости по данным. Проверка, является ли ***S*** пустой, тривиальна в линейной алгебре. Поэтому далее рассматриваем только непустую ***S***. Границы циклов соответствуют ограниченному выпуклому множеству **V** в $\mathbf{R^n}$. Система уравнений (1.3) имеет действительное решение, удовлетворяющее границам циклов тогда и только тогда, когда ***S*** пересекается с **V**. λ-тест может ответить на этот вопрос.

Обычный λ-тест доказывает независимость по данным многомерных массивов, когда **S** не пресекает выпуклое множество **V,** в противном случае λ-тест принимает консервативное решение о существовании зависимости по данным. Банержи-тест не может определить, имеет ли λ-плоскость целочисленные точки пересечения с **V.**

Однако, даже если ***S*** пересекается с **V**, существует вероятность, что система уравнений (1.3) не имеет целочисленных решений в **V**.

Если можно найти новую гиперплоскость, которая содержит ***S*** и не имеет целочисленных точек пересечения с **V**, то это доказывает, что ***S*** не имеет целочисленных значений в **V**, следовательно, зависимости по данным не существует. Данная гиперплоскость является линейной комбинацией уравнений системы.

Теорема 3. $\mathbf{S}\cap\mathbf{V}$ не имеет целочисленных точек тогда и только тогда, когда существует гиперплоскость π, соответствующая линейной комбинации $\left\langle \sum_{i=1}^{m} \lambda_i \vec{a}_i, \vec{x} \right\rangle + \sum_{i=1}^{m} \lambda_i c_i = 0$ системы уравнений (1.3) такая, что $\pi\cap\mathbf{V}$ не имеет целочисленных точек, где $< \vec{a}_i, \vec{x} >$ — скалярное произведение $\vec{a}_i \equiv ($ $a_{i1}\, a_{i2}\, ,\ldots,\, a_{in})$ и $\vec{x} \equiv (x_1, x_2, \ldots x_n)$.

Предложенный алгоритм с помощью λ-теста генерирует множество линейных комбинаций гиперплоскостей. Затем применяется IR-тест для нахождения гиперплоскости из данного множества, которая не имеет целочисленных точек пересечения с **V**.

Алгоритм.

Вход: система уравнений (1.3) и ограничения (1.4), где *n* — количество переменных и *m* — количество уравнений системы.

Выход: **НЕТ ЗАВИСИМОСТИ:** **система** уравнений (1.3) с ограничениями (1.4) не имеет целочисленных решений,
или
ЕСТЬ ЗАВИСИМОСТЬ: система уравнений (1.3) с ограничениями (1.4) имеет целочисленные решения.

Метод:

функция М_Л_ТЕСТ =

1. **для** $i = 1,\ldots, C_n^{m-1}$ **цикл**
2. Вычислить (λ_1, λ_2, ..., λ_m);
3. Сгенерировать новую гиперплоскость.
4. Применить IR-тест к гиперплоскости.
5. **если** (IR-тест дает ответ **NO**), **то**
6. **возврат НЕТ ЗАВИСИМОСТИ;**
7. **все;**
8. **все;**
9. **возврат ЕСТЬ ЗАВИСИМОСТЬ;**

все.

3.1. Сравнение результатов

Проведем сравнение результатов алгоритма с результатами наиболее известных алгоритмов, таких как НОД-тест, Банержи-тест и λ-тест.

Рассмотрим пример:

```
for (i=1;i<=100; i++)
{
    for (j=1;j<=100; j++)
    {
S1:                 A[2*i][ 2*i +3*j]= A[i+j][j-i+6];
    }
}
```

В этом примере оператор **S1** обращается к элементам массива **A**. Если **S1** не имеет зависимости по данным, то оба цикла в примере можно распараллелить.

Экземпляры оператора **S1:** S1(x_1, x_2) и S1(x_3, x_4) будут обращаться к одной ячейке памяти тогда и только тогда, когда следующая система уравнений имеет целочисленные решения:

$$\begin{cases} 2x_1 - x_3 - x_4 = 0, \\ 2x_1 + 3x_2 + x_3 - x_4 = 6 \end{cases} \qquad (3.1.1)$$

где $1 \le x_1, x_3, x_2, x_4 \le 100$.

Сначала попытаемся разрушить потенциальную зависимость с помощью стандартных тестов. Отметим, что обычно на практике в многомерных массивах каждая размерность тестируется отдельно.

Если применить НОД-тест к первому уравнению (3.1.1), он показывает зависимость, поскольку НОД(2, 0, –1, –1) = 1 и 0 делится на единицу. Во втором уравнении (3.1.1) НОД также равен 1. НОД-тест быстрый тест, но на практике не эффективен, так как в большинстве случаев НОД(a_1, a_2, ...a_n) = 1.

Аналогично тест Банержи показывает зависимость, потому что оба уравнения имеют вещественное решение в области пространства итераций, т.е. –198 ≤ 0 ≤ 198 для первого уравнения (4.1) и –94 ≤ 6 ≤ 599 — для второго.

Как было выше отмечено, λ-тест предназначен для многомерных массивов. В случае, когда анализируются двухмерные массивы и учитываются только границы циклов, множество значений (λ_1, λ_2, ..., λ_m) вычисляется по нижеприведенному определению [7].

Определение 3. Дано уравнение вида $a\lambda_1 + b\lambda_2=0$, где *a, b* — одновременно не равны нулю, каноническое решение уравнения определяется следующим образом:

(λ_1, λ_2) = (1, 0), если $a = 0$;
(λ_1, λ_2) = (0, 1), если $b = 0$;
(λ_1, λ_2) = (b, $-a$), если ни один из *a, b* ненулевой и $b > 0$;
(λ_1, λ_2) = ($-b$, a), если ни один из *a, b* ненулевой и $b < 0$.

С помощью определения 3 вычисляем $\mathbf{\Lambda}$={(2, –2), (3, 0), (1, 1), (1, –1)}. Каноническое решение (2, –2) определяет λ-плоскость, которая является линейной комбинацией, и в результате тестирования λ-тест показывает зависимость по данным. Это означает, что гиперплоскость имеет вещественные решения в выпуклом множестве **V**, определяемом границами циклов. Аналогично все λ-плоскости тоже показывают зависимость по данным. Этот случай показывает, что λ-тест не всегда точен.

И наконец, проанализируем данный пример с помощью модернизированного λ-теста. Напомним, что этот алгоритм тоже определяет λ-плоскости, которые являются точной линейной комбинацией, но к ним применяется более точный IR-тест. В нашем примере первое каноническое решение (2, -2) определяет λ-плоскость, т.е. $-6x_2 - 4x_3 = -12$, где $1 \le x_2, x_3 \le 100$. Применим IR-тест к данной λ-плоскости. В результате сложений некоторые коэффициенты исключаются, и поэтому мы сокращаем

интервал решений только по осям x_2 и x_3 . Проектируя уравнение на ось x_2, получаем $x_2 = (-4/6)x_3 + (12/6)$, где $1 \le x_2 \le 100$, $1 \le x_3 \le 100$.

Верхняя и нижняя границы соответственно равняются:

$$l_1^{(1)} = (-4/6)^+(1) - (-4/6)^-(100) + (12/6) = (-388/6)$$
$$u_1^{(1)} = (-4/6)^+(100) - (-4/6)^-(1) + (12/6) = (8/6).$$

Эффективным интервалом решений x_2 становится:
$\left[(-388/6):(8/6)\right] \cap \left[1:100\right] = \left[1:8/6\right]$.

Отсюда, $x_2 \in [\lceil 1 \rceil : \lfloor 8/6 \rfloor] = [1:1]$, т.е. $l_1'^{(1)} = 1$, $u_1'^{(1)} = 1$.

Так как этот интервал не пуст, спроектируем уравнение $-6x_2 - 4x_3 = -12$ с интервалом $1 \le x_2 \le 1$ на ось x_3. Эффективным интервалом решений по x_3 является [6/4:6/4]. Так как $\lceil 6/4 \rceil > \lfloor 6/4 \rfloor$, то уравнение не имеет целочисленных решений в данном интервале, и оператор **S1** не имеет зависимости по данным.

Рассмотрим еще один пример:

```
        for (i=0; i<=10; i++)
        {
            for (j=0;j<=10; j++)
            {
S1:                     A[j-i+5][i-j+20]= i+j;
S2:                 C[i][j]= A[2*i+4*j][7*i+6*j+2];
            }
        }
```

Здесь, если не имеется никакой зависимости по данным между операторами **S1** и **S2**, то оба цикла в примере можно распараллелить. Пусть $x_1 = i$ и $x_2 = j$ для оператора **S1**, и $x_3 = i$ и $x_4 = j$ для **S2**. Уравнения зависимости выглядят следующим образом:

$$\begin{cases} -x_1 + x_2 - 2x_3 - 4x_4 = -5 \\ x_1 - x_2 - 7x_3 - 6x_4 = -18 \end{cases}, \qquad (3.1.2)$$

где $0 \le x_1, x_3, x_2, x_4 \le 10$. Зависимость по данным существует между **S1** и **S2** тогда и только тогда, когда система уравнений (3.1.2) имеет общие целочисленные решения в пределах границ цикла.

Для этого примера стандартные тесты показывают зависимость по данным. Модифицированный тест разрушает потенциальную зависимость. В нашем примере первое каноническое решение (1, 1) определяет λ-плоскость, т.е. $-9x_3 - 10x_4 = -23$, где $0 \le x_3, x_4 \le 10$. Применяем IR-тест к данной λ-плоскости. В результате сложения некоторые коэффициенты исключаются, и поэтому интервал решений сокращается только по осям x_3 и x_4. Проектируя уравнение на ось x_3, получаем $x_3 = (-10/9)x_4 - (23/9)$, где $0 \le x_3 \le 10$, $0 \le x_4 \le 10$.

Верхняя и нижняя границы соответственно равняются:

$$l_1^{(1)} = (-10/9)^+(0)_2 - (-10/9)^-(10) + (23/9) = (-77/9)$$

$$u_1^{(1)} = (-10/9)^+(10) - (-10/9)^-(0) + (23/9) = (23/9).$$

Эффективный интервал решений x_3:

$$\left[(-77/9):(23/9)\right] \cap \left[0:10\right] = \left[0:23/9\right].$$

Отсюда $x_3 \in [\lceil 0 \rceil : \lfloor 23/9 \rfloor] = [0{:}2]$, т.е. $l_1'^{(1)} = 0$, $u_1'^{(1)} = 2$.

Так как этот интервал не пуст, спроектируем уравнение $-9x_3 - 10x_4 = -23$ с интервалом $0 \le x_3 \le 2$ на ось x_4. Выполняя алгоритм IR-теста шаг за шагом, на некотором шаге получаем интервал решений по x_3: $l_1'^{(1)} = 2$, $u_1'^{(1)} = 1$. Это означает, что уравнение не имеет целочисленных решений в данном интервале, и операторы **S1** и **S2** не имеют зависимости по данным.

Таким образом, алгоритм модифицированного λ-теста может разрушить ложные зависимости, где обычный λ-тест принимает консервативное решение о существовании зависимости по данным.

3.2. Временная сложность

Модифицированный λ-тест включает в себя следующие два этапа: (1) вычисление значений λ и (2) тестирование каждой λ-плоскости. По определению 3, вычисление значений λ имеет временную сложность O(y), где y — константа [4]. Сгенерированная λ-плоскость, тестируется **IR-**тестом. Наихудшая временная сложность **IR-**теста: O(kz), где $k = \min\{u_i - l_i + 1 : 1 \le I \le z\}$, z — число переменных в сгенерированной λ-плоскости [8]. Следовательно,

модифицированный λ-тест имеет, в общем случае, наихудшую временную сложностью O(C_n^{m-1} * (k$z + y$)).

В случае анализа двухмерного массива, по Теореме 2 из [7], временная сложность модифицированного λ-теста: O(n * ($kz + y$)).

ЗАКЛЮЧЕНИЕ

При распараллеливании программы одной из основных проблем является выявление зависимости по данным. Особенно вызывает трудности анализ многомерных массивов. В обычной практике каждая размерность массивов тестируется индивидуально. Но имеются алгоритмы, которые специально предназначены для многомерных массивов, например λ-тест.

В данной работе представлен алгоритм модифицированного λ-теста, в котором λ-тест интегрирован с точным IR-тестом. Экспериментальные сравнения результатов показали, что модифицированный алгоритм более точен по сравнению с НОД, Банержи и λ-тестами на зависимость. Таким образом, модифицированный λ-тест является точным тестом для выявления зависимости по данным в многомерных массивах, содержащих сцепленные индексы.

В данное время алгоритм применяется только в тех ситуациях, где границы циклов постоянные и индексные выражения линейны. Дальнейшей целью является расширение теста с учетом более общих критериев, в которых границы циклов являются функциями индексов внешних циклов, а также тестирование зависимости по всем измерениям векторов направлений.

Практическим результатом данной работы является тест, который может быть использован в блоке анализа зависимостей по данным в проектируемой системе быстрого прототипирования компилятора.

СПИСОК ЛИТЕРАТУРЫ

1. Евстигнеев В.А. Анализ зависимостей: состояние проблемы // Системная информатика: Сб. науч. тр. — Новосибирск: Наука, 2000. — Вып. 7. — С. 112–173.
2. Banerjee U. Loop transformations for restructuring compilers: the Foundations. — Boston: Kluwer Academic Publishers, 1993.
3. Kong X., Klappholz D., and Pssaris K. The I Test: An Improved Dependence Test for Automatic Parallelization and Vectorization // IEEE Transactions on Parallel and Distributed Systems. — 1991. — Vol. 2(3) . — P. 342–349.

4. Chang, W.-L., Chu, C.-P., Wu, J. A multi-dimensional version of the I test // Parallel Computing. — 2001. — Vol. 27. — P. 1783–1799.
5. Wolfe M., and Tseng C. The Power Test for Data Dependence // IEEE Transactions on Parallel and Distributed Systems. September 1992.
6. Pugh W. The Omega test: a fast and practical integer programming algorithm for dependence analysis // Communications of the ACM. — 1992. — Vol. 35(8) . — P.102–114.
7. Li Z., Yew P.-C., Zhu C.-Q. An efficient data dependence analysis for parallelzing compilers // IEEE Transaction on Parallel and Distributed Systems. — 1990. — Vol. 1(1) . — P. 26–34.
8. Huang T.-C., Yang C.-M. Data dependence analysis for array references // J. of Systems and Software. — 2000. — Vol. 52. — P. 55–65.
9. Shen Z., Li Z., Yew P.-C. An empirical study of Fortran programs for parallelizing compilers // IEEE Transaction on Parallel and Distributed Systems. — 1992. — Vol. 1 (3) . — P. 356–364

Т.В. Батура, Ф.А. Мурзин

ОБРАБОТКА ПОИСКОВЫХ ЗАПРОСОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ С ПОМОЩЬЮ REFAL-ПОДОБНЫХ КОНСТРУКЦИЙ

В статье кратко обосновывается возможность применения модификаций конструкций языка символьных преобразований REFAL для формирования деревообразного представления предложений на естественном языке и схем «вопрос-ответ» и описан алгоритм использования их в поисковых системах. В действительности, сейчас имеется большой список, более сорока схем типа «вопрос-ответ», которые могут быть полезны при реализации программных систем, ориентированных на обработку текстов.

1. КОНСТРУКЦИИ ЯЗЫКА REFAL

Язык программирования REFAL является одним из языков, созданных для проведения символьных преобразований на компьютерах [1]. Это типичный язык продукций, и он аналогичен языку SNOBOL. Его операторы представляют собой продукции вида $\varphi \to \psi$, которые обозначают, что если слово имеет свойство φ, то необходимо применить действие ψ. Отметим, что так называемый функциональный стиль естественным образом присущ языку REFAL. Ниже дано формальное описание синтаксических свойств этого языка, описана виртуальная REFAL-машина, и даны некоторые примеры программ.

Предположим, что зафиксированы: T — алфавит объектных символов, Q — алфавит вспомогательных символов (например, / , $\perp$, $\to$, скобки, двоеточие, запятая), $T \cap Q = \varnothing$ и F — алфавит функциональных символов.

Имеется три типа переменных s_i, e_i, $i \in \omega$, где ω — множество натуральных чисел.

Если S — произвольный алгоритм, то через S^* обозначим множество всех слов над S, включая пустое слово.

1. Формула языка REFAL определяется индуктивно:

a) переменная является формулой,

b) любое $t \in T$ является формулой,

c) если φ — формула, то (φ) — формула,

d) если φ, ψ — формулы, то их конкатенация $\varphi\psi$ — формула,

e) если φ — формула и $f \in F$, то $/ f / \varphi \perp$ — формула,

f) других формул нет.

Если формула φ получена без применения правила e), то мы назовем ее простой. Множество всех переменных, входящих в формулу φ, обозначим var (φ).

2. Оператором называется слово $\varphi \to \psi$, где φ, ψ — формулы, при этом φ — простая и $\text{var}(\varphi) \supseteq \text{var}(\psi)$.

3. Подпрограммой называется любой столбец, имеющий вид:

$$f : L_1$$
$$\vdots$$
$$L_n,$$

где $f \in F, L_1, ..., L_n$ — операторы, $n \in \omega$. При этом, f называется именем подпрограммы.

4. Программа есть столбец вида:

$$F_1$$
$$\vdots$$
$$F_m,$$

где F_i — подпрограммы, $m \in \omega$.

5. Пусть φ — простая формула, $t \in S^*$, где $S = T \cup \{(,)\}$. Определим функцию i, мы будем называть ее функцией отождествления. Функция i паре $<\varphi, t>$ сопоставляет кортеж, с описанными ниже свойствами, если такой кортеж существует. В противном случае она сопоставляет 0. В этом случае говорят, что отождествление невозможно.

Пусть $\varphi = x_1 ... x_k$ — формула, и для любого j выполнено $x_j \in S$, либо x_j — переменная. Мы полагаем $i(\varphi, t) =< c_1, ..., c_m >$, если $k = m$ и выполнены свойства:

a) $x_j \in S \to c_j = x_j$,

b) $x_j = s_i \to c_j \in T$,

$x_j = e_i \to c_j \in S^*$,

c) $x_j = x_i \to c_j = c_i$,

d) любое c_j имеет правильно построенную скобочную структуру,

e) если $< c'_1 , ..., c'_k >$ — произвольный кортеж со свойствами (a)–(d), то

$$< | c_1 |, ... , | c_k |> \leq < | c'_1 |, ... , | c'_k |>$$

в лексикографическом порядке, где $| c_j |, | c'_j |$ — длины слов c_j , c'_j соответственно.

В итоге мы можем сделать следующие замечания:

- функция i используется, как функция отождествления с образцом,
- переменные типа s_i служат для обозначения символов,
- переменные типа e_i служат для обозначения выражений,
- скобки ( ) используются для того, чтобы фиксировать синтаксическую структуру строк.

6. REFAL-машина состоит из поля зрения и поля памяти и работает в дискретном времени. В текущий момент у нее в поле зрения находится формула, не содержащая переменных, а в поле памяти — программа. Опишем шаг работы машины.

Если в поле зрения находится простая формула, то процесс вычисления на этом заканчивается, и это считается нормальным окончанием.

В противном случае в формуле из поля зрения выделяется самая левая подформула, имеющая вид $/ f / \theta \perp$, такая, что θ простая. После этого отыскивается в программе подпрограмма с именем f. Если такой нет или их несколько, то в поле памяти появляется диагностика этого, и работа машины заканчивается.

В противном случае в подпрограмме с именем f отыскивается самый первый оператор $\varphi \to \psi$ такой, что $i(\varphi, 0) \neq 0$. Если такого оператора нет, то в поле памяти появляется информация об этом, и работа заканчивается.

Допустим теперь, что требуемый оператор имеется. Тогда подформула $/ f / \theta \perp$ в поле зрения заменяется на ψ *. Слово ψ * получается из формулы ψ заменой переменных на значения, которые они получили при отождествлении ψ и θ. Значениями переменных называются c_j из пункта b) в определении функции отождествления.

Рассмотрим теперь несколько примеров. Они показывают, что, используя скобочные структуры специального вида, можно очень коротко записывать довольно сложные программы.

Пример. Пусть в поле зрения REFAL-машины содержится

$/\, f\, /(+a-b+\sin(x))\ \perp$,

а в поле памяти содержится программа

$f : (s_0 e_1 + e_2) \to /\, f\, /(s_0 e_1)\perp /\, f\, /(+e_2)\perp$

$(s_0 e_1 - e_2) \to /\, f\, /(s_0 e_1)\perp /\, f\, /(-e_2)\perp$

$e_0 \to e_0$.

Пусть теперь

$\varphi = (s_0 e_1 + e_2),\ \theta = (+a-b+\sin(x))$.

Тогда имеем

$i(\varphi,\theta) =< (, +, a-b, +, \sin(x),) >$.

Это может быть наглядно представлено в виде

(	s_0	e_1	+	e_2	)
↓	↓	⇓	↓	⇓	↓
(	+	$a-b$	+	$\sin(x)$	).

В этой схеме скобки переходят в скобки, знаку «плюс» соответствует знак «плюс». Переменная s_0 имеет значение +, переменная e_1 имеет значение $a-b$, переменная e_2 имеет значение $\sin(x)$. Поэтому сначала должен быть выполнен первый оператор. В итоге мы получаем в поле зрения REFAL-машины $/\, f\, /(+a-b)\perp /\, f\, /(+\sin(x))\perp$.

Затем исполняется левое вхождение f. Очевидно, что

$$i((s_0 e_1 + e_2),(+a-b)) = 0\,.$$

Поэтому первый оператор программы не может сработать.

Далее имеем

$$i((s_0 e_1 - e_2),(+a-b)) =< (, +, a, -, b,) >,$$

и выполняется второй оператор. Мы можем представить функцию i в виде

(	s_0	e_1	−	e_2	)
↓	↓	↓	↓	↓	↓
(	+	a	−	b	).

После второго шага мы будем иметь в поле зрения:

$$/\, f\, /(+a)\perp /\, f\, /(-b)\perp /\, f\, /(+\sin(x))\perp.$$

Далее может работать только третий оператор. Все дальнейшие изменения содержимого поля зрения изображены ниже, включая последний шаг:

$$(+a)/f/(-b) \perp / f /(+\sin(x)) \perp$$
$$(+a)(-b)/f/(+\sin(x)) \perp$$
$$(+a)(-b)(+\sin(x)).$$

В итоге получаем, что рассмотренная программа осуществляет разбиение выражений на слагаемые.

2. ОБРАБОТКА ПОИСКОВЫХ ЗАПРОСОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Считаем, что поисковый запрос представляет собой совокупность предложений на естественном языке. Эту совокупность предложений можно расширить, используя словарные статьи из толкового словаря (например, словаря Ожегова), т. е. фактически приписать определения отдельных слов. Следующий этап состоит в том, чтобы представить данные предложения в виде помеченных деревьев. Вершины помечаются словами, а ребра — вопросами, задаваемыми от одного слова к другому.

Далее рассмотрим текст достаточно большого объема, из которого необходимо выбрать предложения по тематике поискового запроса и, таким образом, сформировать аннотацию или решить, является ли текст релевантным данному запросу. Для этого предложения данного текста также могут быть представлены в виде деревьев (вообще говоря, необязательно все предложения, а выборочно по некоторым критериям). После этого необходимо сопоставление на похожесть (соответствие) деревьев, полученных из запроса, и деревьев, возникших из текста.

Для аннотации выбираются предложения, которые соотносятся по теме, имеют похожие структуры и т. д. На основании подобных идей можно судить о релевантности.

Обработка поисковых запросов на естественном языке предполагает выполнение ряда действий.

- Семантико-синтаксический разбор запроса.
- Генерация схем возможных ответов.
- Нахождение в тексте фрагментов в соответствии со схемами ответов.
- Анализ контекстной связности между предложениями.

Поисковый запрос может представлять собой либо вопрос, поставленный в явном виде, либо набор фраз, задающих тему. Мы должны в тексте найти ответ на поставленный вопрос, либо выделить фрагменты на заданную тему. Для простоты считаем, что запрос состоит из одного предложения.

2.1. Семантико-синтаксический разбор запроса

Семантико-синтаксический разбор запроса заключается в том, что запросу сопоставляется дерево семантико-синтаксического разбора. В общем случае, это будет размеченное дерево. Пометки ребер будем также называть вопросами. Дерево может быть классического типа, из тех, которые применяются в лингвистике, так и неклассического.

Например, ребра могут помечаться семантическими предикатами, т. е. лексическими функциями по терминологии Мельчука; предикатами, выявленными в процессе изучения структуры словарных статей в словаре Ожегова и т. д. В общем случае можно считать, что ребра помечены двухместными предикатами, которые должны быть истинными на соответствующих пометках вершин.

Результирующее дерево представляется в виде скобочной структуры. Для размеченных деревьев можно использовать следующую конструкцию. Предположим, что (x_1, q, x_2) — помеченное ребро, q — метка. Тогда его скобочное представление будет иметь вид $(x_1\ (q\ (x_2)))$. Понятно, что данное преобразование можно применять рекурсивно.

2.2. Генерация схем возможных ответов

На данной стадии вопросу сопоставляется множество возможных ответов. При этом могут быть использованы конструкции, которые аналогичны конструкциям, применяемым в языке REFAL.

Рассмотрим пример. Вопрос: «Сколько тебе лет?». Ответ: «Мне 15 лет». Договоримся, переменные вида s_i использовать для символов, а переменные e_i использовать для совокупностей слов и для выражений, содержащих скобки.

Тогда может быть предложена схема перехода «вопрос—ответ»

$$e_0\ \textit{тебе}\ e_1 \to \textit{мне}\ s_2 e_1 .$$

Более сложная схема может учесть возможный контекст

$$e_0\ \textit{тебе}\ e_1 \to e_3\ \textit{мне}\ s_2 e_1 e_4 .$$

Если мы будем работать с деревьями синтаксического разбора, то переходу

$$(\textit{сколько}\ (\textit{тебе лет})) \to (\textit{мне}\ (15\ \textit{лет}))$$

соответствует схема

$$(e_0\ (\textit{тебе}\ e_1)) \to (e_3 (\textit{мне}\ (s_2 e_1)) e_4) .$$

Большое количество таких схем можно получить, заглянув в учебники иностранных языков, школьные учебники по конкретным предметам, в учебники по скорочтению. В последних предлагается быстро прочесть текст, а потом с помощью вопросов осуществляется контроль качества его усвоения.

Используя размеченные деревья, можно учесть морфологические особенности слов и их согласование по родам, падежам и т. д. Но в действительности, целесообразно модифицировать REFAL-подобные конструкции, введя новые типы переменных, т. е. сделать их более типизированными и ориентированными на лингвистику. Можно ввести специальные переменные для частей речи. Например, $e[Adj]_0$, где *Adj* — прилагательное и т. д.

В настоящее время выделено более 40 схем, соответствующих стандартным вопросам и ответам. Предложения, в соответствии с которыми строились схемы, взяты из учебника «Do You Speak English?» B.&R. Retman [2]. Каждой из схем сопоставлено представление с вовлечением скобочных структур и использованы расширенные типы переменных языка REFAL. Например,

1. Это красная машина $\rightarrow$ Да, это красная машина

Это $((attr \leftarrow e[Adj]_0))\ sub(e_1)) \rightarrow$ Да, это $((attr \leftarrow (e[Adj]_0))\ sub(e_1))$.

1'. Это красная машина $\rightarrow$ Нет, это не красная машина

Это $((attr \leftarrow e[Adj]_0))\ sub(e_1)) \rightarrow$ Нет, это $((attr \leftarrow (не\ e[Adj]_0))\ sub(e_1))$.

1". Это красная машина $\rightarrow$ Нет, это зеленая машина

Это $((attr \leftarrow e[Adj]_0))\ sub(e_1)) \rightarrow$ Нет, это $((attr \leftarrow (e[Adj]_2))\ sub(e_1))$.

Здесь $attr \leftarrow$ — определение слева от определяемого слова, *sub* — подлежащее.

Более интересными и полезными являются, например, переменные вида f_i, которые будут обозначать, что слова одинаковы с точностью до флексии, т. е. изменений суффиксов и окончаний. При отождествлении левой части продукции с запросом переменная f_i будет отождествлена со словом как s_i. А при отождествлении правой части продукции с предложением в тексте данное слово будет отыскиваться с точностью до флексии.

Можно ввести специальную переменную, которая будет обозначать, что совпадают достаточно длинные начальные части слов, например, не менее 75% каждого из них. Заметим, что слова могут быть разной длины. Такого типа сравнение полезно тем, что оно алгоритмически просто и не требует морфологического разбора. В то же время, для длинных слов указанное частичное совпадение автоматически обозначает, что это одно и то же сло-

во, с точностью до флексии. На ранних стадиях развития ребенка, по-видимому, именно так и происходит.

2.3. Нахождение в тексте фрагментов в соответствии со схемами ответов

Заданному вопросу соответствует несколько возможных ответов. Поэтому можно считать, что схема перехода «вопрос-ответ» имеет вид $\varphi \to \psi_1 \vee \psi_2 \vee \ldots \vee \psi_N$.

После отождествления φ с вопросом переменные, входящие в нее, приобретают значения. Далее в тексте ищем предложения, которые можно отождествить хотя бы с одной из формул ψ_i. Все такие предложения выдаем пользователю как ответы.

Рассмотрим пример, приведенный выше. В нем

φ = Это $((attr \leftarrow e[Adj]_0))\ sub(e_1))$

ψ_1 = Да, это $((attr \leftarrow (e[Adj]_0))\ sub(e_1))$

ψ_2 = Нет, это $((attr \leftarrow (\textit{не}\ e[Adj]_0))\ sub(e_1))$

ψ_3 = Нет, это $((attr \leftarrow (e[Adj]_2))\ sub(e_1))$.

Таким образом, для данного примера схема перехода «вопрос-ответ» кратко запишется $\varphi \to \psi_1 \vee \psi_2 \vee \psi_3$.

Заметим, что, говоря об отождествлении, мы можем рассматривать как сами предложения, так и результаты синтактико-семантического разбора, и работать с ними, что, безусловно, более интересно, и может привести к более качественным результатам.

Целесообразно предусмотреть специальные метки, которые позволят управлять областями отождествления и выдаваемыми областями. Например, формулу ψ_i можно отождествлять не с отдельным предложением, а с целым абзацем. Другой вариант, когда ψ_i отождествляется с предложениями, но после нахождения соответствующего предложения пользователю выдается весь абзац, в котором оно найдено. Поэтому можно считать, что схемы переходов «вопрос-ответ» имеют вид

$l : \varphi \to \psi_1 \vee \psi_2 \vee \ldots \vee \psi_N$, где l — метка.

2.4. Анализ контекстной связности между предложениями

В текстах на естественном языке наблюдается явление, называемое контекстной связностью. Например, мы хотели бы выделить в тексте все пред-

ложения, в которых идет речь о птице вороне. В ряде предложений встречается слово «ворона» с точностью до флексии, а в ряде предложений может встречаться «эта птица». Если слова «эта птица» в соответствующих предложениях заменить на «ворона», то полученные предложения можно анализировать в соответствии с методами, описанными выше.

Связать «ворона» и «эта птица» можно, если заглянуть в толковый словарь. Там написано, что «ворона — большая всеядная птица…».

В ряде случаев способы обнаружения контекстной связности более-менее простые. В данном случае имеется указательное местоимение «эта», слово при нем «птица», как правило, согласовано в роде и падеже со словом «ворона», но вообще говоря, это не обязательно.

Самое главное, что слово «птица» встречается в соответствующей словарной статье толкового словаря. Последнее легко проверить на компьютере. Фактически, в толковом словаре содержится информация о том, что имеет место истинность лексического предиката $Gener(ворона, птица)$, который обозначает, что «птица» является более общим понятием, чем «ворона».

Отметим в заключение, что вопрос о контекстной связности требует дополнительного изучения. В целом он очень сложный, но типовые ситуации можно достаточно легко описать и реализовать на компьютере.

ВЫВОДЫ

На основе изложенного выше могут быть сделаны следующие выводы. При формировании деревообразного представления предложений на естественном языке предлагается использовать модифицированные конструкции языка символьных преобразований REFAL:

1. Целесообразно использовать новые типы переменных, связанные с частями речи, частичным совпадением слов и т. д.;

2. В языке REFAL для любого оператора $\varphi \to \psi$ выполнено $\mathrm{var}(\varphi) \supseteq \mathrm{var}(\psi)$. У нас это нарушается, и таким образом пытаемся учесть контекст.

3. Заданному вопросу соответствует несколько возможных ответов. Поэтому можно считать, что схема перехода «вопрос—ответ» имеет вид $\varphi \to \psi_1 \vee \psi_2 \vee ... \vee \psi_n$.

4. После отождествления φ с вопросом переменные, как и в обычном языке REFAL, входящие в нее, приобретают значения. Далее в тексте ищем

предложения, которые можно отождествить хотя бы с одной из формул ψ_i. Все такие предложения выдаем пользователю в качестве ответов.

СПИСОК ЛИТЕРАТУРЫ

1. Murzin F.A. Syntactic properties of the REFAL language // Int. J. Computer Math. — 1985. — N17. — P. 123 — 139.
2. Retman B.&R. Do You Speak English? — Warszawa: Wiedza Powszechna, 1977. — 160 p.

А. А. Добрынин, Л. С. Мельников*, Х. Вальтер, Й. Шрейер

ЧИСЛО КОСЫХ ПОЛИЭДРАЛЬНЫХ ГРАФОВ С МАЛЫМ ЧИСЛОМ ВЕРШИН

Рассматриваются полиэдральные графы (графы полиэдров), т. е. плоские 3-связные графы. Грань размера k полиэдрального графа имеет тип $\langle a_1, a_2, ..., a_k\rangle$, если инцидентные этой грани вершины, обходимые в циклическом порядке, имеют степени $a_1, a_2, ..., a_k$, и этот набор является лексикографически минимальным среди всех подобных наборов. Если в полиэдральном графе все грани имеют разные типы, то такой граф называется косым (oblique). Для полиэдральных графов с числом вершин не более 12 найдены количества косых графов, в том числе с дополнительными свойствами.

ВВЕДЕНИЕ

Рассматриваются полиэдральные графы (графы полиэдров), т. е. плоские 3-связные графы $G = G(V, E, F)$ с множеством вершин $V = V(G)$, множеством ребер $E = E(G)$ и множеством граней $F = F(G)$. Хорошо известно, что плоские 3-связные графы имеют комбинаторно единственную укладку на плоскости. Число ребер (или граней), инцидентных вершине v, называется степенью v. Число l ребер (или число вершин), инцидентных грани $\alpha \in F$, называется степенью α. Грань α будет также называться l-гранью или l-угольником. Грани $\alpha \in F(G)$ поставим в соответствие последовательность $\langle a_1, a_2, ..., a_l\rangle$, если α является l-угольником и степени инцидентных грани α вершин равны $a_1, a_2, ..., a_l$ при некотором обходе вершин грани в циклическом порядке. Лексикографический минимум среди всех таких последовательностей называется типом грани α. Например, для треугольной грани α типа $\langle a, b, c\rangle$ выполняется $a \le b \le c$.

Полиэдральный граф G называется *косым* (*oblique*), если все его грани имеют разные типы. Косой граф называется *двойным косым*, ес-

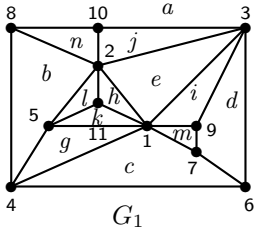
*omeln@math.nsc.ru

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 05–01–00395 и 05–01–00816).

ли его геометрически двойственный граф также является косым. Ясно, что двойные косые графы содержат все косые самодвойственные полиэдральные графы. Граф G называется *суперкосым*, если G и его двойственный граф являются косыми и эти графы не имеют граней совпадающих типов. Пусть $k \geq 1$ — натуральное число. Полиэдральный граф G является *k–косым*, если множество граней $F(G)$ содержит не более k граней одинакового типа независимо от типа. Очевидно, 1-косой граф является косым, и наоборот. Полиэдральный граф называется *триангуляцией*, если все его грани являются треугольными. Б. Грюнбаум и С. Шефард [3] перечислили все гране-транзитивные полиэдральные графы. Ясно, что такие графы имеют грани только одного типа. Х. Вальтер показал [5], что множество косых триангуляций конечно и любая косая триангуляция содержит вершину степени 3. В [4] доказано, что для любого k множество k-косых графов конечно. Из результата О. Бородина [1] следует, что любой косой граф всегда содержит вершину степени 3 или 4.

В [2] получены следующие результаты: любой косой граф состоит не менее, чем из 8 граней и содержит вершину степени 3 (два графа на 10 вершинах имеют точно 8 граней); существуют как самодвойственные косые графы, так и суперкосые графы; любая косая триангуляция содержит не менее 6 вершин с попарно разными степенями; любая косая триангуляция состоит из не менее, чем 16 граней (16 граней имеет единственная триангуляция).

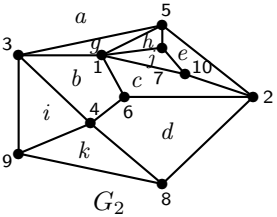
Назовем типом вершины плоского графа лексикографически минимальную последовательность размеров инцидентных ей граней при их циклическом обходе. Очевидно, что набор типов вершин полиэдрального графа совпадает с набором типов граней его двойственного графа. На рис. 1. приводятся примеры суперкосого графа G_1, двойного косого графа G_2 (не самодвойственного) и двойного косого самодвойственного графа G_3. Для графов перечислены типы всех граней и вершин. Грани графа обозначены буквами, а вершины занумерованы числами.



G_1

f	степени вершин				
a	3	3	4	3	5
b	3	4	4	6	
c	3	3	4	7	
d	3	3	3	5	
e	5	6	7		
g	4	4	7		
h	3	6	7		
i	3	5	7		
j	3	5	6		
k	3	4	7		
l	3	4	6		
m	3	3	7		
n	3	3	6		

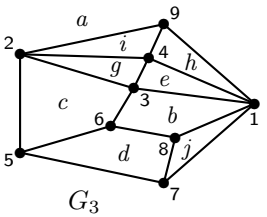
v	размеры граней						
1	3	3	3	3	3	3	4
2	3	3	3	3	3	4	
3	3	3	3	4	5		
4	3	4	5	4			
5	3	3	3	4			
6	4	4	5				
7	3	4	4				
8	3	4	5				
9	3	3	4				
10	3	3	5				
11	3	3	3				



G_2

f	степени вершин				
a	3	3	4	4	4
b	3	4	4	5	
c	3	4	3	5	
d	3	4	3	4	
e	3	3	4	4	
g	4	4	5		
h	3	4	5		
i	3	4	4		
j	3	3	5		
k	3	3	4		

v	размеры граней						
1	3	3	3	4	4		
2	4	4	4	5			
3	3	4	3	5			
4	3	3	4	4			
5	3	3	4	5			
6	4	4	4				
7	3	4	4				
8	3	4	5				
9	3	3	5				
10	3	3	4				



G_3

f	степени вершин				
a	3	3	4	3	5
b	3	3	4	5	
c	3	3	4	4	
d	3	3	3	3	
e	4	4	5		
g	4	4	4		
h	3	4	5		
i	3	4	4		
j	3	3	5		

v	размеры граней				
1	3	3	4	3	5
2	3	3	4	5	
3	3	3	4	4	
4	3	3	3	3	
5	4	4	5		
6	4	4	4		
7	3	4	5		
8	3	4	4		
9	3	3	5		

Рис. 1. Суперкосой, двойной косой и самодополнительный косой графы

1. ЧИСЛО КОСЫХ ГРАФОВ

Ясно, что косые графы являются асимметричными, т. е. имеют тривиальную группу автоморфизмов. Поэтому далее рассматриваются только асимметричные полиэдральные графы. Нас будут интересовать количества косых графов с дополнительными свойствами.

Обозначим через DO (Double Oblique) класс всех двойных косых графов. В этом классе выделим следующие непересекающиеся подмножества: SO (Super Oblique) — множество суперкосых графов (см. граф G_1 на рис. 1), EO (Equal Oblique) — множество не самодвойственных графов, для которых *размеры граней* графа и его двойственного графа совпадают (см. граф G_2 на рис. 1) и SD (Self Dual oblique) — множество самодвойственных графов (см. граф G_3 на рис. 1).

Данные о количестве косых полиэдральных графах с числом вершин не более 12 из указанных множеств приводятся в таблице 1. В клетке таблицы над горизонтальной чертой указаны количества всех полиэдральных графов и асимметричных полиэдральных графов, разделенные наклонной чертой. Под горизонтальной чертой первое число есть количество косых графов. Далее после наклонной черты указано число $|DO|$ двойных косых графов, если они существуют. Для таких графов через запятую перечисляются число $|SO|$ суперкосых графов, число $|EO|$ графов с одинаковыми наборами размеров граней и число $|SD|$ самодвойственных косых графов. Если информация об этих множествах не указывается, то они являются пустыми, а на соответствующем месте таблицы при необходимости указывается ноль.

Утверждение 1. *Множество всех полиэдральных графов с числом граней $f \leq 20$ и числом вершин $v \leq 12$ содержит 80312 косых графов, 1447 двойных косых графов (DO), 50 суперкосых графов (SO), 90 косых графов с одинаковыми наборами размеров граней (EO) и 58 самодвойственных косых графов (SD).*

В ходе компьютерного поиска проверялись дополнительные свойства графов. Так, в косых графах всегда присутствовала вершина степени 4. Среди косых графов встречались такие, у которых: нет 5- и 6-граней; нет вершин степени 5 и 6; нет 6-граней и вершин степени 6; присутствует одна 4-грань и нет 5- и 6-граней. Множество косых триангуляций с не более, чем 22 гранями состоит из 66 графов (1, 3, 13 и 49 триангуляций с 16, 18, 20 и 22 гранями соответственно). Существуют неизоморфные косые самодвойственные графы с полностью совпадающими наборами типов граней.

Таблица 1

Количество всех полиэдральных графов и асимметричных полиэдральных графов (над чертой), количество косых и двойных косых графов (под чертой). Через запятые указано число графов для непустых множеств SO, EO и SD (f — грани, v – вершины).

	4 v	5 v	6 v	7 v	8 v	9 v	10 v	11 v	12 v
4 f	$\frac{1/0}{0}$								
5 f		$\frac{1/0}{0}$	$\frac{1/0}{0}$						
6 f		$\frac{1/0}{0}$	$\frac{2/0}{0}$	$\frac{2/0}{0}$	$\frac{2/0}{0}$				
7 f			$\frac{2/0}{0}$	$\frac{8/2}{0}$	$\frac{11/3}{0}$	$\frac{8/2}{0}$	$\frac{15/0}{0}$		
8 f			$\frac{2/0}{0}$	$\frac{11/3}{0}$	$\frac{42/22}{0}$	$\frac{74/48}{0}$	$\frac{76/44}{2}$	$\frac{38/21}{0}$	$\frac{14/2}{0}$
9 f				$\frac{8/2}{0}$	$\frac{74/48}{1}$	$\frac{296/237}{12/3/0,0,3}$	$\frac{633/533}{26/1}$	$\frac{768/662}{27}$	$\frac{558/449}{6}$
10 f				$\frac{5/0}{0}$	$\frac{76/44}{0}$	$\frac{633/533}{10/1}$	$\frac{2635/2401}{77/3/0,2,1}$	$\frac{6134/5790}{250/2}$	$\frac{8822/8331}{370/2}$
11 f					$\frac{38/21}{0}$	$\frac{768/662}{9}$	$\frac{6134/5790}{139/2}$	$\frac{25626/24888}{759/24/0,6,8}$	$\frac{64439/63080}{2161/38}$
12 f					$\frac{14/2}{0}$	$\frac{558/449}{4}$	$\frac{8822/8331}{107/2}$	$\frac{64439/63080}{1618/38}$	$\frac{268394/265253}{8496/190/0,82,46}$
13 f						$\frac{219/164}{1}$	$\frac{7916/7491}{57}$	$\frac{104213/102524}{1520/32/1}$	$\frac{709302/704267}{16318/313}$
14 f						$\frac{50/16}{0}$	$\frac{4442/4052}{15}$	$\frac{112082/110015}{1084/19/3}$	$\frac{1263032/1255238}{18925/321/2}$
15 f							$\frac{1404/1235}{3}$	$\frac{79773/78169}{409/8/4}$	$\frac{1556952/1548735}{15907/344/19}$
16 f							$\frac{233/137}{1}$	$\frac{36528/35199}{121}$	$\frac{1338853/1329899}{7842/103/19}$
17 f								$\frac{9714/9176}{29}$	$\frac{789749/783543}{3006/2/2}$
18 f								$\frac{1249/970}{3}$	$\frac{306470/301763}{874}$
19 f									$\frac{70454/68697}{111}$
20 f									$\frac{7595/6756}{13}$

Таблица 2

Распределение асимметричных полиэдральных графов по максимальному числу s граней совпадающего типа (f — грани; число вершин не более 12)

$s \to$	1	2	3	4	5	6	7	8	9	10	11	12	13
7 f	–	1	6	–	–	–	–	–	–	–	–	–	–
8 f	2	85	47	6	–	–	–	–	–	–	–	–	–
9 f	72	1068	666	114	11	–	–	–	–	–	–	–	–
10 f	707	9553	5595	1088	139	17	–	–	–	–	–	–	–
11 f	3068	52293	30228	7140	1483	218	11	–	–	–	–	–	–
12 f	10225	167447	114166	35550	8172	1375	171	9	–	–	–	–	–
13 f	17895	367424	292458	102201	27022	6128	1171	139	8	–	–	–	–
14 f	20024	556122	524947	198555	53450	13028	2646	474	72	3	–	–	–
15 f	16319	594693	653579	263836	75050	19224	4310	908	167	44	9	–	–
16 f	7964	449573	567342	242990	70207	20146	5236	1362	335	73	7	–	–
17 f	3035	238113	333591	150477	46763	14992	4217	1121	298	85	20	6	1
18 f	877	82369	129416	60951	19392	6832	1985	647	186	65	10	3	–
19 f	111	17249	29324	14542	4870	1793	529	188	62	22	4	2	1
20 f	13	1610	2986	1443	469	179	45	9	2	–	–	–	–
всего	80312	2537600	2684351	1078893	307028	83932	20321	4857	1130	292	50	11	2

2. БЛИЗОСТЬ АСИММЕТРИЧНЫХ ГРАФОВ К КОСЫМ ГРАФАМ

Как можно описать близость других асимметричных полиэдральных графов к косым графам? Будем характеризовать такую близость двумя антиподальными величинами. Пусть число граней f фиксированно. Первая величина s есть максимальное число граней графа одинакового типа, вторая — число u различающихся типов граней графа. В частности, для косого графа $u = f$ и $s = 1$. Ясно, что $s + u - 1 \leq f$, где равенство достигается на косых графах и графах с $s = f$.

В таблице 2 приведено распределение асимметричных графов по величине s. Первый столбец таблицы содержит количества косых графов. Существуют ли асимметричные полиэдральные графы для которых выполняется $s = f$ (или $u = 1$)? Ответ на этот вопрос будет отрицательным, так как в [5] показано, что кроме гране-транзитивных полиэдральных графов существует в точности один полиэдральный граф с единственным типом граней, который, однако, имеет симметрии.

Таблица 3

Распределение асимметричных полиэдральных графов по числу u различающихся типов граней (f — грани; число вершин не более 12)

$u \rightarrow$	3	4	5	6	7	8	9	10	11
7 f	–	–	3	4	–	–	–	–	–
8 f	–	1	35	61	41	2	–	–	–
9 f	–	16	167	593	718	365	72	–	–
10 f	2	23	360	2082	5035	5695	3195	707	–
11 f	4	62	682	4062	13723	27166	29512	16162	3068
12 f	11	82	891	5745	23047	62421	98864	89573	46256
13 f	2	56	878	6341	30866	90149	171051	222578	185283
14 f	3	38	760	5970	28039	91016	199421	313438	347380
15 f	1	83	520	4229	21232	68020	159824	290254	385318
16 f	6	74	311	2743	11879	37709	95595	188683	270740
17 f	2	24	232	1364	5381	17073	43568	83114	123853
18 f	–	14	88	522	2063	5595	13069	24274	37581
19 f	–	1	19	165	483	1339	2592	4256	7005
20 f	–	–	1	17	40	64	162	323	603
всего	31	474	4947	33898	142547	406614	816925	1233362	1407087

$u \rightarrow$	12	13	14	15	16	17	18	19	20
7 f	–	–	–	–	–	–	–	–	–
8 f	–	–	–	–	–	–	–	–	–
9 f	–	–	–	–	–	–	–	–	–
10 f	–	–	–	–	–	–	–	–	–
11 f	–	–	–	–	–	–	–	–	–
12 f	10225	–	–	–	–	–	–	–	–
13 f	89346	17896	–	–	–	–	–	–	–
14 f	254762	108470	20024	–	–	–	–	–	–
15 f	363195	228808	90336	16319	–	–	–	–	–
16 f	297242	250791	148037	53461	7964	–	–	–	–
17 f	154657	157374	119568	63442	20032	3035	–	–	–
18 f	51598	58137	51292	34981	17220	5422	877	–	–
19 f	9978	11603	11485	9646	6418	2827	769	111	–
20 f	814	1028	1038	1039	851	494	209	60	13
всего	1231817	834107	441780	178888	52485	11778	1855	171	13

Как видно из таблицы 2 лучшее приближение величины s к числу граней f дает граф с $f = 17$ гранями, из которых $s = 13$ граней одного типа.

Количество k–косых полиэдральных графов с f гранями получается суммированием чисел в столбцах 1, 2, ..., k в соответствующей строке таблицы 2.

В таблице 3 представлено распределение асимметричных графов по количеству u различающихся типов граней. Числа косых графов образуют верхнюю огибающую непустых элементов таблицы. Оказалось, что не существует асимметричных полиэдральных графов с числом вершин не более 12, которые имели бы два типа граней ($u = 2$).

СПИСОК ЛИТЕРАТУРЫ

1. **Borodin O.** Structural properties of planar maps with the minimum degree 5 // Math. Nachr. — 1992. — Vol. 158. — P. 109–117.
2. **Dobrynin A. A., Mel'nikov L. S., Schreyer J., Walther H.** Some news about oblique graphs // Discuss. Math. Graph Theory. — 2002. — Vol. 22, N 1. — P. 39–50.
3. **Grünbaum B., Shephard G. C.** Spherical tilings with transitivity properties // The Geometric Vein: The Coxeter Festschrift. — Springer–Verlag, 1982. — P. 65–98.
4. **Voigt M., Walther H.** Polyhedral graphs with restricted number of faces of the same type // Discrete Math. — 2002. — Vol. 244, N 1–3. — P. 473–478.
5. **Walther H.** Polyhedral graphs with extreme numbers of types of faces // Discrete Appl. Math. — 2002. — Vol. 120, N 1–3. — P. 263–274.

А.А. Дунаев, Т.Ф. Валеев, Е.А. Тарасов

ИССЛЕДОВАНИЕ МЕТОДОВ ОРГАНИЗАЦИИ ВИЗУАЛЬНОЙ ОБРАТНОЙ СВЯЗИ В АППАРАТНО-ПРОГРАММНОМ КОМПЛЕКСЕ «БОСЛАБ»

ВВЕДЕНИЕ

Специалистами НИИ молекулярной биологии и биофизики СО РАМН[1] разработан аппаратно-программный комплекс «Бослаб», позволяющий регистрировать физиологические параметры испытуемого человека, такие как температура кожи, мышечная активность (миограммы), кардиоинтервалы, биоэлектрическая активность головного мозга (электроэнцефалограммы) и т. п. «Бослаб» организует работу с испытуемым в виде сеанса, состоящего из сессий — различных задач (тестов, лечебных процедур), во время выполнения которых испытуемым программа непрерывно регистрирует состояние параметров, интересующих экспериментатора. После окончания сеанса экспериментатор имеет возможность просмотреть ход работы и сопоставить с ним изменения зарегистрированных параметров. Параметры процедур в «Бослабе», заданные до начала сеанса, лишь частично могут быть изменены в ходе выполнения. Например, пороговые значения, относительно которых в тестах необходимо изменять физиологические параметры (повышать температуру, снижать мышечное напряжение) для эффективного выполнения задания представлены, в основном, в простейшем графическом варианте — в виде линий, столбиков. Они могут быть изменены лишь в ручном режиме в процессе тестирования. К сожалению, такая форма представления сигналов не способствует повышению мотивации испытуемого и не использует современные возможности обработки изображений. С другой стороны, существующие в «Бослабе» игровые представления тестов трудоемки в создании, и их настройки не могут быть изменены в ходе выполнения. Данное обстоятельство существенно ограничивает свободу действий экспериментатора, вынуждая его прерывать работу с испытуемым для корректировки параметров.

В рамках данной работы создано приложение для проведения тестов, обеспечивающее визуальную и звуковую обратную связь с испытуемым в

[1] Ранее — Институт медицинской и биологической кибернетики СО РАМН.

комплексе «Бослаб» и позволяющее изменять параметры теста непосредственно во время проведения сеанса.

Программа представляет собой приложение для ОС Microsoft Windows 2000/XP и запускается «Бослабом» в качестве внешнего приложения для игрового представления рабочей сессии. Предусмотрены два режима работы программы — настройка и воспроизведение. Используя совместно функции «Бослаба» и описываемой программы, экспериментатор имеет возможность создать набор готовых рабочих сессий и использовать их в дальнейшей работе.

1. ОБЩЕЕ УСТРОЙСТВО СИСТЕМЫ

1.1. Принцип работы

«Бослаб» регистрирует одновременно до 14 различных типов сигналов: ЭЭГ, ЭМГ, температура, альфа-, бета- и тета-ритмы и т. п., причём для некоторых сигналов предусмотрены два канала. Общее количество регистрируемых параметров — 22. Один из параметров может быть выбран для передачи внешнему приложению, в роли которого может выступать описываемая программа.

Приложение получает от «Бослаба» данные о текущем состоянии выбранного сигнала, используя функции, экспортируемые специальной динамической библиотекой. Для каждого вида сигнала существует характерный диапазон значений (например, значение сигнала ЭЭГ лежит в диапазоне –100÷100 мкВ), в пределах которого экспериментатором задаётся эталонное значение параметра. Вычисляемое значение отклонения регистрируемого значения от эталона используется в качестве параметра обратной связи с испытуемым.

Обратная связь организуется следующим образом. Испытуемому на дисплее ПК предлагается изображение, в которое программа вносит помехи или искажения. Сила помех или степень искажений зависит от параметра обратной связи. Разработаны разнообразные методы генерации помех и искажений (далее — «эффекты»), и в зависимости от особенностей теста и индивидуальных качеств испытуемого может быть выбран наиболее подходящий эффект. Применение эффекта может также сопровождаться звуком.

1.2. Интеграция с «Бослабом»

«Бослаб» даёт возможность вызвать внешнее приложение в двух режимах: настройки и воспроизведения. Требуемый режим устанавливается передачей программе ключа через командную строку.

В режиме настройки «Бослаб» создаёт в рабочей директории программы специальный инициализационный файл, после чего запускает приложение. Далее экспериментатор формирует будущую рабочую сессию, для чего используется весь пользовательский интерфейс программы. Сессия содержит последовательность изображений, каждое из которых хранится в отдельном файле в одном из общепринятых форматов[2].

Для каждого элемента последовательности задаётся длительность интервала времени, в течение которого этот элемент будет работать в тесте, и устанавливается один из доступных эффектов. Помимо последовательности изображений, экспериментатор задаёт диапазон значений, который будет использоваться при тестировании.

Тест сохраняется во внешнем файле, имя которого записывается в инициализационный файл, копия которого затем сохраняется «Бослабом» в его внутренней базе данных.

В режиме воспроизведения «Бослаб» извлекает из базы данных требуемый инициализационный файл и копирует его в рабочую директорию программы. После этого запускается в режиме воспроизведения описываемое приложение, которое автоматически открывает указанный в инициализационном файле тест.

В режиме воспроизведения доступны только элементы пользовательского интерфейса, управляющие граничными и пороговыми значениями сигнала. Экспериментатор имеет возможность подстроить границы диапазона и пороговое значение, после чего вручную запустить тестирование, в ходе которого изображения, заданные в тесте, будут показаны на дисплее ПК с применением соответствующих эффектов в течение заданных интервалов времени.

[2] Для загрузки изображений используется сторонняя библиотека NexgenIPL, свободно распространяемая компанией Binary Technologies. Эта библиотека распознаёт более 20 различных графических форматов, включая форматы семейства JPEG, а также такие популярные форматы, как PNG, GIF, TIFF, TGA и др.

2. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ АЛГОРИТМОВ

2.1. Постановка задачи

Поскольку элементы теста полностью однотипны, без ограничения общности можно рассматривать один отдельно взятый элемент, а точнее — заданные в этом элементе файл изображения, эффект и длительность воспроизведения.

Задача программы состоит в применении выбранного эффекта к изображению в течение всего периода воспроизведения. Результатом применения эффекта является другое изображение, которое отображается на дисплее. Параметр обратной связи для эффекта постоянно запрашивается от «Бослаба», причём для эффектов, учитывающих случайные факторы, результат может пересчитываться постоянно.

Важным условием, которому должны удовлетворять алгоритмы обработки, является минимизация задержки, возникающей между моментом изменения сигнала и появлением результата. Для испытуемого реакция программы на его действия должна быть субъективно мгновенной. Известно, что типичное время осознанной реакции человека на визуальные раздражители составляет не менее 100 мсек. Следовательно, чтобы испытуемый не замечал задержки, время полной обработки изображения не должно превышать 100 мсек.

Другое важное качество, которым должен обладать эффект, можно сформулировать так: испытуемый должен уверенно идентифицировать изменения изображения. Иными словами, не должно возникать сомнений в том, проявился ли эффект. С другой стороны, визуальное восприятие разных людей существенно отличается, поэтому универсального эффекта, одинаково хорошо воспринимаемого всеми людьми, может не быть.

Кроме того, эффекты должны удовлетворять ряду простых условий. При значении параметра обратной связи, равном нулю, в изображение не должно вноситься никаких искажений. При возрастании или убывании значения параметра сила эффекта не должна, соответственно, убывать или возрастать (хотя допускается ступенчатое изменение).

2.2. Обозначения

Будем считать изображением прямоугольный растр точек, каждая из которых может иметь свой цвет. Как уже отмечалось выше, эффект является,

по сути, преобразованием исходного изображения, в результате которого получается другое изображение.

При обработке изображений в программе используется система цветовых координат RGB. Цветное изображение размером $m \times n$ может быть представлено матрицей $S[m,n]$, в которой каждый элемент представляет одну точку изображения и является трёхкомпонентным вектором (r,g,b), в котором каждая компонента соответствует одной из компонент цвета точки и может принимать значения в интервале $[0,255]$.

Поместим начало координат в верхний левый угол изображения. Оси координат направим влево и вниз. Будем обозначать произвольную точку изображения P так: P_{xy}, где x — смещение точки влево от начала координат (номер столбца), а y — вниз (номер ряда).

2.3. Эффект «случайный шум»

Эффект «случайный шум» изменяет цвет каждой точки, добавляя к каждой компоненте её цвета значение, вычисляемое по следующей формуле:

$$a = (rand(0,512) - 256) \cdot \frac{p}{100},$$

где $rand(0,512)$ обозначает псевдослучайное целое число в интервале $[0,512]$, а p является параметром обратной связи эффекта и принимает только целые значения от 0 до 100.

Для каждой точки добавка вычисляется один раз и затем суммируется к каждой из компонент. В случае, если результат оказывается за пределами интервала $[0,255]$, выполняется корректировка, и компонента принимает минимальное или максимальное значение в допустимом интервале.

Несмотря на простоту алгоритма, эффект «случайный шум» имеет важное свойство: шум становится заметен уже при значениях параметра обратной связи, равных единицам (то есть, единицы процентов от максимальной силы эффекта).

2.4. Эффект размытия («Blur»)

Эффект размытия достигается применением линейного однородного фильтра [1] с размером окрестности 11x11 пикселей и весовыми коэффициентами, равными значениям функции Гаусса. Результирующий цвет точки определяется следующим образом:

$$P^*_{xy} = \frac{\sum_{i=-5}^{5}\sum_{j=-5}^{5} \alpha_{ij} P_{x+i,y+j}}{\sum_{i=-5}^{5}\sum_{j=-5}^{5} \alpha_{ij}}, \qquad (1)$$

где P_{xy} — цвет точки до обработки. Красная, зелёная и синяя компоненты цвета обрабатываются независимо по формуле (1). Для корректной обработки краёв исходное изображение увеличивается во всех направлениях на 5 пикселей, которые закрашиваются чёрным цветом. Коэффициенты α_{ij} определяются по формуле:

$$\alpha_{ij} = \exp\left(-\frac{r(i,j)^2}{2\sigma^2}\right), \qquad (2)$$

где $r(i,j) = \sqrt{i^2 + j^2}$ — расстояние до центра, а $\sigma > 0$ — степень размытия, переменный параметр фильтрации. Из (2) видно, что чем меньше σ, тем сильнее влияние центрального элемента окрестности. Если σ приближается к нулю, то полученное после фильтрации изображение практически не отличается от исходного. При увеличении σ веса всех точек окрестности становятся более близкими, и изображение размывается сильнее.

Подставим (2) в (1) и преобразуем:

$$P^*_{xy} = \frac{\sum_{i=-5}^{5}\sum_{j=-5}^{5} e^{-\left(i^2+j^2\right)/2\sigma^2} P_{x+i,y+j}}{\sum_{i=-5}^{5}\sum_{j=-5}^{5} e^{-\left(i^2+j^2\right)/2\sigma^2}} = \frac{\sum_{i=-5}^{5}\sum_{j=-5}^{5} e^{-i^2/2\sigma^2} e^{-j^2/2\sigma^2} P_{x+i,y+j}}{\sum_{i=-5}^{5}\sum_{j=-5}^{5} e^{-i^2/2\sigma^2} e^{-j^2/2\sigma^2}}$$

$$= \frac{\sum_{i=-5}^{5}\left(e^{-i^2/2\sigma^2} \sum_{j=-5}^{5} e^{-j^2/2\sigma^2} P_{x+i,y+j} \right)}{\left(\sum_{i=-5}^{5} e^{-i^2/2\sigma^2}\right)\left(\sum_{j=-5}^{5} e^{-j^2/2\sigma^2}\right)}.$$

Обозначим

$$\beta_i = \frac{e^{-i^2/2\sigma^2}}{\sum_{j=-5}^{5} e^{-j^2/2\sigma^2}}, \qquad (3)$$

тогда

$$P^*_{xy} = \sum_{i=-5}^{5} \left(\beta_i \sum_{j=-5}^{5} \beta_j P_{x+i,y+j} \right). \quad (4)$$

Используя формулу (4), фильтрацию можно проводить в два этапа. Сначала вычисляется промежуточное изображение

$$Q_{xy} = \sum_{j=-5}^{5} \beta_j P_{x,y+j}\ , \quad (5)$$

а затем конечное

$$P^*_{xy} = \sum_{i=-5}^{5} \beta_i Q_{x+i,y}\ . \quad (6)$$

Таким образом, для обработки одной точки по (5) и (6) требуется 22 умножения и сложения, тогда как по формуле (1) их требовалось не менее 121.

Величина σ определяется из параметра p, передаваемого фильтру, как $\sigma = p/20$. Параметр p является параметром обратной связи эффекта и принимает только целые значения от 0 до 100. Поэтому величины β_i вычисляются для всех возможных σ по формуле (3) при инициализации фильтра. Во время вычисления очередного кадра используется набор значений, соответствующий текущему p. Если $p = 0$, то фильтрация не производится, т.е. $P^*_{xy} = Q_{xy} = P_{xy}$. Это достигается, если положить при $\sigma = 0$

$$\beta_i = \begin{cases} 1, & \textit{при } i = 0 \\ 0, & \textit{иначе} \end{cases}.$$

Чтобы процесс фильтрации был достаточно быстрым, вычисление Q_{xy} и P^*_{xy} реализовано на ассемблере процессора Intel Pentium с использованием технологии MMX. Для ускорения вычислений полученные при инициализации фильтра значения β_i умножаются на 256 и округляются до целого значения. Таким образом, вычисления по формулам (5) и (6) сводятся к умножению однобайтовых целых и суммированию результатов, которые затем сдвигаются на 8 бит вправо (таким сдвигом достигается деление на 256). С помощью инструкций MMX все компоненты цвета одной точки умножаются, складываются и сдвигаются параллельно.

2.5. Препарирование

Препарированием называют класс поэлементных преобразований полутонового изображения, выделяющих фрагменты изображения с определёнными яркостными характеристиками. К классу препарирующих преобразований относятся пороговая обработка, выделение яркостного среза, линейное контрастирование и другие методы [1].

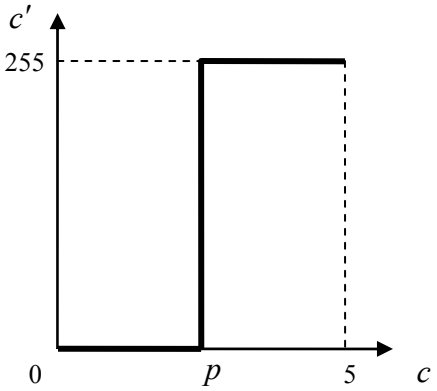


Рис. 1. Пороговое преобразование

Пример препарирующего преобразования — порогового преобразования — показан на рис. 1. Жирной линией показана зависимость яркости c' точки результирующего изображения от яркости c точки исходного изображения. Пунктирными линиями выделены максимальные значения яркости, равные 255[3]. Пороговое значение p прямо пропорционально параметру обратной связи.

Таким образом, пороговая обработка может быть описана следующей формулой:

$$c' = \begin{cases} 0, & 0 \le c < p \\ 255, & p \le c \le 255 \end{cases}, \quad 0 \le p \le 255 .$$

В описываемой программе используются цветные изображения, что даёт возможность препарирования двумя способами: покомпонентно, с пре-

[3] Из соображений удобства обработки для хранения значения яркости одной точки выделяется один байт.

парированием каждой компоненты цвета независимо, и с препарированием значения яркости, вычисленного по значениям компонент. В первом случае результатом преобразования является цветное изображение, во втором — полутоновое. Для второго случая возможны также различные способы вычисления яркости.

Все запрограммированные препарирующие преобразования описаны в приложении 1.

2.6. Эффект «чёрная дыра»

Результат применения эффекта «чёрная дыра» показан на рис. 2. Яркость точки результирующего изображения изменяется пропорционально расстоянию от центра и параметру обратной связи.

2.7. Геометрические эффекты

Реализованы несколько вариантов геометрических искажений, основанных на сдвиге строк исходного изображения.

Первый способ сдвигает строки изображения поочерёдно влево и вправо на величину, пропорциональную параметру обратной связи. Другие алгоритмы этой группы сдвигают строки со смещением, изменяющимся в зависимости от номера строки по определённому закону (синусоидальная зависимость, пилообразное изменение и т. п.).

Кроме того, создан эффект «вихрь», закручивающий исходное изображение вокруг центра (рис. 2).

Цвет точки результирующего изображения при применении «вихря» рассчитывается следующим образом. Координаты результирующей точки (x_r, y_r) переводятся из декартовых в

Рис. 2. Эффекты «чёрная дыра» и «вихрь». Вверху показано исходное изображение, эффекты показаны при максимальной силе

полярные (ρ_r, φ_r), причём начало координат устанавливается в центр изображения. Затем выполняется поворот на угол, зависящий от параметра обратной связи, после чего координаты переводятся назад в декартову систему. Полученные координаты (x_s, y_s) идентифицируют точку исходного изображения, цвет которой копируется в точку результирующего изображения. В случае, если точка (x_s, y_s) оказывается за границами исходного изображения, выбирается цвет точки, лежащей на границе.

2.8. Преобразования цветового пространства

В данной группе запрограммированы два эффекта: сведение к полутоновому изображению и «поворот цвета».

Алгоритм первого преобразования сводится к вычислению среднего значения компонент цвета точки. Полученное значение затем присваивается компонентам точки результирующего изображения.

«Поворот цвета» работает следующим образом. Цвет точки переводится в систему цветовых координат HSV[4]. После этого значение тона изменяется пропорционально параметру обратной связи, и выполняется обратное преобразование в координаты RGB цвета результирующей точки. Тон в системе координат HSV можно интерпретировать как угол, и всё преобразование в целом тогда будет соответствовать повороту на угол, пропорциональный параметру обратной связи.

3. РЕЗУЛЬТАТЫ ПРЕДВАРИТЕЛЬНЫХ ИСПЫТАНИЙ ПРОГРАММЫ

3.1. Цели и методика испытаний

Предварительные испытания проводились для того, чтобы выбрать из всего множества запрограммированных преобразований те, которые лучше всего подходят для визуальной обратной связи (см. п. 2.1.).

Скорость обработки изображения измерялась автоматически. Эффекты, применение которых занимало более 50 мсек, отбраковывались. (Очевидно, время обработки изображения зависит от нескольких параметров, таких как размер изображения, вычислительная сложность эффекта и общее быстродействие ЭВМ; эффекты, единственным недостатком которых является

[4] HSV — Hue, Saturation, Value — тон, насыщенность и яркость, соответственно.

высокая вычислительная сложность, могут с успехом применяться на более мощных ЭВМ.)

Другие критерии, такие как уверенная идентификация изменений, оценивались субъективно группой из трёх человек[5].

3.2. Результаты

Первый важный вывод, который можно сделать после предварительного исследования, — для большинства эффектов следует подбирать изображения со специальными особенностями. Так, препарирующие эффекты лучше выглядят на изображениях с плавными изменениями яркости, для гауссова размытия нужны, наоборот, чёткие контуры, а «вихрь» хорошо работает только на таких изображениях, изменение геометрии которых вызывает ощущение «неправильности» у испытуемого.

С уверенностью можно сказать, что в комплексе наилучшие показатели имеют препарирующие эффекты. Простота вычислительного алгоритма обуславливает высокую скорость обработки, а для того, чтобы эффект уверенно идентифицировался, подходят многие обыкновенные фотографические изображения (например, живой природы).

Эффекты, основанные на преобразовании цветового пространства изображения, малопригодны для организации обратной связи, поскольку плавные изменения цветовой гаммы всего изображения крайне трудно отследить визуально, особенно при медленном изменении.

3.3. Перспективы

В настоящее время планируется эксперимент с участием группы добровольцев, в котором будет выявлена эффективность различных эффектов в бета- и релаксационных тренингах. В случае, если на практике будет показано, что описанные методы визуальной обратной связи в сравнении с существующими средствами качественно улучшают результаты тренингов, будет поставлена задача разработки ПО для широкого применения в комплексе «Бослаб».

Кроме того, прорабатывается возможность использования аудиального канала обратной связи с испытуемым. Звуковое сопровождение тренинга в описываемом образце ПО весьма примитивно: приложение оповещает испытуемого звуком о том, что значение регистрируемого сигнала превысило

[5] Вполне достаточно для того, чтобы получить бинарную оценку «подходит — не подходит» каждого отдельного эффекта.

установленный порог. Таким образом, возможность проведения тренинга «на слух» присутствует, но в очень простой форме. Дальнейшее развитие приложения будет продолжено после успешного испытания первого опытного образца.

Приложение 1

ПРЕПАРИРУЮЩИЕ ПРЕОБРАЗОВАНИЯ

№ п.п.	Формула преобразования	Примечание
1	$c' = \begin{cases} 0, c < p \\ 255, c \geq p \end{cases}, \quad p \in [0,255]$	Пороговая обработка.
2	$c' = \begin{cases} 0, c \in (0,128-p) \bigcup (128+p,255) \\ 255, c \in [128-p,128+p] \end{cases}, \quad p \in [0,127]$	Яркостные срезы.
3	$c' = \begin{cases} c, c \in (0,128-p) \bigcup (128+p,255) \\ 255, c \in [128-p,128+p] \end{cases}, \quad p \in [0,127]$	
4	$c' = \begin{cases} c, c < p \\ 255, c \geq p \end{cases}, \quad p \in [0,255]$	
5	$c' = \begin{cases} 0, c \in (0,128-p) \\ (c-p) \cdot \frac{255}{255-2p}, c \in [128-p,128+p], p \in [0,127] \\ 255, c \in (128+p,255) \end{cases}$	Линейное контрастирование с насыщением.
6	$c' = \begin{cases} 255, c \in (0,128-p) \\ 255-(c-p) \cdot \frac{255}{255-2p}, c \in [128-p,128+p], p \in [0,127] \\ 0, c \in (128+p,255) \end{cases}$	Инверсия линейного контрастирования с насыщением.
7	$c' = \begin{cases} 0, c \in (0,128-p) \bigcup (128+p,255) \\ (c-p) \cdot \frac{255}{255-2p}, c \in [128-p,128+p] \end{cases}, \quad p \in [0,127]$	Линейное контрастирование в диапазоне. В зависимости от разновидности преобразования, вне диапазона
8	$c' = \begin{cases} 255, c \in (0,128-p) \bigcup (128+p,255) \\ (c-p) \cdot \frac{255}{255-2p}, c \in [128-p,128+p] \end{cases}, p \in [0,127]$	

№ п.п.	Формула преобразования	Примечание
9	$c' = \begin{cases} 127, c \in (0,128-p) \bigcup (128+p,255) \\ (c-p) \cdot \dfrac{255}{255-2p}, c \in [128-p,128+p] \end{cases}, p \in [0,127]$	выполняется подавление до нуля (7), насыщение до белого (8) и выравнивание до серого (9).
11	$c' = \begin{cases} 0, c \in (0,128-p) \bigcup (128+p,255) \\ c, c \in [128-p,128+p] \end{cases}, p \in [0,127]$	
12	$c' = \begin{cases} 255, c \in (0,128-p) \bigcup (128+p,255) \\ c, c \in [128-p,128+p] \end{cases}, \quad p \in [0,127]$	

СПИСОК ЛИТЕРАТУРЫ

1. **Грузман И.С., Киричук В.С., Косых В.П. и др.** Цифровая обработка изображений в информационных системах. Учебное пособие. — Новосибирск: Изд-во НГТУ, 2000. — 168 с.
2. **Binary Technologies** http://www.binary-technologies.com/
3. **Carey Bunks.** Grokking the GIMP. — New Riders Publishing, 2000. — ISBN 0-7357-0924-6

А.А. Дунаев

ИССЛЕДОВАТЕЛЬСКАЯ СИСТЕМА ДЛЯ АНАЛИЗА ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

ВВЕДЕНИЕ

Исследования в области автоматической обработки текста (АОТ) и формализации естественных языков, планомерно продвигаясь от самых простых методов анализа к более сложным, постепенно приближаются к такому уровню обработки текста, на котором уже возможно представление текста не просто в виде последовательности слов, а единым целым, обладающим неким смыслом, что уже соответствует человеческому восприятию.

Стремительное увеличение вычислительных мощностей сделало возможным применение трудоёмких лингвистических алгоритмов на больших объемах данных. Но, несмотря на то, что в области формализации естественных языков и систем АОТ, в частности, задействовано большое количество людей и мощностей, работающих в самых разных направлениях, результаты пока довольно скудны, так как ни одна из существующих моделей не может перекрыть структуру языка в целом, а объёмы данных, с которыми имеет дело лингвистика, очень большие.

Такое положение вещей само собой рождает задачу создания системы, удобной для отработки различных решений анализа с целью нахождения наиболее оптимальных и эффективных. Этому способствует то, что как сам анализ, так и программные комплексы, реализующие данные подходы, достаточно легко поддаются фрагментации, т.е. делению на функциональные блоки, выполняющие изолированную функциональность. Исходя из данной специфики проблемной области наиболее естественной задачей является создание модульного программного испытательного стенда, дающего возможность разрабатывать реализации отдельных функциональных блоков, применяя для каждого из них последние достижения в данной области, а затем исследовать их совместную работу путём точной настройки каждого из них в отдельности и гибкой компоновки между собой.

1. ОБЗОР СУЩЕСТВУЮЩИХ СИСТЕМ

В настоящее время лингвистами сформулированы различные теории, позволяющие в какой-то степени формализовать естественный язык. В основном, суть этих теорий сводится к тому, что предложению в тексте сопоставляются различные конечные объекты — графы, или, в общем случае, конечные модели, которые, как принято считать [1, 2], отражают смысл предложений.

1.1. Общие принципы систем обработки текстов

Компоненты, составляющие структуру систем анализа текстов — лингвистические процессоры, которые последовательно обрабатывают входной текст. Вход одного процессора является выходом другого [3, 4, 6].

Выделяются следующие компоненты:

- графематический анализ — выделение слов, цифровых комплексов, формул и т.д.;
- морфологический анализ — построение морфологической интерпретации слов входного текста;
- синтаксический анализ — построение дерева зависимостей всего предложения;
- семантический анализ — построение семантического графа текста.

Для каждого уровня разрабатывается свой язык представления. Язык представления, как полагается, состоит из констант и правила их комбинирования. На графематическом уровне константами являются графематические дескрипторы (ЛЕ — лексема, ЦК — цифровой комплекс и т.д.) На морфологическом уровне — граммемы (рд — родительный падеж, мн — множественное число). На синтаксическом — названия отношений (subj — отношение между подлежащим и сказуемым, circ — обстоятельство). О семантическом анализе будет сказано ниже.

Основой для построения уровней служат результаты работы предыдущих компонентов, но, что важно, последующие компоненты также могут улучшить представление предыдущих. Например, если для какого-то предложения синтаксический анализатор не смог построить полного дерева зависимостей, тогда, возможно, семантический анализатор сможет спроектировать построенный им семантический граф на синтаксис.

1.2. Графематический анализ

Графематический анализ — достаточно простой компонент, выполняющий первые предварительные действия над текстом. На вход компоненту подается текст, на выходе строится графематическая таблица, в которой на каждой строке стоит слово или разделитель из входного текста. Компонент выделяет некоторые аббревиатуры, имена с инициалами, даты и пр. Кроме деления текста на слова, компонент разбивает текст на абзацы и предложения (макросинтаксический анализ).

Графематическая таблица[1] состоит из двух столбцов. В первом столбце стоит некоторый кусок входного текста (выделенный по правилам, о которых будет сказано ниже), во втором столбце стоят графематические дескрипторы, характеризующие этот кусок текста. Например, для текста «Иван спал» будет построена таблица из трех строк:

Кусок входного текста	Графематические дескрипторы
Иван	ЛЕ Бб ПРД1
_	РЗД ПРБ
Спал	ЛЕ бб ПРД2

Так или иначе дескрипторы создают формальное описание текста на уровне графематики, которое уже поддаётся автоматизированной обработке в терминах лингвистических теорий.

1.3. Морфологический анализ

Морфологический компонент осуществляет морфоанализ и лемматизацию русских словоформ. Морфоанализ — приписывание словоформам морфологической информации, лемматизация — приведение текстовых форм слова к словарным. При лемматизации для каждого слова входного текста морфологический процессор выдает множество морфологических интерпретаций следующего вида:

- лемма,
- морфологическая часть речи,
- множество наборов граммем.

[1] Здесь и далее по тексту приводятся примеры реализаций, как это сделано в системе Диалинг [5].

Лемма — это нормальная форма слова. Например, для существительных — это единственное число (если оно есть у существительного), именительный падеж.

Граммема — это элементарный морфологический описатель, относящий словоформу к какому-то морфологическому классу, например, словоформе *стол* с леммой СТОЛ будут приписаны следующий набор граммем: «**мр, ед, им, но**», «**мр, ед, вн, но**». Таким образом, морфологический анализ выдает два варианта анализа словоформы *стол* с леммой СТОЛ внутри одной морфологической интерпретации: с винительным (**вн**) и именительным падежами (**им**).

Также большую роль здесь играет омонимичность словоформ. Например, у словоформы *стали* могут быть следующие интерпретации:

- сталь — существительное;
- стать — глагол.

Таким образом, видно, что морфологического анализа явно не достаточно для выбора одной конкретной морфологической интерпретации слова, к тому же, выбор одной интерпретации может повлиять на выбор интерпретации для соседних слов. Поэтому программы работают с целым набором возможных морфологических интерпретаций, постепенно выделяя наиболее вероятные на следующих этапах анализа.

1.4. Фрагментационный анализ

Фрагментационный анализ — деление предложения на неразрывные синтаксические единства (фрагменты), большие или равные словосочетанию (синтаксической группе), и установление частичной иерархии на множестве этих единств. Фрагменты — это главные и придаточные предложения в составе сложного, причастные, деепричастные и другие обособленные обороты. Иерархия отражает тот факт, что в предложении некоторые фрагменты синтаксически зависимы от других. Так, фрагмент «причастный оборот» будет подчиняться фрагменту, содержащему определяемое слово, придаточное предложение — главному.

Необходимость фрагментационного анализа в системе АОТ вызвана, в первую очередь, техническими причинами.

1.5. Синтаксический анализ

Следующим этапом, после морфологического и фрагментационного анализов, является этап синтаксической обработки текста. Цель синтакси-

ческого анализа — построение групп на предложении. Синтаксическая группа — это отрезок (первое слово группы — последнее слово группы) в предложении, для которого указан подотрезок — его главная группа. В частном случае группа — одно слово. Как видно из определения, синтаксические группы неразрывны, а из того, что две группы пересекаются, следует, что одна лежит в другой (т.е. является ее подотрезком).

Синтаксическую структуру предложения можно представить в виде дерева: корень (нулевой уровень) — само предложение; узлы — синтаксические группы (далее просто группы); листья — элементарные группы (слова); ребра — отношение «лежать непосредственно в» (A->B значит, что B лежит в A и при этом нет такой группы C, что B лежит в C и C лежит в A). До начала работы анализатора каждое слово — группа первого уровня (группы первого уровня не входят ни в какие группы кроме предложения) и кроме корня других групп нет. Результатом работы является «дерево» предложения, описывающее лингвистические отношения подчинения. По сути, это и есть математическая модель предложения на естественном языке.

2. ПОСТАНОВКА ЗАДАЧИ

В настоящее время ведутся активные исследования в области разработки алгоритмов анализа текстов. Результатом этих исследований являются десятки моделей и готовых алгоритмов, которым необходима проверка. При этом до сих пор не существует инструмента, предоставляющего удобные средства для разработки в данной области. Это вынуждает разработчика-лингвиста сосредотачивать внимание не только на написании алгоритма, но и на создании системы, способной запустить этот алгоритм, обеспечить его взаимодействие с остальными и предоставить необходимую информацию о его работе. Таким образом, главной задачей данной работы ставится создание исследовательского стенда для анализа текстов на естественном языке.

Важно отметить, что результатом работы должен быть законченный продукт, подходящий для применения его в качестве полноценного анализатора текстов, а именно стенд, предоставляющий необходимые функции для ведения исследований в области разработки алгоритмов анализа.

Система должна обеспечивать:

- возможность загрузки и редактирования анализируемых текстов;
- анализ текста посредством программируемого конвейера, составленного из разрабатываемых независимо компонентов;

- просмотр результатов анализа текста каждым из компонентов;
- обеспечение замера производительности работы компонентов и визуализацию этих данных;
- возможность независимой разработки компонентов анализатора с последующей возможностью включения в конвейер;
- функции работы со словарями — нахождение словарных статей, возможность создания и подключения новых словарей;
- приемлемое время работы

Отметим, что морфологический и синтаксический анализы производятся посредством использования внешних модулей (системы Диалинг).

В результате проведённого исследования современных технологий и средств разработки, был решён вопрос выбора инструментов для решения поставленной задачи. Кратко основные положения можно представить следующим образом.

- Совместимость с самыми современными технологиями и их использование:
  - язык реализации исследовательского стенда — C#[2];
  - описание и реализация бизнес-логики программируемых модулей анализатора.
- Расширяемость — исследовательский стенд предоставляет возможность изменять существующие блоки анализатора и создавать новые.
- Простота использования — использование графического представления для создания моделей компонентов анализа.
- Безопасность и защищённость — архитектура предоставляет возможность разрабатывать и подключать модули любой сложности (никак не ограничивая их внутреннюю структуру архитектурными особенностями), без предоставления при этом исходного кода.
- Поддержка языков — система позволяет использовать для разработки компонентов анализатора модули, написанные на следующих языках: C, C++, Managed C++, Pascal, Visual Basic и др.

[2] На момент написания программы для исполнения приложений на языке C# в среде операционной системы Microsoft Windows использовались ряд дополнениний.

Принципиальная схема архитектуры

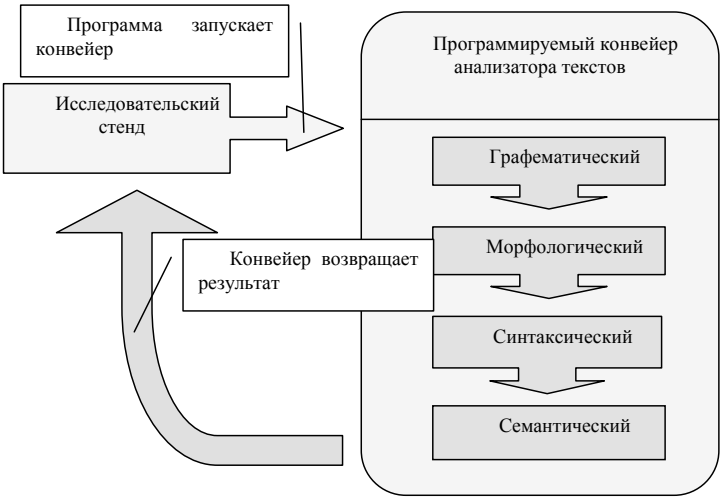


Рис. 1

Как уже упоминалось ранее, для реализации данной архитектуры (рис. 1) был выбран язык C#. Исследовательский стенд является Windows-приложением, предоставляющим функциональность работы со стендом со стороны пользователя, реализуя такие возможности, как загрузка, отображение и редактирование текста, запуск анализа текста и отображение результатов анализа.

Он взаимодействует с программируемым конвейером[3], который и является изменяемой компонентой программного комплекса. Программируемый конвейер предоставляет функциональность работы со стендом со стороны исследователя — разработчика алгоритмов анализа — реализуя такие возможности, как подключение модулей анализатора к программе, а также связывание их в единый конвейер и обеспечение всей функциональности, необходимой для их совместной работы.

[3] Программируемый конвейер — приложение, реализованное на основе технологии Microsoft Framework.

2.2. Приложение исследовательского стенда

Внешний вид пользовательского интерфейса представлен на рис. 2.

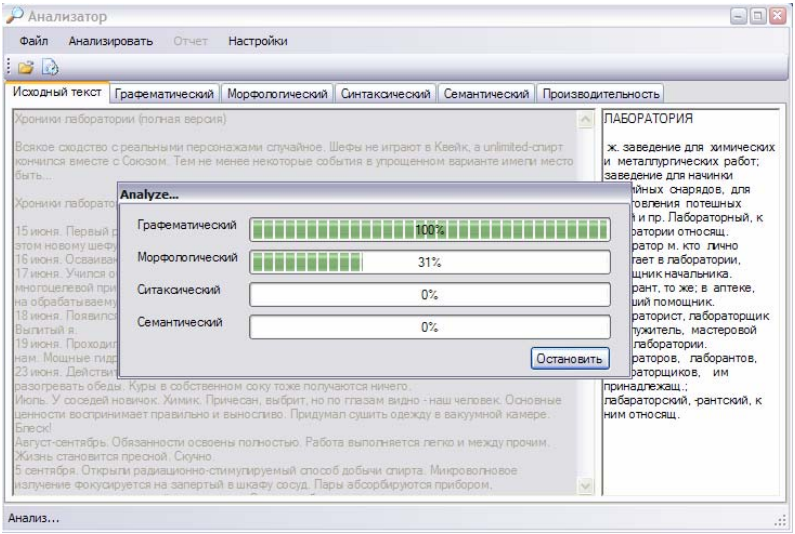


Рис. 2

2.2.1. Результаты оптимизации

Разработка данного приложения проводилась на основе предыдущих работ, проведённых в данном направлении. Для обеспечения приемлемой производительности были разработаны структуры хранения и управления данными.

Применение данных моделей привело к значительному ускорению работы программы. Для тестирования производились выборки одних и тех же слов из оптимизированных и неоптимизированных словарей (рис. 3).

Важно отметить, что в данном тесте не использовался механизм кэширования, т.е. и в оптимизированных словарях каждый раз происходило обращение к жесткому диску. При выборке же слов из памяти, время можно считать равным нулю.

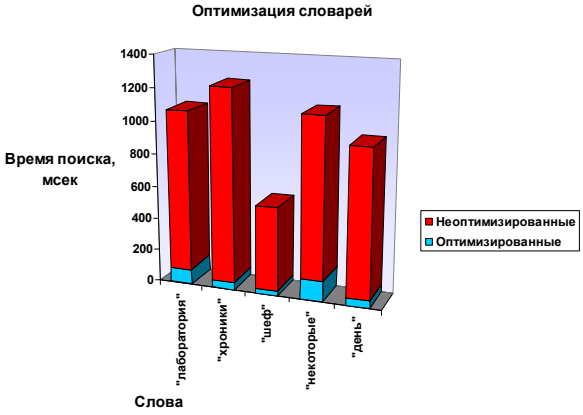
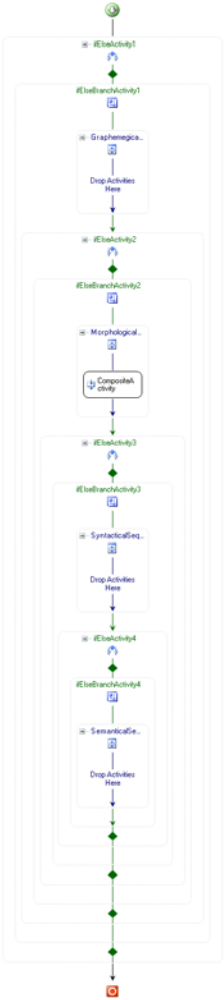


Рис. 3



2.3. Программируемый конвейер

Основной функцией данного программного модуля является предоставление конвейера анализатора в виде четырёх последовательно исполняемых модулей: графематического, морфологического, синтаксического и семантического. Эти модули реализуют логику, описанную в разд. 1. Эта часть программы реализована на основе технологии Microsoft Framework. Схема конвейера отображена на схеме.

Данная система позволяет разрабатывать элементы анализатора на уровне WYSIWYG. Для потенциальных пользователей, специалистов в области лингвистики, но не программистов, эта возможность, безусловно, имеет большое значение. Разработка анализаторов, с учётом возможностей данной технологии, сводится к написанию функциональных элементарных блоков и последующей их компоновке с использованием графического представления.

2.4. Блоки программируемого конвейера

В рамках работы были реализованы и протестированы два первых компонента конвейера анализатора.

2.4.1. Графематический анализатор

Это первый компонент программируемого конвейера анализатора текстов. Его задача описана в разд. 1.2. В рамках задачи данный компонент был реализован на основе Microsoft Framework. Результаты работы компонента можно увидеть на рис. 4.

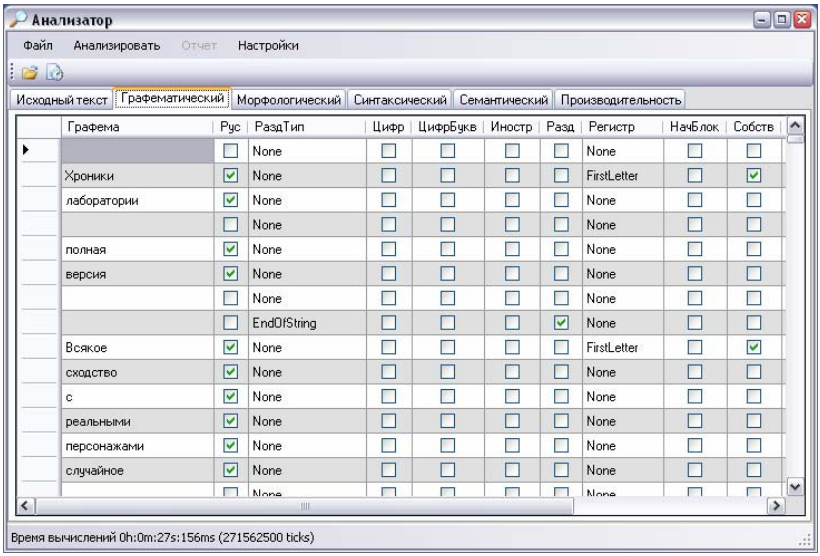
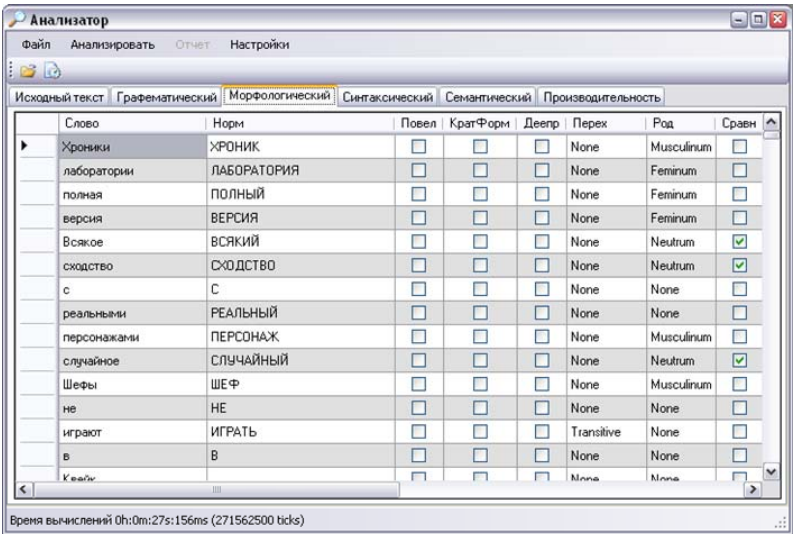


Рис. 4

2.4.2. Морфологический анализатор

Это второй блок программируемого конвейера анализатора текстов. Его задача описана в разд. 1.3. В рамках задачи данный компонент был также реализован на основе Microsoft Framework. Результаты работы компонента можно увидеть на рис. 5.



	Слово	Норм	Повел	КратФорм	Деепр	Перех	Род	Сравн
▶	Хроники	ХРОНИК	☐	☐	☐	None	Musculinum	☐
	лаборатории	ЛАБОРАТОРИЯ	☐	☐	☐	None	Feminum	☐
	полная	ПОЛНЫЙ	☐	☐	☐	None	Feminum	☐
	версия	ВЕРСИЯ	☐	☐	☐	None	Feminum	☐
	Всякое	ВСЯКИЙ	☐	☐	☐	None	Neutrum	☑
	сходство	СХОДСТВО	☐	☐	☐	None	Neutrum	☑
	с	С	☐	☐	☐	None	None	☐
	реальными	РЕАЛЬНЫЙ	☐	☐	☐	None	None	☐
	персонажами	ПЕРСОНАЖ	☐	☐	☐	None	Musculinum	☐
	случайное	СЛУЧАЙНЫЙ	☐	☐	☐	None	Neutrum	☑
	Шефы	ШЕФ	☐	☐	☐	None	Musculinum	☐
	не	НЕ	☐	☐	☐	None	None	☐
	играют	ИГРАТЬ	☐	☐	☐	Transitive	None	☐
	в	В	☐	☐	☐	None	None	☐
	Квейк		☐	☐	☐	None	None	☐

Рис. 5

ЗАКЛЮЧЕНИЕ

В результате работы был создан исследовательский стенд для анализа текстов на естественном языке. Более или менее успешно были решены все задачи, поставленные в рамках данной работы. Созданный инструмент уже на настоящий момент времени позволяет вести на нём работу по испытанию разрабатываемых лингвистических моделей. Это крайне важно, потому что именно практическое применение было одним из основных факторов, повлиявших на решение о разработке данного исследовательского стенда.

Выполненные работы по оптимизации и разработке удобного интерфейса позволили предоставить не только функциональный, но и комфортный интерфейс. А разработка гибкой архитектуры дала возможность разработчикам алгоритмов анализа самим выбирать как средства реализации (в том числе и основанные на современной технологии Microsoft Framework, так и способы отображения информации при диагностике алгоритмов.

Также в результате работы были теоретически проанализированы и технически оценены приоритетные направления развития данного программного комплекса. Среди которых можно выделить следующие.

- Стандартизация хранения словарных данных путём использования распространённых баз данных (также является шагом к оптимизации скорости работы).
- Обеспечение возможности использования Интернет в качестве информационной базы.
- Реализация механизма отождествления выражений на базе языка REFAL.
- Реализация компонента синтаксического анализа.
- Реализация системы распределённого анализа на основе данного программного комплекса.
- Создание интерфейса для возможности использования функций данного приложения другими программами в своих целях.
- Добавление функции исполнения скриптов — программ на псевдоязыке — позволяющее автоматизировано обрабатывать массивы текстов (анализ множества файлов, складирование результатов).
- Создание универсального формата отчётов о проведённых анализах с целью сравнения результатов работы различных алгоритмов.

Некоторые их этих задач могут лечь в основу дальнейшей исследовательской деятельности в данном направлении. Таким образом, помимо решения конкретной задачи, была подготовлена база для последующих работ и определены перспективы развития.

СПИСОК ЛИТЕРАТУРЫ

1. Мельчук И.А. Опыт теории лингвистических моделей типа «Смысл⇔Текст». — М.: Наука, 1974. — 315 с.
2. Леонтьева Н.Н. Система французско-русского автоматического перевода (ФРАП): лингвистические решения, состав, реализация // МП и ПЛ. Проблемы создания системы автом. перевода / Сб. научн. трудов МГПИИЯ им. М. Тореза. — Вып. 271. — М., 1987. — С. 6–25.
3. Леонтьева Н.Н. ПОЛИТекст: информационный анализ политических текстов // Сб. НТИ. — 1995. — Сер. 2, N 4.
4. Кудряшова И.М. О семантическом словаре в системе ФРАП // Сб. научн. трудов. — М.: МГПИИЯ им. М. Тореза, 1986. — Вып. 271. — 8 с.
5. Сокирко А.В. Семантические словари в автоматической обработке текста. // Канд. дисс., МГПИИЯ. — М., 2000. — 108 с.
6. Кулагина О.С. Исследования по машинному переводу. — М.: Наука, 1979. — 127 с.

В.Н. Касьянов

МУЗЕИ И ИНТЕРНЕТ

ВВЕДЕНИЕ

С появлением сети Интернет и развитием сетевых технологий музеи и другие учреждения культурного наследия начинают переосмысливать свои задачи и возможности. Все большее число музеев принимает решение поддерживать свой сайт (цифровой или электронный музей), чтобы расширить предоставление полезной информации о себе и привлечь новых пользователей. Поэтому для посещения любого из них уже не надо покидать собственной комнаты, не говоря даже о том, чтобы тратиться на поездку, покупать билеты на самолет и лететь в Москву, Париж, Лондон или Нью-Йорк. Для посещения величайших сокровищниц мировой культуры теперь достаточно иметь компьютер с выходом в Интернет, причем не надо скачивать никакого дополнительного дорогостоящего программного обеспечения или каким-либо образом менять настройки компьютера.

Преимущество цифрового музея понятно. Посетители цифрового музея могут наслаждаться культурными реликвиями без ограничений на время и место, и полная безопасность памятников культуры гарантирована. Более того, с помощью мультимедийного взаимодействия пользователи могут даже «прикасаться» к объектам и «манипулировать» ими, что может быть важным для профессионалов. Поэтому в настоящее время наряду с традиционными музеями широкое развитие получают электронные или виртуальные музеи различной тематики, доступные в среде Интернет, в том числе и по информатике [13–20, 23–24, 30–34, 36–42]. Так, поиск по запросу «Virtual Museum» в поисковике Google дает свыше 31 млн. результатов, а «Virtual Computer Museum» — 6300000. В рунете Яndex находит более 150 тыс. документов со словами «виртуальный музей». Не меньше ссылок можно также получить на запрос «электронная коллекция».

Те, кто хочет посетить существующие цифровые сайты российского Интернета, могут воспользоваться коллекцией ссылок на музейные сайты и виртуальные музеи, расположенной на сайте «Музеи России» [39]. Поискать цифровые сайты зарубежных музеев, можно воспользовавшись специализированным каталогом MUSEE [29].

Среди всех этих электронных музеев есть как вполне солидные и серьезные сайты известных музеев [36–38], так и довольно экзотичные музеи, такие как музей печали [33], музей «Белорусская соломка» [14] или музей сувенирных спичек [34]. Порой под названием «музей» в сети можно обнаружить подборку из нескольких текстов или фотоснимков [31].

Виртуальные музеи, являясь органической частью сети Интернет, своим присутствием в этой информационной среде открывают огромные возможности по развитию сети Интернет как среды по сохранению культурной истории, ее осмыслению и интерпретации, а также по формированию нового общественного сознания.

Современное развитие музейного дела становиться все более тесно связанным с сетью Интернет. Интернет открывает новые возможности для решения многих актуальных проблем реальных музеев. Он приводит к переосмыслению и совершенствованию музейной деятельности, в том числе и лежащей в основе музейной коммуникации знаковой системы, основанной на использовании предметных реалий культуры.

В данной статье мы остановимся на некоторых новых возможностях, связанных с представлением музеев в сети Интернет. В разд. 1 среди всех цифровых музеев мы выделяем так называемые музейные сайты и виртуальные музеи и описываем их основные свойства. Подходы к унификации доступа и интеграции музейных информационных ресурсов в сети Интернет с целью их объединения в единое музейное информационное пространство рассматриваются в разд. 2. Разд. 3 посвящён методам адаптивной гипермедиа, позволяющей создавать электронные музеи, настраиваемые на конкретные потребности каждого отдельного пользователя музея. В разд. 4 исследуются возможности музеев нового типа, так называемых открытых виртуальных музеев, в которых нет закрытых фондов и любой человек может быть не только посетителем, но и музейным работником. Разд. 5 содержит краткое описание работ по созданию открытого виртуального адаптивного музея по истории информатики в Сибири.

1. МУЗЕЙНЫЕ САЙТЫ И ВИРТУАЛЬНЫЕ МУЗЕИ

В большинстве случаев цифровые музеи, являющиеся представительством реальных музеев в сети Интернет, ориентируются на тех людей, которые придут в них лично, и предстают перед ними в виде своеобразных электронных буклетов или путеводителей, иногда дополненных обновляе-

мой информацией с музейного автоответчика. Мы их будем называть музейными сайтами.

Типовая структура российского музейного сайта включает разделы с информацией о возможностях посещения музея, с описанием истории музея и с презентацией основных коллекций и отдельных известных экспонатов, хранящихся в музее, а также разделы с презентациями постоянной экспозиции и выставок.

Разделы с информацией о возможностях посещения музея и описанием истории музея наиболее просты в подготовке, редко нуждаются в изменениях и могут ограничиваться чисто текстовым представлением. Хотя нужно отметить, что использование изображений при изложении истории или интерактивных схем проезда к музею не вызывает особых трудностей и весьма полезно. Что касается разделов, посвященных постоянной экспозиции музея и его коллекциям, то они тоже достаточно стабильны, но, как правило, являются мультимедийными презентациями, позволяющими посетителям ознакомиться со схемами музейных зданий, видами экспозиционных залов и отдельными экспонатами. Наиболее изменяемым разделом из перечисленных является раздел выставок, который по сути требует постоянного обновления для показа недавно открывшихся выставок и анонсирования будущих. Однако он также содержит существенную статическую часть — описание истории выставочной деятельности музея или архив уже прошедших выставок.

Помимо перечисленных выше разделов иногда сайты российских музеев содержат раздел образовательных программ, обычно состоящий из текстовых описаний экскурсий по музею и его разделам, а также раздел с научной информацией, как правило, имеющий вид набора персональных вебстраниц музейных сотрудников со справками о их научной деятельности и со списками публикаций.

Одним из основных отличий зарубежных музейных сайтов от российских является то, что они предоставляют посетителям не только информацию, но и возможность осуществления покупок. Каждый из них обязательно содержит музейный электронный магазин, где посетитель может осуществить свои покупки по сети (on-line), факсу или почте.

Помимо «классических» цифровых музеев, являющихся музейными сайтами, в сети существует громадное количество так называемых виртуальных музеев. Под виртуальным музеем в данном контексте мы понимаем репозиторий цифровых культурных и научных ресурсов, к которым есть доступ и которые могут использоваться в любое время и из любого места через Интернет. Это означает, что виртуальный музей — это веб-сайт (циф-

ровой музей), который может, но не обязан иметь в основе какой-нибудь один реальный музей и который содержит виртуальные экспонаты, являющиеся мультимедийными цифровыми представлениями произвольных артефактов без каких-либо ограничений на их природу или текущее состояние. Например, виртуальный экспонат может представлять уже исчезнувшею картину, самого художника, художественную школу или отдельное культурное событие.

Таким образом, мы видим, что музейный сайт является частью реального музея, в рамках которого этот музей реализует часть своих программ, в первую очередь, популяризационных. Он ориентируется на посетителей соответствующего реального музея и является лишь техническим средством для дополнительного предоставления музейной информации для них. В отличие от музейного сайта виртуальный музей является не дополнением к экспозиции реального музея, а ее электронным аналогом. Он создается не для реальных, а для виртуальных посетителей, многие из которых потому и прибегают к услугам сети Интернет, что не имеют возможности посетить музей лично.

Существуют музеи, сайты которых выходят за рамки простого представительства в сети и являются по существу полноправными виртуальными музеями. Ярким примером такого музея является сайт [35] одной из величайших сокровищниц мира, Государственного Эрмитажа.

Любой виртуальный посетитель сайта «Государственный Эрмитаж» может посетить раздел «Виртуальные выставки», где у него есть возможность познакомиться с экспонатами, хранящимися, в силу их ценности и редкости, в запасниках музея и поэтому зачастую являющимися вообще недоступными обычным посетителям музея.

Другая интересная возможность — это «Галерея увеличенных изображений», в рамках которой виртуальному посетителю предоставляется возможность рассмотреть в мельчайших деталях произведения искусства, которые обычным посетителям музея приходится рассматривать через стекло или из-за ограждения. Любой желающий может на сайте «Государственный Эрмитаж» совершить трехмерную интерактивную экскурсию по залам музея, знакомясь с панорамой залов и осматривая экспонаты музея с разных углов и при разном увеличении. Кроме того, для виртуальных посетителей доступна «Цифровая коллекция», позволяющая осуществлять поиск нужной картины не только по названию или автору, но и по содержанию изображения (расположению цветовых пятен на холсте), а затем рассматривать найденную картину как целиком в полноэкранном режиме (на выбор

800x600 или 1024x786), так и по частям (фрагментами), как если бы, находясь в зале, удалось приблизиться к картине с лупой в руках.

Главным недостатком виртуальных музеев считается то, что его виртуальные экспонаты, как правило, трудно признать оригиналами и легко подвергнуть копированию. Есть мнение, что виртуальный музей — это профанация, поскольку там, где нет подлинного музейного предмета, не может быть музея. Но что, например, является большей ценностью для большинства посетителей музея архитектуры: подлинные чертежи, выполненные для того, чтобы воплотиться в камне, или визуализация тех идей, которые они представляют, в виде изображений чертежей вместе с трехмерной реконструкцией архитектурного сооружения, дополненных представлением культурно-исторического контекста (воспоминаний, мифов, слухов и т.д.).

Посетитель виртуального музея может детально познакомиться с теми экспонатами (из фондов реального музея), которые в реальном музее ему совершенно недоступны и о наличие которых в музее он практически никак не может узнать. Здесь он может прочитать антикварные книги, прослушать уникальные пластинки и посмотреть фильмы из архивной фильмотеки. У него есть возможность детально изучить изображения картин, хранящихся под стеклом или за ограждением.

2. УНИФИКАЦИЯ И ИНТЕГРАЦИЯ СЕТЕВЫХ МУЗЕЙНЫХ РЕСУРСОВ

Важным преимуществом электронных музеев является также та простота, с которой пользователь осуществляет движение по музею и поиск экспонатов в нем. Вместе с тем, в настоящее время в сети Интернет представлено большое число разнородных и обособленных информационных ресурсов по культурному наследию (различные электронные музеи, архивы, коллекции и каталоги) самого разнообразного характера (описательные базы данных, базы данных изображений, видео- и аудиоматериалов и др.). Эти ресурсы принадлежат различным организациям, которые проводят самостоятельную политику в отношении их описания, использования и публичного доступа к ним. Большинство существующих информационно-поисковых систем поддерживают совершенно разные структуры хранения данных, способы доступа и форматы представления информации и, как следствие, имеют свой собственный пользовательский интерфейс.

С середины 80-х годов прошлого столетия в мире в области разработки распределенных информационно-поисковых систем по культурному наследию ведутся интенсивные исследования по унификации доступа и интегра-

ции информационных ресурсов. Цель — объединение имеющихся информационных ресурсов по культурному наследию в единую распределенную информационную систему со сквозным поиском. К настоящему времени уже разработана технология, основанная на международных стандартах ANSI/NISO Z39.50 (ISO 23950) [12, 22, 25] и профиле CIMI [3, 22]. Стандарт Z39.50 позволяет унифицировать сетевой доступ к любым базам данных, а профиль CIMI регламентирует доступ к информации о культурном наследии по протоколу Z39.50. Информационные системы по культурному наследию, организованные на основе Z39.50 серверов с использованием CIMI-профиля, становятся независимыми от конкретных систем хранения данных и, следовательно, могут быть интегрированы с другими подобными системами.

Стандарт Z39.50 (Information Retrieval (Z39.50): Application Service Definition and Protocol Specification) определяет прикладную службу и спецификацию протокола для поиска и извлечения информации из баз данных. Он предназначен для унификации сетевого доступа к базам данных и определяет процедуры поиска, извлечения и форматы представления информации.

Первая версия протокола Z39.50 была подготовлена Комитетом организации по национальным информационным стандартам США — NISO (National Information Standards Organization) и введена в 1988 г. стандартом Z39.50-1988, действие которого распространялось только на работу с библиографической информацией. В 1989 г. было организовано Агентство поддержки протокола Z39.50 (Maintenance Agency Z39.50) под административным управлением лаборатории Конгресса США, а в 1990 г. сформирована Группа исполнителей Z39.50 (Z39.50 Implementors Group — ZIG), членами которой стали производители, продавцы, распространители разнородных видов информации (библиографической, финансовой, химической и др.). Агентство поддержки Z39.50 является постоянно действующим органом, занимающимся сопровождением и развитием этого стандарта и поддерживающим свой сайт [12], на котором находятся вся информация по протоколу, новости, документация, реестры объектов и т.п. В 1992 г. указанными организациями была разработана версия 2 стандарта (Z39.50-1992), заменившая стандарт 1988 г. Версия 3 стандарта (Z39.50-1995) была разработана в 1995 г. Поскольку стандарт Z39.50-1995 является развитием версии 1992 г. и совместим с ней, он определяет протокол версий 2 и 3. В 1995 г. протокол Z39.50 был принят как американский национальный стандарт ANSI (ANSI/NISO Z39.50-1995), а в ноябре 1998 г. — как международный стандарт ISO-23950. Стандарт ANSI/NISO Z39.50-2003, действующий в

настоящее время, был утвержден в ноябре 2002 г. Он является технической переработкой стандарта ANSI/NISO Z39.50-1995 и также определяет версии 2 и 3, но дополнительно включает различные пояснения, исправления и соглашения, рекомендованные группой ZIG.

В первые годы своего существования протокол Z39.50 использовался преимущественно для организации доступа к библиографическим ресурсам, на сегодняшний день область его применения существенно расширена. Сейчас он применяется для доступа к научно-технической и финансовой информации, к геоинформационным ресурсам, к глобальным базам метаданных, тезаурусам и рубрикаторам, а также к цифровым коллекциям и музейной информации.

Профиль CIMI служит спецификацией использования Z39.50 для поиска и извлечения информации о культурном наследии. Он определяет подмножество характеристик, опций и параметров Z39.50, необходимых для поддержки функциональности и требований пользователя при поиске и извлечении информации о культурном наследии. Элементы этого профиля имеют глобальные идентификаторы и являются частью международного стандарта ISO-23950.

Профиль CIMI был разработан в 1998 г. Консорциумом по компьютерному обмену музейной информацией — CIMI (Consortium for the Computer Interchange of Museum Information) [3]. Это некоммерческая инициатива, занимающаяся развитием коммуникативных стандартов, сохранением и обменом музейной информацией в электронном виде. При создании профиля рабочая группа CIMI Z39.50 объединила вместе экспертов по Z39.50, экспертов в области музейного дела и музейной информации, разработчиков программного обеспечения и специалистов в области коммерции.

Еще одним показателем процесса унификации и интеграции сетевых музейных информационных ресурсов является появление открытых для доступа специалистов всех стран единых национальных баз данных по музейным коллекциям. Например, такая канадская база CHIN содержит информацию о нескольких миллионах музейных предметов всех музеев Канады.

3. АДАПТИВНАЯ ГИПЕРМЕДИА

Подавляющее большинство из представленных в настоящее время в сети Интернет виртуальных музеев представляет собой электронные публикации, основанные на использовании традиционных гипермедиа-

технологий. Одним из ограничений традиционных гипермедиа-систем является то, что они предоставляют одно и то же информационное содержание и один и тот же механизм навигации всем пользователям. Вместе с тем, виртуальные музеи, не ограничивают круг своих пользователей, и посетители музея с различными предпочтениями, целями, знаниями, интересами могут нуждаться в различных частях содержащейся информации и использовать различные пути для навигации.

Начинают появляться публикации, в которых обсуждается, как конструировать веб-сайты, чтобы информация, содержащаяся на музейном сайте, могла настраиваться на потребности конкретного пользователя. В статье [10] описывается использование технологии порождения естественного языка при построении персональных виртуальных каталогов для различных приложений, включая цифровые музеи. В статье [4] описывается интеллектуального помечающего обозревателя (ILEX) — системы, которая динамически генерирует персонифицированные текстовые метки в музейной галерее драгоценностей. Статья [11] описывает переносную систему (адаптивный музейный гид, реализованный на карманном компьютере), который предоставляет посетителю персональную навигационную помощь и информацию о посещаемых объектах.

Адаптивная гипермедиа (adaptive hypermedia) возникла в начале 90-х годов прошлого столетия как альтернатива традиционному подходу «стричь всех под одну гребенку» в разработке гипермедиа-систем [2]. *Адаптивные гипермедиа-системы* (АГС) строят модели целей, предпочтений и знаний каждого своего индивидуального пользователя и используют эту модель во время взаимодействия с пользователем для того, чтобы адаптироваться под потребности этого пользователя. Цель — персонализация гипермедиа-систем, их настройка на особенности индивидуальных пользователей. Таким образом, каждый пользователь имеет свою собственную картину и индивидуальные навигационные возможности при работе с адаптивной гипермедиа-системой.

Поддержка адаптивных методов в гипермедиа-системах оказывается весьма полезной в тех случаях, когда имеется одна система, обслуживающая множество пользователей с различными целями, уровнем знаний и опытом, и когда лежащее в ее основе гиперпространство является относительно большим.

Например, обучающие гипермедиа-системы, в которых пользователь или ученик имеет конкретную цель обучения (включая и такую цель, как общее образование), являются типичным приложением адаптивных гипермедиа-систем. В этих системах основное внимание уделяется знаниям обу-

чающихся, которые могут сильно различаться. Состояние знаний изменяется во время работы с системой. Таким образом, корректное моделирование изменяющегося состояния знаний, надлежащее обновление модели и способность делать правильные заключения на базе обновленной оценки знаний являются важнейшей составляющей обучающей гипермедиа-системы. Другим важным приложением являются онлайновые информационные системы, а также онлайновые справочные системы. К онлайновым информационным системам относятся, например, электронные энциклопедии, хранилища документов или туристические справочники. Чтобы выдать правильную информацию пользователям с различным уровнем квалификации, этим системам также требуется модель знаний пользователя. Важен также контекст запроса: нужна ли информация пользователю для краткой справки, для разработки презентации, для освежения знаний? Онлайновые справочные системы принимают во внимание конкретную среду, например, место вызова (контекстно-зависимые справочные системы). Для ограничения вариантов навигации адаптивные гипермедиа-системы могут быть объединены со средствами поиска информации в гипермедиа-системы с поиском информации. Ссылки в таких системах не вводятся автором системы, а формируются на основе сходства: ссылка между двумя документами устанавливается в том случае, если оба документа удовлетворяют некоторому условию похожести.

Подробное рассмотрение различных областей приложения адаптивных гипермедиа-систем, а также методов и технологий адаптивной гипермедиа можно найти в работах [2, 21, 27].

АГС-системы в общем случае поддерживают следующие три вида адаптации: к данным пользователя, к рабочим характеристикам и к данным окружения. Данные пользователя включают такие различные его характеристики, как знания, цели, подготовку, опыт в гиперпространстве, предпочтения, интересы и индивидуальные особенности пользователя. Рабочие характеристики включают данные о взаимодействии пользователя с системой, которые не могут быть сведены к характеристикам пользователя, но все еще могут использоваться для принятия решений адаптации. Данные окружения включают все аспекты пользовательского окружения, которые не связаны с пользователями (например, аппаратные средства, программное обеспечение, пропускная способность сети или местонахождение пользователя).

АГС-системы обеспечивают *адаптивное представление* (adaptive presentation), т. е. адаптацию содержания гипердокументов, и *адаптивную поддержку навигации* (adaptive navigation support), т. е. адаптацию структу-

ры гиперссылок. Смысл методов адаптивного представления состоит в том, чтобы адаптировать содержание страницы, к которой в данный момент обращается пользователь, к его текущему знанию, предпочтениям, интересам, целям и другим характеристикам пользователя. Основные методы адаптивного представления текста — это дополнительные, предварительные и сравнительные объяснения, варианты объяснения и сортировка. Следующие технологии используются для реализации выше перечисленных методов адаптивного представления текста: условный текст, стрейч-текст, варианты фрагментов и варианты страниц, фреймовая технология. Смысл методов адаптивной навигационной поддержки состоит в том, чтобы помочь пользователям найти путь в гиперпространстве с помощью адаптации способа представления ссылок к целям, знанию и другим характеристикам индивидуального пользователя. Методы адаптивной навигационной поддержки используются для достижения нескольких целей адаптации: обеспечить глобальное руководство, локальное руководство, глобальную ориентацию, локальную ориентацию и управление индивидуализированными представлениями в информационных пространствах. Для достижения этих целей применяются следующие технологии: полное руководство, сортировка ссылок, сокрытие ссылок, аннотирование ссылок, генерирование ссылок и адаптация карты.

На абстрактном уровне АГС состоит из следующих трех компонентов: модель предметной области, модель пользователя и модель адаптации. *Модель предметной области* (МО) представляет собой описание информационного содержания и структуры ссылок предметной области на концептуальном уровне, например, используя представление множеств концептов и концептуальных связей в виде ориентированного ациклического графа. *Модель пользователя* (МП) представляет предпочтения, знания, цели, историю навигации и возможно другие релевантные аспекты пользователя, информацию о которых система получает в явном виде от пользователя или неявном — посредством отслеживания взаимодействий пользователя с системой. Основной частью МП является представление знаний пользователя предметной области посредством концептов МО, например, с помощью оверлейной модели. Основой адаптивной функциональности АГС служит *модель адаптации* (МА). Она состоит из правил адаптации, которые формируют связь между МО и МП и определяют представление генерируемой информации.

Адаптивные виртуальные музеи поддерживают модель пользователя и используют ее для своей настройки на потребности конкретного пользователя. Например, они позволяют организовывать выставки по запросу лю-

бого своего посетителя и с учетом его индивидуальных интересов, а также проводить экскурсии для каждого отдельного посетителя с учетом его интересов, предпочтений и ограничений (таких, как время).

4. ОТКРЫТЫЕ ВИРТУАЛЬНЫЕ МУЗЕИ

С точки зрения посетителей музея реальный музей — это его экспозиция, среда для экскурсий и выставок. С другой стороны, музей — это институт культурного наследия, обладающий огромными фондами, закрытыми для обычных посетителей, и предназначенный для поддержки коллекционирования, исследований, создания каталогов и выставок артефактов, но посетители музеев не могут участвовать в этой важной музейной работе.

Мы полагаем, что виртуальные цифровые музеи могут и должны быть открытыми для своих посетителей: дать им доступ по сети ко всем своим фондам и разрешить им все виды музейной работы через Интернет. В частности, было бы полезным, чтобы пользователь всегда мог предложить представление некоторого реального артефакта в открытый виртуальный музей в качестве виртуального экспоната. Кроме того, открытый виртуальный музей может также иметь средства для снабжения экспонатов авторскими описаниями, для предложения авторских экскурсий по музею, а также для создания авторских выставок. Эти возможности особенно важны для музеев современной истории.

Открытый виртуальный музей (open virtual museum) — это гипермедиа-система, которая предназначена быть одновременно и доступным репозиторием для коллекций артефактов, и институтом культурного наследия, поддерживающим коллективную работу широкого круга людей по коллекционированию, аннотированию, организации, исследованиям, созданию каталогов и выставок этих артефактов [8]. Таким образом, открытый виртуальный музей ориентируется не только на виртуальных посетителей, но и виртуальных музейных работников.

Открытые виртуальные музеи могут круглосуточно поддерживать доступность и активное использование цифровых культурных и научных ресурсов для всех людей планеты без каких-либо ограничений на их местожительство, национальность, образование, возраст и т.д.

5. ВИРТУАЛЬНЫЙ МУЗЕЙ ИСТОРИИ ИНФОРМАТИКИ В СИБИРИ

Хотя понятие открытого виртуального музея было впервые сформулировано в 2005 г., проект по созданию первого открытого музея стартовал существенно раньше. В 2001 г. при поддержке РГНФ в ИСИ СО РАН начались работы над виртуальным адаптивным музеем SVM по истории информатики в Сибири, и его основой послужили работы авторов проекта SVM над страницами по истории информатики в Сибири, размещенные ими в рамках системы поддержки гуманитарных исследований SIMICS [9].

Информатика сформировалась как наука в середине 50-х годов прошлого столетия и менее чем за полувековой период шагнула далеко вперед. С годами от нас уходят активные участники и свидетели ее первых шагов, многое забывается, становится труднодоступным или безвозвратно утерянным. Постоянное развитие информатики и сверхмощное давление зарубежной вычислительной науки усиливают этот процесс, и нужны целенаправленные действия, чтобы богатый отечественный опыт не забывался и мог быть востребован. Без понимания прошлого трудно двигаться вперед.

Нужно сказать, что исследования по истории информатики в передовых странах мира ведутся достаточно широко. Вместе с тем, до недавнего времени история информатики в бывшем Советском Союзе была практически неизвестна на Западе, хотя отдельные работы, посвященные этим вопросам, публиковались [6–7], а в 1996 г. компьютерное общество IEEE Computer Society, в связи с 50-й годовщиной своего основания, наградило самой престижной медалью «Computer Pioneer» Виктора Михайловича Глушкова, Сергея Алексеевича Лебедева и Алексея Андреевича Ляпунова. Одновременно этой награды был также удостоен ряд ученых из стран Восточной Европы (Григоре Моисил, Иван Пландер, Антонин Свобода и другие). Медаль «Computer Pioneer» была учреждена в 1981 г., чтобы признать и представить общественности выдающихся ученых, усилиями которых создавалась и развивалась сфера вычислительной и информационной науки и техники. Среди 55 лауреатов этой награды такие замечательные личности, как Артур Беркс, Джон Маккарти, Марвин Минский, Никлаус Вирт, Хайнц Земанек. Согласно формулировке награждения В.М. Глушков «основал первый в СССР Институт Кибернетики на Украине, разработал теорию цифровых автоматов и компьютерной архитектуры, а также рекурсивный макроконвейерный процессор», С.А. Лебедев «разработал и построил первый советский компьютер и основал советскую компьютерную промышленность», А.А. Ляпунов «разработал теорию операторных методов для абст-

рактного программирования и основал советскую кибернетику и программирование» [5].

Начало работ по программированию и информатике в Сибирском отделении АН СССР относится к моменту приезда в новосибирский Академгородок в начале 60-х годов прошлого столетия А.А. Ляпунова и его ученика — Андрея Петровича Ершова. Сибирская школа информатики и программирования была третьей по значимости в СССР после школ Москвы и Киева и, несмотря на сегодняшние трудности, переживаемые наукой и образованием в России, продолжает играть свою роль и поныне, более чем через десять лет после смерти ее основателя Андрея Петровича Ершова [1, 43]. Это позволяет самостоятельно исследовать становление и развитие информатики в Сибири, точнее в Новосибирском научном центре, на фоне российского и мирового процессов.

Основная цель создания виртуального музея SVM — это сохранение историко-культурного наследия, связанного с созданием и развитием информационных ресурсов, являющихся важнейшим национальным богатством, а также обеспечение свободного повсеместного доступа к ним с целью повышения общеобразовательного и культурного уровня широких слоев населения. Он предназначен для использования различными категориями пользователей, и посетители музея с различными предпочтениями, целями, знаниями, интересами могут нуждаться в различных частях содержащейся информации и использовать различные пути для навигации. Поэтому при создании музея истории информатики в Сибири авторы проекта уделили особое внимание вопросам адаптации. Другое важное отличие музея SVM от традиционных (виртуальных и обычных) состоит в том, что авторы проекта с самого начало стремились сделать его открытым: предоставить широкому кругу пользователей удобные возможности по пополнению и развитию музея. Поэтому в виртуальном музее SVM пользователи могут не только пополнять экспонатами музей и высказывать предложения и замечания, но и создавать свои авторские экскурсии и экспозиции.

Минимальной структурной единицей музея SVM является *экспонат*; в качестве экспонатов выступают следующие объекты: ученые-информатики, коллективы, документы архива, публикации, проекты, события, конференции и вычислительная техника. Типы экспонатов могут пополняться.

Следующими структурными единицами являются экскурсии и экспозиции, образованные из множеств экспонатов, объединенных по тематическому, хронологическому или типологическому критериям. *Экскурсия* — это протекающий во времени рассказ о музее, в ходе которого происходит демонстрация экспонатов в определенной последовательности. В отличие

от экскурсии *экспозиция* предполагает, что с составляющими ее экспонатами посетитель знакомится сам, причем только в режиме on-line. Обычно экспозиция предоставляет пользователю несколько способов навигации, в том числе возможность свободно перемещаться по множеству экспонатов.

Следующей структурной единицей музея является зал. В общем случае, *зал* представляет собой совокупность экспонатов одного типа, при этом каждому типу экспонатов соответствует одноименный зал. В музее имеются *открытые* залы, доступные для просмотра всем посетителям, и *запасники* — залы, доступные только для зарегистрированных пользователей. Открытые залы содержат зал экспозиций и зал экскурсий, а запасники включают следующие залы: библиотеку, архив, хронику событий, зал ученых-информатиков, зал коллективов, зал проектов, зал вычислительной техники, зал конференций, зал новых поступлений и зал подготовки экспозиций и экскурсий.

В библиотеке собраны книги, монографии, сборники статей, учебные и методические пособия, статьи из научных журналов, тезисы конференций и т.д. Архив представляет собой совокупность текстовых, графических, звуковых и видеоматериалов. Хроника событий включает описания наиболее выдающихся событий из истории развития информатики в Сибири. Зал информатиков содержит информацию о наиболее выдающихся ученых-информатиках, включая биографии, основные печатные труды и достижения, фото и пр. В зале коллективов содержатся данные о коллективах: группах, лабораториях и институтах. В зале проектов размещены данные о проектах, создаваемых в рамках работ по информатике (темы, системы). В зале вычислительной техники расположены экспонаты, имеющие отношение к вычислительной технике, которая использовалась и разрабатывалась с начала создания Сибирского отделения Академии наук. Зал конференций содержит информацию о различных научных мероприятиях. Новые экспонаты, добавляемые пользователями музея, помещаются в зал новых поступлений. В зале подготовки экспозиций и экскурсий размещаются экспозиции и экскурсии, создаваемые пользователями музея.

Интерфейс музея разрабатывается с учетом его использования различными категориями пользователей. Все пользователи музея подразделяются на две основные категории: *незарегистрированные* пользователи («посетители») и *зарегистрированные* пользователи («специалисты»), которые различаются по правам доступа к информационным ресурсам.

«Посетители» имеют только возможность просмотра информации, которая открыта для публичного доступа (например, в виде экскурсий и экспозиций). «Специалистам» доступны для просмотра все имеющиеся в музее

информационные ресурсы, включая информацию запасников, закрытую для публичного доступа. Все «специалисты» подразделяются на две группы в зависимости от уровня прав доступа к ресурсам: группу «простых специалистов», работающих только в зале новых поступлений, и группу «музейных работников».

В группе «простых специалистов» выделяются «волонтеры», имеющие права на добавление новых экспонатов, а также «экскурсоводы» и «экспозиторы», которые могут создавать собственные экскурсии и экспозиции. Добавленные или созданные ими объекты сначала помещаются в зал новых поступлений, впоследствии администраторы соответствующих ресурсов принимают решение об их включении в музей. Волонтеры, экскурсоводы и экспозиторы не имеют прав на редактирование информационных ресурсов музея.

Группу «музейных работников» можно представить в виде иерархической структуры, на самом верху которой находится «директор» (или «главный администратор»), обладающий полными правами на администрирование всех информационных ресурсов музея, включая администрирование пользователей музея. На втором уровне иерархии находятся администраторы отдельных ресурсов музея, которые назначаются «директором»: «главный экспозитор», «главный экскурсовод», «главный библиотекарь», «главный архивариус», «главный хронолог», «главный биограф», «главный коллективовед», «главный проектант», «главный инженер», «главный секретарь». Администраторы ресурсов имеют полные права на администрирование соответствующих типов ресурсов. В их полномочия также входит администрирование специалистов, работающих с соответствующими типами ресурсов. Третий уровень иерархической структуры включает «музейных работников», назначаемых администраторами соответствующих типов ресурсов: «библиотекарей», «архивариусов», «хронологов», «биографов», «коллективоведов», «проектантов», «инженеров», «секретарей». Они имеют ограниченные права на редактирование соответствующих типов ресурсов.

К настоящему времени разработан и реализован пользовательский интерфейс виртуального музея SVM, который реализует функции управления информационными ресурсами и пользователями, и начата работа по наполнению музея. Интерфейс управления информационными ресурсами предназначен для обеспечения механизма навигации и просмотра информации, поиска, ввода и редактирования информационных ресурсов. Интерфейс управления пользователями служит для регистрации, аутентификации, авторизации и администрирования пользователей.

ЗАКЛЮЧЕНИЕ

Будущее развитие музеев становится все более тесно связанным с развитием сети Интернет и музейных сайтов. Уже сегодня количество виртуальных гостей музеев сопоставимо с числом их реальных посетителей. Музейные сайты открывают перед музеями дополнительные возможности для презентации своих коллекций и их интеграцию в мировую музейную систему. Информация, размещенная на музейных сайтах, становится доступной громадной аудитории людей (в том числе специалистам, работающим в разных музеях), которые получают возможность сопоставлять музеи друг с другом, оценивать претензии на приоритеты, выявлять аналоги, находить партнеров и т.д.

Тенденция развития такова, что все большее число музейных сайтов начинают жить по законам Интернета, все более открывая себя для свободного и активного обращения с ними виртуальных посетителей и вовлекая все большее число людей в процессы комплектования, хранения, изучения и популяризации артефактов, представляющих материальную и духовную историю в виртуальных музеях.

Виртуальные музеи нового типа начинают формировать глобальную виртуальную музейную среду, доступную для всех и настраиваемую на нужды каждого. Эта среда позволяет каждому из нас быть как посетителем музея, так и музейным работником и является инструментом структурной гармонизации, согласования и корректировки наших спонтанных интересов и проектов в области сохранения культурного наследия.

СПИСОК ЛИТЕРАТУРЫ

1. Bjørner D., Kotov V. Images of Programming. Dedicated to the Memory of A.P. Ershov. — Amsterdam: North-Holland, 1991.
2. Brusilovsky P. Adaptive hypermedia. // User Modelling and User-Adapted Interaction. — 2001. — Vol. 11, N 1. — P. 87–110.
3. CIMI — консорциум по компьютерному обмену музейной информацией (Consortium for the Computer Interchange of Museum Information). — <http://www.cimi.org>
4. Cox R., O'Donnell M. and Oberlander J.: Dynamic versus static hypermedia in museum education: an evaluation of ILEX, the intelligent labelling explorer // Proc. of the 9th Internat. Conf. on Artificial Intelligence and Education, Le Mans, 1999. — P.181—188.
5. CS Recognizes Pioneers in Central and Eastern Europe // IEEE Computer. — 1998. — N 6. — P. 79–84

6. Ershov A.P. A history of computing in the USSR // Datamation. — 1975. — Vol. 21, N 9. — P. 80–88.
7. Ershov A.P., Shura-Bura M.R. The early development of programming in the USSR // A History of Computing in the Twentieth Century. — New York: Acad. Press, 1980. — P. 137–196.
8. Kasyanov V.N. SVM — Siberian Virtual Museum of Informatics History // Innovation and the Knowledge Economy: Issues, Applications, Case Studies. — Amsterdam: IOS Press, 2005. — Part 2. — P.1014–1021.
9. Kasyanov V.N. SIMICS: information system on informatics history // Proc. of Intern. Conf. on Educational Uses of Information and Communication Technologies (ICEUT). 16th IFIP World Computer Congress. — Beijing, PHEI, 2000. — P.168.
10. Milosavljevic M. Electronic Commerce via Personalised Virtual Electronic Catalogues // Proc. of 2nd Annual CollECTeR Workshop on Electronic Commerce (CollECTeR'98), Sydney, 1998. — <http://www.dynamicmultimedia.com.au/papers/collecter98/>.
11. Not E., Petrelli D., Sarini M., Stock O., Strapparava C. and Zancanaro M. Hypernavigation in the physical space: adapting presentation to the user and to the situational context, New Review of Multimedia and Hypermedia. — 1998. —Vol. 4. — P. 33—45.
12. Агентство поддержки протокола Z39.50 (Maintenance Agency Z39.50) . — <http://lcweb.loc.gov/z3950/agency>
13. Архив академика А. П. Ершова. — <http://ershov.iis.nsk.su>
14. Белорусская соломка. — <http://straw.iatp.by>
15. Виртуальный компьютерный музей. Проект Эдуарда Пройдакова. — <http://www.computer-museum.ru/>
16. Виртуальный музей авиации. — <http://avia.russian.ee>
17. Виртуальный музей Диего Риверы. — <http://www.usaccess-inc.com/museum/diego-home-eng.htm>
18. Виртуальный музей истории информатики в Сибири. — <http://pco.iis.nsk.su/svm>
19. Виртуальный музей компьютерной техники. — <http://museum.iu4.bmstu.ru/project.shtml>
20. Виртуальный школьный музей информатики. — <http://schools.keldysh.ru/sch444/MUSEUM>
21. Волянская Т.А. Методы и технологии адаптивной гипермедиа // Современные проблемы конструирования программ. — Новосибирск, 2002. — С. 38–68.
22. Волянская Т.А. Международные стандарты представления в сети Интернет информационных ресурсов по культурному наследию: стандарт ANSI/NISO Z39.50 и профиль CIMI // Программные средства и математические основы информатики. — Новосибирск, 2004. — С. 7—42.
23. Дармштадский виртуальный музей информатики. — <http://www.fbi.fh-darmstadt.de/~vmi>

24. Европейский музей компьютерной науки и техники. Экспозиция по истории компьютерной науки и техники в Украине. — http://www.icfcst.kiev.ua/museum/
25. Жижимов О.Л. Введение в Z39.50. — Новосибирск: Изд-во НГОНБ, 2000.
26. Канадская база данных CHIN по музейным коллекциям. — <http://www.chin.gc.ca>
27. Касьянов В.Н, Касьянова Е.В. Адаптивные системы и методы дистанционного обучения // Информационные технологии в высшем образовании. — 2004. — Т.1, N 4. — С. 40—60.
28. Касьянов В.Н., Несговорова Г.П., Волянская Т.А. Виртуальный музей истории информатики в Сибири // Проблемы программирования. — 2003. — N 4. — С. 82—91.
29. Каталог ссылок MUSEE на музейные сайты зарубежных музеев. — <http://www.musee-online.org/directo.htm>
30. Манчестерский виртуальный музей вычислений. — <http://www.computer50.org/kgill/>
31. Музей деревянного зодчества «Малые Карелы» . — <http://karely.narod.ru>
32. Музей Николая Рериха. — <http://www.roerich.ru>
33. Музей печали. — <http://sorrow.hotmail.ru>
34. Музей сувенирных спичек. — <http://matches.yaroslavl.ru>
35. Очерки по истории советской вычислительной техники и школ программирования. — <http://www.osp.ru/museum>
36. Сайт «Британский музей». — <http://www.metmuseum.org/home.asp>
37. Сайт «Государственный Эрмитаж» . — <http://www.hermitage.ru/>
38. Сайт «Лувр». — <http://www.louvre.fr/louvre.htm>
39. Сайт «Музеи России». — <http://www.museum.ru/>
40. Сайт «Русский музей». — <http://www.rusmuseum.ru/>
41. Сайт «Третьяковская галерея». — <http://www.tretyakov.ru/>
42. Сайт Метрополитен-музея. — <http://www.museum.ru/>
43. Специальный выпуск памяти академика Андрея Петровича Ершова. Программирование. — 1990. — N 1.

Е.В. Касьянова

АДАПТИВНАЯ СИСТЕМА ПОДДЕРЖКИ ДИСТАНЦИОННОГО ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ

ВВЕДЕНИЕ

В не столь далеком прошлом хороший почерк уже являлся гарантией спокойной и обеспеченной жизни до старости. Последние десятилетия характерны ускорением обновляемости технологий и знаний в различных сферах деятельности человека. Поэтому школьного и даже вузовского образования надолго уже не хватает. Сегодня особенно актуальна концепция непрерывного образования на протяжении всей жизни или, как говорят, пожизненного обучения (long-life education). Поиск соответствующей организационной структуры и учреждений образования (особенно образования взрослых), которые обеспечили бы переход от принципа «образование на всю жизнь» к принципу «образование через всю жизнь» — важнейшая проблема XXI века. Открытое образование — это образование, доступное всем. Его развитие неизбежно приведет к существенному пересмотру традиционных методик и технологий учебного процесса, а также к формированию единого открытого образовательного пространства.

Системы дистанционного обучения в настоящее время активно исследуются и развиваются и уже успели пройти путь в пять поколений, начиная от систем обучения по переписке, больше известных в СССР как системы заочного обучения, и кончая системами гибкого обучения и интеллектуального гибкого обучения, определяющими настоящее и будущее дистанционного образования и базирующимися на Web-технологиях.

Выгоды сетевого обучения ясны: аудиторная и платформенная независимости. Сетевое обучающее программное обеспечение, один раз установленное и обслуживаемое в одном месте, может использоваться в любое время и по всему миру тысячами учащихся, имеющих компьютеры, подключенные к Интернету. Тысячи программ сетевого обучения и других образовательных приложений стали доступны в сети за последние годы. Проблема состоит в том, что большинство из них являются не более чем статичными гипертекстовыми страницами.

Появившиеся в последнее время *адаптивные гипермедиа-системы* существенно повышают возможности обучающих систем [1–3]. Целью адап-

тивных систем является *персонализация* гипермедиа-системы, ее настройка на особенности индивидуальных пользователей. Поддержка адаптивных методов в гипермедиа-системах оказывается весьма полезной в тех случаях, когда имеется одна система, обслуживающая множество пользователей с различными целями, уровнем знаний и опытом, и когда лежащее в ее основе гиперпространство является относительно большим. Поэтому области применения адаптивной гипермедиа выходят далеко за границы обучающих систем [1] и включают, например, такие, казалось бы далекие от обучения, области применения гипермедиа-систем, как открытые адаптивные виртуальные музеи [4].

Статья посвящена проекту WAPE, работа над которым ведется в Институте систем информатики СО РАН [5–7]. Цель проекта — разработка адаптивной среды дистанционного обучения WAPE, поддерживающей активное индивидуальное обучение программированию в рамках проблемного подхода и соединяющей возможности адаптивных гипермедиа-систем и интеллектуальных обучающих систем.

Статья начинается с описания основных свойств адаптивных гипермедиа-систем (разд. 1). В разд. 2 дается общее описание системы WAPE с упором на ее возможности, предоставляемые студентам. Возможностям, предоставляемым системой другим пользователям (администраторам, лекторам и инструкторам), посвящен разд. 3. В разд. 4 рассматриваются модель знания курса, а в разделах 5 и 6 — вопросы моделирования знаний студента и использования при этом моделировании механизма сетей Байеса. Моделям тестирования и видам тестов посвящен разд. 7. В разд. 8 и 9 рассматриваются проекты (упражнения и задания) подсистемы CLASS, играющей роль виртуальной семинарской аудитории в системе WAPE. Разд. 10 содержит описание подсистемы PRACTICE, предназначенной для прохождения студентами компьютерного практикума по курсу, поддерживаемому системой WAPE. Виды аннотирования проектов и других указаний, помогающих работе студентов, рассматриваются в разд. 11. Разд. 12 посвящен режимам работы студентов, а вопросы обновления модели знаний студента и развития курса описаны в разд. 13 и 14. В разд. 15 представлен разработанный вузовский курс программирования на базе языка Zonnon. Завершает изложение заключение.

1. АДАПТИВНЫЕ ГИПЕРМЕДИА-СИСТЕМЫ

Класс адаптивных гипермедиа-систем состоит из всех таких гипертекстовых и гипермедиа-систем, которые отражают некоторые особенности пользователя в его модели и применяют эту модель для адаптации различных видимых для пользователя аспектов системы. Таким образом, каждый пользователь имеет свою собственную картину и индивидуальные навигационные возможности для работы с адаптивной гипермедиа-системой.

Отметим, что области применения адаптивных систем далеко выходят за границы обучающих систем. Например, другим важным приложением являются *онлайновые информационные системы* (on-line information systems), а также *онлайновые справочные системы* (on-line help systems). К онлайновым информационным системам относятся, например, электронные энциклопедии, хранилища документов или туристические справочники. Чтобы выдать правильную информацию пользователям с различным уровнем квалификации, этим системам также требуется модель знаний пользователя. Важен также контекст запроса: нужна ли информация пользователю для краткой справки, для разработки презентации, для восстановления знаний? Онлайновые справочные системы принимают во внимание конкретную среду, например, место вызова (контекстно-зависимые справочные системы).

Вместе с тем, обучающие гипермедиа-системы, в которых пользователь или ученик имеет конкретную цель обучения (включая и такую цель, как общее образование), являются типичным приложением адаптивных гипермедиа-систем. В этих системах основное внимание уделяется знаниям обучающихся, которые могут сильно различаться. Состояние знаний изменяется во время работы с системой. Таким образом, корректное моделирование изменяющегося уровня знаний, надлежащее обновление модели и способность делать правильные заключения на базе обновленной оценки знаний являются важнейшей составляющей обучающей гипермедиа-системы. Эти свойства стали особенно важны для Web-систем дистанционного обучения с тех пор, как обучаемые стали учиться в основном самостоятельно (обычно дома). Интеллектуальное и личное содействие, которое могут дать учитель или студент-сокурсник при обычном (аудиторном) обучении, при дистанционном обучении нелегко достижимо. Адаптивность важна для программного обеспечения дистанционного обучения еще и потому, что оно должно использоваться намного более разнообразным по уровню знаний множеством студентов, чем любое «однопользовательское» учебное приложение. Сетевое программное обеспечение, разработанное для одного

класса пользователей (с определенным складом ума), может совсем не подойти другим обучаемым.

Выделяются следующие характеристики пользователя обучающей системы, важные для ее адаптации: цель (или задача) пользователя, уровень его знаний, уровень его подготовки, имеющийся опыт работы пользователя с данной гипермедиа-системой, набор (система) предпочтений пользователя, личностные характеристики пользователя и характеристики пользовательской среды.

Сетевые обучающие системы успешно объединяют технологии адаптации, используемые в интеллектуальных обучающих системах и адаптивных гипермедиа-системах.

Целью различных интеллектуальных обучающих систем является использование знаний о сфере обучения, обучаемом и о стратегиях обучения для обеспечения гибкого индивидуализированного изучения и обучения. Для ее достижения традиционно используются следующие основные технологии: построение последовательности курса обучения, интеллектуальный анализ ответов обучаемого и интерактивная поддержка в решении задач. В группу технологий интеллектуальных адаптаций сетевых обучающих систем входит также технология, получившая название *подбора моделей обучаемых* (или просто *подбор моделей*).

Что касается гипермедиа-систем, то в них адаптация в адаптивной гипермедиа может состоять в настройке содержания очередной страницы (*адаптация на уровне содержания*) или в изменении ссылок с очередной страницы, индексных страниц и страниц карт (*адаптация на уровне ссылок*).

Основные цели (методы) адаптации на уровне содержания гипермедиа-систем — это дополнительные, предварительные и сравнительные объяснения, варианты объяснений и сортировка.

Для достижения целей адаптации: на уровне адаптации разработаны такие техники, как условный и эластичный тексты, варианты страниц и фрагментов, а также технология, основанная на фреймах. Основные цели (методы) адаптации навигации — это глобальное и локальное руководство, поддержка локальной и глобальной ориентаций, управление индивидуализированными представлениями, а основные технологии адаптивной навигационной поддержки — это полное руководство, адаптивная сортировка (упорядочение) ссылок, адаптивное сокрытие ссылок, адаптивное аннотирование ссылок, адаптивное генерирование ссылок и адаптация карты.

2. СИСТЕМА WAPE

Система WAPE ориентирована на поддержку дистанционного обучения и предполагает четыре типа пользователей: студенты, инструкторы, лекторы и администраторы. Все пользователи осуществляют доступ к системе через стандартный Web-браузер, который представляет HTML-документы, предоставляемые HTML-сервером на стороне сервера.

После авторизации пользователя в качестве студента открывается подходящее меню команд.

WAPE система поддерживает три уровня процесса обучения:

- когда студент изучает теоретический материал в некоторой специфической области с использованием гипертекстовых учебников и задачников,
- когда система тестирует концептуальные знания студента, соответствующие изученному теоретическому материалу,
- когда студент под управлением системы выполняет учебные проекты, решая задания и упражнения.

Третий уровень рассматривается как основной в использовании WAPE системы; для того чтобы изучить курс, поддерживаемый WAPE системой, студент должен справиться с набором *проектов* (заданий и упражнений), который инструктор подбирает студенту строго индивидуально.

Другой тип задач, поддерживаемый системой WAPE, — это так называемые *тесты*. В отличие от проектов, решение о выполнении (или невыполнении) которых принимается инструктором, тесты — это вопросы, правильность ответов студентов на которые система оценивает полностью автоматически.

Ориентация на цели обучения является одним из важных свойств нашей WAPE среды. Поскольку мы не хотим фиксировать путь обучения студента или студенческой группы от начала до конца, студенты свободны в определении своих собственных целей обучения и своих собственных последовательностей обучения. На каждом шаге они могут обращаться за помощью к системе, запрашивая подходящий материал, последовательности обучения и советы по примерам и проектам. Если студенту необходим совет по нахождению своего собственного пути обучения, он может спросить систему о следующей подходящей цели обучения.

Система WAPE предназначена для обслуживания многих студентов с различными целями, знанием и опытом. В нашей системе основной упор делается на знание студентов, уровень которого может весьма сильно варьироваться у разных студентов. Более того, состояние знаний студента из-

меняется в процессе работы с системой. Поэтому большое внимание уделяется возможностям адаптивности в нашем проекте.

Система WAPE предоставляет лектору и инструкторам средства для управления мониторингом взаимодействия студентов с системой. Возможно определять те действия студента, которые нуждаются в реакции со стороны преподавателя. Например, когда студент завершает выполнение задания (или упражнения), сообщения посылаются инструктору, отвечающему за мониторинг работы данного студента.

Открытые дискуссии, поддерживаемые WAPE системой, обеспечивают полную виртуальную атмосферу телекласса, включая возможности кооперативного изучения курса вместе с другими студентами и средства кооперативного преподавания для инструкторов и лекторов.

3. ВОЗМОЖНОСТИ АДМИНИСТРАТОРОВ И ПРЕПОДАВАТЕЛЕЙ

Помимо студентов, система WAPE поддерживает три типа пользователей: инструкторы, лекторы и администраторы. Эти типы пользователей различаются как по своим правам, так и по возможностям работы с системой. Каждому типу пользователей соответствует свой интерфейс, поддерживаемый системой.

Интерфейс администратора поддерживает ряд административных функций организации учебного процесса, которые разбиваются на следующие две части.

1. Управление курсами и преподавателями.

В этой части можно осуществлять создание и удаление курсов и преподавателей, а также связывать преподавателей с курсами, потоками и группами в качестве лекторов и инструкторов, а также заменять одного лектора курса или инструктора учебной группы на другого.

2. Управление студентами.

В этой части можно создавать и удалять потоки, группы и отдельных студентов, а также переводить студентов из одной группы в другую.

Интерфейс лектора поддерживает следующие основные функции.

1. Редактирование учебной информации курса.

Сюда входят возможности по включению новых учебников в курс, пополнению гиперкниг курса новыми примерами, созданию или улучшению примеров проектов, по пополнению заданий новыми тестами и эталонными решениями, а также пополнению пространств тестов новыми тестами.

2. Общение со студентами и инструкторами.

Она реализовано в виде общих и преподавательских форумов, администратором которых является лектор. В общих форумах могут участвовать как студенты, так и преподаватели, а в преподавательских — только лектор и инструкторы. После включения лектором некоторой общей темы для обсуждения любой студент и любой инструктор могут написать свое мнение по обсуждаемому вопросу, но у лектора есть возможность удалять и редактировать любые сообщения.

3. Просмотр статистики.

Практически любые действия студента и инструктора заносятся в таблицу статистики и могут быть рассмотрены лектором. В частности, лектор может посмотреть, как часто и сколько времени студенты его потока тратят на обучение, сколько раз и какие тесты они проходили (с фиксацией времени и успеха прохождения). Здесь же он может узнать текущее состояние модели знаний каждого студента, а также какие задачи были им уже решены, а какие нет. Для каждой отдельной задачи можно также узнать количество раз, которое студент пытался ее решить, и посмотреть все варианты решений, которые студент предложил, вместе с комментариями инструктора.

4. Управление мониторингом.

Система предоставляет лектору возможности управления мониторингом взаимодействия студентов и инструкторов с системой. Здесь он может определить те действия студентов и инструкторов, которые нуждаются в его реакции. Каждый раз, когда выбранные действия будут происходить, лектор будет получать соответствующие сообщения.

Интерфейс инструктора поддерживает следующие основные функции.

1. Проверка и оценка заданий студентов.

Каждое задание, решенное студентом, должна быть проверено и оценено инструктором. После того как студент решает задание и проверяет его на имеющихся тестах, он отправляет ее решение на проверку инструктору. Для проверки правильности и оценки качества студенческого решения инструктор может использовать эталонные решения данного задания, если они есть. При этом он должен не только оценить данное решение (отклонить или принять с некоторой положительной оценкой), но и написать комментарий с разъяснением причин такой оценки.

2. Общение со студентами и преподавателями.

Указанные возможности образуют общую и приватную части. Общая часть реализована в виде форумов, администратором которых является лек-

тор, а также форумов, которые администрирует инструктор и создает их для своих студентов. Приватная часть дает инструктору возможность приватного общения с любым своим студентом. Она выполнена в виде гостевой книги, т.е. выводится просто последовательный список сообщений в порядке, обратном к порядку их написания (т.е. последнее сообщение всегда выводится первым). Преподаватель может отвечать на сообщения студентов, и наоборот. Ответы выводятся под соответствующими сообщениями и выделяются другим цветом и шрифтом.

3. Просмотр статистики.

Интерфейс просмотра статистики у инструктора имеет те же самые возможности, что и у лектора. Различие лишь в том, что инструктор может просматривать только свою статистику и статистику студентов своих групп.

4. Управление режимом работы и мониторингом.

Система предоставляет инструктору возможности управления режимом работы каждого своего студента, а также мониторингом взаимодействия этого студента с системой. Здесь он может определить, как знания студента будут влиять на его возможность решать проекты, а также выбрать те действия студента, которые нуждаются в его реакции. Каждый раз, когда студент будет совершать выбранные инструктором действия, соответствующие сообщения будут посылаться инструктору. Например, когда студент завершает выполнение задания (или упражнения), сообщения всегда посылаются инструктору, отвечающему за группу, в которой работает данный студент.

4. МОДЕЛЬ ЗНАНИЙ КУРСА

Центральным объектом модели обучения в WAPE является некоторый учебный курс (или просто курс), который представляет реальный курс, читаемый в некотором университете, реальном или виртуальном.

Каждый курс имеет своего лектора, который создает и поддерживает глоссарий, проекты, тесты, учебники и задачники по курсу (в качестве учебного материала), а также курирует проблемное обучение групп студентов, объединенных в студенческий курс (или поток), осуществляемое под руководством группы инструкторов (ассистентов лектора). Информация о завершении обучения одного потока студентов сохраняется в системе и может использоваться лектором для совершенствования его курса до того, как будет набран новый поток студентов для обучения.

В основе курса лежит его *модель знаний*, которая представляет собой конечное непустое множество *единиц знаний* S с двумя бинарными отношениями U и W на S, удовлетворяющими следующим свойствам для любых p, $q \in S$:

1) $(p, q) \in U$ тогда и только тогда, когда p является составной единицей знания, которая является объемлющей для q;

2) $(p, q) \in W$ тогда и только тогда, когда единица p должна быть изучена до изучения q.

Предполагается, что в модели знаний (S, U, W) пара (S, U) образует лес, а (S, W) является ациклическим графом.

На основе модели знаний курса строится *глоссарий* курса, в котором каждая единица знаний представлена одним (или несколькими) ключевым словом (или фразой) и дополнена множеством ссылок на те элементарные информационные ресурсы курса (элементарные информационные ресурсы учебников и задачников, а также примеры, тесты и проекты), содержание которых относится к данной единице знаний.

Каждый учебник или задачник имеет вид гипертекстовой книги (*гиперкниги*), обладающей иерархической структурой: книга, глава, параграф.

В общем случае гиперкнига содержит последовательность глав, каждая из которых может в свою очередь состоять из последовательности параграфов.

Элементарный информационный ресурс гиперкниги — это либо параграф, либо глава, не содержащая параграфов.

Особую группу элементарных информационных ресурсов гиперкниги образуют *примеры*. Они не содержат теоретического материала курса и служат для его пояснения.

Каждый элементарный информационный ресурс курса представлен отдельной Web-страницей и индексируется некоторым набором единиц знаний, описывающих содержание этого ресурса. Происхождение информационного ресурса несущественно для индексирования, только содержание определяет его информационный индекс.

Пусть $S \neq \varnothing$ — множество всех единиц знаний, $P(S)$ — множество всех подмножеств множества S, а H – множество всех информационных ресурсов, тогда *карта содержания* — это функция $I : H \to P(S) \setminus \varnothing$, которая каждому информационному ресурсу $h \in H$ сопоставляет *информационный индекс* этого ресурса $I(h)$ — непустое множество всех тех единиц знаний, которые связаны с данным информационным ресурсом: если h является тестом или проектом, то $I(h)$ состоит из тех единиц знания, которые используются в h, а если h является информационным элементом гиперкниги, то $I(h)$

состоит из тех единиц знания, объяснение которых в той или иной степени содержится в данном информационном ресурсе *h*.

5. МОДЕЛИРОВАНИЕ ЗНАНИЙ СТУДЕНТА

Знания студента *x* моделируются *вектором знаний* $K(x) = (p_1, \ldots p_n)$, где $n = |S|$ — число единиц знаний в модели знаний курса, а для любого i, $1 \le i \le n$, $p_i = p(s_i / E_i)$ — условная вероятность, описывающая предположение системы о том, что студент *x* обладает знанием единицы знания $s_i \in S = \{s_1, s_2, \ldots, s_n\}$ на базе тех свидетельств E_i, которые система собрала о знании студентом *x* единицы знания s_i в процессе мониторинга его работы над курсом.

Каждый элемент p_i вектора знаний $K(x)$ выражает степень знания единицы знания s_i, которым обладает студент *x* в данный момент.

Мы используем четыре степени такого знания:

- знания эксперта в области данной единицы знания — обозначаются *E* (отличные знания),
- знания продвинутого пользователя – обозначаются *F* (несколько затруднительных моментов в понимании s_i, но в целом понятие вполне усвоено),
- знания начинающего — обозначаются *A* (много затруднительных моментов, понятие s_i плохо усвоено),
- знания новичка — обозначаются *N* (пользователь еще не готов для работы с данным понятием, оно им не усвоено).

Таким образом, единицы знания являются с одной стороны понятиями, описывающими предметную область курса, а с другой — случайными переменными с четырьмя дискретными значениями *E*, *F*, *A* и *N*, кодирующими степень знаний данного студента.

Свидетельства, получаемые системой в процессе мониторинга действий студента, изменяются со временем. В типичном случае знание студента возрастает во время работы с курсом, хотя недостаток знания также воспринимается системой как свидетельство. Поскольку каждый отслеживаемый результат производимого наблюдения за студентом сразу же заносится в свидетельства, вектор знаний в любой момент дает моментальный снимок текущего уровня знаний студента.

Обновление свидетельств о студенте происходит только при выполнении им тестов и проектов. При этом никакое тестирование не позволяет студенту получить от системы оценки “отличные знания”, и он может дос-

тичь уровня знаний эксперта только за счет успешного выполнения проектов.

Пусть $P = (p_1, \ldots p_n)$ и $Q = (q_1, \ldots q_n)$ — два произвольных вектора знаний, тогда:

- $P \geq Q$, если $p_i \geq q_i$ для любого I;
- $min\ (P,\ Q)$ – это такой вектор знаний $(w_1, \ldots w_n)$, что $w_i = min(p_i, q_i)$ для любого I;
- $max\ (P,\ Q)$ – это такой вектор знаний $(w_1, \ldots w_n)$, что $w_i = max(p_i, q_i)$ для любого i.

6. МЕХАНИЗМ СЕТЕЙ БАЙЕСА: ВЫЧИСЛЕНИЕ ВЕРОЯТНОСТЕЙ

Сети Байеса являются мощным инструментом вывода в графах с зависимыми случайными вершинами. Мы используем такую сеть для вычисления распределения вероятностей для каждой единицы знаний и, следовательно, для вычисления векторов знаний пользователей.

Сеть Байеса представляет собой ориентированный ациклический граф со следующими свойствами (рис. 1):

- каждая вершина графа представляет случайную переменную;
- имеется дуга из X в $Y \neq X$ в каждом случае, когда Y зависит от X;
- каждая вершина помечена таблицей условной вероятности, которая определяет воздействие на вершину ее непосредственных предшественников.

Чтобы построить сеть Байеса, которая вычисляет распределение вероятностей для каждой единицы знаний курса для конкретного студента, требуется выполнить два действия. Во-первых, сгенерировать некоторый ациклический граф, имеющий единицы знаний в качестве вершин и обучающие зависимости между ними в качестве дуг, и, во-вторых, определить таблицы вероятности для всех вершин.

Для наших целей мы строим сеть Байеса со случайными переменными, которые дают распределение вероятностей для вычисления знаний пользователя. Мы используем в качестве вершин случайные переменные с четырьмя дискретными значениями из множества $\{E, F, A, N\}$ и проводим дугу из вершины X в вершину Y в том и только том случае, когда $Y < X$ и не существует такого Z, что $Y < Z < X$, где $Y < X$, если Y зависит от X. т.е. если $(X, Y) \in W$ или $(Y, X) \in U$.

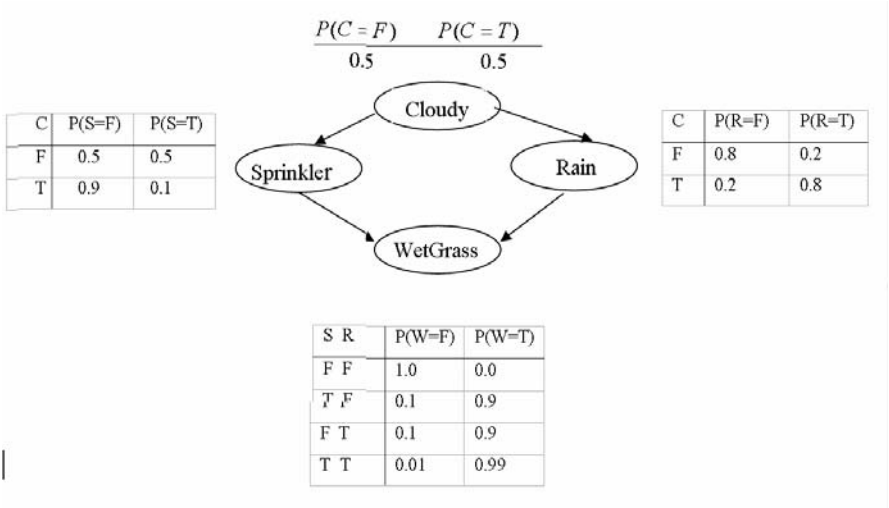


Рис. 1. Простая сеть Байеса, вершины которой представляют случайные логические переменные, принимающие значения: T (истина) или F (ложь)

Известно, что точный вывод в сетях Байеса является *NP*-трудной проблемой [8]. Вместе с тем существуют линейные по времени алгоритмы для тех сетей Байеса, базовые графы которых не содержат циклов, а также существует несколько методов обработки таких не полностью ориентированных циклов: кластеризация, кондиционирование и стохастическая симуляция.

7. МОДЕЛЬ ТЕСТИРОВАНИЯ

Тесты – это вопросы студенту, ответы на которые оцениваются системой без участия преподавателя.

По виду различается три типа тестовых вопросов: выборные, мультивыборные и наборные. Выборный тест предполагает выбор одного варианта ответа из предлагаемого списка (рис. 2), а мультивыборный — нескольких вариантов (рис. 3). Наборный тест требует, чтобы студент ввел правильный ответа в специальное текстовое поле (рис. 4). По существу, каждый мультивыборный тест предлагает студенту список вариантов отве-

тов, некоторое (возможно пустое) множество из которых является правильными.

Мы используем компьютерное тестирование со случайным позиционированием ответов для выборных и мультивыборных тестов, чтобы ответы по номерам позиций могли привести к ошибкам студентов, заучивающих позиционные номера правильных ответов. Таким образом, вместо запоминания вопросов и номеров строк вариантов правильных ответов студенты вынуждены в вопросах и ответах на вопросы фокусироваться на содержательной стороне дела. Поэтому подход к тестированию, используемый нашей системой, включает случайную генерацию вопросов в заданной предметной области и случайное расположение вариантов ответов.

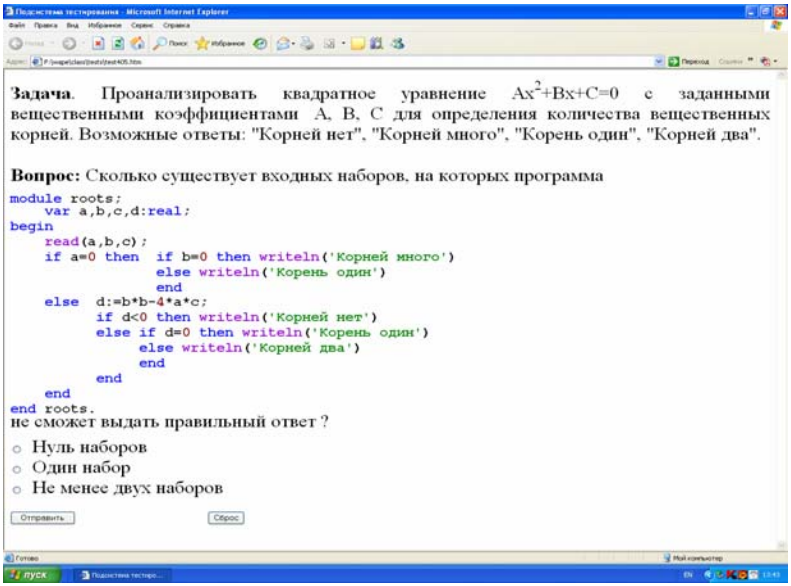


Рис. 2. Выборный тест

Предполагается, что в процессе тестирования студент, как правило, получает не один, а целую серию тестов, ответы на которые оцениваются суммарно. Такой подход еще более усложняет работу тех студентов, которые пытаются пройти тестирование путем заучивания позиционных номеров правильных ответов.

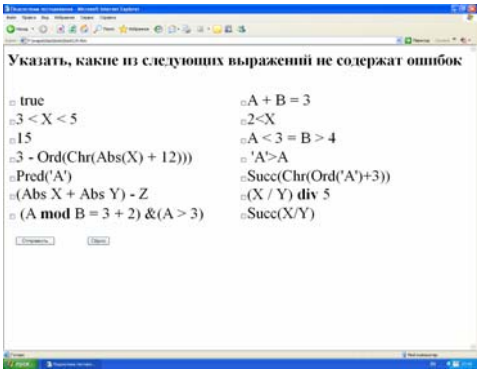


Рис. 3. Мультивыборный тест

Каждый тест измеряет вербальные или аналитические умения, относящиеся к конкретной области изучаемого курса. Различные временные ограничения ассоциируются с каждым вопросом. Мы различаем три вида тестов: вербальные, качественные и аналитические.

Вербальный тест определяет некоторую конкретную концепцию определения и имеет временное ограничение в 60 секунд. Вербальный тест проверяет способность анализировать и оценивать написанный материал и синтезировать информацию, полученную из него, для анализа отношений между отдельными частями предложений и для распознавания отношений между словами и понятиями.

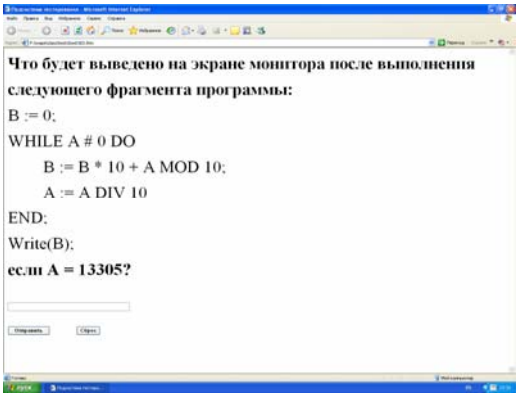


Рис. 4. Наборный тест

В случае *качественных* тестов, в которых должны быть объяснены более сложные понятия, обычно ответ ожидается между 120 и 240 секундами. Качественный тест проверяет базисные умения и понимание элементарных понятий, а также способность студента качественно мыслить и решать задачи в качественной постановке или объяснять более сложные понятия.

Для *аналитического* теста, при котором должно быть объяснено некоторое трудное понятие, ответ обычно ожидается внутри временного диапазона в 360 — 480 секунд. Аналитический тест проверяет способность студента понимать структурированные множества отношений, выводить новую информацию из множеств отношений, анализировать и оценивать аргументы, идентифицировать центральные вопросы и гипотезы, делать правильные выводы и опознавать хорошо обоснованные объяснения. Тесты аналитического вида обычно измеряют умение рассуждать, связанное с несколькими разделами изучаемого курса.

С каждой единицей знания курса связываются два пространства тестов, первое из которых используется для тестирования студентов, претендующих на знание данной единицы знаний на уровне “начинающего”, а второе — на уровне “продвинутого” студента.

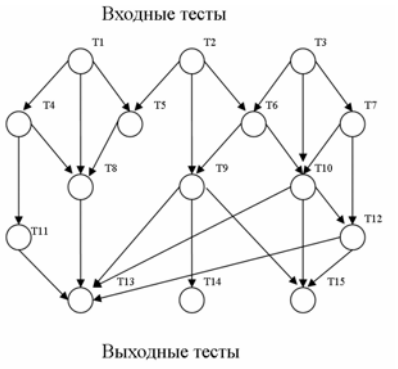


Рис. 5. Пространство из 15 тестов

Пространство тестов — это ациклический ориентированный граф $G = (X, V)$, вершинами которого являются тесты, а дуги отражают последовательность их прохождения: $(p, q) \in V$ тогда и только тогда, когда после выполнения студентом теста p ему может быть предложен тест q.

В пространстве тестов выделяются множества входных и выходных тестов. Тест p является входным, если он не имеет предшественников в G, и является выходным, если p не имеет преемников в G.

С каждым пространством тестов лектор связывает две константы границ знаний C_1 и C_2, где C_1 — минимальный уровень знаний для начинающего студента в данном пространстве тестов, а C_2 — минимальный уровень знаний для продолжающего студента для данного пространства тестов. По умолчанию предполагается, что пространство тестов для начинающих студентов состоит из вербальных и качественных тестов и имеет $C_2 > 1$, а пространство для продвинутых студентов состоит из аналитических тестов, и в нем $C_2 = 1$.

Путь тестирования в пространстве тестов — это путь по G от некоторого входного теста до некоторого выходного. В процессе тестирования студенту предлагается серия тестов $T = \{p_1, \ldots, p_n\}$, образующих путь тестирования $(p_1, \ldots, p_n)$, генерируемый случайным образом. Например, последовательность (T3, T6, T9, T13) образует один из возможных путей тестирования для пространства тестов рис. 5.

Пусть студент выполнил серию тестов T и

$$\gamma_x = \left(\sum_{t \in T} \alpha(t)\right) / \left(\sum_{t \in T} \beta(t)\right), \text{ где}$$

$\alpha(t)$ — число правильных ответов данного студента в серии тестов T,

$\beta(t)$ — максимальное число правильных ответов для серии тестов T, которые мог бы дать студент. Заметим, что в случае выборного или наборного ответа студент может дать только один правильный ответ, а в мультивыборном тесте максимальное число правильных ответов совпадает с длиной списка вариантов ответов.

Студент x демонстрирует на серии тестов T знания

- новичка, если $\gamma_x < C_1$,
- начинающего студента, если $C_1 \le \gamma_x < C_2$, и
- продвинутого студента, если $C_2 \le \gamma_x$.

8. ПРОЕКТЫ СИСТЕМЫ CLASS

Системы CLASS и PRACTICE являются основными подсистемами системы WAPE.

Подсистема CLASS является виртуальной семинарской аудиторией, которая предназначена для получения опыта программирования на некотором

языке высокого уровня. Это среда проблемного обучения, в которой студенты группы под руководством инструктора обучаются конструировать корректные и эффективные программы для решения достаточно простых задач.

Любой курс, поддерживаемый системой CLASS, включает набор проектов, предназначенных для выполнения студентами. Каждый проект P — это упорядоченное множество однотипных задач $\{P_1, P_2,\ldots,P_n\}$, где $n \geq 0$ — количество вариантов проекта P.

В системе CLASS имеются проекты двух видов: упражнения и задания, различающиеся по виду предполагаемого решения.

Каждое *упражнение* P — это набор вопросов $\{P_1, P_2,\ldots,P_n\}$, предполагаемые ответы на которые не рассматриваются системой как исполняемые программы.

В отличие от упражнений любое *задание* P — это задача на программирование; так что решить вариант i задания P, т.е. P_i, — это значит написать исполняемую программу.

По уровню сложности решаемых задач различаются проекты двух типов: стандартной и повышенной сложности.

Проекты стандартной сложности не имеют пометок и образованы из тех задач, которые проверяют понимание студентом изложенного в учебнике теоретического материала. В частности, к проектам стандартной сложности относятся те упражнения и задания, которые направлены на обучение студентов использованию новых языковых конструкций совместно с уже усвоенными, а также на отработку описанных в учебнике схем решения задач.

Другой тип проектов выделен специальной пометкой «проект повышенной сложности» и содержит задачи, которые добавляют новую или требующую размышлений информацию к материалу, изложенному в учебнике. Такие проекты заставляют студентов обдумывать некоторую важную концепцию, относящуюся к теоретическому материалу учебника, или находить ответ на вопрос, который может возникать у студента во время чтения учебника.

Каждый проект стандартной сложности требует, чтобы у студента, допускаемого к его выполнению, уровень знания для всех тех единиц, которые образуют индекс данного проекта, был бы не ниже уровня продвинутого студента, а любой проект повышенной сложности может начать решать только тот студент, который является экспертом во всех единицах знаний, входящих в индекс данного проекта.

В каждом проекте можно выделить некоторый вариант в качестве примера. Решение такого варианта, который получает номер 0, оформляется в

виде гипертекстового документа, содержащего формулировку задачи, ее решение и обоснование решения. В частности, в случае задания он содержит описание задания, текст программы и комментарии к программе, поясняющие и обосновывающие данное решение.

Для поддержки данной возможности используются виртуальные и пользовательские номера вариантов проектов.

Виртуальные номера используются при описании проекта лектором на языке описания проектов, а пользовательские — это номера, в терминах которых варианты данного проекта используются студентами в курсе, поддерживаемом системой CLASS. В частности, вариант, имеющий в качестве пользовательского номера нуль, является примером данного проекта.

Лектор, описывая проект в терминах виртуальных номеров, одновременно задает функцию преобразования виртуальных номеров в пользовательские. Указанная функция используется подсистемой генерации проектов каждый раз, когда происходит обращение к проекту за формулировкой некоторого его варианта или за его тестами.

9. ЗАДАНИЯ СИСТЕМЫ CLASS

Каждое задание системы CLASS предполагает составление программы, читающей входные данные из файла *input.txt* и записывающие результаты своего исполнения (выходные данные) в файл *output.txt.*

С каждым заданием α связывается множество эталонных решений $S(\alpha)$, множество тестов задания $T(\alpha)$, множество правил вывода знаний студента $R(\alpha)$ и два вектора знаний: $P(\alpha)$ — вектор предварительных знаний и $F(\alpha)$ — вектор итоговых знаний.

Эталонные решения $S(\alpha)$ — это прокомментированные программы решения задания α. В комментариях описываются особенности данного решения задания α и приводится мотивированная его оценка. Эталонные решения используются преподавателями (лекторами и инструкторами) и не доступны студентам.

Тесты задания $T(\alpha)$ используются системой для проверки правильности понимания студентом условия задания и автоматической проверки правильности составленной студентом программы. Каждый тест из множества $T(\alpha)$ — это пара, состоящая из входных и соответствующих выходных данных.

Множество тестов $T(\alpha)$ задания α распадается на два непересекающихся подмножества $T_1(\alpha)$ и $T_2(\alpha)$, т.е. $T(\alpha) = T_1(\alpha) \cup T_2(\alpha)$ и $T_1(\alpha) \cap T_2(\alpha) = \varnothing$,

называемых тестами *правильности понимания задания* и тестами *правильности программы* (*решения задания*).

Перед тем как начать решать задание, студент должен подтвердить, что он правильно понимает условие задания. Для этого он должен для каждого теста $t \in T_1(\alpha)$ правильно ввести выходные данные теста t по заданным входным. Тесты из $t \in T_1(\alpha)$ могут использоваться студентом также на начальном этапе отладки программы решения задания α.

Перед тем, как студенту будет разрешено отдать программу (решение задания α) на проверку инструктору, он должен продемонстрировать системе ее правильность на всех тестах из $T_2(\alpha)$. Для этого его программа должна на входных данных каждого теста $t \in T_2(\alpha)$ выдавать выходные данные, которые соответствуют ожидаемым.

Каждому тесту $t \in T(\alpha)$ могут быть сопоставлены два текста: диагностическое сообщение и общедоступный комментарий. *Диагностическое сообщение* — это тот текст, который выдается студенту тогда, когда программа студента является неправильной относительно теста t. В тексте комментария указываются предполагаемые ограничения на рабочие характеристики программы для данного теста.

Каждое *правило вывода* знаний студента из множества $R(\alpha)$ имеет вид: $B \to N$, где

- B — логическое выражение, содержащее литералы вида x или $\neg x$,
- $x \in T_2(\alpha)$,
- N — вектор знаний.

Правило $B \to N$ работает в два этапа.

1. На первом этапе в B вместо каждого вхождения теста $x \in T_2(\alpha)$ подставляется либо значение *true,* если на входных данных теста x программа выдает предполагаемые результаты, либо значение *false* в противном случае.
2. На втором этапе вычисляется выражение B, и, если оно принимает истинное значение, то происходит перевычисление вектора знаний $K(x)$ студента x, решающего данное задание, по следующему правилу $K(x) := min(N, K(x))$.

Вектора предварительных и итоговых знаний $P(\alpha)$ и $F(\alpha)$, приписанные заданию α, имеют следующий смысл.

Предполагается, что студент x может приступить к выполнению проекта α только в том случае, если $K(x) \geq P(\alpha)$.

Успешное выполнение проекта α приводит к перевычислению вектора знаний $K(x)$ студента x, решающего данное задание, по следующему правилу $K(x) := max(F(\alpha), K(x))$.

10. СИСТЕМА PRACTICE

Система PRACTICE является виртуальной лабораторией, предназначенной для прохождения студентами компьютерного практикума по курсу. Основная цель, которая ставится перед студентом при выполнении индивидуальных заданий (интегральных проектов), составляющих компьютерный практикум, — это практическое освоение всех этапов разработки надежной и наглядной интерактивной (диалоговой) программы для компьютерного решения некоторой нетривиальной задачи, требующей разработки алгоритма, обработки сложных структур данных и создания дружественного интерфейса.

С каждым интегральным заданием α связывается множество эталонных решений $S(\alpha)$ и два вектора знаний: $P(\alpha)$ — вектор предварительных знаний и $F(\alpha)$ — вектор итоговых знаний. Они имеют тот же смысл, что и у заданий системы CLASS, с тем отличием, что *эталонные решения* $S(\alpha)$ — это прокомментированные решения задания α, которые имеют вид гипертекстовых отчетов (см. ниже).

Тематика индивидуальных заданий для компьютерного практикума определяется, в первую очередь, всеми видами работ, которые должен освоить студент, чтобы научиться создавать качественные (эффективные, наглядные и надежные) нетривиальные программы. В большинстве случаев задача, решаемая во время выполнения индивидуального задания, — это задача невычислительного характера, имеющая краткую и точную (содержательную) формулировку и допускающая большое разнообразие решений.

При выполнении проектов практикума студенты должны интегрировать все, что они изучили ранее (при работе в системе CLASS), а также получить навыки, являющиеся базовыми для разработки программного обеспечения на всех уровня и для программирования как дисциплины. Это включает использование в той или иной мере формальных методов анализа задач и конструирования программ с упором на создание эффективных и надежных программ, которые удовлетворяют заданным спецификациям и поддерживают дружественный интерфейс.

Работая в виртуальной лаборатории, студент изучает методический материал, содержащийся в задачнике курса, тестирует свои знания теоретического материала, прочитанного им, или решает интегральные проекты.

В процессе решения проекта в системе PRACTICE студент составляет гипертекстовый отчет, который включает:

1) формулировку задачи;

2) описание программы для пользователя, ее внешнюю спецификацию, т.е. описание способа задания входных данных, вида результатов программы при заданных входных данных и сценария диалога в процессе исполнения программы;

3) словесное описание алгоритма и обоснование его правильности и эффективности;

4) текст программы;

5) описание тестового набора и его обоснование.

Для удобства все интегральные задания разбиты на три группы:

- обычной сложности (они не помечены),
- повышенной сложности,
- пониженной сложности.

При оценке сложности задания рассматриваются три показателя:

- сложность структур данных,
- сложность вычислений,
- изобретательность.

В показатель “изобретательность” включаются такие свойства задания, как непривычность для студента понятий, используемых в задании, сложность извлечения из определений тех свойств, на которых должен базироваться алгоритм решения задания, а также сложная связь между структурами данных и вычислениями. Каждый из указанных трех показателей задания оценивается в баллах 0, 1 или 2. Пометку “задание повышенной сложности” получают задания, набирающие в сумме по трем показателям всего 1 балл, а пометку “задание пониженной сложности ”— задания, суммарный балл которых больше 3.

11. АННОТАЦИЯ ПРОЕКТОВ

Для вычисления релевантности доступного по ссылке проекта текущему уровню знаний студента используется простая метафора «семафора» для аннотаций. Ссылки на проекты, которые должен решить в курсе студент, помечены одним из трех цветовых кружков: готов-для-решения (зеленый

кружок рядом с текстом ссылки), не-готов-для-решения (красный кружок) или уже-решен (серый кружок), что позволяет студенту выбрать подходящие проекты.

Студенту часто требуется информация по определенным темам, но не хватает предварительных знаний для понимания этой информации. К примеру, студент хочет работать над проектом построения ветвящихся программ, но не понимает таких тем, как логические выражения; в таком случае ему не стоит начинать чтение сразу со страниц, содержащих описание условных операторов. Для поддержки студента система сравнивает его текущий уровень знаний с необходимым для понимания рассматриваемой темы. Если у студента не хватает некоторых предварительных знаний, система может сгенерировать последовательность информационных элементов (след), который направляет его обучение в рамках данной темы.

Генерация такого следа реализована как алгоритм обхода в глубину, который проверяет оценку системой знаний студента о тех единицах знаний, которые предварительно необходимы для его текущей цели. Алгоритм проверяет, все ли предварительные знания достаточно усвоены студентом. Если нет, алгоритм находит единицы знаний, нуждающиеся в изучении. После этого генерируется последовательность подцелей и последовательность информационных элементов, которая последовательно направляет работу студента через необходимые темы вплоть до выбранной им.

Если студент хочет получить более конкретные указания во время работы с системой, он может запросить у системы следующий разумный шаг обучения. Этот запрос на прямое руководство выполняется путем определения подходящей цели обучения, зависящей от текущего состояния знаний студента и состояния его проектов. Цель определяется как набор единиц знаний. Чтобы определить следующую подходящую цель обучения, вычисляется последовательный след, покрывающий все нерешенные проекты студента. Для каждого элемента этого следа проверяется оценка системой уровня знаний студента. Если студенту не хватает знания какой-либо единицы знаний, тогда она предлагается в качестве следующей подходящей цели.

12. РЕЖИМЫ РАБОТЫ СТУДЕНТА

Инструктор может управлять тем, как знания студента влияют на возможность его работы с системой.

В *жестко контролируемом* режиме студент x не может выполнять ни один из тех проектов, для которых он не имеет достаточно знаний. Другими словами, при этом режиме разрешается студенту x приступить к выполнению некоторого проекта α только в том случае, когда $K(x) \geq P(\alpha)$.

В *слабо контролируемом* режиме студент x может выполнять не только те проекты α, для которых у него хватает знаний (т.е. $K(x) \geq P(\alpha)$), но и любой такой проект α, что для любого i либо $K(x)(i) \geq P(\alpha)(i)$, либо $K(x)(i) = F$ и $P(\alpha)(i) = E$.

В *свободном режиме* студент может выполнять любой проект.

При этом вне зависимости от того режима, который установлен студенту x инструктором, студент всегда получает предупреждение о недостаточности своих знаний, если он пытается начать выполнять такой проект α, для которого у него не хватает компетентности в некоторой единице знаний i (т.е. $K(x)(i) < P(\alpha)(i)$). Одновременно с предупреждением студенту демонстрируются те единицы знаний (вместе с их уровнями), знание которых он должен повысить до выполнения данного проекта.

13. ОБНОВЛЕНИЕ МОДЕЛИ ЗНАНИЙ СТУДЕНТА

Многие адаптивные системы фиксируют факт чтения пользователем определенной информации и на этой основе обновляют оценку его знаний. Некоторые из них учитывают время чтения или последовательность прочитанных страниц для углубления этой оценки. Хотя это оправданный подход, его недостатком является трудность измерения знания, приобретенного пользователем во время чтения Web-страницы. Вместо этого мы используем для обновления модели знаний только тесты и проекты. Это мотивировано проблемным подходом к обучению в тех курсах, на поддержку которых ориентирована система WAPE.

Предполагается, что после чтения того или иного теоретического материала система организует обратную связь. Студент последовательно указывает те темы, которые были целью изучения данного материала. Для каждой из этих тем (единиц знаний) студент переоценивает свои изменившиеся знания, выбирая одну из двух категорий: «тема была несложной — я овладел материалом без труда», «тема была непростой — у меня были некоторые проблемы в понимании». Эти категории соответствуют знаниям продвинутого студента и начинающего. После этого система генерирует индивидуальный набор тестов, успешное прохождение которых позволяет студенту изменить желаемым образом оценку своих знаний.

Обновление модели знаний автоматически происходит во время работы студента над некоторым проектом каждый раз, когда решение студента не проходит некоторый тест, а также тогда, когда инструктор завершает оценку текущего (посылаемого инструктору на проверку) варианта решения проекта студентом.

Студент имеет право в любой момент понизить оценку уровня своих знаний любой единицы знаний, а также попытаться повысить её. В последнем случае в зависимости от его претензий он получает индивидуальный набор тестов, сгенерированный системой, успешное прохождение которых позволяет студенту изменить желаемым образом оценку своих знаний.

14. РАЗВИТИЕ КУРСА

В процессе функционирования предполагается развитие каждого курса, поддерживаемого системой, по следующим направлениям:

1) включение новых учебников в курс,

2) создание или улучшение примеров решения проектов,

3) пополнение заданий новыми тестами и эталонными решениями.

Указанные усовершенствования осуществляются лектором и не требуют от него каких-либо специальных программистских знаний. При включении некоторого нового гипер-учебника в курс нет необходимости его преобразования. Единственное, что требуется — это построение информационных индексов элементарных информационных элементов, составляющих данный учебник. Эти индексы можно строить как автоматически (путем поиска вхождений ключевых фраз), так и вручную (путем просмотра учебника лектором). В основе других направлений развития курса лежат наработки студентов, которые при включении их преподавателем в курс могут подвергнуться редактированию и комментированию.

Для описания новых проектов можно использовать следующий специальный *язык задания проектов* (ЯЗП), позволяющий задавать одновременно все варианты проекта компактным образом.

Ниже для описания синтаксиса языка ЯЗП используется Расширенный Бекуса-Наура Формализм (Extended Backus-Naur Formalism, EBNF), который характеризуется следующими свойствами:

- альтернативы разделяются символом |;
- скобки [и] обозначают факультативность выражения в скобках;
- скобки { и } обозначают повторение (возможно 0 раз);
- скобки (и) используются для формирования групп элементов;

- нетерминальные символы начинаются с прописной буквы (например, Statement);
- терминальные символы либо начинаются со строчной буквы, либо записывается целиком прописными буквами (например, BEGIN), или представляются в виде строк (например, ":=").

Правила описания проекта на языке ЯЗП имеют следующий вид, где Символ — это произвольный символ, отличный от знака # :

Проект={Условие Образец}.
Образец={{Символ}{#Выражение}{Символ}}.
Условие= ИстинноеУсловие | ВычисляемоеУсловие.
ВычисляемоеУсловие = #<Выражение, Интервал> |
#<Выражение, Число>.
ИстинноеУсловие = ##.
Выражение= Номер | Номер * Число | Номер DIV Число.
Интервал=ОткрывСкобка Число, Число ЗакрСкобка.
ОткрывСкобка = (| [.
ЗакрСкобка =) |].
Число = {Цифра}.
Номер = # Число.

Каждый Проект на языке ЯЗП представляет собой фактически последовательность строк вида Условие Образец, по которым строятся все конкретные варианты проекта по следующим правилам. Вначале *N* подставляется в каждое Выражение вместо Номера, преобразуя их в константные. Затем полученные константные выражения вычисляются, и вычисленные значения заменяют вхождения выражений. Данная подстановка преобразует образцы в строки факультативных и обязательных частей текста формулировки соответствующего варианта. Обязательная часть образуется из образца, которому предшествует ИстинноеУсловие. Условие, соответствующего образца, при котором данная факультативная строка должна включаться в формулировку конкретного варианта проекта, — это принадлежность вычисленного значения выражения интервалу, заданному в вычисляемом условии, или его равенство заданному значению.

15. КУРС ПРОГРАММИРОВАНИЯ НА БАЗЕ ЯЗЫКА ZONNON

Для включения в систему нами разработан курс начального обучения программированию на базе нового языка Zonnon [9, 10], работа над которым ведется в Цюрихском институте информатики. Язык Zonnon задуман

как дальнейшая эволюция хорошо известного и широко применяемого на западе в учебных целях языка Оберон, являющегося преемником языков Паскаль и Модула-2. Язык Zonnon сохраняет стремление к простоте, ясному синтаксису и независимости концепций, а также уделяет внимание параллельности и легкости композиции и выражения. Унификация абстракций является стержнем проектирования языка Zonnon, и она отражается в его концептуальной модели, основанной на модулях, объектах, определениях и реализациях. Язык Zonnon содержит такие новые черты, как активность в объектах, основанный на межобъектном взаимодействии диалог, перегрузка операций и обработка исключительных ситуаций. Он специально разрабатывается как платформно-независимый язык, и его первая реализация выполнена для платформы .NET.

Разработанный курс базируется на ряде методических и технологических принципов, основными из которых являются следующие:

- принцип концентрического изложения материала, когда обучаемый осваивает языковые средства и приемы программирования постепенно, слой за слоем;
- принцип обучения конструированию программ на подробно комментированных образцах решения тщательно подобранных задач;
- принцип доказательного программирования, когда программа строится вместе с доказательством ее правильности;
- принцип пошаговой разработки программ, когда программа строится из формальной спецификации задачи с помощью мелких формально проверяемых шагов преобразования;
- принцип модульного программирования, позволяющий проектировать, разрабатывать и собирать программу по частям и с использованием библиотек уже готовых частей;
- принцип объектно-ориентированного программирования, позволяющий разработчикам программ легко создавать все более сложные приложения с помощью инкапсуляции, наследования и полиморфизма.

Подготовлены два гипертекстовых учебных пособия «Введение в программирование» и «Практикум по программированию», поддерживающие курс, которые размещены на сайте русскоязычной библиотеки учебных курсов международной программы MSDN Academic Alliance [11].

ЗАКЛЮЧЕНИЕ

В статье описывается архитектура адаптивной среды дистанционного обучения, которая поддерживает активное индивидуальное обучение программированию в рамках проблемного подхода и соединяет возможности адаптивных гипермедиа-систем и интеллектуальных обучающих систем. Среда нацелена на поддержку обучения конструированию алгоритмов и -- разработки эффективных и надежных программ, в процессе которой обучаемый, решая поставленные ему индивидуальные задачи, действует вполне самостоятельно, но постоянно имеет возможность получения квалифицированной помощи, корректирующей и направляющей его усилия, начиная с этапа понимания условия задачи и кончая этапом оценки правильности решения.

СПИСОК ЛИТЕРАТУРЫ

1. Brusilovsky P. Adaptive hypermedia // User Modeling and User-Adapted Interaction. — 2001. — Vol 11. — P. 87–110.
2. Касьянов В.Н., Касьянова Е.В. Дистанционное обучение: методы и средства адаптивной гипермедиа // Программные средства и математические основы информатики. — Новосибирск, ИСИ СО РАН, 2004. — С. 80–141.
3. Касьянов В.Н, Касьянова Е.В. Адаптивные системы и методы дистанционного обучения // Информационные технологии в высшем образовании. — 2004.— Т.1, N 4. — С. 40–60.
4. Kasyanov V. SVM — Siberian Virtual Museum of Informatics History // Innovation and the Knowledge Economy: Issues, Applications, Case Studies. — Amsterdam, IOS Press, 2005. — Part 2. — P. 1014–1021.
5. Kasyanov V.N., Kasyanova E.V. Web-based systems for supporting computer-science teaching and learning // SIGCSE Bulletin. — New York: ACM Press, 2002. — Vol.34, N 3. — P. 238. — (Proc. of the 7th ACM SIGCSE Conf. on Innovation and Technology in Computer Science Education).
6. Kasyanov V.N., Kasyanova E.V. An environment for Web-based education of programming // HCI Internat. 2003. — Heraklion, Crete University Press, 2003. — P. 179–180.
7. Kasyanova E.V. WAPE: an adaptive environment for Web-based education of programming // Proc. of the 17th IMACS World Congress. — Paris, 2005. — P. 681–685.
8. Cooper G. The computational complexity of probabilistic inference using Bayesian belief networks // Artificial Intelligence. — 1990. — Vol. 42. — P. 393–405.

9. Касьянова Е.В. Язык программирования Zonnon для платформы .NET // Программные средства и математические основы информатики. — Новосибирск, ИСИ СО РАН, 2004. — С. 189–205.
10. Касьянова Е.В. Вводный курс программирования на базе языка Zonnon // Методы и инструменты конструирования и оптимизации программ. – Новосибирск, ИСИ СО РАН, 2005. — С. 95–116.
11. http://www.microsoft.com/Rus/Msdnaa/Curricula/Default.msp

А.В. Козырева

ОПРЕДЕЛЕНИЕ КООРДИНАТ МОБИЛЬНОГО УСТРОЙСТВА В ПРОСТРАНСТВЕ НА ОСНОВЕ ИЗОБРАЖЕНИЙ, ПОЛУЧАЕМЫХ ОТ ЕГО ВИДЕОКАМЕРЫ

ВВЕДЕНИЕ

В статье описаны алгоритмы, необходимые для решения задачи позиционирования мобильного устройства в пространстве относительно окружающих объектов. Под мобильными устройствами будем понимать смартфоны (мобильные телефоны, обладающие операционной системой) и КПК (карманные компьютеры), обладающие «зрением», т.е. видеокамерой. Видеокамера — наш единственный источник восприятия окружающего пространства. На основе поступающих с камеры снимков будет проходить анализ изменения положения в пространстве.

Нам с камеры в постоянном потоке поступают изображения. Будем сравнивать изображения попарно и на основе различий между ними вычислять на сколько и как сдвинулось мобильное устройство.

Таким образом в каждый момент времени имеем два изображения. Нам совершенно не нужно сравнивать два полученных изображения полностью. Нам достаточно рассмотреть лишь некоторую их часть. Т.е. сначала мы должны найти на изображении точки, следя за которыми мы поймем как изменилось положение. Такие точки называют «опорными» или «особыми». Найдя особые точки на первом изображении нам необходимо отыскать их на втором. Здесь нам потребуются алгоритмы слежения за особенностями. Следует отметить, что если на втором изображении «исчезли» все «опорные» точки первого изображения, то мы не сможем решить поставленную задачу. Чтобы этого избежать нам необходимо либо очень часто делать снимки, либо следить за большим числом точек. Для решения задачи нам требуется минимум четыре точки на одном изображении и соответствующие четыре точки на втором. Максимальное число точек ограничено всем изображением. Так как наши «опорные» точки на получаемых изображениях имеют две координаты (изображение является плоским), а «опорная» точка в пространстве имеет 3 координаты, то нам необходимо знать правила по которым происходит данное преобразование. Данные правила зависят от внутренних параметров фотокамеры, т.е. изначально

нам необходимо провести так называемую калибровку камеры. Данную процедуру можно проводить различными способами, которые отличаются лишь точностью. В процессе разработки нашей программной системы необходимо будет отыскать наиболее оптимальное сочетание качества и быстроты работы. Само определение изменилось ли положение в пространстве или нет, производится путем поиска неизвестных коэффициентов в матрице вращения и векторе сдвига.

Таким образом задача определения изменения положения в пространстве мобильного устройства разбивается на три части, при этом внутренние параметры камеры ищутся лишь один раз:

- Поиск внутренних параметров камеры
- Найти соответствующие «особые» точки на изображениях
- Посчитать матрицу поворотов и вектор сдвига.

1. ВНУТРЕННИЕ ПАРАМЕТРЫ КАМЕРЫ

1.1. Модель регистрирующей камеры

Строго говоря, различные точки пространства предметов отображаются оптической системой камеры в пространстве изображений на различных расстояниях от фокальной плоскости. Однако, если расстояние между камерой и наблюдаемой сценой значительно превышает фокусное расстояние оптической системы, можно считать, что изображение строится в ее фокальной плоскости. В этом случае можно воспользоваться проективной моделью камеры, в которой изображение трехмерного объекта получается проектированием его в фокальную плоскость (плоскость изображения) через единственную точку, называемую оптическим центром. Прямая линия, перпендикулярная плоскости изображения и проходящая через эту точку, называется оптической осью камеры, а точка пересечения оптической оси с плоскостью изображения — главной точкой.

Определим в трехмерном пространстве ортогональную правую систему координат OXYZ, начало которой совпадает с оптическим центром, ось OZ — с оптической осью камеры. Такая система называется стандартной системой координат камеры. Пусть плоскость изображения находится на расстоянии f от оптического центра. В этой плоскости зададим систему координат oxy с началом в главной точке и осями ox и oy, параллельными осям OX и OY соответственно (рис. 1). Легко убедиться, что в стандартной

системе координат проекцией точки трехмерного пространства M с координатами (X,Y,Z) является точка m в плоскости изображения с координатами (x,y), причем $x = fX/Z$, $y = fY/Z$.

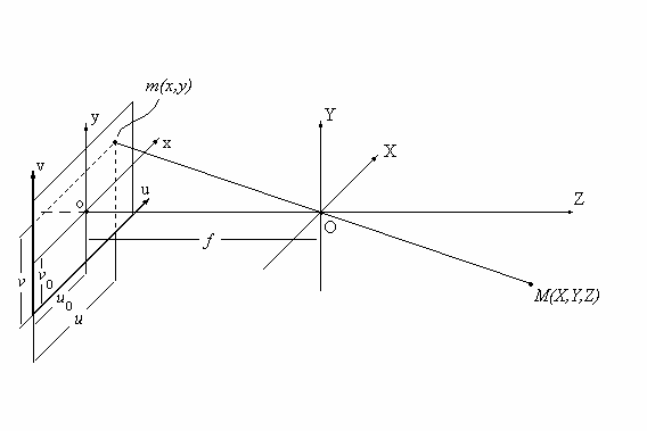


Рис. 1. Система координат проективной камеры

Для полного описания камеры следует учесть, что для регистрации изображения в плоскости изображения камеры помещается какой-либо фотоприемник. В общем случае измерение координат в фотоприемнике осуществляется в единицах, отличных от единиц, задающих координаты в стандартной системе. Поэтому для полного описания камеры необходимо выразить координаты точки m в естественных единицах фотоприемника. В достаточно общем для любых фотоприемников виде (рис. 1) это может выглядеть как

$$u = x/w + u_0 \ , \quad v = y/h + v_0 \, ,$$

где (u_0, v_0) — координаты главной точки относительно начала координат фотоприемника (в естественных координатах фотоприемника); w и h — масштабы вдоль осей ox и oy (например, расстояния между ячейками матричного фотоприемника вдоль строк и столбцов).

В новой системе координаты проекции точки m примут вид:

$$u = \frac{fX}{wZ} + u_0 \, , \ v = \frac{fY}{hZ} + v_0 \, . \quad (1.1)$$

Для последующего изложения введем трехмерный вектор $\mathbf{M} = (X,Y,Z)^T$, соответствующий точке M, и двумерный вектор

$\mathbf{m} = (x, y)^T$, соответствующий точке m. Определим также вектор однородных внутренних координат камеры $\mathbf{v} = (u, v, 1)^T$. Используя эти обозначения, соотношения (1.1) можно представить в компактной векторно-матричной записи:

$$Z\mathbf{v} = \mathbf{AM}, \qquad (1.2)$$

где $\mathbf{A} = \begin{bmatrix} f/w & 0 & u_0 \\ 0 & f/h & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ — матрица, известная под названием матрицы внутренних параметров камеры, поскольку она содержит только параметры оптической системы и фотоприемника камеры.

1.2. Калибровка камеры

Внутренние параметры камеры находим самокалибровкой камеры по плоскому шаблону.

Пусть у нас есть точка на плоскости $m = [u, v]^T$, которая является проекцией некоторой точки $M = [X, Y, Z]^T$ в пространстве. Соотношение между этими двумя точками можно представить в следующем виде:

$$\mathbf{s}m = \mathbf{A[R\ t]}M \qquad (1.3)$$

где s — некоторый скаляр, пара (R t) — так называемые внешние параметры камеры, и A — матрица, представляющая собой внутренние параметры камеры: $A = \begin{bmatrix} \alpha & \gamma & x_0 \\ 0 & \beta & y_0 \\ 0 & 0 & 1 \end{bmatrix}$, где (x_0, y_0) — координаты основной точки, α и β — коэффициенты сжатия по осям Ox и Oy, соответственно, и γ — коэффициент асимметрии между осями изображения.

Без потери общности можем принять Z = 0. Обозначим через r_i i-й столбец матрицы вращения R. Тогда из (1.3) для точки на объекте $\tilde{M} = [X, Y, 1]^T$ и соответствующей точки на изображении $\tilde{m}$ имеет место следующее соотношение:

$$\mathbf{s}\,\tilde{m} = \mathbf{H}\,\tilde{M}, \qquad (1.4)$$

где H — матрица гомографии изображения, H = A [r_1 r_2 t].

В [1] приводится способ расчета матрицы гомографии. Мы предположим, что данная матрица нами уже вычислена.

Положим, что H = [h_1 h_2 h_3], тогда из (3.4) мы получим:

$$[\,h_1 \;\; h_2 \;\; h_3\,] = \lambda\, \mathrm{A}\, [\,r_1 \;\; r_2 \;\; \mathrm{t}],$$

где λ — скаляр.

Используя, что r_1 и r_2 ортогональны, получаем

$$h_1^T A^{-T} A^{-1} h_2 = 0 \quad (1.5)$$

$$h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \qquad (1.6)$$

Положим

$v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T$, тогда (1.5) и (1.6) можно переписать в следующем виде:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} \mathrm{b} = 0. \qquad (1.7)$$

В (1.7) b — это 6D вектор, равный [$B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}$], где B_{ij} — элементы матрицы B= $A^{-T}A^{-1}$. Используя n изображений шаблона и ниже указанные уравнения (1.8), мы можем вычислить внутренние параметры камеры (матрицу A) при помощи (1.7).

$$\begin{aligned} v_0 &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \\ \lambda &= B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \\ \alpha &= \sqrt{\lambda / B_{11}} \\ \beta &= \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \\ \gamma &= -B_{12}\alpha^2\beta/\lambda \\ u_0 &= cv_0/\alpha - B_{13}\alpha^2/\lambda \end{aligned} \qquad (1.8)$$

Более подробно процесс расчета матрицы внутренних параметров камеры и матриц гомографии изображений можно прочитать в [1] и [2]. Там же приведены результаты работы алгоритмов на реальных объектах.

2. «ОСОБЕННОСТИ» ИЗОБРАЖЕНИЙ

Точечная особенность изображения m — это такая точка изображения, окрестность которой o(m) можно отличить от окрестности любой другой

точки изображения o(n) в некоторой другой окрестности особой точки o2(m). Точечной особенности сцены должна соответствовать точечная особенности изображений. Обратное неверно: существуют такие особые точки изображения, которым не соответствует никакие особые точки сцены. Такие точки называется ложными особенностями сцены.

Отделение ложных особенностей от настоящих является одной из основных проблем при отслеживании движении камеры или при реконструкции трехмерной сцены. Усугубляется она тем, что надежно решить ее, работая только с одним изображением, невозможно. Необходимо устанавливать соответствие между особенностями нескольких изображений, а затем построить некоторую модель, которой эти соответствия будут удовлетворять наилучшим образом. Особые точки, соответствия которых не будут удовлетворять такой модели, будут признаны ложными, или выбросами.

Методы построения таких моделей в последнее десятилетия бурно развивались. Алгоритмы, о которых рассказывается далее, применимы для всех точечных особенностей изображений. Однако необходимо дополнительно отметить, что при слежении за ложной точечной особенностью изображения, на разных кадрах последовательности ей будут соответствовать разные точки сцены.

Для простоты в качестве окрестности точки изображения берется прямоугольное окно небольшого размера. Для сравнения таких прямоугольных окон могут использоваться различные меры на изображениях (например, обычная кросс-корреляция).

Большинство детекторов точечных особенностей работают сходным образом: для каждой точки изображения вычисляется некоторая функция от ее окрестности. Точки, в которых эта функция достигает локального максимума, очевидно можно отличить от всех точек из некоторой ее окрестности.

Существует целый набор функций, которые можно использовать для обнаружения точечных особенностей. Чаще всего для задач отслеживания точек сцены применяться функции, находящие в изображении структуры, похожие на угол, — уголки (corners). Детекторы, использующие такие функции, называются детекторами углов. Именно они чаще всего применяются для решения задач отслеживания точечных особенностей сцены.

2.1. Слежение за точечными особенностями

В общем случае под слежением за точечными особенностями сцены понимается определение координат проекции точки сцены в текущем кадре,

если известны координаты ее проекции в предыдущем, и неизвестно ничего о камере, с которых получены изображения. Однако ничего о точечных особенностях сцены нам не известно. В изображениях мы можем выделить только набор особенностей изображения, которые могут как соответствовать, так и не соответствовать каким-то особенностям сцены. Поэтому, работая исключительно в пространстве изображений, мы можем отслеживать только точечные особенности изображений, а не сцены. Разумеется, предполагаемая природа точечных особенностей изображения должна учитываться. Например, т.к. точечная особенность считается лежащей на плоском сегменте сцены, то ее изображение может претерпевать перспективные искажения, которые в свою очередь можно приблизить аффинными. Учитывая это обстоятельство, задачу слежения за особенностями формулируется следующим образом: дана последовательность изображений одной и той же сцены S, полученная с движущейся или неподвижной камеры, и набор точечных особенностей {N}, выделенных в первом кадре последовательности. Для каждой точечной особенности n из {N} найти такие точки n(t) на всех изображениях, что их окрестности будут максимально близки к окрестности n(0), с учетом предполагаемой природы искажения ее окрестности и движения точки.

В простейшем случае в новом кадре находится ближайшая к предыдущему положению точка с наиболее близкой в используемой мере окрестностью. Такое слежение за особенностями сцены может рассматриваться как дискретизация оптического потока, т.е. определение его величины не на всем изображении, а в нескольких отдельных точках. Для каждой точки вычисляется только ее смещение от кадра к кадру (2 параметра).

При изменении точки зрения меняется и освещенность окрестности точки сцены, а значит, меняется и яркость пикселей изображения этой окрестности. Эти изменения можно минимизировать, если воспользоваться нормализацией изображения окрестности особенности. С другой стороны, изменения освещенности при небольших смещениях камеры от кадра к кадру можно описать в форме аффинных искажений a * I + b. При поиске нового положения особенности в некоторых алгоритмах рассчитывается и параметры изменения освещения ее окрестности a и b.

Чаще всего параметры искажения окрестности особых точек в дальнейшем совершенно не используются. Вычисление этих параметров производятся только для того, чтобы получить как можно более точные и истинные положения проекций особых точек на кадре.

Необходимо упомянуть, что любой алгоритм слежения за особенностями может быть применен к любым точкам изображения, а не обязательно

только к особенностям. Однако в этом случае невозможно корректно определить ее положение в новом кадре. Например, если применить слежения за точкой на некоторой границе изображения, то скорее всего будет найдена тоже точка на границе, но она может на самом деле и не соответствовать той же точке сцены.

2.2. Простая и расширенная схема работы системы слежения

Схема работы любой системы слежения за особенностями состоит из двух основных этапов:

Схема А.

Этап 1 (Детектирование)

Определить в первом кадре особенности изображения.

Этап 2 (Слежение)

Для каждого последующего кадра: для каждой особенности Feature(i) найти новое положение особенности в кадре t.

Особенности обычно выделяются в первом кадре последовательности и отслеживаются на протяжении всех остальных. Но при движении камеры или сцены часть особенностей может время от времени пропадать из вида, или изображения их окрестностей будут искажаться настолько, что перестанут быть особенностями. В этом случае слежение за такими точками становится невозможным.

В реальных практических системах слежения за особенностями постоянно вычисляется качество отслеживаемых особенностей (степень их особенности). Если оно падает ниже некоторого порога, то такие особенности отбрасываются. Вместо них, а также вместо пропавших из вида, в текущем изображении ищутся новые особые точки, которые в дальнейшем будут отслеживаться вместе со старыми, еще качественными особенностями.

С учетом оценки качества особенностей, и дополнения множества отслеживаемых особенностей схема А расширяется до:

Схема B.

Этап 1 (Детектирование и оценка)

1. Найти набор особенностей {Features}
2. Определить качество всех особенностей — Quality({Features})
3. Оставить только особенности, чье качество выше некоторого заранее или динамически определенного порога, получив множество {GoodFeatures}

Этап 2 (Слежение и оценка)

Для каждого последующего кадра:

1. Найти в текущем кадре новое положение всех особенностей из {GoodFeature} — слежение
2. Определить текущее качество всех {GoodFeatures}
3. Оставить только те особенности, чье качество удовлетворяет некоторому критерию
4. Если число отслеживаемых точек падает ниже требуемого, применить детектор к текущему изображению и добавить в {GoodFeatures} новые точки.

2.3. Нахождение набора особенностей

За последние 20 лет было создано громадное количество различных детекторов точечных особенностей изображений. Все они используют разные подходы к формированию функции оценки пикселей для нахождения особенностей. В нашем случае, в системах слежения за особенностями, определяющим параметром качества детектора будет качество последующего отслеживания.

Чаще всего используется детектор Харриса [3]. Для каждого пикселя изображения вычисляется значение особой функции отклика угла, оценивающая степень похожести изображения окрестности точки на угол.

Для этого вначале рассчитывается матрица:

$$M = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix},$$

где I(x, y) — яркость изображения в точке (x, y)

Если оба ее собственных значения велики, то даже небольшое смещение точки (x, y) в сторону вызывает значительные изменения в яркости. Что и

соответствует особенности изображения. Функция отклика угла записывается в следующем виде:

$$R = \det M - k(traceM)^2$$

Параметр k обычно полагается 0.04 (предложено Харрисом). Точки изображения, соответствующие локальным максимумам этой функции и признаются особенностями. Для достижения субпиксельной точности может использоваться квадратичная интерполяция.

Для снижения влияния шумов на найденные особенности используется сглаживание по Гауссу, но не в самом изображении, а в картах частных производных: $\left(\frac{\partial I}{\partial x}\right)^2, \left(\frac{\partial I}{\partial y}\right)^2, \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right)$.

Во многих случаях находится чересчур большое количество углов, из-за чего в дальнейшем их будет сложно отслеживать. Поэтому вводится ограничение на минимальное расстояние между найденными особенностями, и все лишние отбрасываются.

2.4. Развитие алгоритмов слежения

Все современные алгоритмы слежения за особенностями опираются на работу 1981 году Лукаса и Канаде. В 1991 году математическая формулировка этого алгоритма была изменена, и стала основой для всех последующих обобщений с учетом аффинных искажений окрестности и освещенности. Путем замены соответствующих переменных на константы любой из них превращается в обычный алгоритм *Lucas—Kanade*.

- *Lucas—Kanade* — особенность считается только смещающейся, без искажений [5];
- *Tomasi—Kanade* — переформулирование *Lucas—Kanade*. Движение считается смещением, и рассчитывается путем итеративного решения построенной системы линейных уравнений [6];
- *Shi—Tomasi—Kanade* — учитывает аффинные искажения особенности [4];
- *Jin—Favaro—Soatto* — модификация *Shi—Tomasi—Kanade* с учетом аффинных изменений освещенности особенности [8].

2.5. Алгоритм «Lucas—Kanade»

Этот алгоритм в принципе применим для функций любой размерности n. Пусть x — особенность первой функции F, необходимо найти такую точку x+h функции G, что разность окрестностей этих точек по мере — минимальна.

Расстояние между окрестностями записывается в виде: $E = \sum_{x \in R} [F(x+h) - G(x)]^2$,где F(x), G(x) — две функции.

Функцию F(x) с помощью разложения в ряд Тейлора можно приближенно представить в виде: $F(x+h) \approx F(x) + h\frac{\partial}{\partial x}F(x)$, где $\frac{\partial}{\partial x} = [\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_2}...\frac{\partial}{\partial x_n}]^T$ — градиент.

Используя это приближение, ищется минимум E путем дифференцирования и приравнивания производной к нулю:

$$0 = \frac{\partial}{\partial h}E \approx \frac{\partial}{\partial h}\sum_x [F(x) + h\frac{\partial F}{\partial x} - G(x)]^2 = \sum_x 2\frac{\partial F}{\partial x}[F(x) + h\frac{\partial F}{\partial x} - G(x)].$$

Отсюда смещение h можно получить как

$$h = \left[\sum_x \left(\frac{\partial F}{\partial x}\right)^T [G(x) - F(x)]\right]\left[\sum_x \left(\frac{\partial F}{\partial x}\right)^T \left(\frac{\partial F}{\partial x}\right)\right]^{-1}.$$

Как было указано ранее, задача слежения за особенностями без учета аффинных искажений является поиском величины оптического потока в наборе точек. Поэтому метод Lucas—Kanade часто применяется для поиска оптического потока во всем изображении.

2.6. Алгоритм «Tomasi—Kanade»

В этом алгоритме движение особых точек также описывается смещением вида:

$$\text{delta}(x) = x + d.$$

Как и в предыдущем алгоритме, задача заключается в поиске такого d, при котором минимизируется разность окон особенностей: $\varsigma = \sum_W [I(x+d, t+\tau) - I(x,t)]^2$

Опять же аналогично функция изображения раскладывается с помощью ряда Тейлора:

$$I(x+d,t+\tau) \approx I(x,t) + \nabla I(x,t)^T d + I_t(x,t)\tau \text{ , где}$$
$$\nabla I^T = [I_u, I_v] = [\partial I / \partial u, \partial I / \partial v],\ I_t = \partial I / \partial t .$$

Тогда разницу между окнами по мере можно переписать в виде:

$$\varsigma \approx \sum_W \left(\nabla I(x,t)^T d + I_t(x,t)\tau\right)^2 .$$

Продифференцировав это выражение по d и приравняв производную к нулю, получаем линейную систему относительно d: Cd=g, где:

$$C = \sum_W \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix} \text{ и } g = -\tau \sum_W I_t \begin{bmatrix} I_u I_v \end{bmatrix}^T .$$

Из этой системы d получается как: $d_k = C^{-1} g$.

С учетом приближения функции изображения с помощью ряда Тейлора, решение получается неточным. Для его уточнения удобно применить итеративную процедуру Ньютона—Рафсона. Т.е. полученное на первом шаге решение берется за новое первое приближение, уравнение снова и снова.

На каждом шаге рекомендуется пересчитывать окрестность особенности с помощью билинейной интерполяции для достижения субпиксельной точности нахождения положения особенности в новом кадре.

Если интервал времени между кадрами принять за 1, получается следующий алгоритм:

$$\begin{cases} d_0 = 0 \\ d_{k+1} = d_k + C^{-1} \sum_W \left[\left(I(x,t) - I(x+d_k, t+1) \right) \nabla I(x,t) \right] \end{cases}.$$

Таким образом, этот алгоритм слежения фактически является поиском точки, в которой достигается минимум некоторой функции, методом градиентного спуска. Во время каждой итерации мы сдвигаемся вдоль направления градиента изображения в текущей точке.

2.7. Алгоритм «Shi—Tomasi—Kanade»

В этом алгоритме впервые учитываются аффинные искажения изображения окрестности особых точек, поэтому движение пикселей окна особенности описывается в виде Ax + d, где A — матрица (2×2), а d — смещение (2×1).

Задача слежения за особенностью сводится к проблеме проблема определения параметров движения и искажения окна особенности, при которой минимизируется разность:

$\varsigma = \iint_W \left[J(\Delta x + d) - I(x)\right]^3 w(x)dx$, где W — окно особенности, а w — весовая функция (может использовать, а может и быть равна 1 во всем окне), J(x) и I(x) — два изображения.

Выражение дифференцируется относительно параметров движения, и производная приравнивается к 0. Затем система линеаризуется с помощью разложения функции изображения в ряд Тейлора: $J(\Delta x + d) = J(x) + g^T(u)$. Это дает нам линейную 6x6 систему: Tz = a, где в векторе z объединены все искомые параметры: $z^T = \left[d_{xx} d_{yx} d_{xy} d_{yy} d_x d_y\right]$.

Вектор ошибки a записывается в виде:

$$a = \iint_W [I(x) - J(x)] \begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix} wdx .$$

А матрицу размерности 6×6 T можно представить следующим образом:

$$T = \iint_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} wdx ,$$

$$U = \begin{bmatrix} x^3 g_x^3 & x^3 g_x g_y & xyg_x^3 & xyg_x g_y \\ x^3 g_x g_y & x^3 g_y^3 & xyg_x g_y & xyg_y^3 \\ xyg_x^3 & xyg_x g_y & y^3 g_x^3 & y^3 g_x g_y \\ xyg_x g_y & xyg_y^3 & y^3 g_x g_y & y^3 g_y^3 \end{bmatrix},$$

$$V^T = \begin{bmatrix} xg_x^3 & xg_xg_y & yg_x^3 & yg_xg_y \\ xg_xg_y & xg_y^3 & yg_xg_y & yg_y^3 \end{bmatrix},$$

$$Z = \begin{bmatrix} g_x^3 & g_xg_y \\ g_xg_y & g_y^3 \end{bmatrix}.$$

Полученная система решается также итеративно по методу Ньютона—Рафсона.

Если движение считается не аффинным, а просто смещением, то первые четыре элемента искомого вектора z обращаются в 0, и значимыми остаются только последние два. Алгоритм превращается в алгоритм *Tomasi—Kanade* .

2.8. Автоматический отбор некачественных особенностей

Авторы алгоритма *Shi—Tomasi—Kanade* в своей работе показывают, что непосредственно для слежения за особенностями больше всего подходит обычный *Tomasi—Kanade* алгоритм. Алгоритм с учетом аффинных искажений в этом случае можно использовать для определения текущего качества особенности, т.е. степени близости ее окна к окну детектированной особенности. Однако они не предлагают никаких автоматических методов определения порога, когда особенность признается плохой и перестает отслеживаться. Этот недостаток попытались исправить авторы [5–7], с помощью введения правила *X84*.

Предположим, что после точного вычисления движения особенности, яркость соответствующих пикселей будет совпадать с точностью до некоторой ошибки, распределенной по нормальному закону (гауссову шуму): $I(\delta(x),t) - I(x,0) \approx \eta(0,1)$.

Т.к. квадрат нормального распределения представляет собой распределение хи-квадрат, мы получаем: $[I(\delta(x),t) - I(x,0)]^2 \approx \chi^2(1)$.

Сумма n переменных, с распределением по хи-квадрат с одной степенью свободы, распределена как хи-квадрат с n степенями свободы. Поэтому, разница между особенностями по окну N×N W имеет вид: $\zeta = \sum_W [I(\delta(x),t) - I(x,0)]^2 \approx \chi^2(N^2)$.

При возрастании степени свободы, распределение хи-квадрат приближается к нормальному распределению. При степени свободы больше 30, нормальное распределение можно использовать как приближение хи-

квадрат. Если размер окна особенности, по крайней мере, 7×7, то можно спокойно утверждать, что: $\zeta \approx \eta(N^2, 2N^2)$.

Поэтому, если два окна, которые мы сравниваем, принадлежат плохой особенности, то разница должна быть выбросом по отношению к нормальному распределению разницы между хорошими особенностями. Для определения таких выбросов предлагается использовать правило *X84*, являющееся устойчивым за счет использования медианы и медианной дисперсии, вместо обычного среднего и дисперсии.

Согласно правилу, все значения, отклоняющиеся от медианы на более чем k Медианных абсолютных дисперсий (Median Absolute Deviation — MAD):

$$MAD = \underset{i}{mad}\{|\zeta_i - \underset{j}{mad}\zeta_j|\},$$

где ζ — разница между текущим окном особенности и ее окном в первом кадре.

При значении k = 5.2 это соответствует примерно 3.5 стандартным дисперсиям, и интервал [u-3.5sigma, u+3.5sigma) содержит более 99.9% значений нормального распределения.

Как утверждается, правило позволяет эффективно определять уменьшение качества особенности и отсекать ее во всех случаях, при которых количество выбросов меньше 50%.

3. МАТРИЦА ПОВОРОТОВ И ВЕКТОР СДВИГА

Рассмотрим общий случай, когда оптические оси камер не параллельны, и направление смещения оптического центра одной камеры относительно оптического центра другой произвольно (рис. 2). Введем для каждой камеры свою стандартную систему координат. Пусть первой камере соответствует система координат O′X′Y′Z′, а второй — O″X″Y″Z″ (рис. 2). Пусть вектор $\mathbf{M}' = (X', Y', Z')^T$ характеризует координаты некоторой точки M трехмерного пространства в системе первой камеры, а вектор $\mathbf{M}''(X'', Y'', Z'')^T$ — в системе второй. Переход от глобальной системы координат к стандартным системам первой и второй камер осуществляется с помощью преобразований $\mathbf{M}' = \mathbf{R}'\mathbf{M} + \mathbf{t}'$ и $\mathbf{M}'' = \mathbf{R}''\mathbf{M} + \mathbf{t}''$ соответственно. Учитывая это, легко показать, что связь между векторами $\mathbf{M}'$ и $\mathbf{M}''$ задается соотношением

$$\mathbf{M}'' = \mathbf{R}\mathbf{M}' + \mathbf{t}, \qquad (3.1)$$

где $\mathbf{R} = \mathbf{R}''\mathbf{R}'^{T}$ — ортогональная матрица, описывающая ориентацию системы координат второй камеры относительно первой, а $\mathbf{t} = -\mathbf{R}''\mathbf{R}'^{T}\mathbf{t}' + \mathbf{t}''$ — вектор трансляции, определяющий положение оптического центра второй камеры в системе координат первой. Матрицу $\mathbf{R}$ и вектор $\mathbf{t}$ принято называть внешними параметрами системы регистрации.

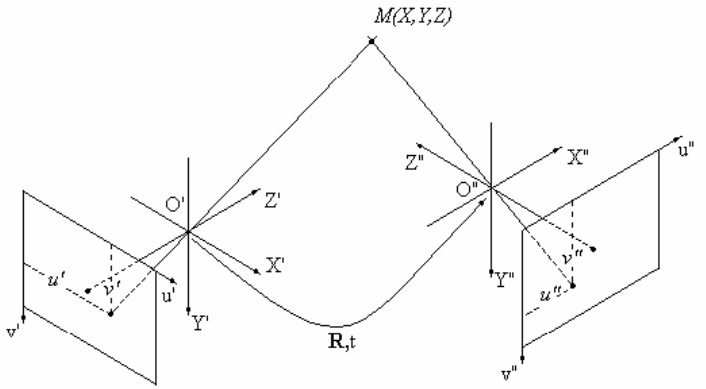


Рис. 2. Система двух произвольно ориентированных камер

В общем случае трехмерные координаты точки могут быть заданы в системе, не совпадающей со стандартной системой координат камеры (назовем ее глобальной). Пусть OXYZ — глобальная система координат, а O'X'Y'Z' — стандартная система координат камеры. Переход от системы OXYZ к системе O'X'Y'Z' можно осуществить поворотом координатных осей к системе $OX''Y''Z''$ и последующим смещением начала координат. Тогда связь между координатами точки M в глобальной и стандартной системе может быть представлена как

$$\mathbf{M}' = \mathbf{R}\mathbf{M} + \mathbf{t}, \qquad (3.2)$$

где $\mathbf{M}$ и $\mathbf{M}'$ — векторы пространственных координат точки M в глобальной и стандартной системах, соответственно; $\mathbf{R}$ — матрица размерности 3×3, описывающая поворот стандартной системы координат относи-

тельно глобальной; компонентами матрицы являются направляющие косинусы осей глобальной системы в стандартной системе координат ; **t** — трехмерный вектор смещения начала координат глобальной системы относительно начала координат стандартной.

Теперь у нас есть как минимум четыре «особые» точки (см. ранее) и мы знаем как они связаны ((3.2)). Найдем матрицу поворота и вектор сдвига. У нас есть уравнения вида:

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = R \begin{pmatrix} U_i \\ V_i \\ W_i \end{pmatrix} + t$$

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = A^{-1} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \quad , i=1..4,$$

$$\begin{pmatrix} U_i \\ V_i \\ W_i \end{pmatrix} = A^{-1} \begin{pmatrix} u_i \\ v_i \end{pmatrix}$$

где $(x_i, y_i)^T$ — точка в плоскости первого изображения, являющаяся проекцией точки в пространстве $(X_i, Y_i, Z_i)^T$, $(u_i, v_i)^T$ — это точка в плоскости второго изображения, являющаяся проекцией точки $(U_i, V_i, W_i)^T$ в пространстве, A^{-1} — матрица внутренних параметров камеры, R и t — искомые матрица вращения и вектор сдвига соответственно.

Каждое уравнение указано в «своих» координатах: первое в глобальной системе, второе в системе камеры в первом положении, третье в системе камеры, находящейся во втором положении.

С учетом того, что нам известны двухмерные координаты точки на каждом изображении и матрица внутренних параметров, мы можем считать, что и трехмерные координаты нам также известны. Таким образом, получаем четыре уравнения вида $\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = R \begin{pmatrix} U_i \\ V_i \\ W_i \end{pmatrix} + t$. Распишем их более подробно:

$$\begin{cases} X_1 = r_{11}U_1 + r_{12}V_1 + r_{13}W_1 + t_1 \\ X_2 = r_{11}U_2 + r_{12}V_2 + r_{13}W_2 + t_1 \\ X_3 = r_{11}U_3 + r_{12}V_3 + r_{13}W_3 + t_1 \\ X_4 = r_{11}U_4 + r_{12}V_4 + r_{13}W_4 + t_1 \end{cases}, \begin{cases} Y_1 = r_{21}U_1 + r_{22}V_1 + r_{23}W_1 + t_2 \\ Y_2 = r_{21}U_2 + r_{22}V_2 + r_{23}W_2 + t_2 \\ Y_3 = r_{21}U_3 + r_{22}V_3 + r_{23}W_3 + t_2 \\ Y_4 = r_{21}U_4 + r_{22}V_4 + r_{23}W_4 + t_2 \end{cases},$$

$$\begin{cases} Z_1 = r_{31}U_1 + r_{32}V_1 + r_{33}W_1 + t_3 \\ Z_2 = r_{31}U_2 + r_{32}V_2 + r_{33}W_2 + t_3 \\ Z_3 = r_{31}U_3 + r_{32}V_3 + r_{33}W_3 + t_3 \\ Z_4 = r_{31}U_4 + r_{32}V_4 + r_{33}W_4 + t_3 \end{cases}.$$

Мы видим, что неизвестные каждой системы не зависят от неизвестных других систем. В каждой системе 4 неизвестных: три из матрицы вращения и одна неизвестная из вектора сдвига. Поэтому нам и требовалось лишь четыре «особых» точки. Каждая система представляет линейные уравнения. Распишем более подробно первую систему:

$$\begin{cases} X_1 = r_{11}U_1 + r_{12}V_1 + r_{13}W_1 + t_1 \\ X_2 = r_{11}U_2 + r_{12}V_2 + r_{13}W_2 + t_1 \\ X_3 = r_{11}U_3 + r_{12}V_3 + r_{13}W_3 + t_1 \\ X_4 = r_{11}U_4 + r_{12}V_4 + r_{13}W_4 + t_1 \end{cases} \Rightarrow \begin{pmatrix} U_1 & V_1 & W_1 & 1 \\ U_2 & V_2 & W_2 & 1 \\ U_3 & V_3 & W_3 & 1 \\ U_4 & V_4 & W_4 & 1 \end{pmatrix} \begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \\ t_1 \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} \Rightarrow A \begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \\ t_1 \\ -1 \end{pmatrix} = 0,$$

где $A = \begin{pmatrix} U_1 & V_1 & W_1 & 1 & X_1 \\ U_2 & V_2 & W_2 & 1 & X_2 \\ U_3 & V_3 & W_3 & 1 & X_3 \\ U_4 & V_4 & W_4 & 1 & X_4 \end{pmatrix}$.

Приведем A к верхнетреугольному виду и получим решение:

$$\begin{aligned} t_1 &= -A_{45} \\ r_{13} &= -A_{35} - t_1 \\ r_{12} &= -A_{23}r_{13} - t_1 - A_{25} \\ r_{11} &= -A_{12}r_{12} - A_{13}r_{13} - t_1 - A_{15} \end{aligned}.$$

Или в общем виде:

$$
\begin{aligned}
t_i &= -A_{45} \\
r_{i3} &= -A_{35} - t_i \\
r_{i2} &= -A_{23} r_{i3} - t_i - A_{25} \\
r_{i1} &= -A_{12} r_{i2} - A_{13} r_{i3} - t_i - A_{15}
\end{aligned}
\qquad i=1,2,3,
$$

где A — это верхнетреугольная матрица, полученная для каждой системы уравнений.

Таким образом, мы нашли как изменилось положение камеры в пространстве на основе двух снимков.

СПИСОК ЛИТЕРАТУРЫ

1. **Harker M., O'Leary P.** Computation of Homographies // Department of Product Engineering University of Leoben, Австрия.
2. **Zhang Z.** A Flexible New Technique for Camera Calibration // Microsoft Research, One Microsoft Way, США, Рэдмонд — 1998.
3. **Pollefey M.** Tutorial on 3d reconstruction // 2000 http://www.esat.kuleuven.ac.be/~pollefey/tutorial/
4. **Shi J., Tomasi C.** Good features to track. // IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94), IEEE Computer Society, Сиэтл — 1994. http://citeseer.ist.psu.edu/shi94good.html
5. **Lucas B., Kanade T.** An Iterative Image Registration Technique with an Application to Stereo Vision // Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI), 1981
6. **Kanade T.** Detection and tracking of point features // TR Carnegie-Melon University, 1991
7. **Smith P., Sinclair D., Cipolla R., Wood K.** Effective Corner Matching. // In Proc. of BMVC'98, V.2 — P.545–556, Великобритания — 1998.
8. **Jin H., Favaro P., Soatto S.** Real-time Feature Tracking and Outlier Rejection with Changes in Illumination. // Intl. Conf. on Computer Vision, 2001
9. **Конушин А.** Слежение за точечными особенностями сцены, http://ict.edu.ru/ft/002409/num4pntrac.pdf

А.В.Козырева

О НЕКОТОРЫХ СПОСОБАХ КАЛИБРОВКИ ВИДЕОКАМЕРЫ

В данной статье речь идет о некоторых способах калибровки видеокамер. Одной из центральных задач в области машинного зрения является задача определения ориентации и расположения камеры в пространстве по изображению, полученному с ее помощью. Эту задачу также называют задачей калибровки камеры или определения ее параметров. Решение данного вопроса требуется в различных областях: картографии, системах распознавания объектов, системах управления компьютером посредством определения положения рук или направления взгляда, системах управления роботами и т.д.

Технологии калибровки можно разделить на 2 вида.

1. Фотограмметрическая калибровка.

Калибровка камер производится наблюдением за калибровочным объектом, геометрия которого в 3D пространстве известна с большей точностью. Калибровка может быть сделана очень рационально. Калибровочный объект обычно состоит из 2 или 3 плоскостей ортогональных друг другу. Иногда плоскости подвергаются точно заданному преобразованию. Эти подходы нуждаются в дорогих калибровочных аппаратах и их скрупулезной установке.

2. Самокалибровка.

Технологии этой категории не используют калибровочных объектов. Только движение камеры в статической сцене. Если изображения будут браться от тех же самых камер с фиксированными внутренними параметрами, соответствия между тремя картинками достаточно для получения и внутренних, и внешних параметров, которые позволят реконструировать 3D структуру.

Способ 1

Технология, представленная Zhengyou Zhang [1] требует только камеру для наблюдения за плоским объектом, показанным с нескольких сторон. Объект может быть отпечатан на лазерном принтере и прикреплен к приемлемой плоской поверхности (рис. 1). Камера или объект могут двигаться, при этом характер движения не известен заранее. Предложенный подход лежит между фотограмметрической калибровкой и самокалибровкой, потому что мы используем больше 2D информацию, нежели 3D или вообще

только её одну. По сравнению с самокалибровкой этот метод более устойчив.

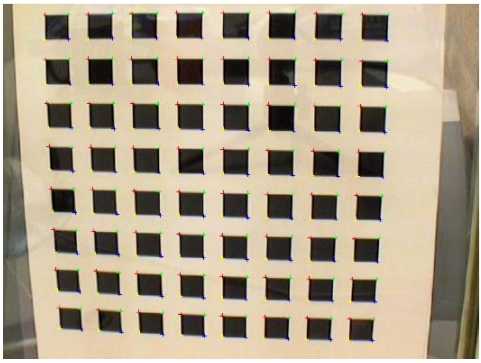


Рис.1. Шаблон для самокалибровки камеры

Пусть у нас есть точка на плоскости m = $[u,v]^T$, которая является проекцией некоторой точки M = $[X,Y,Z]^T$ в пространстве. Соотношение между этими двумя точками можно выписать в следующем виде:

$$\text{sm} = \text{K}[\text{R t}]\text{M}, \tag{1.1}$$

где s — некоторый скаляр, пара (R t) — так называемые внешние параметры камеры, и K — искомая матрица, представляющая собой внутренние параметры камеры: $K = \begin{bmatrix} \alpha & \gamma & x_0 \\ 0 & \beta & y_0 \\ 0 & 0 & 1 \end{bmatrix}$, где (x_0, y_0) — координаты основной точки, α и β — коэффициенты сжатия по осям Ox и Oy, соответственно, и γ — коэффициент асимметрии между осями изображения.

Без потери общности можем принять Z = 0. Обозначим через r_i i-й столбец матрицы вращения R. Тогда из (1.1) для точки на объекте $\tilde{M} = [X,Y,1]^T$ и соответствующей точки на изображении $\tilde{m}$ имеет место следующее соотношение:

$$\text{s}\,\tilde{m} = \text{H}\,\tilde{M}, \tag{1.2}$$

где H — матрица гомографии изображения, H = K [r_1 r_2 t].

Существует много способов вычисления гомографии, один из них приведен в [2]. Мы подсчитаем матрицу гомографии, используя критерий максимального правдоподобия. Введем ковариационную матрицу Λ_{m_i} :

$$\sum_i (m_i \hat{-} \hat{m}_i)^T \Lambda_{m_i}^{-1} (m_i \hat{-} \hat{m}_i)$$

$\hat{m}_i = \frac{1}{\bar{h}_3^T M_i} \begin{bmatrix} \bar{h}_1^T M_i \\ \bar{h}_2^T M_i \end{bmatrix}$, где $\bar{h}_i$ — i-я строка матрицы H.

На практике $\Lambda_{m_i} = \sigma^2$ для всех i. В этом случае выражение выше преобразуется в более простое нелинейное: $\min_H \sum_i \| m_i \hat{-} \hat{m}_i \|^2$. Это выражение минимизируем методом Левенберга (алгоритм представлен в [3]).

Положим, что H = [h_1 h_2 h_3], тогда из (1.2) мы получим:

$$[\, h_1 \;\; h_2 \;\; h_3 \,] = \lambda \, \mathrm{K} \, [\, r_1 \;\; r_2 \;\; \mathrm{t} \,],$$

где λ — скаляр.

Используя то, что r_1 и r_2 ортогональны, получаем

$$h_1^T K^{-T} K^{-1} h_2 = 0 \tag{1.3}$$

$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 \tag{1.4}$$

Положим

$$v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T,$$

тогда (1.3) и (1.4) можно переписать в следующем виде:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} \mathrm{b} = 0. \tag{1.5}$$

В (1.5) b — это 6D вектор, равный [$B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}$], где B_{ij} — элементы матрицы B = $K^{-T}K^{-1}$. Используя n изображений шаблона и ниже указанные уравнения (1.6), мы можем вычислить внутренние параметры камеры (матрицу K) при помощи (1.5).

$$
\begin{aligned}
v_0 &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \\
\lambda &= B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \\
\alpha &= \sqrt{\lambda / B_{11}} \\
\beta &= \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \\
\gamma &= -B_{12}\alpha^2\beta/\lambda \\
u_0 &= cv_0/\alpha - B_{13}\alpha^2/\lambda.
\end{aligned}
\qquad (1.6)
$$

С.О. Новиков, О.А. Лебедев и Г.В. Захаренко в своей статье «Измерение и исследование трехмерных объектов в условиях неполной информации» [2] предлагают иной способ расчета матрицы H. Желающие могут о нем прочитать самостоятельно.

Способ 2

Рассмотрим вариант калибровки камеры при помощи объектива с переменным фокусным расстоянием. Этот способ предложила Марина Колесник в своей статье «Техника калибровки для объективов с переменным фокусным расстоянием» [3].

Введем понятие фрейма. Под фреймом будем понимать множество изображений, полученных при одном фокусном расстоянии.

Рассмотрим камеру, оснащенную объективом с переменным фокусным расстоянием. Камера выполняет последовательную съемку сцены, что позволит рассмотреть подпоследовательность фреймов N и N + 1. Сделаем следующие предположения относительно камеры и изображений.

Оптический центр камеры постоянен вне зависимости от изображений и фреймов.

Камера может вращаться, но изображения фреймов N и N + 1 должны в достаточной мере перекрывать друг друга для установления соответствия между точками.

Оптическое искажение удалено с обоих изображений. Внутренние параметры камеры известны для изображения фрейма N.

Эти условия позволят нам решить уравнения, описывающие вариацию фокальной линзы между изображениями фреймов N и N + 1.

Будем работать в координатной системе камеры (стандартная координатная система). Перспективная проекция матрицы P моделируется четырьмя внутренними параметрами:

$$\mathbf{P} = \begin{vmatrix} -\alpha_u & 0 & u_0 \\ 0 & -\alpha_v & v_0 \\ 0 & 0 & 1 \end{vmatrix}, \qquad (2.1)$$

где u_0, v_0 — координаты центральной точки и α_u, α_v — скалярные множители для строк и столбцов соответственно. Положим $\alpha_u, \alpha_v, u_0, v_0$ как известные внутренние параметры для изображения рамки N. Заметим, что α_u и α_v — линейные функции фокальной линзы, параметры $\alpha'_u, \alpha'_v, u'_0, v'_0$ рамки N + 1 определены как:

$$\{f' = \mu f\} \Rightarrow \{\alpha'_u = \mu\alpha_u, \alpha'_v = \mu\alpha_v, u'_0 = u_0, v'_0 = v_0\}. \qquad (2.2)$$

Фокусное расстояние f, f′ соответствующих рамок N, N + 1 и μ — известные переменные. Уравнение оптического луча <C, a> определено как пиксель (точка) **a** и оптический центр камеры C связаны уравнением:

$$A = \lambda \mathbf{P}^{-1}\tilde{a}, \qquad (2.3)$$

где $\tilde{a}$ = [a_1, a_2, 1] — однородный координатный вектор точки **a**, 3D вектор A определен как точка на оптическом луче <C, a> и λ — изменения между $-\infty$ и $+\infty$.

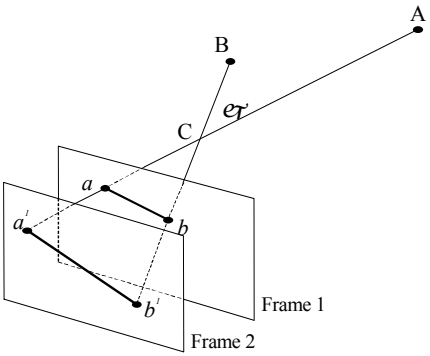


Рис. 2. Геометрия двух последовательных изображений для камеры с переменным фокусом

Будем рассматривать геометрию двух последовательных изображений, представленную на рис. 2. Выберем две 3D точки A и B на сцене и их проекции на изображения a, a^1 и b, b^1 рамок N и N+1 соответственно. Угол, определенный оптическими лучами <a, C> и <b, C>, и угол, определенный соответствующими точками <a^1, C> и <b^1, C>, физически равны между собой и выражены следующим уравнением:

$$\cos\varphi = \frac{A^T B}{|A||B|}, \tag{2.4}$$

где AB — скалярное произведение 3D векторов A и B. Комбинируя (2.3) и (2.4) для рамок N и N+1, запишем:

$$\frac{(\mathbf{P}_1^{-1}\tilde{a})^T(\mathbf{P}_1^{-1}\tilde{b})}{\left|\mathbf{P}_1^{-1}\tilde{a}\right|\left|\mathbf{P}_1^{-1}\tilde{b}\right|} = \frac{(\mathbf{P}_2^{-1}\tilde{a}^1)^T(\mathbf{P}_2^{-1}\tilde{b}^1)}{\left|\mathbf{P}_2^{-1}\tilde{a}^1\right|\left|\mathbf{P}_2^{-1}\tilde{b}^1\right|}, \tag{2.5}$$

где: $\tilde{a},\tilde{a}^1,\tilde{b},\tilde{b}^1$ — однородные координаты точек изображения a, a^1, b и b^1, и P_1 и P_2 — перспективная проекция матрица для рамок N и N+1, выраженных в (2.1). Положим M в качестве значения левой стороны уравнения (2.5), что не содержит неизвестных переменных. На основе (2.5), используя выражения (2.1) и (2.2) для матриц P_1 и P_2, имеем:

$$\frac{\left(u_0-a_1^1\right)\left(u_0-b_1^1\right)}{\mu^2\alpha_u^2}+\frac{\left(v_0-a_2^1\right)\left(v_0-b_2^1\right)}{\mu^2\alpha_v^2}+1=$$
$$=M\sqrt{\frac{\left(u_0-a_1^1\right)^2}{\mu^2\alpha_u^2}+\frac{\left(v_0-a_2^1\right)^2}{\mu^2\alpha_v^2}+1}\times\sqrt{\frac{\left(u_0-b_1^1\right)^2}{\mu^2\alpha_u^2}+\frac{\left(v_0-b_2^1\right)^2}{\mu^2\alpha_v^2}+1}.$$

Из этого получаем квадратичное уравнение для неизвестных $\mu_1 = \mu^2$:

$$\mu_1^2(1-M^2)+\mu_1(2\Delta-M^2(\Delta_1+\Delta_2)) \quad +\Delta^2-M^2\Delta_1\Delta_2=0$$

где:

$$\Delta=\frac{\left(u_0-a_1^1\right)\left(u_0-b_1^1\right)}{\alpha_u^2}+\frac{\left(v_0-a_2^1\right)\left(v_0-b_2^1\right)}{\alpha_v^2} \qquad (2.6)$$

$$\Delta_1=\frac{\left(u_0-a_1^1\right)^2}{\alpha_u^2}+\frac{\left(v_0-a_2^1\right)^2}{\alpha_v^2}$$

$$\Delta_2=\frac{\left(u_0-b_1^1\right)^2}{\alpha_u^2}+\frac{\left(v_0-b_2^1\right)^2}{\alpha_v^2}.$$

Резюмируя, получаем следующую процедуру обновления калибровки для камеры с переменным фокусом.

1. Вычисляем соответствие для пар выбранных точек на двух изображениях фреймов N и N + 1, полученных камерой с переменным фокусным расстоянием.
2. Выпишем уравнение (2.6) для пар соответствующих точек.
3. Решим (2.6) для вещественного положительного корня и обновим скаляр α_u, α_v.

Способ 3

Опишем идею алгоритма, представленную Ричардом Хартли [4]. Допустим, у нас дано N перекрывающихся изображений $J_0, J_1, ..., J_N$, где N $\geq$ 2. Все изображения получены одной и той же камерой либо несколькими камерами с одинаковыми внутренними параметрами. Основные шаги алгоритма.

1. Установить точечное соответствие между изображениями.

2. Для каждого j = 1, . . . , N посчитаем 2D проективное преобразование P_j от J_0 к J_j.

3. Найдем верхнетреугольную матрицу K такую, что $K^{-1}P_jK=R_j$ является матрицей вращения для всех j > 0. Матрица K будет калибровочной матрицей для камер (или камеры), а R_j представляет собой ориентацию j-й камеры по отношению к 0-й.

4. Улучшим расчетную матрицу, используя итеративное приближение Levenberg—Marquardt [5, 6].

Остановимся более подробно на шаге 3, оставив остальные выкладки желающим изучить материал самостоятельно.

Итак, нам известны преобразования P_j для j = 1, . . . , N, и мы желаем найти калибровочную матрицу K, которая должна представлять собой верхнетреугольную матрицу, удовлетворяющую условию

$$K^{-1}P_jK = R_j, \tag{3.1}$$

где R_j является матрицей вращения для любого j. Матрица P должна быть сопряженная (conjugate) к матрице вращения, это значит, что P довольно специфична, как будет показано далее. Из соотношений (3.1) и $R = R^{-T}$ следует

$$R_j = K^T P_j^{-T} K^{-T}. \tag{3.2}$$

Из эквивалентности (3.1) и (3.2) следует

$$(KK^T)P_j^{-T} = P_j(KK^T). \tag{3.3}$$

Положим

$$C = KK^T = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}.$$

Уравнение (3.3) дает нам девять линейных уравнений с шестью неизвестными. Чтобы найти матрицу C, необходимо знать как минимум три различные матрицы P_j. В частности, для каждого изображения и соответствующей матрицы P_j для j = 1, . . . , N будет 9 уравнений. Эту переопределенную систему уравнений можно записать в виде Xa = 0, где X — матрица размерности 9Nx6 и вектор **a** состоит из неизвестных матрицы C. Это проблема минимизации и решается методом наименьших квадратов, на котором мы не будем заострять внимание.

Найдя матрицу C, матрицу K можно найти, используя факторизационный метод Choleski [7]. Решение для матрицы K возможно лишь при условии положительно определенной матрицы C.

Способ 4

Процесс калибровки камеры, предложенный в [8], проходит в два шага. На первом шаге вычисляется изменение эпиполярных линий. В статье рассматриваются два метода расчета эпиполяр, один основанный на проективных инвариантах, второй — на обобщении основной матрицы. На втором шаге используются так называемые уравнения Круппа, которые связывают эпиполярное преобразование с изображением. Уравнения Круппа могут быть решены после как минимум трех перемещений камеры. Затем, уже можно найти внутренние параметры камеры.

Способ 5

Идея, предложенная в статье Колесник Марины [9], представляет собой способ вычисления внутренних параметров камеры, используя угловую информацию для множества соответствующих точек так, как они видятся на плоскости изображения. Такую калибровку автор называет угловой. Для описываемого процесса необходим лазер для генерации образцов с известными угловыми характеристиками.

Способ 6

В статье Питера Стурма [10] рассматривается проблема самокалибровки движущейся камеры, внутренние параметры которой известны за исключением фокусного расстояния, которое может изменяться в течение времени наблюдения за сценой. Условия, под действием которых определяется значение фокусного расстояния для последовательности изображений, не являются необходимыми, но вторичны. Это зависит только от характера движения камеры. Расчет фокусного расстояния высчитывается на основе так называемых критических последовательностей движения. Рассмотрим последовательность изображений. Положим (R_i, t_i) — внешние параметры изображения i. Если евклидово преобразование вырождено для последовательности изображений, то мы говорим, что $\{(R_i, t_i) \mid i = 1, ..., n\}$ является критической последовательностью движения для преобразования Евклида.

Способ 7

Совместная статья [11] Пауло Мендонца и Роберто Сиполла знакомит с расширенной техникой самокалибровки Хартли [12], основанной на свойствах основной матрицы, позволяющей стабильно рассчитывать изменяющееся фокусное расстояние и принципиальные точки. Известно, что три сингулярных значения основной матрицы должны удовлетворять двум условиям: одно из значений должно быть нулевым, а другие два — идентичными. Основная же матрица получается из фундаментальной матрицы пу-

тем преобразования входящих в нее внутренних параметров пар камер, связанных двумя сценами. Таким образом, работа с основной матрицей сводится к работе с внутренними параметрами пар камер. Это позволяет искать пространство внутренних параметров камер в порядке минимизации функции стоимости. Этот подход показывает более простой, нежели другие, метод с аналогичной точностью в результате. Другое преимущество данной техники в том, что ей не требуется последовательное направление слабо откалиброванной матрицы камеры для всей последовательности изображений, т.е. множества согласующихся камер.

СПИСОК ЛИТЕРАТУРЫ

1. Zhang Z. A Flexible New Technique for Camera Calibration // Microsoft Research, One Microsoft Way, США, Рэдмонд. — 1998.
2. Новиков С.О., Лебедев О.А., Захаренко Г.В. Измерение и исследование трехмерных объектов в условиях неполной информации. — www.imvs.ru/imvs/itvs/03_1_2/novikov.pdf
3. Kolesnik M. Calibration Update Technique for a Zoom Lens // CAIP, 1999, — P.435–443.
4. Hartley R.I. Self-Calibration of Stationary Cameras // G.E. CRD, Schenectady, NY, 12301.
5. Levenberg K. A Method for the Solution of Certain Problems in Least Squares // Quart. Appl. Math. V.2, 1944, P.164–168.
6. Marquardt D. An Algorithm for Least-Squares Estimation of Nonlinear Parameters // SIAM J. Appl. Math. V.11, 1963, — P.431–441.
7. Atkinson K.E. An Introduction to Numerical Analysis, 2nd Edition // John Wiley and Sons, New York — 1989.
8. Luong Q.-T., Faugeras O., Maybank S.J. Camera Self-Calibration: Theory and Experiments // Internat. J. of Computer Vision, 1992. — P.123–151.
9. Kolesnik M. Using Angles for Internal Camera Calibration. — viswiz.gmd.de/~marina/pubs.html
10. Sturm P.F. Critical Motion Sequences for the Self-Calibration of Cameras and Stereo Systems with Variable Focal Length // Computational Vision Group, Department of Computer Science. — The University of Reading, 1992.
11. Mendonca P.R.S., Cipolla R. A Simple Technique for Self-Calibration // Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Colorado V.I, 1999 — P. 500–505.
12. Hartley R. Estimation of relative camera positions for uncalibrated cameras // Proc. 2nd European Conf. on Computer Vision. — Lect. Notes Comput. Sci. — 1992. — Vol. 588. — P. 579–587.

Л. С. Мельников*, И. В. Петренко

ПУТЕВЫЕ РАЗБИЕНИЯ В НЕОРИЕНТИРОВАННЫХ ГРАФАХ

Количество вершин в наиболее длинном простом пути графа G обозначается $\tau(G)$. Подмножество S множества вершин $V(G)$ называется P_{n+1}-*свободным множеством* в G, если $\tau(G[S]) \leq n$. P_{n+1}-свободное множество максимального порядка в графе G называется *максимальным* P_{n+1}-*свободным множеством* графа G.

Известна гипотеза о том, что для любых G-графа и $n < \frac{\tau(G)}{2}$ существует P_{n+1}-свободное множество M в G, такое что $\tau(G - M) \leq \tau(G) - n$.

В настоящей работе приводится доказательство этой гипотезы для $n \leq 8$.

ВВЕДЕНИЕ

В настоящей работе речь пойдет о конечных неориентированных графах.

Записью P_n мы будем обозначать путь порядка n, т. е. путь, содержащий n вершин. Соответственно, через C_n будет обозначаться цикл на n вершинах. Через $g(G)$ и $c(G)$ будем обозначать длину кратчайшего *(обхват)* и длиннейшего циклов в G соответственно. Количество вершин в наиболее длинном простом пути графа G обозначается $\tau(G)$.

Пусть S — некоторое подмножество множества вершин $V(G)$ графа G. Подграфом графа G, порожденным множеством S, называется граф $G[S]$, множество вершин которого $V(G[S]) = S$, а множество ребер $E(G[S]) = E(G) \cap S \times S$.

Открытой окрестностью вершины $v \in V(G)$ называется множество вершин вида $N(v) = \{u \in V(G) | (u, v) \in E(G)\}$. *Открытой окрестностью подмножества* A *множества вершин* $V(G)$ называется множество вершин вида $N(A) = \bigcup_{a \in A} N(a)$, а *замкнутой окрестностью* того же множества называется множество вершин вида $N[A] = N(A) \cup A$.

Пусть $G = (V, E)$ — конечный неориентированный простой граф. Для некоторой пары натуральных чисел (a, b) разбиение множества вершин $V(G)$ на два непересекающихся подмножества $\{A, B\}$ называется (a, b)-*разбиением* , если $\tau(G[A]) \leq a$, а $\tau(G[B]) \leq b$. Граф, для которого имеется такое (a, b)-разбиение, называется (a, b)-*разбиваемым.*

*omeln@math.nsc.ru

Работа автора поддержана Российским фондом фундаментальных исследований (коды проектов 05-01-000395 и 05-01-00816).

Если G является (a,b)-разбиваемым для любых a и b таких, что имеет место равенство $a+b=\tau(G)$, то такой граф называется *τ-разбиваемым.*

В работе [12] Г. Чартрандом, Д.П. Геллером и С. Хедетниеми было введено так называемое *k-хроматическое число графа,* $\chi_k(G)$, которое определяется как наименьшее количество множеств $\{V_1, V_2, \ldots, V_n\}$, разбивающих множество $V(G)$ таким образом, что имеет место $\tau(G[V_i]) \le k$ для каждого i. Там же для k-хроматического числа некоторого графа G была получена следующая верхняя оценка:

$$\chi_k(G) \le \lfloor \frac{\tau(G)-1-k}{2} \rfloor + 2,$$

потенциально улучшаемая до $\chi_k(G) \le \lceil \tau(G)/k \rceil$, в случае, если гипотеза о τ-разбиваемости произвольного графа G справедлива.

В дальнейшем k-раскраски изучались в работах [6, 10], а затем, уже в терминах путевых разбиений, в работах [8, 11], а также в последовавших за ними [9, 13].

В общем случае эта гипотеза может быть сформулирована в следующем виде.

Гипотеза 1. *Каждый граф является τ-разбиваемым.*

Гипотеза 1 тесно связана с открытой гипотезой Михока [20] о ядрах и работой Хайнала [16]. Эта гипотеза нашла отражение в диссертациях [17, 21]. Широкую известность гипотеза получила в 1997 г., будучи опубликована практически одновременно в работах [8, 11].

Вариант этой гипотезы для ориентированных графов сформулирован в [19], аналогичные постановки рассматривались в [18, 7]. В 2005 г. вышла работа [15], полностью посвященная версии гипотезы для ориентированных графов.

В связи с гипотезой 1 выдвигались различные метагипотезы, изучению которых посвящены, в частности, работы [13, 1, 2, 3, 4, 5].

Еще одна такая метагипотеза была предложена в работе [14]. Она использует понятие так называемого *P_n-свободного множества.* Подмножество S множества вершин $V(G)$ называется *P_{n+1}-свободным множеством* в G, если $\tau(G[S]) \le n$. P_{n+1}-свободное множество максимального порядка в графе G называется *максимальным P_{n+1}-свободным множеством* графа G.

Гипотеза 2. *Для любых G-графа и $n < \frac{\tau(G)}{2}$ существует P_{n+1}-свободное множество S в G, такое что $\tau(G - S) \leq \tau(G) - n$.*

Легко видеть, что если гипотеза 2 справедлива для некоторого графа G, то и гипотеза 1 для него справедлива.

В работе [14] доказаны некоторые результаты в пользу гипотезы 2. Так, например доказано, что для некоторого графа G и максимального P_{n+1}-свободного множества M в графе G имеет место неравенство

$$\tau(G - M) \leq \tau(G) - \frac{2}{3}n.$$

Кроме того, приведено доказательство, что гипотеза 2 верна для $n = 5$.

В настоящей работе доказано, что гипотеза 2 справедлива для $n \leq 8$.

1. ОБЩИЙ СЛУЧАЙ

В любом графе имеется P_{n+1}-свободное множество. Если M — максимальное P_{n+1}-свободное множество графа G, тогда для каждой вершины x из подграфа $G - M$ имеет место одна из перечисленных ниже альтернатив.

1. В M имеется путь P порядка n такой, что x смежна с концевой вершиной P.
2. В M имеется пара непересекающихся путей P и Q, у каждого из которых концевая вершина смежна с x, причем $|V(P) \cup V(Q)| = n$.

Предположим теперь, что некоторый путь L является самым длинным путем, лежащим в $G - M$, т. е. $\tau(G) = |L|$. Если для x выполняется первый вариант, то в графе G имеется путь PL, состоящий из вершин P и L, и тогда $\tau(G) \geq |L| + |P| \geq \tau(G - M) + n$. Это на самом деле означает, что гипотеза 2 справедлива для графа G.

Таким образом, далее нам потребуется рассмотреть лишь случай, когда для вершины x выполняется альтернатива 2 (рис. 1). Легко видеть в этом случае, что поскольку $|P| + |Q| = n$ и, следовательно, хотя бы один из путей не превосходит $\frac{n}{2}$, то имеет место неравенство $\tau(G - M) \leq \tau(G) - \frac{1}{2}n$.

Предположим, L — путь порядка $\tau(G - M)$, вершины x и y — концевые вершины этого пути. Предположим, что для обеих вершин x и y выполняется альтернатива 2. Тогда в M имеются пути P, Q, R, S, такие что имеют место соотношения $|P| + |Q| = n$ и $|R| + |S| = n$, причем вершина x смежна с концевыми вершинами P, Q, а вершина y смежна

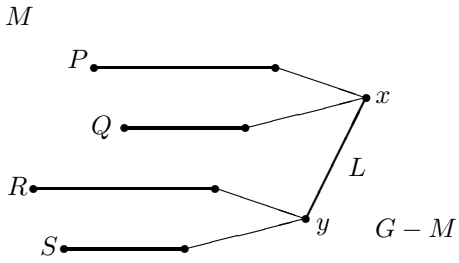


Рис. 1. Пути P, Q, R, S в M (альтернатива 2)

с концевыми вершинами R, S. Не ограничивая общности, можем предполагать, что $|P| \geq |R| \geq \lceil \frac{n}{2} \rceil \geq |S| \geq |Q|$.

В этих условиях верны следующие утверждения.

Утверждение 1. *Пусть пути P и R такие, как описано выше, пересекаются между собой таким образом, что имеет место $p_1 = r_1$. Тогда имеет место неравенство $\tau(G - M) \leq \tau(G) + n$.*

Доказательство. Предположим, P и R пересекаются, причем $p_1 = r_1$, тогда в G имеется путь $P - L - Q$, и требуемое равенство выполняется.

Утверждение 2. *Пусть пути P и R такие, как описано выше, пересекаются между собой таким образом, что имеет место $p_q = r_q$. Тогда имеет место неравенство $\tau(G - M) \leq \tau(G) + n$.*

Доказательство. Пусть P и R пересекаются, причем таким образом, что $p_1 = r_q$. При этом в M, очевидно, имеется путь длины $2q - 1$. Рассмотрим путь $P - R - L - q_1$, длина которого не менее, чем $|L| + n$, при условии, что путь R не пересекается с Q по вершине q_1.

Предположим, что Q пересекается с R по вершине q_1, причем $q_1 = r_i$ для некоторого i. При этом предположение о том, что $i < q-1$ приводит нас к противоречию с условием $\tau(M) \leq q$. Если же $i = q - 1$, то в G, очевидно, имеется путь $Q - P - L$, и тем самым утверждение доказано.

Утверждение 3. *Пусть пути P и R такие, как описано выше, пересекаются между собой таким образом, что имеет место $p_1 = r_q$. Тогда имеет место неравенство $\tau(G - M) \leq \tau(G) + n$.*

Доказательство. Пусть P и R пересекаются, причем таким образом, что $p_1 = r_q$. При этом в графе G имеется путь $R - L - S$, для которого требуемое неравенство, очевидно, выполняется, и тем самым утверждение доказано.

Эти результаты в силу своей тривиальности, естественно, носят лишь вспомогательный характер. Дальнейшее доказательство потребует от нас раздельного рассмотрения двух различных случаев, в зависимости от четности n.

2. СЛУЧАЙ $|P| \geq |R| > \frac{N}{2} > |S| \geq |Q|$

Теорема 4. *Пусть G — некоторый граф, M — максимальное P_{n+1}-свободное множество в нем. Пусть P, R, Q, S — пути в M, такие как описано выше. Тогда, если $P = n - q$ и $n \geq 2q + 1$ при $q \leq 3$, справедливо неравенство*

$$\tau(G - M) \leq \tau(G) - n\,.$$

Доказательство. Если $R \cap P = \emptyset$, то в графе G имеется путь R, L, P, причем в силу нашего предположения $|R| \geq \frac{n}{2}$, имеет место неравенство $\tau(G) \geq |P| + |L| + |R| = n - 1 + |L| + |R| \geq n + |L|$.

Предположим теперь, что P и R пересекаются, т. е. для некоторых вершин p_i и r_j имеет место равенство $p_i = r_j$. Обозначим через P_1 часть пути P, состоящую из вершин $p_1, \ldots, p_{i-1}$, а через P_2 — $P - P_1$. Аналогично введем обозначения R_1 и R_2 для частей пути R. Предположим, $j > q$, тогда в G имеется путь $P_2 - R_1 - L - P_1$ и, следовательно,

$$\tau(G) \geq |P_2| + |R_1| + |L| + |P_1| = |P| + j - 1 + |L| \geq |P| + |Q| + |L| = n + \tau(G - M).$$

Отметим, что аналогичные рассуждения справедливы и для S. Предположим, что $j \leq q$, но при этом S пересекает P, т.е. имеет место равенство $p_k = s_m$ для некоторых m и $k \neq i$ (в силу того, что пути R и S не пересекаются между собой). Если при этом $m > q$, то путь $P_2 - S_1 - L - P_1$, где S_1 определяется аналогично R_1 и обладает тем же свойством, что и рассмотренный выше.

Таким образом, нам необходимо изучить случаи, когда и S и R пересекаются с P и при этом имеют место $j \leq q$ и $m \leq q$. Очевидно, количество этих случаев зависит от значения параметра q. Рассмотрим различные случаи для значений, которые может принимать q. Далее мы будем предполагать, что имеет место неравенство $i > k$. Такое предположение мы делаем исключительно для краткости изложения. Оно не ограничивает общности наших рассуждений, так как все то, что будет доказано для S, является справедливым и для R, т. е. для случая $i < k$.

Случай 1. Пусть $q = 1$, т. е. $s_1 = p_k$, где $k \geq 1$. Тогда в G имеется путь $\mathcal{L} = S - P_1 - L - R$, откуда

$$\tau(G) \geq |S| + |P_1| + |L| + |R| \geq |L| + n \geq \tau(G - M) + n.$$

Случай 2. Пусть $q = 2$. Если предположить, что S пересекает P по вершине s_1 или по вершине s_2, но при этом выполняется условие $k \geq 2$, то мы оказываемся в условиях случая 1, и тогда путь $\mathcal{L}$ является искомым.

Предположим, $s_2 = p_1$. Тогда в графе G имеется путь $\mathcal{L} = P - s_1 - L - q_1$ и требуемое неравенство для $\tau(G)$ выполняется. Однако, может оказаться, что пути S и Q пересекаются, а именно, что имеет место равенство $q_1 = s_1$. Однако в этом случае в качестве пути $\mathcal{L}$ может быть рассмотрен путь $s_{|S|}, \ldots s_2 - q_1 - L - R$. Легко видеть, что $|\mathcal{L}| = n + |L|$, а это означает, что для случая 2 теорема верна.

Случай 3. Пусть $q = 3$ и S пересекает P, т. е. имеет место равенство $p_k = s_m$. Как и в предыдущем случае, если предположить, что $m \leq k$, то в графе G имеется путь $\mathcal{L} = S_2 - P_1 - |L| - R$, и все доказано. Если $m > k$, то в качестве $\mathcal{L}$ может быть рассмотрен путь $P_2 - S_1 - L - Q$, однако при этом необходимо учесть, что S и Q могут пересекаться.

Предположим, $p_1 = s_2$ и $q_1 = s_1$. Это не приводит нас к противоречию, так как в этом случае в G имеется путь $S - L - R$, подробно рассмотренный в предыдущем случае.

Допустим, $p_1 = s_3$, в этом случае, очевидно, $q_1 \neq s_1$, так как это противоречило бы предположению о том, что $\tau(M) \leq n$. Если предположить, что $q_1 = s_2$, то в графе G имеется путь $P - L - s_1 - q_1 - q_2$, который в этом случае может быть выбран в качестве $\mathcal{L}$.

Наконец, предположим, что $p_2 = s_3$. Тогда S может пересекаться с Q по вершинам q_1, q_2. Если $s_1 = q_1$, то в G имеется путь $P - L - Q$ порядка $|L| + n$, а если $s_1 = q_2$, то в G имеется путь $S - q_1 - L - R$, очевидно, что его порядок не менее, чем $|L| + n$. Пусть $s_2 = q_1$, тогда в G имеется путь $\mathcal{L} = P - L - s_1 - Q$, наконец, если $s_2 = q_2$, то в G имеется путь $\mathcal{L} = P - L - s_1 - q_2 - q_1$. Отметим, что в обоих последних случаях $|\mathcal{L}| \geq |L| + n$. Тем самым, теорема доказана полностью.

Отметим, что из доказанного утверждения следует и результат, полученный в [14]. Непосредственным следствием из теоремы 4 является следующая теорема.

Теорема 5. *Пусть G — некоторый граф, M — максимальное P_8-свободное множество. Тогда имеет место неравенство:*

$$\tau(G - M) \leq \tau(G) + 7.$$

3. СЛУЧАЙ $|P| = |R| = |Q| = |S| \leq 4$

В настоящем разделе мы рассмотрим подробно случай $|P| = |R| = |S| = |Q| = q$ при $n = 2q$. Этот случай является самым сложным с точки зрения построения доказательства.

Лемма 6. *Пусть G — граф, M — максимальное P_7-свободное множество. Пусть $|P| = |R| = |Q| = |S| = 3$, тогда*

$$\tau(G - M) \leq \tau(G) - 6.$$

Доказательство. Ясно, что если никакие два из путей P, Q, R, S не пересекаются, то лемма доказана. Аналогично, легко понять, что, если пути R и S не пересекают оба путь P, то также доказательство вполне очевидно. Предположим, что и R и S пересекают путь P.

В силу утверждений 1, 2 и 3, для доказательства этой леммы нам достаточно рассмотреть лишь случаи, когда пути P и R пересекаются и при этом имеет место одно из равенств: $p_2 = r_2$, $p_2 = r_3$ или $p_1 = r_2$. Рассмотрим поочередно эти случаи. Они изображены на рис. 2, 3 и 4. Для случаев 1 и 3 отметим сразу, что, если S пересекает P по вершине p_3, то, очевидно, в G имеется путь $S - L - p_1 - p_2 - r_3$, и тем самым лемма доказана, так как $\tau(S - L - p_1 - p_2 - r_3) = |L| + 6$. Таким образом, нам достаточно рассмотреть подслучаи, когда S пересекает P по вершине p_1.

Отметим также, что в случае, когда S пересекает P по вершине p_1 и при этом имеет место одно из равенств $p_1 = s_1$ или $p_1 = s_3$, мы оказываемся в условиях утверждений 1 и 3 соответственно. Это еще более упрощает нашу задачу.

Случай $p_2 = r_2$. Предположим, что при этом $s_2 = p_1$. В этом случае в G имеется путь $P - s_1 - L - q_1 - q_2$, который удовлетворяет нашим требованиям. Необходимо проверить, что будет, если при этом окажется, что Q пересекает S по вершине s_1. Поскольку случай $q_1 = s_1$ тривиален в силу утверждения 1, достаточно рассмотреть лишь только случай, когда $s_1 = q_2$. Легко видеть, что в этом случае в G имеется путь $P - s_1 - q_1 - L - r_1$ и при этом требуемое неравенство имеет место.

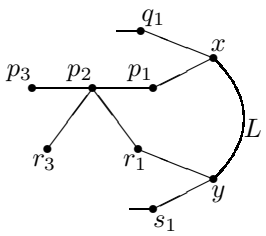


Рис. 2. Пути P и R пересекаются, причем $p_2 = r_2$

Случай $p_2 = r_3$. Этот случай в общем аналогичен предыдущему, даже немного проще. Итак, предположим, что $p_2 = r_3$ и при этом имеет место равенство $p_1 = s_2$. Тогда в G имеется путь $s_1 - p_1 - L - r_1 - r_2 - p_2 - p_3$, для которого, очевидно, имеет место требуемое неравенство. Таким образом, в этом случае лемма также верна.

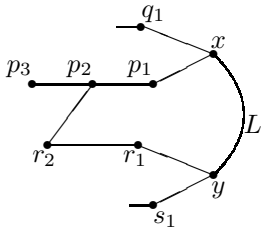


Рис. 3. Пути P и R пересекаются, причем $p_2 = r_3$

Случай $p_1 = r_2$. В этом случае можно было бы сделать такой же перебор, как и в двух предыдущих. Однако заметим, что если P и R пересекаются по p_1, то S пересекает P по p_2 или по p_3, а значит, имеет место один из случаев, разобранных выше, либо мы оказываемся в условиях одного из утверждений 1, 2 или 3 для путей P и S.

Таким образом, лемма доказана.

Подобный перебор, если бы его пришлось осуществлять для доказательства следующей леммы, занял бы гораздо больше места и сил. Поэтому здесь мы прибегли к иной технике доказательства.

Лемма 7. *Пусть G — граф, M — максимальное P_9-свободное множество. Пусть $|P| = |R| = |Q| = |S| = 4$, тогда $\tau(G - M) \leq \tau(G) - 8$.*

Доказательство. Отметим, что в силу того, что имеет место $|P| = |Q| = |R| = |S|$, можно сделать следующий вывод. Если какие-то два пути не пересекаются между собой, то лемма верна, поскольку, очевидно,

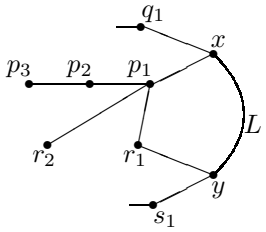


Рис. 4. К доказательству леммы 6.
Пути P и R пересекаются, причем $p_1 = r_2$

что в этом случае в графе G имеется путь длины $n+8$. Действительно, предположим, что P не пересекает ни R, ни S. Тогда в G имеется путь $P-L-S$ или $P-L-R$, который обеспечивает выполнение требуемого неравенства.

Предположим поэтому, что все 4 пути P, Q, R, S пересекаются между собой. Однако, поскольку нам известно, что P и Q, а также R и S между собой пересекаться не могут, очевидно, имеется 4 различных вершины v, w, u, z в M, такие что $w = P \cap R, v = P \cap S, u = Q \cap R, z = Q \cap S$. Для определенности (никак не ограничивая этим общности) предположим, что v — ближайшая к x и y вершина.

Ясно, что эти 4 вершины лежат на некотором цикле C, длина которого не превосходит 8, поскольку $\tau(M) \leq 8$.

Предположим, что $|V(C)| = 8$. Поскольку $v \in V(C)$, в G имеется путь, состоящий из вершин $V(C)$, вершин пути из v в x и далее из вершин пути L. Требуемое неравенство при этом выполняется.

Прежде, чем перейти к следующему шагу доказательства, заметим, что в силу утверждения 1 можем предположить, что хотя бы одна вершина из $N(x) \cap M$ имеет степень 2, т.е. не является одной из вершин v, w, u, z.

Допустим теперь, что имеет место неравенство $5 \leq |V(C)| \leq 7$. В этом случае один из путей P, Q, R, S пересекается с циклом C в точности по одному ребру. Предположим, что это путь P и (p_i, p_{i+1}) и есть это самое ребро. В таком случае, в графе G имеется путь длины $|P| + |V(C)| - 2 + |L| + 1 = |V(C)| + 3 + |L| \geq |L| + 8$. Следовательно, в этом случае лемма также верна.

Рассмотрим теперь случай $|V(C)| = 4$. Легко видеть, что в этом случае путь требуемой длины также может быть легко построен. Доказательство может быть получено исчерпывающим перебором всех воз-

можных значений для вершин v, w, u, z.

Лемма доказана.

Из леммы 6, 7 и теоремы 4 легко следует следующая теорема.

Теорема 8. *Пусть G — некоторый граф, M — максимальное P_{n+1}-свободное множество, где $n \leq 8$. Тогда имеет место неравенство:*

$$\tau(G - M) \leq \tau(G) - n.$$

ЗАКЛЮЧЕНИЕ

Полученные здесь результаты могут интерпретироваться следующим образом. Как уже говорилось, справедливость гипотезы 2 для всех $k \leq n$, где k — некоторое целое число, для произвольного графа G немедленно влечет справедливость гипотезы о τ–разбиваемости для $\tau \leq 2n - 1$ для произвольного графа G. Таким образом, верна следующая теорема.

Теорема 9. *Любой граф G является 15 – разбиваемым.*

Этот результат не является принципиально новым, он уже был получен прежде авторами в работе [3] как следствие теоремы о существовании P_8-ядра, что позволяет использовать его как своего рода проверку результатов работы [3].

Однако, следует заметить, что использование данной техники и понятия P_n-свободного множества значительно упростило доказательства и снизило их объем.

Предполагаемая же авторами возможность обобщения теоремы 4 на большие значения q (а в перспективе — на произвольные значения q) при соответствующих значениях n демонстрирует далеко не исчерпанные в данной работе возможности использованной методики.

СПИСОК ЛИТЕРАТУРЫ

1. **Мельников Л. С., Петренко И. В.** Существование путевых ядер и разбиений в неориентированных графах // Тез. докладов XIV Междунар. конф., посвященной 80-летию С.В.Яблонского “Проблемы теоретической кибернетики” (Пенза, 23–28 мая 2005 г.) / Под ред. О.Б.Лупанова. — М.: Изд-во механико–математического факультета МГУ, 2005. — С. 95.
2. **Мельников Л. С., Петренко И. В.** Путевые ядра и разбиения в графах с малыми длинами циклов // Методы и инструменты конструирования и оптимизации программ. — Новосибирск, 2005. — С. 145–160.

3. **Мельников Л. С., Петренко И. В.** О путевых ядрах и разбиениях в неориентированных графах // Дискретный анализ и исследование операций. — 2002. — Т. 9, Сер. 1, № 2. — С. 21–35.
4. **Мельников Л. С., Петренко И. В.** Путевые ядра и длины циклов в неориентированных графах // Современные проблемы конструирования программ. — Новосибирск, 2002. - С. 222–231.
5. **Aldred R. E. L., Thomassen C.** Graphs with not all possible path-kernel // Discrete Math. — 2004. — Vol. 285, N 1–3. — P. 297–300.
6. **Benade G., Broere I., Jonck B., Frick M.** Uniquely $(m, k)^\tau$-colorable graphs and $k - \tau$-saturated graphs // Discrete Math. — 1996. — Vol. 162, N 1–3. — P. 13–22.
7. **Berge C., Duchet P.** Recent problems and results about kernels in directed graphs // Discrete Math. — 1990. — Vol. 86, N 1–3. — P. 27–31.
8. **Borowiecki M., Broere I., Frick M., Mihók P., Semanišin G.** A survey of hereditary properties of graphs // Discussiones Mathematicae Graph Theory. — 1997. — Vol. 17, N 1. — P. 5–50.
9. **Broere I., Dorfling M., Dunbar J. E., Frick M.** A path(ological) partition problem.// Discussiones Mathematicae Graph Theory. — 1998. — Vol. 18, N 1. — P. 113–125.
10. **Broere I., Frick M.** On the order of uniquely k, m-colorable graphs // Discrete Math. — 1990. — Vol. 82, N 3. — P. 225–232.
11. **Broere I., Hajnal P., Mihók P.** Partition problems and kernels of graphs// Discussiones Mathematicae Graph Theory. — 1997. — Vol. 17, N 2. — P. 311–313.
12. **Chartrand G., Geller D. P., Hedetniemi S. T.**, A generalization of the chromatic number // Proc. Camb. Phil. Soc. — 1968. — Vol. 64, N 2. — P. 265–271.
13. **Dunbar J. E., Frick M.** Path kernels and partitions // J. Combin. Math. Combin. Comput. 1999. V. 31. pp. 137–149.
14. **Dunbar J. E., Frick M., Bullock F.** Path partitions and P_n–free sets // Discrete Math. — 2004. — Vol. 289, N 1–3. — P. 145–155.
15. **Frick M., van Aardt S., Dlamini G., Dunbar J., Oellermann O.** The directed path partition conjecture // Discussiones Mathematicae Graph Theory. — 2005. — Vol. 25, N 3. — P. 331–343.
16. **Hajnal P.** Partitions of graphs with condition on the connectivity and minimum degree// Combinatorica. — 1984. — Vol. 3, N 1. — P. 95–99.
17. **Hajnal P.** Graph partitions. — Ph. D. Thes. Szeged Attila Jósef University, 1984.
18. **Kapoor S. F., Kronk H. V., Lick D. R.** On deturs in graphs // Canad. Math. Bull. — 1966. — Vol. 11, N 2. — P. 195–201.
19. **Laborde J. M., Payan C., Xuong N. H.** Independent sets and longest directed paths in digraphs // Graphs and other combinatorial topics (Prague, 1982). — Leipzig: Teubner, 1983. — P. 173–177.
20. **Mihók P.** Problem 4 // Graphs, hypergraphs and mathroids. — Zielon Góra:Higher College Engrg., 1985. — P. 86.
21. **Vronka J.** Vertex sets of graphs with prescribed properties. — Ph. D. Thes. Košice, Safarik University, 1986.

Г. П. Несговорова

СОВРЕМЕННЫЕ ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ И ЦИФРОВЫЕ ТЕХНОЛОГИИ В СОХРАНЕНИИ КУЛЬТУРНОГО И НАУЧНОГО НАСЛЕДИЯ И РАЗВИТИИ МУЗЕЙНОГО ДЕЛА

ВВЕДЕНИЕ

Использование информационных технологий в различных сферах культуры — это достаточно новая область деятельности, начавшая развиваться в 80-х годах прошлого века. Одной из таких сфер культуры является музейное дело (как часть общего культурного пространства), развитие которого в мире насчитывает несколько сот лет, начиная с XV века. И только с конца XX века началось активное внедрение информационных технологий в культуру вообще и в музейную деятельность в том числе. В этой деятельности участвуют люди разных профессий: от инженеров-электронщиков и программистов до археологов и искусствоведов (включая администрацию музея и смотрителей музейных залов). Это сообщество специалистов, занимающихся поддержанием и развитием музейного дела и процессом информатизации культуры в целом и музеев в частности.

В российских высших учебных заведениях даже появилась новая специальность — «Музейная информатика». Под таким названием подразумевается применение информационных технологий в музейном деле. Специалисты этого профиля должны сочетать в себе знание музейного дела и овладение навыками применения современных информационных технологий для развития и усовершенствования организации музейной деятельности.

На современном этапе модернизация в культурной сфере связана скорее с организацией работы, чем с изменением характера используемых ресурсов. Компьютер, пришедший на смену картотечному ящику и пишущей машинке, является лишь удобным инструментом и не более того. Он не поменял природы гуманитарных знаний. Происходящие у нас на глазах перемены в области культуры связаны не с появлением баз данных и персональных компьютеров, а с появлением новой **среды коммуникации**. Эта среда диктует особые формы взаимоотношений, называемых **сетевыми**. Описывать сетевые отношения начали только с появлением Интернета, хотя сети как системы человеческого взаимодействия были известны задолго до компьютерной эры.

Цель данной статьи — провести обзор применения информационно-коммуникационных технологий как в реально существующих музеях и их сайтах, так и в развитии достаточного нового явления в сети Интернет — виртуальных музеев. Как реальные, так и виртуальные музеи служат одной цели — сохранению культурного и научного наследия.

1. ИНФОРМАЦИОННЫЙ МЕНЕДЖМЕНТ В КУЛЬТУРЕ И МУЗЕЙНОМ ДЕЛЕ

Зачем нужны информационные ресурсы? Ответ на этот вопрос дает новая дисциплина — «**информационный менеджмент**» (ИМ), это и **управление информацией** и **управление с помощью информации** в любых областях человеческой деятельности и, в частности, в области культуры.

Информационный менеджмент — это

1) управление информацией, т.е. информационными потоками и информационными ресурсами (в нашем случае — музейными), другими словами: технология работы с информацией в определенной предметной области (области культуры);

2) управление с помощью информации, т.е. управленческая технология, менеджмент в собственном смысле этого слова.

Сегодня есть все основания рассматривать информационные технологии как неотъемлемый компонент управленческих технологий во многих сферах человеческой деятельности, в том числе, и в культуре. Среди элементов новых управленческих технологий, проникающих в культурную сферу и музейное дело, можно выделить следующие:

1) средства оперативной коммуникации (электронная почта, списки рассылок, новостные разделы музейных сайтов и др.);

2) распределенные ресурсы и средства доступа к ним (базы данных, порталы, терминалы компьютерных сетей и пр.);

3) средства координации деятельности (электронные доски объявлений, форумы, электронные опросы и т.д.);

4) формы обратной связи и организации сотрудничества (гостевые книги, форумы, телеконференции);

5) средства «производства» (инструментарий поиска ресурсов и партнеров, стандартные и специализированные программные средства).

В настоящее время в виртуальную среду все чаще перемещаются места делового общения, обмена идеями и взаимного консультирования (веб-клубы, интернет-кафе), средства совместного проектирования и продвиже-

ния проектов (веб-лаборатории, обмен баннерами). Возникают целые виртуальные поселения с проблемно-ориентированной социальной структурой и специализированными вспомогательными структурами (Geocity[1], Fortunecity[2] и др.). Информационные технологии становятся неотъемлемой частью культуры, и несомненно, что информационный менеджмент найдет свое применение и в такой отрасли культуры, как музейное дело.

2. РАСПРОСТРАНЕННОСТЬ ВИРТУАЛЬНЫХ МУЗЕЕВ В ИНТЕРНЕТЕ И ИХ ЗНАЧЕНИЕ ДЛЯ КУЛЬТУРНОЙ ЖИЗНИ РОССИИ

Понятие виртуальный музей вошло в нашу жизнь начиная с 90-х годов XX века. В настоящее время виртуальными музеями называют как представительства реальных музеев — музейные сайты, так и собственно виртуальные музеи [1]. При кажущейся аналогии с реальным музеем, виртуальный музей — это **новая реальность**, которая выходит за рамки традиционного представления о реальном музее с его постоянной экспозицией и временными выставками. Экспозиция виртуального музея постоянна лишь в своем развитии, время работы его выставок может длиться годами, и их количество связано лишь с новыми идеями, а ограничено только тематикой данного музея, да и то которую при желании можно расширить или изменить. Экспонаты реального музея со временем приходят в негодность, а оцифрованные коллекции виртуального музея можно хранить бесконечно долго.

Таким образом, виртуальный музей — это не просто сайт реально существующего музея, а созданный в сети оригинальный сайт, не имеющий своего аналога среди реально существующих музеев и отражающий любую тематику, если по ней находятся экспонаты и материалы, имеющие свое физическое или идейное воплощение в реальном мире. В настоящее время в сети Интернет с каждым годом появляется все больше виртуальных музеев различных тематик, информационную ценность которых трудно переоценить, поскольку посетить виртуальный музей (правда, при наличии компьютера и доступа к сети Интернет) значительно проще и быстрее, чем

[1] Geocity — это виртуальный город, являющийся громадным собранием различных по тематике веб-страниц, которые может бесплатно завести на данном сервере любой пользователь сети. При объединении самых разнообразных страниц использована метафора города, в котором отдельные веб-страницы соответствуют домам, сгруппированным в кварталы и районы города.

[2] Fortunecity — ведущий веб-хостинговый оператор (с 1997 г.). Предлагает бесплатный хостинг и осуществляет услуги в области электронной коммерции.

собраться посетить реальный музей. К тому же реальные музеи, как правило, отражают ограниченную традиционную тематику, а виртуальные отличаются разнообразием и необычностью своих экспозиций.

Примером виртуального музея является разрабатываемый в лаборатории конструирования и оптимизации программ ИСИ СО РАН виртуальный музей истории информатики в Сибири [2]. Надо отметить, что реального музея с такой тематикой не существует. Особенностью этого виртуального музея является его открытость. Открытость виртуального музея по определению означает его доступность любым пользователям из любой точки земного шара в любое время. При этом посетитель виртуального музея может не только ознакомиться с экспонатами виртуальных экспозиций, но и сам принять участие в его развитии, став зарегистрированным пользователем и получив тем самым возможность пополнять новыми экспонатами экспозиции музея, вносить свои поправки. Таким образом, контент нашего музея будет создаваться множеством людей, которые смогут помещать в Сибирский виртуальный музей истории информатики новые тексты, фотографии, видео- и аудиофайлы и любые другие документы, соответствующие заявленной тематике виртуального музея.

Такая деятельность вписывается в рамки международного проекта электронных музеев «Музей без границ». Поэтому наш виртуальный музей является частицей открытых информационно-коммуникационных обществ и обществ знаний в сети Интернет. Мы не ограничиваемся распространением знаний, собранных в виртуальном музее истории информатики в Сибири, сугубо региональными или государственными рамками, а делаем его открытым для всех желающих из любых стран мира через международную сеть, которой является Интернет.

3. ИСПОЛЬЗОВАНИЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В РЕАЛЬНЫХ МУЗЕЯХ

Информационные технологии начали внедряться в музейное дело с конца 80-х годов прошлого столетия. Сначала это были автоматизированные информационные системы (АИС), представляющие собой базы данных с элементами поиска в них, пришедшие на смену картотеке и ручному поиску нужной карточки в ней. Затем стали создаваться музейные сайты, отражающие экспозиции музея, рассказывающие о его деятельности и играющие роль своего рода рекламы с анонсированием новых экспозиций, будущих выставок, указанием времени работы музея и другой полезной инфор-

мации, направленной на привлечение посетителей в музей. С этой же целью — привлечение посетителей — в музейном деле недавно стали применяться новые технологии с использованием карманных персональных компьютеров (КПК). Весной 2005 г. в Политехническом музее г.Москвы стартовал новый проект — «Музейное ориентирование» — с использованием КПК. Это ролевой квест, который называется «Quazitec или секрет гениальности». Ролевая игра «Музейное ориентирование» разработана московскими десятиклассниками при поддержке сотрудников Политехнического музея в экспозиции «Автоматика и робототехника». Она является частью проекта, который предполагает коллективное использование общественных сетевых сервисов и разнообразие игровой и учебной деятельности, связанной с мобильными устройствами.

Ходить по музею просто так, без экскурсовода, — достаточно скучное дело, а поиграть в музее, да еще и с пользой, — занятие увлекательное и познавательное. Суть этой ролевой компьютерной игры в следующем. Игроки должны быть оснащены КПК и цифровыми аппаратами. На миникомпьютерах указаны все задания и вопросы по тематике выбранной экспозиции, которые надо выполнить, и обмануть при этом КПК невозможно: пока не пройдешь один уровень, до другого — компьютер не допустит. Командам надо пройти по пять уровней в пяти залах музея. Чтобы выполнить задания, надо внимательно осмотреть экспозицию и найти ответы на викторинные вопросы. Кроме тестовых заданий, предлагались и практические, например, расшифровать царскую телеграмму, закодированную азбукой Шиллинга, и т.д. Каждый неверный шаг отслеживался, а КПК считал ошибки. Побеждала команда, ответившая на большее число вопросов за минимальное время и при наименьшем количестве ошибок.

Описанная выше игра — это еще один пример того, как компьютер и информационные технологии применяются в музейном деле. Играя в эту игру, посетители, в первую очередь школьники, узнают много нового, чего бы они не узнали при простом посещении музея. Кроме того, учащиеся сами участвуют в разработках навигационного программного обеспечения для перемещения по залам музея и составлении заданий, что несомненно помогает им лучше узнать и запомнить содержание «игровой» экспозиции. Далее предполагается развивать это направление и устраивать такие игры в залах других музеев, например, Государственного исторического музея.

4. ВЗАИМОДЕЙСТВИЕ РЕАЛЬНЫХ И ВИРТУАЛЬНЫХ МУЗЕЕВ

Помимо внедрения автоматизированных информационных систем и создания музейных сайтов, в жизни реальных музеев в настоящее время существует тенденция к созданию объединенных музейных ресурсов, что само по себе станет побудительным мотивом как к использованию современных информационных технологий в музейном деле, так и к ускорению роста объемов информационных музейных ресурсов. Это одинаково касается как реальных, так и виртуальных музеев. При этом виртуальные музеи относятся скорее к номинации «Сетевое искусство», а реальные — к номинации «Музеи». Создатели сайтов Эрмитажа, Государственного музея изобразительных искусств им. А.С. Пушкина, Русского музея, Кунсткамеры и других не сами придумали эти музеи, а лишь отразили их в электронном виде с применением информационных технологий. Виртуальный же музей создается авторами в уникальном, не существующем в реальности варианте. В [1] сделана попытка дать определение, что такое виртуальный музей и чем он отличается от сайтов реально существующих музеев. Сайты реальных музеев и собственно виртуальные музеи, помещенные в сети Интернет, обобщенно будем называть **электронными музеями**.

Цели **электронных музеев** состоят в следующем:

1) создание механизмов свободного и эффективного доступа граждан к информации о культурном, научном и историческом наследии, предоставление широкого спектра информационных услуг на базе современных телекоммуникационных технологий;

2) содействие развитию образования, сохранению памятников истории и культуры;

3) вовлечение разного рода организаций и отдельных граждан в работу по сохранению, исследованию и популяризации культурного наследия страны, используя современные информационные технологии;

4) содействие формированию единого мирового культурно-информационного пространства.

5. ПРОЕКТ КОМИССИИ ЕВРОПЕЙСКОГО СООБЩЕСТВА MINERVA PLUS ЕГО РЕАЛИЗАЦИЯ В РОССИИ

В этом разделе остановимся на обзоре проекта Комиссии Европейского сообщества MINERVA PLUS[3], посвященного оцифровке[4] культурного и

[3] MINERVA — Ministerial NEtwoRk for VAlorising Activities in digitisation

научного наследия и являющегося шагом к сохранению национальных информационных ресурсов. Современные цифровые технологии приходят сейчас на смену аналоговым во всем мире. Аналоговое видео, например, имеет свойство стареть. Каждые 5 лет качество видеозаписи падает примерно в 2 раза. А еще видеокассеты занимают много места и пылятся. Решение проблемы — оцифровка видео и запись на DVD, так как в любой момент и без потери качества можно сделать копии. Оцифрованное видео может храниться сколь угодно долго, оно экологически безвредно.

Немного из истории развития проекта MINERVA. В 2000 г. Европейским Союзом (ЕС) была принята 10-летняя рабочая стратегия экономического, социального, экологического и культурного обновления. В отношении культуры была поставлена цель — создание Европейского культурного пространства (European Cultural Area) посредством выдвижения на передний план единого европейского культурного наследия и стимулирования плодотворного международного сотрудничества в этой области.

Появление и развитие в Европе «цифровой культуры» происходит одновременно с трансформацией традиционных культурно-просветительских учреждений, в том числе и музеев, в общеевропейское культурное пространство. В этом контексте выработан широкий спектр различных программ и инициатив, среди них программа IST (Information Society Technologies — Технологии информационного общества), план мероприятий по программе «Электронная Европа» (e-Europe), программа "e-Content" по созданию европейских информационных ресурсов по культуре и науке и обеспечению доступа к ним. В том, что касается развития сферы культуры, Евросоюзом ведется активная работа с широким спектром задач: от поддержки традиционных видов деятельности в этой сфере до исследований в области «цифровой культуры». Совет Министров культуры Европы все отчетливее осознает проблемы и перспективы, связанные с созданием «цифровой культуры», и все чаще поддерживает начинания в этой области.

Оцифровка национального культурного и научного наследия является одной из первоочередных задач правительств европейских стран, и принято решение разработать механизм координации национальных программ представления информации по культуре и науке в цифровом виде для создания

[4] Оцифровка (цифрование) — технология и процесс переноса разного рода информации на компьютер, иными словами — перевод аналоговых данных в цифровую форму, доступную для обработки в цифровой машинной среде и для хранения на машинно-читаемых устройствах с помощью дигитайзера — устройства для преобразования готовых (бумажных, видео и пр.) изображений в цифровую форму. Производится оцифровка как видео-, так и аудиоматериалов.

единой общеевропейской платформы доступа к этим информационным ресурсам.

Проект MINERVA и стал конкретным механизмом реализации этих планов, он выполнялся в течение 2002-2003 гг. Далее решено было продолжить эту деятельность в рамках проекта MINERVA PLUS с привлечением стран, вступающих в Евросоюз в 2004 г., а также Израиля и России. Задачей этого проекта является координация национальных программ в областях науки и культуры, а деятельность по проекту базируется на принципе встроенности в национальную деятельность по оцифровке.

Министерство культуры Российской Федерации стало полноправным участником общеевропейского проекта MINERVA PLUS в 2003 г. В России развитие информационного общества, обеспечение доступа граждан к информации, интеграция в общемировое информационное пространство являются приоритетными направлениями. В этом отношении информационные ресурсы по культуре и науке, примером которых может являться наш виртуальный музей истории информатики в Сибири, имеют ключевое значение, поэтому участие в проекте MINERVA PLUS служит катализатором процессов оцифровки культурного и научного наследия России и разработки механизмов, обеспечивающих всеобщий открытый доступ к информации. Таким образом, этот проект является тематической сетью Министерства культуры РФ для обсуждения, корреляции и гармонизации деятельности по оцифровке культурного и научного наследия в нашей стране, для создания согласованной общеевропейской платформы, разработки и популяризации рекомендаций и методик по оцифровке, метаданным, долговременному доступу и сохранению цифрового наследия.

Участие России в проекте MINERVA PLUS дает возможность выработать предложения для формулирования российской государственной политики по оцифровке культурного и научного наследия, согласованной с общеевропейской политикой в этом вопросе; разработать план действий по оцифровке, реализация которого приведет к созданию в России высококачественных информационных ресурсов по культуре и науке и обеспечению широкого доступа к ним; будет способствовать включению российских ресурсов в общеевропейское информационное культурное пространство, облегчит предоставление доступа к общеевропейским информационным ресурсам гражданам России. Цифровые музейные ресурсы — это наполнение как сайтов реальных музеев, так и виртуальных, которые можно использовать и в учебных коммуникациях.

ЗАКЛЮЧЕНИЕ

В заключение хочется отметить, что развитие реальных музеев, связанное с применением информационно-коммуникационных и цифровых технологий (в частности, создание музеями собственных сайтов в сети Интернет), а также развивающиеся и вновь появляющиеся виртуальные музеи различной тематики служат, по большому счету, развитию культуры и увеличению знаний у большого количества людей. Все описанные в статье технологии работы с реальными и виртуальными музеями, новыми направлениями, такими как музейное ориентирование, оцифровка национального культурного и научного наследия, подтверждают то, что музеи, существующие столетия, переживают сейчас новый этап своего развития. Это развитие направлено на дальнейшее увеличение интереса к музеям и способствует просвещению не только школьников и студентов, но и широких слоев общества, а также сохранению культурного и научного наследия в мировом масштабе.

СПИСОК ЛИТЕРАТУРЫ

1. Несговорова Г.П. Обзор виртуальных музеев в сети Интернет // Методы и инструменты конструирования и оптимизации программ. — Новосибирск, 2005. — С. 161–172.
2. Касьянов В.Н., Несговорова Г.П., Волянская Т.А. Виртуальный музей истории информатики в Сибири // Современные проблемы конструирования программ. — Новосибирск, 2002. — С. 169–181.
3. http://www.future.ru
4. http://svr.museum.ru
5. http://www.adit.ru

Р. А.Осмонов, Д. Н. Штокало

ПРЕОБРАЗОВАНИЯ ЦИКЛОВ, ОСНОВАННЫЕ НА НЕСИНГУЛЯРНЫХ МАТРИЦАХ

ВВЕДЕНИЕ

В статье предложен способ использования несингулярных матриц вместо унимодулярных для преобразований гнезд циклов. Несингулярные матрицы, детерминант которых не ограничен значением ±1, включают унимодулярные матрицы как частный случай и дают возможность включить новое преобразование, называемое в системе несингулярного преобразования масштабированием цикла. Полученный результат показывает, что проще работать с несингулярными матрицами, чем с унимодулярными.

Главное преимущество унимодулярного преобразования состоит в том, что оно дает подход к решению так называемой «упорядоченности преобразований» для многих задач, в которых не существует явного порядка выполнения преобразований. Можно получить унимодулярную матрицу, из которой может быть просто определен нужный порядок преобразований цикла [1, 2].

Проще сгенерировать несингулярную матрицу, чем унимодулярную, так как во втором случае существует несколько ограничений, которые должны быть удовлетворены. В этой статье приводится процедура, которая генерирует несингулярную матрицу, данную изначально с первыми несколькими строками. Эта процедура нетривиальная, так как необходимо обеспечить, чтобы результирующая матрица сохраняла зависимости в гнезде цикла. В общем случае генерация преобразованного гнезда цикла более проста при использовании несингулярных матриц, чем при использовании унимодулярных, и это главная идея статьи.

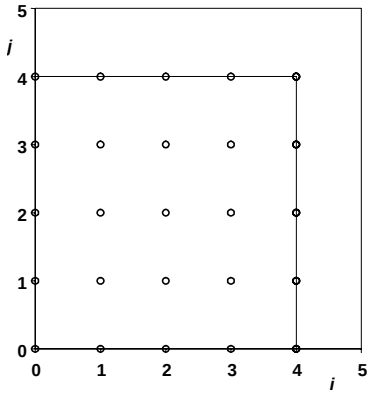
1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Гнездо цикла на рис. 1(б) имеет следующее свойство: каждая целочисленная точка в пределах границ цикла — это точка в пространстве итераций гнезда цикла. Такое пространство называется *плотным* пространством итераций. Для сравнения, в *разреженном* пространстве итераций целочис-

ленная точка может находиться в пределах границ цикла, но не быть итерационной точкой в пространстве итераций цикла. Понятие плотности и разреженности может быть формально определено следующим образом [3].

Определение 1. Пространство итераций плотное, если для любых двух целочисленных векторов v_1 и v_2, представляющих итерации цикла, любой целочисленный вектор $v_3 = \lambda v_1 + (1 - \lambda)v_2$ для некоторого $0 \le \lambda \le 1$ также представляет итерацию цикла. Пространство итераций разреженное, если оно не плотное.

```
for (i1=0; i1<5; i1++)
      for (i2=0; i2<5; i2++)
      {
          a[i1+1][i2]=b[i1][i2]+c[i1][i2];
          b[i1][i2+1]=a[i1][i2+1]+1;
      }
```



(*а*) (*б*)

Рис. 1. Гнездо цикла (а), плотное пространство итераций (б)

Смысл этой классификации пространства итераций состоит в том, что значительно труднее сгенерировать код для гнезда цикла в случае, когда пространство итераций разрежено, чем если пространство итераций является плотным, как будет показано в разд. 3.

1.1. Преобразования цикла

В этой статье рассматриваются преобразования, которые могут быть представлены линейными идентичными отображениями из пространства итераций исходной программы в пространство итераций конечной программы. Эта система преобразований включает перестановку, скашивание и обращение, а также новое преобразование — масштабирование цикла. Все эти преобразования являются стандартными за исключением масштабирования. Масштабирование цикла — преобразование, приводящее к разреживанию пространства итераций, и как следствие, — к изменению шага

цикла. Изменению подвергается внутренний цикл. Масштабирование цикла изменяет размер шага цикла, масштабируемая матрица не является унимодулярной. Линейное идентичное отображение между итерационными пространствами может быть получено при использовании целочисленных несингулярных матриц. Обратно, можно показать, что преобразование, представленное любой целочисленной несингулярной матрицей, может быть показано как композиция четырех основных преобразований. Более строго это может быть выражено в виде следующей теоремы.

Теорема 1. Преобразование, представленное любой целочисленной несингулярной матрицей, может быть представлено как композиция перестановки, скашивания, обращения и масштабирования.

Доказательство. Путем применения соответствующих элементарных строковых и столбцовых операций, целочисленная несингулярная матрица может быть приведена к диагональной матрице [4]. Элементарные операции можно представить путем умножения перестановочной, обращающей и скашивающей матриц с правой и с левой сторон несингулярной матрицы.

Теорема 2. Унимодулярные преобразования отображают плотное (разреженное) пространство итераций в другое плотное (разреженное) пространство итераций.

Доказательство. Пространство остается таким же, меняется только базис [5].

1.2. Генерация кода

Чтобы сгенерировать код для конечного гнезда цикла, необходимо сгенерировать циклы, которые просматривают точки конечного пространства итераций в лексикографическом порядке и переводят в операторах тела цикла старые индексы цикла к новым индексам цикла.

Если вектора S_i и S_j представляют начальные и конечные итерационные переменные, тогда имеем $S_i = T^1 S_j$. Это множество уравнений выражает старые индексы в терминах новых. Они могут быть использованы для замены исходных индексных переменных в теле цикла на новые индексные переменные. Для текущего примера это множество уравнений следующее:

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} \frac{1}{5} & -\frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}.$$

Заметим, что $T^{-1}S_j$ всегда будет целочисленной точкой, даже в случае, если элементы T^{-1} являются рациональными.

2. ЗАДАЧИ ГЕНЕРАЦИИ КОДА

Проблема при генерации циклов для получения конечного пространства итераций заключается в том, что данное преобразование, в основном, не сохраняет лексикографический порядок (две итерации могут быть выполнены в одном порядке в исходном гнезде цикла, но в другом порядке — в конечном гнезде цикла). Поэтому не существует явного пути для использования исходного гнезда цикла, чтобы сгенерировать код. В первую очередь, можно найти отображение исходных границ и затем сгенерировать гнездо цикла, которое посещает в лексикографическом порядке все целочисленные точки в области, ограниченной отображением. Этот подход работает хорошо, когда конечное пространство итераций плотное, разреженное пространство итераций потребует дополнительного аппарата, который позволил бы перепрыгивать через точки, не входящие в пространство итераций.

2.1. Вычисление отображения границ

Существует много путей для подсчета отображения исходных границ, один из простых методов использует исключение Фурье—Моцкина.

Для вычисления соответствующих границ цикла в данной работе используется алгоритм исключения Фурье—Моцкина [5, 6]. Алгоритм метода исключения Фурье—Моцкина достаточно прост. Рассмотрим систему линейных неравенств

$$\sum_{j=1}^{n} a_{ij} x_j \le b_j, i = (1,...,m).$$

Эта система может быть представлена в виде неравенств в соответствии со знаком коэффициента x_n.

$$x_n \le D_i(\overline{x}), i = (1,...,p),$$
$$x_n \ge E_j(\overline{x}), j = (1,...,q),$$
$$0 \le F_k(\overline{x}), k = (1,...,r),$$

где D_i, E_j и F_k — линейные функции от $\overline{x} = (x_1,..x_{(n-1)})$.

Теперь можно исключить x_n из системы для получения следующей преобразованной системы:

$$E_j(\bar{x}) \le D_i(\bar{x}), i = (1,...,p), j = (1,...,q),$$
$$0 \le F_k(\bar{x}), k = (1,...,r).$$

Этот процесс повторяется до тех пор, пока останется только одна переменная. Границы для этой переменной могут быть определены путем рассмотрения преобразованной системы уравнений.

Возвращаясь к задаче, рассмотрим систему неравенств для S_j. Границы цикла для j_n могут быть вычислены посредством решения неравенств относительно j_n. Границы для j_k могут быть вычислены путем исключения $j_{(k+1)},...,j_n$ из системы, используя метод исключения Фурье—Моцкина и т.д.

Рассмотрим данный пример. Пространство итераций на рис. 1(б) ограничено системой линейных неравенств:

$$0 \le i1 \le 4,$$
$$0 \le i2 \le 4.$$

Вычислим *i, j* в терминах *u, v*, используя матрицу преобразования, затем, заменяя *i, j* на *u, v* в неравенствах и применяя метод исключения Фурье—Моцкина, получим следующую картину границ пространства итераций:

$$0 \le u \le 4,$$
$$max(-2u, (u-20)/2) \le v \le min(20 - 2u, u/2).$$

К сожалению, нельзя использовать эти неравенства непосредственно при генерации кода для конечного гнезда цикла. Существуют две проблемы. Первая заключается в том, что нижняя и верхняя границы могут быть даже нецелочисленными, например, когда $u = 3$, верхняя граница для $v = 3/2$. Вторая заключается в том, что, даже если бы начальное пространство итераций было плотным, конечное пространство итераций будет разреженным. Это означает, что необходимо найти некоторые пути для перепрыгивания точек, которые не представляют итерации в пространстве конечного гнезда цикла.

2.2. Плотное пространство итераций

Для частного случая, когда конечное пространство итераций является плотным (в случае использования унимодулярной матрицы для преобразования гнезда цикла с плотным пространством итераций (теорема 2)), обе эти задачи могут быть решены просто. Если конечное пространство итераций плотное, нет необходимости перепрыгивать через точки, которые не

входят в пространство итераций конечного гнезда цикла. Более того, можно использовать операции «нижняя целая часть» и «верхняя целая часть» для получения ближайших целых чисел в пределах отображения границ.

2.3. Разреженное пространство итераций

Для разреженного пространства итераций операции «нижняя целая часть» и «верхняя целая часть» не могут решить проблему. Например, на рис. 6 (б) *(5, –7)* — это целочисленная точка, ближайшая к границе v при $u = 5$, но начальная точка, соответствующая $u = 5$, есть $v = 5$. В этом случае возможно только использование условного теста в теле цикла для избежания выполнения тела цикла в точках, которые не соответствуют точкам конечного пространства итераций. Условный тест предполагает посещение целочисленных точек, в которых нет необходимости, кроме того, условный тест дорог.

3. АЛГОРИТМ ГЕНЕРАЦИИ КОДА

Основная идея в понимании решения главной задачи состоит в том, что целочисленная несингулярная матрица T может быть разложена на составные части в виде произведения нижней треугольной матрицы H с положительными диагональными элементами и унимодулярной матрицы U. Это разложение относится к нормальным формам Эрмита матричного преобразования [7]. Действительно, если U используется для преобразования программы, результирующая программа выполняет итерации в том же самом лексикографическом порядке, как и программа, полученная путем использования T как матрицы преобразования. Можно заметить, что диагональные элементы H соответствуют размерам шагов цикла. Данное разложение дает алгоритм, который генерирует эффективный код для общего случая несингулярных матриц.

3.1. Вычисление матрицы полного преобразования

Данная процедура дополняет матрицу частичного преобразования до несингулярной квадратной, используя матрицу векторов зависимости, поэтому требуется совпадение количества строк матрицы частичного преобразования с её рангом.

В первом цикле удаляются некоторые столбцы матрицы зависимости, далее не участвующие в алгоритме. Во втором цикле процедуры происхо-

дит поиск некоторого вектора, линейно независимого от всех строк существующей на данный момент матрицы частичного преобразования, отклоненного от векторов матрицы зависимости не более чем на 90 градусов. Этот вектор дополняет матрицу частичного преобразования, причем соответствующие ему по определённому правилу вектора матрицы зависимости вычеркиваются из нее. Данная техника применяется до тех пор, пока матрица зависимости не пуста. На завершающей стадии алгоритма выполняется произвольное дополнение матрицы частичного преобразования до несингулярной квадратной.

На входе: транспонированная $m \times n$ матрица частичного преобразования $P^T_{m \times n}$, матрица зависимостей $D_{n \times n}$.

На выходе: матрица полного преобразования $T_{n \times n}$.

for i=1 to m {
$f^T := P_i^T D$;
Вычёркиваем из матрицы D столбцы D^j, где $f_j > 0$;
}
r:=m;
while D не пуста {
Пусть k — первая ненулевая строка D;
$x := c(I - P(P^T P)^{-1} P^T)e_k$, где число $c > 0$, приводящее x к вектору целых чисел с взаимопростыми компонентами;
Вычёркиваем из D столбцы D^j, где $D_{kj} > 0$;
Дополняем матрицу $P^T_{r+1} = x^T$;
r=r+1;
}
Пусть $R_{n \times n}$ — матрица перехода, полученная в результате работы алгоритма *HermiteNormalForm(P^T) (рис. 3)*;
Дополняем матрицу $P^T_{r \times n}$ до квадратной присоединением вниз нижних строк матрицы $R_{n \times n}$
($P^T_{n \times n} = appendBottom(P^T_{r \times n}, R[r+1:n, 1:n])$;

Результат $T := P^T_{n \times n}$.

Рис. 2. Алгоритм генерации матрицы

Несингулярная матрица $T = \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$, сгенерированная алгоритмом в соответствии с матрицей зависимости гнезда цикла $D = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$.

3.2. Вспомогательное пространство итераций

С помощью применения операций со столбцами к целочисленной несингулярной матрице T можно привести ее к целочисленной нижней треугольной матрице с положительными диагональными элементами [8]. Эта нижняя треугольная матрица относится к нормальной форме Эрмита [4, 7] матрицы T. Из этого следует, что T может быть записана как произведение нижней треугольной матрицы H с положительными диагональными элементами и унимодулярной матрицы U, которая представляет композицию операций со столбцами. Это разложение не уникальное, но для преобразования цикла можно использовать любое такое разложение.

Рис. 4 показывает алгоритм вычисления H и U [9].

Пусть $T = HU$, S_i — исходное пространство и S_j — конечное пространство. Определим $S_k = US_i$. Тогда $S_j = TS_i = HUS_i = HS_k$

Определение 2. Пространство итераций S_k называется *вспомогательным* пространством итераций к S_i по отношению к разложению HU.

Теорема 3. Вспомогательное пространство итераций является плотным пространством итераций, если начальное пространство плотное.

Доказательство: Следует из Теоремы 2, так как U — унимодулярная матрица.

Важное свойство вспомогательного пространства заключается в том, что оно выполняет итерации в том же лексикографическом порядке, что и конечное пространство итераций.

3.3. Нахождение формы Эрмита

Алгоритм представляет стандартную процедуру приведения матрицы к треугольному виду. Специфика алгоритма заключается в сохранении целочисленности элементов матрицы.

На входе: целочисленная матрица $T_{n\times n}$.
На выходе: форма Эрмита H матрицы T, матрица перехода $U_{n\times n}$ такая, что $HU=T$.

Пусть $U:=I$ — единичная матрица;
for i=1 to n {
while $T[i, i+1:n]\neq\vec{0}$ {
Найдём наименьший по модулю отличный от нуля элемент t_{ij} вектора $T[i, i : n]$;
if $i \neq j$ then меняем i,j столбцы T и i,j строки U ($T = TU_{ij}$, $U = U_{ij}^{-1} U$);
for j=i+1 to n {
Складываем столбец $[-\frac{t_{ij}}{t_{ii}}]\, T_i$ со столбцом T_j ($T = TU_{ij}$, $U = U_{ij}^{-1} U$);
}
}
}
$H:=T$;
Результат: матрицы H,U.

Рис. 3. Алгоритм вычисления формы Эрмита

Важное свойство вспомогательного пространства заключается в том, что оно выполняет итерации в том же самом лексикографическом порядке, как и конечное пространство итераций.

$$T=\begin{pmatrix}2 & 1\\ 1 & -2\end{pmatrix}\ H=\begin{pmatrix}1 & 0\\ -2 & 5\end{pmatrix}\ U=\begin{pmatrix}2 & 1\\ 1 & 0\end{pmatrix}.$$

Рассмотрим унимодулярное преобразование $U = \begin{pmatrix}2 & 1\\ 1 & 0\end{pmatrix}$ на текущем примере.

$$\begin{pmatrix}p\\ q\end{pmatrix}=U\begin{pmatrix}i\\ j\end{pmatrix}.$$

Для исходных границ на рис. 1(а) можно вычислить отображение границ, показанное на рис. 4(а). Так как вспомогательное пространство плотное, можно использовать операции «нижняя целая часть» и «верхняя целая часть» для подсчета точных границ.

Теорема 4. Если вспомогательное пространство итераций обходится в лексикографическом порядке, тогда конечное пространство итераций также обходится в лексикографическом порядке.

Доказательство. $S_j = HS_k$, где H — нижняя треугольная матрица с положительной диагональю. Пусть $\vec{k}_1 \prec \vec{k}_2$ — две итерации во вспомогательном пространстве итераций, $\vec{d}_1 = \vec{k}_2 - \vec{k}_1$ — разница этих двух векторов. Ясно, что $\vec{d}_1 \succ 0$. Чтобы увидеть, что лексикографический порядок сохраняется, рассмотрим новый дистанционный вектор $\vec{d}_2$

$$\vec{d}_2 = \vec{j}_2 - \vec{j}_1 = H\vec{k}_2 - H\vec{k}_1 = H\vec{d}_1 .$$

Если $\vec{d}_1(i)$ – i-й элемент $\vec{d}_1$, то ведущий ненулевой элемент $\vec{d}_1(i)$ должен быть положительным, так как $\vec{d}_1 \succ 0$. Тогда ведущий ненулевой элемент $\vec{d}_2$ — это $h_{ii}\vec{d}_1(i)$, который также положителен. Следовательно, $\vec{d}_2 \succ 0$ и $\vec{j}_1 \prec \vec{j}_2$.

Этот результат представляет собой способ генерации кода. Разложение T на HU — это генерация циклов для прохождения вспомогательного пространства с использованием унимодулярной матрицы и вычислением переменных конечного пространства итераций в теле цикла.

Используя границы начального пространства, получаем промежуточный код для текущего примера, показанный на рис. 4.

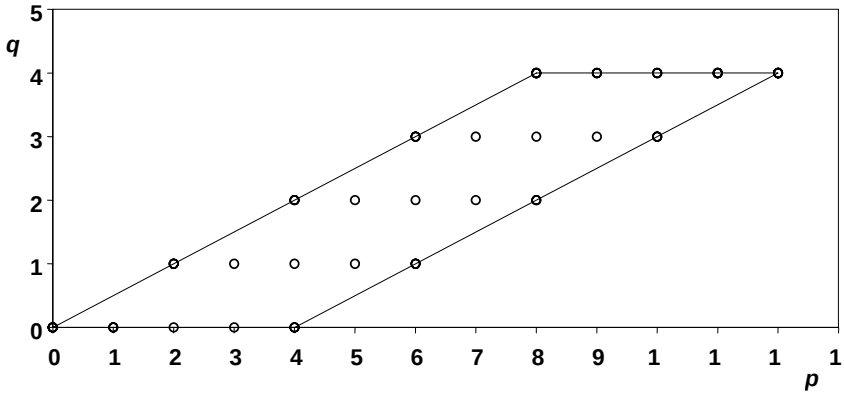
Данный код избегает выполнения условного теста, и его можно значительно улучшить. Можно заметить, что вычисление u постоянно во внутреннем цикле, кроме того, u — это линейная функция индекса внешнего цикла, и она может быть уменьшена в силе. Подобным образом, v — это линейная функция от p и q, и она может быть уменьшена в силе. Данные преобразования могут быть оставлены для дальнейшей оптимизационной стадии, предпочтительно использовать индукционные переменные u и v непосредственно как переменные цикла вместо p и q.

```
for (p=0; p<12; p++)
    for (q=max(0, ⌈ (p-4)/2 ⌉); q<min(4, ⌊ p/2 ⌋); q++)
    {
    /*функция перехода от промежуточных итерационных переменных к конечным */
```

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -2 & 5 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}$$

```
    a[(v+2*u)/5+1][(u-2*v)/5]=b[(v+2*u)/5][(u-2*v)/5]+c[(v+2*u)/5][(u-
    2*v)/5];
    b[(v+2*u)/5][(u-2*v)/5+1]=a[(v+2*u)/5][(u-2*v)/5+1]+1;
    }
```

(*а*)



(*б*)

Рис. 4. Промежуточное гнездо цикла (а),
вспомогательное пространство итераций (б)

3.4. Конечное пространство итераций

Так как H — это нижняя треугольная матрица, то легко преобразовать границы цикла вспомогательного пространства в границы цикла конечного пространства. Для текущего примера отношение между этими двумя пространствами выражается следующим уравнением:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -2 & 5 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} \quad \Rightarrow \quad \begin{cases} u = p \\ v = -2p + 5q \end{cases}.$$

Из первого уравнения следует, что границы для u — это границы для p. Следовательно, границы для u следующие: $0 \le u < 12$. Из второго уравнения следует, что границы для v — это границы для q, умноженные на *5* со смещением *(–2*u)*. Границы для u постоянные, границы для v зависят только от u. Следовательно, эти границы могут быть использованы непосредственно для построения конечного гнезда цикла. Алгоритм нахождения границ приведен на рис. 5.

3.5. Вычисление границ конечного пространства итераций

На входе: форма Эрмита H, границы вспомогательного пространства S_k.
На выходе: границы конечного пространства S_j.
for i=1 to n {
Вычислить смещение путем замены $k_1,\ldots,k_{(i-1)}$ на $j_1,..,j_{(i-1)}$
$v_i = h_{i1}k_1 + \ldots + h_{i(i-1)}k_{(i-1)}$
Вычислить нижнюю границу $l^k{}_i$, заменяя $k_1,..,k_{(i-1)}$ на $j_1,\ldots j_{(i-1)}$
$l_i^j = v_i + h_{ii}l_i^k$
Вычислить верхнюю границу $u^k{}_i$, заменяя $k_1,..,k_{(i-1)}$ на $j_1,\ldots j_{(i-1)}$
$u_i^j = v_i + h_{ii}u_i^k$
}
Результат S_j.

Рис. 5. Алгоритм нахождения границ цикла

Чтобы закончить генерацию кода, необходимо перепрыгнуть через точки в пределах границ цикла, которые не представляют итерации в конечном пространстве. Фактически оказывается, что данные точки удовлетворяют использованию циклов с постоянными размерами шагов, и эти шаги представляются числами на главной диагонали формы Эрмита.

Для текущего примера H имеет диагональ *[1, 5],* что соответствует шагу *1* для внешнего цикла и шагу *5* для внутреннего цикла. Таким образом, можно сформулировать следующую теорему.

Теорема 5. Положительные целые числа на диагонали формы Эрмита — это интервалы в каком-либо измерении.

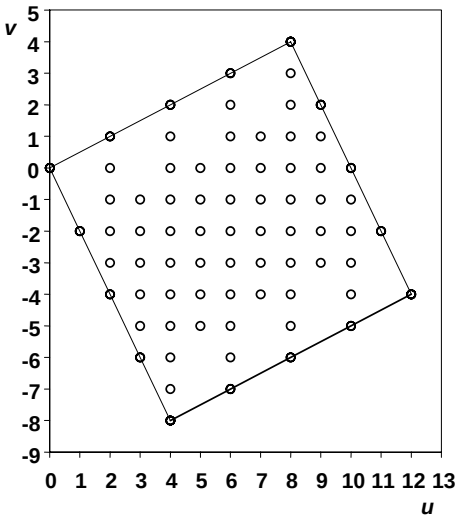
Доказательство. Рассмотрим две точки: $P1 = (k_1, k_2,\ldots k_i, k_{(i+1)}\ldots k_n)$ и $P2 = (k_1, k_2,\ldots k_i, a_{(i+1)}\ldots a_n)$ во вспомогательном пространстве, которые име-

ют одинаковые координаты в первых *(i – 1)* измерениях. Пусть *P3* и *P4* будут их отображениями в конечном пространстве. *P3* и *P4* имеют одинаковые координаты для первых *(i – 1)* измерений, кроме этого, их разница в *i*-ом измерении h_{ii}, так как H — это нижняя треугольная матрица.

Поэтому гнездо цикла может быть построено прямым прохождением конечного пространства. Для текущего примера на рис. 6 показано конечное гнездо цикла.

```
for (u=0; u<12; u++)
    for (v=-2*u+5*max⌈0,(u-4)/2⌉;  v<-2*u+5*min⌊4, u/2⌋; v=+5)
    {
        a[(v+2*u)/5+1][(u-2*v)/5]=b[(v+2*u)/5][(u-2*v)/5]+c[(v+2*u)/5][(u-2*v)/5];
        b[(v+2*u)/5][(u-2*v)/5+1]=a[(v+2*u)/5][(u-2*v)/5+1]+1;
    }
```

(а)



(б)

Рис. 6. Преобразованное гнездо цикла (а), разреженное пространство итераций (б)

Внутренний цикл в данном гнезде может быть распараллелен. Заметим, что вспомогательное пространство используется только для подсчета границ для конечного пространства.

4. СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Рассмотрим матрицу дистанционных векторов, представляющую зависимости по данным в цикле. Применим к этой матрице поочередно унимодулярное [1] и несингулярное преобразования. Полученные результаты представлены в следующей таблице.

преобразование	**D**	**T**	**H**	**U**	**D'**
несингулярное	$\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$	$\begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ -2 & 5 \end{pmatrix}$	$\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 3 & -2 \end{pmatrix}$
унимодулярное	$\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$			$\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$

Сравнивая несингулярное преобразование с унимодулярным, видим, что унимодулярное является частным случаем несингулярного. Алгоритм хорошо работает с унимодулярными преобразованиями, которые рассматриваются как частный случай. Если в качестве преобразования T сгенерирована унимодулярная матрица, то матрица H является единичной и, следовательно, шаг цикла равен единице, т.е. преобразование масштабирования цикла отсутствует. Дистанционные вектора конечного гнезда цикла различны (**D'**), но они, тем не менее, удовлетворяют условию о распараллеливании k-го цикла.

ЗАКЛЮЧЕНИЕ

Представленная система преобразования цикла основана на целочисленных несингулярных матрицах. Идея состоит в том, что сгенерированная несингулярная матрица может быть представлена в виде произведения нижней треугольной и унимодулярной матрицы. Посредством унимодулярной матрицы начальное гнездо цикла преобразуется к промежуточному

виду. Далее при помощи нижней треугольной матрицы уточняются границы, и на выходе имеем преобразованное гнездо цикла, в котором один из циклов может быть распараллелен. Также представлен простой законченный алгоритм, который получает на входе матрицу преобразования с первыми несколькими строками и генерирует полную матрицу преобразования.

СПИСОК ЛИТЕРАТУРЫ

1. Осмонов Р. А. Повышение степени параллелизма в циклах с помощью матричных преобразований // Тез. докл. конф.-конкурса работ студентов, аспирантов и молодых ученых «Технологии Microsoft в информатике и программировании». — Новосибирск, 2005. — С. 133–134.
2. Гантмахер Ф. Р. Теория матриц. — М.: Наука, 1998.
3. Schrijver A. Theory of Linear and Integer Programming. — John Wiley & Sons, 1986.
4. Ancourt C., Irigoin F. Scanning polyhedra with DO loops // Third ACM Symp. on Principles and Practice of Parallel Programming, April 1991. —P. 39–50.
5. Banerjee U. Loop transformations for restructuring compilers. The foundations. — Kluwer Academic Publishers, 1993.
6. Nemhauser G., Wolsey L. Lnteger and Combinatorial Optimization. Wiley-Interscience series in Discrete Mathematics and Optimization. — New York: John Wiley and Sons, 1988.
7. Xue J. An algorithm to automate non-unimodular transformations of loop nest // The 5th IEEE symp. on parallel distributed processing. — Dallas: IEEE computer society press, 1993.
8. Cohen H. A course in computational algebraic number theory. — Springer-Verlag, 1996.

А. Л. Серебренников

ОБЗОР ВОЗМОЖНОСТЕЙ СРЕДЫ SIGNIFICO НА ПРИМЕРЕ РЕШЕНИЯ ПРИКЛАДНОЙ ЗАДАЧИ

ВВЕДЕНИЕ

Со временем мы сталкиваемся с увеличивающимся потоком информации в различных областях знаний. Порой полученные нами данные неудобны для восприятия или обработки. В основном, это данные, полученные с различных регистрирующих и сканирующих устройств. Другими словами, мы получаем некоторую проекцию реального мира, и тут-то начинаются сложности. Общеизвестно, что с помощью стандартных методов довольно трудно производить, например, распознавание образов, классификацию, тем более прогнозирование. Все эти задачи могут быть решены с помощью нейросетевых технологий, о которых и пойдет речь в этой статье.

Создано достаточно много нейросетевых пакетов, но, как показывает практика, их возможностей хватает для решения довольно ограниченного круга задач. Существующие нейросетевые пакеты обладают недостаточным инструментарием для предварительной обработки данных, не позволяют организовывать системы нейронных сетей и содержат недостаточное число алгоритмов обучения. В этих условиях специалистам приходится создавать специализированный комплекс программного обеспечения для решения задачи.

В статье описывается создаваемая интегрированная среда по работе с нейронными сетями. В среде Significo сделана попытка создать инструментарий, с помощью которого можно довольно удобно решать комплексные задачи. Используя мощную систему предварительной обработки данных, возможность построения систем нейронных сетей и обширный набор алгоритмов обучения, аналитик или специалист могут создать в результате специфичный конвейер обработки данных.

В статье представлен пример решения задачи с помощью среды. Предлагается полностью пройти цикл решения задачи. Это позволит показать

преимущества и недостатки нейросетевых технологий и нововведения, имеющие место в среде Significo.

В данном случае решение задачи позволит производить анализ крови человека, не производя забор крови, а основываясь только на отражённом спектре света от тела человека. Эта методика могла бы значительно увеличить скорость анализа (с нескольких часов до секунды), уменьшить требуемое оборудование (с лаборатории до портативного устройства). Также с помощью метода можно было бы открыть новые возможности диагностики, применяя динамический анализ крови.

Решение задачи будет проходить в пять этапов.

1. Описание физических процессов, проходящих при отражении света, и создание их физической модели.
2. Анализ спектров, в результате которого, нужно выделить необходимые для обучения нейросети участки спектра.
3. Предварительная обработка данных.
4. Выбор наиболее подходящей архитектуры сети и алгоритма её обучения.
5. Подведение итогов проведённой работы и оценка результатов.

1. ОПИСАНИЕ ЗАДАЧ И ЦЕЛЕЙ ЭКСПЕРИМЕНТА

Под воздействием различных электромагнитных полей промышленного и природного происхождения физиологические системы организма человека претерпевают сложные процессы перестройки. К естественным природным электромагнитным полям, в основном, относится солнечная радиация, воздействие на организм человека которой до сих пор не имеет четкой и ясной оценки.

Связь между внешней средой, поверхностными покровами и внутренними органами живого организма многообразна. Она осуществляется через сосудистую, лимфатическую и нервную системы. Кожа человека, являясь эктодермальной производной, связана теснейшим образом со всеми внутренними органами человека. Кожа сама является комплексным органом, покрывающим все тело человека, она соприкасается с внешней средой и,

прежде всего, на ней возникает реакция от внешнего воздействия, переходящая на внутренние органы [1].

Ранее влияние солнечной радиации на организм человека изучалось чаще всего либо с точки зрения адаптации к различным климатическим условиям [2], либо с точки зрения исследования биологических ритмов [3]. В биологии и медицине влияние солнечной радиации на организм человека изучается достаточно давно. Значительные успехи достигнуты в лечении людей ультрафиолетовым и красным светом. Хотя механизмы взаимодействия человеческого организма с электромагнитным полем оптического диапазона частот остаются до конца не изученными, следует отметить также отсутствие медицинской аппаратуры, регистрирующей это взаимодействие.

Для решения этой задачи был использован спектрофотометр «Спектротон», способный регистрировать отраженный сигнал в диапазоне от 380 до 720 нм. Спектральные измерения коэффициента отражения кожи проводились с использованием техники интегрирующей сферы. Такой метод позволяет собирать рассеянное кожей излучение с разных слоев кожи и количественно оценивать коэффициент отражения в широком спектральном диапазоне.

Световой сигнал прибора, посылаемый на кожу, действует кратковременно, что не приводит к изменению физических свойств кожи и не вызывает повышения температуры тканей в зоне облучения. При облучении, например, красным лазером повышение температуры сопровождается выделением свободной воды, а при облучении ультрафиолетовой лампой повышение температуры не сопровождается потоотделением, что говорит о различных механизмах взаимодействия отдельных составляющих спектра с кожей человека.

Для исследования был выбран контингент практически здоровых мужчин и женщин в возрасте от 40 до 80 лет, проживающих в одной географической зоне (г. Новосибирск). Исследования проводились несколько лет с ноября по февраль месяц в первой половине дня, с 10 до 14 часов местного времени. Всего было обследовано 111 человек, из которых 67 человек — женщины и только 44 — мужчины. Спектральная характеристика снималась с внутренней поверхности предплечья левой руки, при этом обследуемый человек находился в комфортных температурных условиях в положе-

нии сидя. Исследования проводились в затемненной комнате. Эта область руки была выбрана для исследования преднамеренно как зона, не содержащая активно функционирующих органов.

2. ОПИСАНИЕ ФИЗИКИ ПРОЦЕССА ОТРАЖЕНИЯ/ПОГЛОЩЕНИЯ СВЕТА ЧЕЛОВЕЧЕСКИМ ТЕЛОМ

Измеряемый параметр достаточно сложная величина. С оптической точки зрения кожа представляет собой поглощающую среду с ярко выраженными рассеивающими свойствами. Взаимодействие света с кожей носит сложный характер, который начинает проявляться уже при прохождении света сквозь границу раздела «воздух—кожа». Граница раздела воздух-кожа не является гладкой. Она представляет собой плотный слой кератиноцитов, на котором располагаются фрагменты эпидермиса, находящиеся в стадии десквамации. Падающее излучение частично отражается (френелевское отражение, составляющее величину порядка 5%), не меняя своего спектрального состава. Значительная часть света (95%) входит в кожу, где свет поглощается и рассеивается. Энергия поглощенного света тратится на протекание различных фотохимических реакций. Спектр поглощения любой биологической ткани, в том числе и кожи, определяется наличием у практически всех биологически активных молекул сопряженных двойных связей (хромофорных групп) [4].

В измеряемом спектральном диапазоне 380-720 нм. роль хромофоров выполняют окисленные формы флавиновых соединений (λ = 450нм.). При связывании флавина с белком максимум сдвигается в сторону больших длин волн и проявляется при λ = 480–490 нм. [5]. Кроме того, помимо флавиновых соединений доминирующим хромофором эпидермиса, определяющим его поглощающие свойства в видимой области спектра, является пигмент меланин [6–10]. Спектр меланина, содержащегося в коже, имеет максимум поглощения около 335 нм. проявляется во всем измеряемом диапазоне и плавно уменьшается с увеличением длины волны.

Основными компонентами дермы, определяющими ее поглощение в видимой области спектра, являются хромофоры дермальной крови (оксигемоглобин, деоксигемоглобин, билирубин), каротиноиды и порфирины. Оксигенированная и деоксигенированная формы гемоглобина поглощают

свет следующим образом. Оксигемоглобин наиболее сильно поглощает свет в области 405нм., с увеличением длины волны его поглощение уменьшается, при этом на длинах волн 535 и 575 нм. имеются характерные максимумы поглощения. Деоксигемоглобин наиболее сильно поглощает свет в области 430 нм. и менее сильно вблизи 550 нм. Поглощение обеих форм гемоглобина в области свыше 600 нм. мало [6, 11, 12]. Для билирубина характерна полоса поглощения с максимумом в диапазоне 450–460 нм.

Помимо поглощения кожная ткань характеризуется значительным светорассеянием. Причиной рассеяния является неоднородность показателя преломления в объеме биоткани, отражающей ее физическую неоднородность [8, 16]. В кожной ткани есть рассеиватели, размеры которых меньше длины волны света, порядка длины волны, и рассеиватели, размеры которых значительно превышают длину волны света. В результате, в коже имеют место различные виды рассеивания: от рэлеевского рассеяния до рассеивания Ми.

3. ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

Предварительная обработка данных имеет очень большое значение при работе с нейросетевыми технологиями, по сути, задачи решаются именно на этом этапе. Искусственные нейронные сети практически помогают в компенсации различного рода помех, распознавании образов, классификации и прогнозировании. Таким образом, на этапе предварительной обработки требуется изучить сущность решаемой проблемы, описать физический смысл имеющихся данных, построить физико-математическую модель обследуемого объекта и относительно полученной модели извлечь из данных принципиально важные параметры, которые впоследствии будут применяться при обучении нейронных сетей.

Среда Significo представляет достаточно обширный инструментарий для осуществления предварительной обработки данных. В Significo представляется возможным составлять из различных элементов среды систему, которой под силу осуществлять предварительную обработку данных в автоматическом режиме, не используя дополнительных инструментариев. В данном аспекте среда Significo имеет ряд принципиальных отличий по

сравнению с имеющимися нейросетевыми пакетами, они заключаются в следующем.

- Возможность построения системы из различных элементов, представляющих собой отдельные специальные алгоритмы, предназначенные для предварительной обработки. В эти алгоритмы, помимо детерминированно направленных методов, входят и некоторые нейронные сети, которые способны компенсировать помехи, находящиеся в данных.
- Наличие значительно расширенного инструментария предварительной обработки, который включает в себя методы линейного статистического анализа, элементы спектрального разложения и др.
- Реализация прозрачности системы предобработки при работе пользователя на этапе обучения сети.

Модель облучаемых биологических тканей целесообразно представлять в упрощенном виде, так как реальная система очень сложна, а множество деталей не известны. Таким образом, представим модель тканей как четырёхслойную структуру.

Первый роговой слой обладает рассеивающими свойствами, и при его прохождении теряется порядка 5% излучения. Этот слой не изменяет состава спектра.

Второй слой представляет собой однородную светопроводящую среду с наличием клеток, которые поглощают отдельные участки спектров, энергия которых уходит на прохождение многочисленных химических реакций.

Третий слой — это совокупность светонепроницаемых образований, расстояние между которыми на порядок ниже длин волн видимого спектра. Таким образом, этот слой представляет собой интерференционную решётку, которая разлагает попавшее на неё излучение на спектр.

Четвёртый слой подобен второму, за исключением того, что в силу другого характера химических реакций поглощает другие участки спектра.

В результате становится ясно, как в рамках принятой нами модели могут изменяться полученные спектральные данные.

4. ОБЩИЙ АНАЛИЗ СПЕКТРОВ ОТНОСИТЕЛЬНО ГЕМОГЛОБИНА

Представленные в литературе относительно гемоглобина данные являются недостаточными, так как в этом случае мы имеем отраженный спектр с поверхности кожи, а известная информация была получена непосредственно из спектра крови. Поэтому необходимо провести общий анализ имеющихся спектров.

С помощью средств предобработки среды Significo строим следующий набор данных. В нём имеют место закономерности:

- все строки упорядочены по возрастанию гемоглобина,
- первый столбец — это изменения гемоглобина между двумя смежными столбцами, все остальные столбцы построены по аналогии, только для различных длин волн спектра,
- для более удобного графического представления первый столбец умножен на 10.

В таблице 3.1 включены участки спектров, которые отмечаются в литературе при анализе спектров крови. Применяя включённый в систему предобработки алгоритм получаем данные, находящиеся в таблице 3.2, которая была получена при анализе данных. Выбранные участки спектров соответствуют длинам волн, которые наибольшим образом коррелируют с параметром динамики гемоглобина.

В результате, из представленных таблиц можно сделать вывод, что большая часть представленных в источниках участков спектров или сильно зашумлены процессами, имеющими место в тканях человека, или сдвинулись по тем же причинам. В итоге совпало менее 50% участков спектра.

Таблица 1

160	10	10	14	20	22	32	35
30	-45	-37	-33	-52	-59	-56	-63
30	63	48	39	53	57	37	32
0	-21	-8	-6	-7	-8	-2	-6
30	4	3	8	13	16	18	27
20	-79	-76	-74	-79	-78	-50	-29
0	130	123	120	123	116	91	59
10	2	-4	-9	4	13	12	42
10	-39	-32	-31	-40	-41	-56	-65
0	-73	-64	-54	-87	-101	-51	-79
40	79	70	61	89	98	57	70
0	-1	-2	2	1	2	6	22
40	-57	-49	-46	-43	-38	-56	-82
20	51	44	43	38	30	55	64
40	-33	-36	-41	-40	-37	-47	-30
20	2	10	19	9	4	21	14
70	67	63	51	71	77	66	65
	400	410	430	450	460	550	600

Таблица 2

160	12	10	10	14	25	26
30	-65	-45	-37	-33	-59	-46
30	96	63	48	39	54	9
0	-47	-21	-8	-6	-10	5
30	8	4	3	8	21	1
20	-96	-79	-76	-74	-75	8
0	152	130	123	120	113	6
10	11	2	-4	-9	18	58
10	-43	-39	-32	-31	-22	-58
0	-116	-73	-64	-54	-134	-55
40	120	79	70	61	111	49
0	-4	-1	-2	2	-2	45
40	-84	-57	-49	-46	-43	-71
20	77	51	44	43	27	17
40	-24	-33	-36	-41	-32	-10
20	-21	2	10	19	1	16
70	83	67	63	51	95	29
	380	400	410	430	480	720

Полученные участки спектров являются лишь первоначальной предобработкой данных, заключающейся в выделении необходимых и достаточных данных для решения задачи.

Допустим, что в результате последующей нормировки все входные и выходные переменные отображаются в единичном кубе.

Задача нейросетевого моделирования, найти статистически достоверные зависимости между входными и выходными переменными. Единственным источником информации для статистического моделирования являются примеры из обучающей выборки. Чем больше бит информации принесет каждый пример, тем лучше используются имеющиеся в распоряжении данные.

Рассмотрим произвольную компоненту нормированных (предобработанных) данных — $\tilde{x}_i$. Среднее количество информации, приносимой каждым примером $\tilde{x}_i^\alpha$, равно энтропии распределения значений этой компоненты $H(\tilde{x}_i)$. Если эти значения сосредоточены в относительно небольшой области единичного интервала, информационное содержание такой компоненты мало. В пределе нулевой энтропии, когда все значения переменной совпадают, эта переменная не несет никакой информации. Напротив, если значения переменной $\tilde{x}_i^\alpha$ равномерно распределены в единичном интервале, информация такой переменной максимальна. Общий принцип предобработки данных для обучения, таким образом, состоит в максимизации энтропии входов и выходов. Этим принципом следует руководствоваться и на этапе кодирования нечисловых переменных.

5. ВЫБОР ОПТИМАЛЬНОЙ АРХИТЕКТУРЫ, ДОПОЛНИТЕЛЬНЫЕ СРЕДСТВА ДЛЯ УСПЕШНОГО РЕШЕНИЯ ЗАДАЧИ

Среда Significo к своему завершению должна иметь обширный набор нейросетевых парадигм, который мог бы позволить решать разнообразные задачи, связанные с компенсацией шумов, аппроксимацией, распознаванием образов, кластеризацией, классификацией. Для решения всех этих задач существуют наиболее удачные нейросетевые парадигмы. Предполагается, что в среде будет реализовано 6 архитектур, которые можно будет применять для решения вышеприведённых задач. Рассмотрим их кратко.

- Самой популярной архитектурой на данный момент является многослойный персептрон. Это наиболее универсальная архитектура, позволяющая применять к ней большой спектр обучающих алгоритмов. Эта сеть в основном применяется для распознавания различных образов.
- Сети с обратными связями. Это несколько преобразованный многослойный персептрон, который имеет обратные связи. Применяются данные сети для задач прогнозирования.
- Рекуррентные сети. Эти сети наиболее популярны в задачах прогнозирования, так как обладают свойством долговременной памяти, т.е. запоминают последовательности.
- Сети Кохонена. Данные сети наиболее популярны в задачах классификации. Это самоорганизующаяся сеть, и обучается она без учителя, т.е. позволяет проводить автоматическую классификацию.
- Вероятностные нейронные сети. Этот тип сетей знаменит своей способностью обучаться на ограниченном наборе данных. Может быть использован для классификации и кластеризации.
- Нейронные сети с общей регрессией. Данный тип нейронных сетей отличается тем, что может обучаться на ограниченном наборе данных и используется обычно для аппроксимации непрерывных функций.

Принципиальными отличиями среды Significo по сравнению с существующими на данное время нейросетевыми пакетами являются следующие.

- Способность объединять отдельные нейросети в системы, что позволяет снизить время обучения, по сравнению с применением одной нейросети. Каждую локальную задачу можно решать с помощью специфичной для этой задачи нейросетевой архитектуры.

- Возможность контрастирования нейронных сетей. Это позволит представлять сети не как чёрный ящик, а как логически понятную структуру. Операция контрастирования также минимизирует количество связей и нейронов в сети, что при моделировании снижает время обучения сети и количество потребляемых ресурсов. Сеть становится наиболее предсказуемой в зависимости от степени контрастирования.
- Реализовано большее количество алгоритмов обучения, что особо полезно при практическом использовании среды.
- Сделаны оригинальные доработки алгоритмов обучения, которые позволяют более гибко управлять процессом обучения, анализировать его и ускорять.

Вернёмся к решаемой задаче. В данном случае от нейросети требуется работа с зашумленными данными по распознаванию образов. Существует достаточно много нейросетевых парадигм для решения данной задачи, но целесообразнее выбрать именно многослойный персептрон по следующим причинам:

- успешно работает с зашумлёнными данными,
- применяется в задачах распознавания образов,
- для обучения многослойного персептрона имеется очень большое количество обучающих алгоритмов,
- возможно применять операцию контрастирования,
- легко анализировать процесс обучения.

При работе с многослойным персептроном целесообразно использовать сначала алгоритм обратного распространения ошибки, так как он менее ресурсоёмкий и для обучения требуется меньше времени. Даже в случае неудачного обучения можно извлечь пользу, например, оценить насколько сложные закономерности существуют в обучающих векторах, на это указывает уровень локальных минимумов ошибки. Если сеть не выходит из локального минимума, в процессе обучения можно применять различные

средства для исправления ситуации, например метод шока сети, или перейти на один из стохастических алгоритмов обучения, например Больцмановское обучение или обучение Коши.

6. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

На завершающем этапе работы производится обучение сети и подведение итогов.

Начнём с краткого анализа процесса обучения и указания особенностей работы среды в нём. Особенностями Significo являются следующие возможности.

- Выбор различных алгоритмов формирования списка калибровочных векторов: последовательный выбор, псевдослучайный выбор или ручной.
- Процесс обучения имеет несколько видов управления: ручной, по истечению определённого числа итераций, при достижении определённого значения ошибки на тренировочных или на калибровочных данных.
- Возможность отката процесса обучения на произвольную позицию с учётом выбираемого пользователем шага.

На рис. 1 мы видим картину процесса обучения, график серого цвета отображает ошибку обучения, черным цветом, отображается ошибка на калибровочных данных.

График на рис. 1 говорит о том, что на начальном этапе обучения сеть зашла в локальный минимум, через 40000 итераций вышла и далее продолжала обучаться в штатном режиме. По завершению обучения ошибка на тренировочных векторах составляла порядка 3%, а на калибровочных — 9%.

Рис. 1. График процесса обучения нейронной сети

Рассмотрим полученные результаты, они приведены в таблице 3. Вся таблица разделена на отдельные элементы, в которых указаны: прогнозируемое значение, значение, полученное классическими методами, абсолютная ошибка прогноза. Учитывая, что допустимая ошибка в стандартных анализах крови может достигать 15мг, все прогнозируемые значения гемоглобина удовлетворяют поставленным условиям. При обучении для калибровки было выбрано 20% векторов, 2 из которых не имеют отклонений, а 2 были спрогнозированы с достаточно большой ошибкой в 10%. Это объяснятся тем, что в наборе тренировочных векторов не было сходных с ними векторов. В тренировочном наборе имеются два прогноза с умеренной ошибкой, скорее всего их статистическая составляющая была довольно низкой. Основная же часть прогнозов была предельно точной.

Таким образом, это небольшое исследование можно считать успешным. Для достижения лучшего результата необходимо большее количество обучающих векторов.

Таблица 3

Результаты обучения нейронной сети

136,9993 137 -0,0007		135,0345 135 0,034522		125,0992 136 -10,9008	8,80%	142,055 142 0,054997	
131,0209 131 0,020897		144,4476 148 -3,55237	2,40%	118,9158 119 -0,08418		163,6293 164 -0,37074	
124,8767 125 -0,12329		138,9955 139 -0,00455		134,9771 135 -0,02287		127,0232 127 0,023163	
145,0121 145 0,012106		142,0218 142 0,021806		131,0731 131 0,073131			
124,5533 112 12,55333	10%	124,5512 137 -12,4488	10%	121,0053 121 0,005333			

общее число векторов	18
предельно корректный результат	12
хороший результат	2
Удовлетворительный результат	1
результат укладывающийся в норму 15мг	3

СПИСОК ЛИТЕРАТУРЫ

1. Владимиров Ю.А., Потапенко А.Я. Физико-химические основы фотобиологических процессов. — М.: Высш.шк., 1989. — 189 с.
2. Кочубей В.И., Конюхова Ю.Г. Методы спектрального исследования крови и костного мозга. — Саратов: Изд-во Сарат. ун-та., 2000. — 72 с.

3. А.Н.Горбань, В.Л.Дунин-Барковский, А.Н.Кирдин и др. Нейроинформатика. — Новосибирск: Наука. Сибирское предприятие РАН, 1998. — 296 с.
4. Ф. Уоссермен Нейрокомпьютерная техника, теория и практикаю — Norwood, 1992. — 118 с.
5. Масалович А.И. От нейрона к нейрокомпьютеру. — Журнал доктора Добба. — 1992. — 48 с.
6. Poli R., Cagnoni S., Livi R. et al. A Neural Network Expert System for Diagnosing and Treating Hypertension. — Computer, 1991. — 71 с.
7. Baxt W.G. A neural network trained to identify the presence of myocardial infarction bases some decisions on clinical associations that differ from accepted clinical teaching. — Med. Decis. Making, 1994. — 222 с.

А. Л. Серебренников

СРАВНИТЕЛЬНЫЙ АНАЛИЗ НЕЙРОСЕТЕВЫХ ПАКЕТОВ И МЕСТО СРЕДЫ SIGNIFICO СРЕДИ НИХ. КРАТКОЕ ОПИСАНИЕ СРЕДЫ

1. ОБЩИЕ СВЕДЕНИЯ О ПРИЛОЖЕНИИ НЕЙРОСЕТЕВОГО МОДЕЛИРОВАНИЯ

Программу моделирования нейронной сети обычно называют нейропакетом, понимая под этим программную оболочку, эмулирующую для пользователя среду нейрокомпьютера на обычном компьютере. В настоящее время на рынке программного обеспечения имеется множество самых разнообразных программ для моделирования нейронных сетей. Можно выделить несколько основных функций, которые реализованы во всех этих программах.

1.1. Формирование нейронной сети

Для решения разных практических задач требуются различные модели нейронных сетей. Модель нейронной сети определяется моделями нейронов и структурой связей сети.

Для построения нейронной сети, ориентированной на решение конкретной задачи, используются процедуры формирования нейронных сетей, которые обеспечивают ввод указанных характеристик моделей нейронов и структур нейронных сетей.

Каждая группа моделей нейронных сетей может быть использована для решения лишь некоторого ограниченного класса практических задач. Так, многослойные и полносвязные нейронные сети с сигмоидальными передаточными функциями используются для распознавания образов и адаптивного управления; нейронные сети с локальными связями — для обработки изображений и некоторых других частных задач. Для решения задач линейной алгебры используются многослойные сети с особыми передаточными функциями.

Лишь для небольшого числа моделей нейронных сетей существует строгое математическое обоснование возможности их применения для решения конкретных практических задач. В наибольшей степени теоретически проработаны двухслойные нейронные сети с сигмоидальными переда-

точными функциями. На основе теоремы Колмогорова-Арнольда доказано, что такие сети могут реализовывать любые отображения входного сигнала в выходной. К построению многопараметрических отображений сводится большинство задач распознавания, управления, идентификации.

1.2. Обучение нейронной сети

В большинстве нейропакетов предлагаются стандартные процедуры обучения нейронных сетей, ориентированные на конкретные нейропарадигмы.

Как правило, в нейропакетах реализуется возможность задания различных типов данных и различных размерностей входных и выходных сигналов в зависимости от решаемой задачи. В качестве входных данных в обучающей выборке могут использоваться растровые изображения, таблицы чисел, распределения. Типы входных данных — бинарные (0 и 1), биполярные (-1 и +1) числа, целые или действительные числа из некоторого диапазона. Выходные сигналы сети — векторы целых или действительных чисел.

Для решения практических задач часто требуются обучающие выборки большого объема. Поэтому в ряде нейропакетов предусмотрены средства, облегчающие процесс формирования и использования обучающих примеров. Однако в настоящее время отсутствует универсальная методика построения обучающих выборок, и набор обучающих примеров, как правило, формируется индивидуально для каждой решаемой задачи.

В качестве функции ошибки, численно определяющей сходство всех текущих выходных сигналов сети и соответствующих требуемых выходных сигналов обучающей выборки, в большинстве случаев используется среднеквадратичное отклонение. Однако в ряде нейроимитаторов существует возможность либо выбора, либо задания своей функции ошибки.

Реализуемые в нейропакетах алгоритмы обучения нейронных сетей можно разделить на три группы: градиентные, стохастические, генетические. Градиентные алгоритмы (первого и второго порядков) основаны на вычислении частных производных функции ошибки по параметрам сети. В стохастических алгоритмах поиск минимума функции ошибки ведется случайным образом. Генетические алгоритмы комбинируют свойства стохастических и градиентных алгоритмов: на основе аналога генетического наследования реализуют перебор вариантов, а на основе аналога естественного отбора — градиентный спуск.

1.3. Тестирование обученной нейронной сети

Для проверки правильности обучения построенной нейронной сети в нейроимитаторах предусмотрены специальные средства ее тестирования. В сеть подается некоторый сигнал, который, как правило, не совпадает ни с одним из входных сигналов примеров обучающей выборки. Далее анализируется получившийся выходной сигнал сети.

Тестирование обученной сети может проводиться либо на одиночных входных сигналах, либо на тестовой выборке, которая имеет структуру, аналогичную обучающей выборке, и также состоит из пар (<вход>, <требуемый выход>). Обычно обучающая и тестовая выборки не пересекаются. Тестовая выборка строится индивидуально для каждой решаемой задачи.

2. СРАВНИТЕЛЬНЫЙ АНАЛИЗ НЕЙРОСЕТЕВЫХ ПАКЕТОВ

2.1. NeuroSolutions

Универсальный нейропакет NeuroSolutions фирмы NeuroDimension Inc предназначен для моделирования широкого круга искусственных нейронных сетей. По оценкам специалистов он является лучшим нейросетевым пакетом. Основное достоинство описываемого нейропакета состоит в его гибкости: помимо традиционно используемых нейросетевых парадигм (типа полносвязных многослойных нейронных сетей или самоорганизующихся полей Кохонена) нейропакет включает в себя мощнейший редактор визуального проектирования нейронной сети, позволяющий создавать практически любые собственные нейронные структуры.

Нейропакет NeuroSolutions снабжен мощными и хорошо продуманными средствами визуализации: отображать в каком угодно виде и визуально контролировать можно все, начиная от структуры нейронной сети и кончая процессом и результатом обучения. Наличие мощных средств визуализации выводит данный нейропакет на уровень CAD-систем (систем автоматизированного проектирования), т.е. NeuroSolutions можно считать системой проектирования и моделирования нейронной сети. Помимо грамотно организованных средств взаимодействия с операционной системой (поддерживается OLE2) нейропакет снабжен также генератором исходного кода и средствами, позволяющими использовать внешние модули при проектировании и обучении нейронной сети.

Нейропакет NeuroSolutions имеет также достаточно мощные средства для организации обучающих выборок. Встроенные конверторы данных поддерживают графические изображения в формате BMP, обычные текстовые файлы с числовыми или символьными данными, а также функции непрерывного аргумента (например, времени), заданные в аналитическом виде или в виде выборки значений. Нейропакет позволяет использовать любые внешние конверторы данных, определяемые пользователем.

Для тех, кто использует только традиционные нейронные парадигмы и не пользуется средствами собственного проектирования нейронной сети, нейропакет NeuroSolutions содержит генератор стандартных архитектур (Neural Wizard). С помощью этого генератора можно быстро задать архитектуру нейронной сети, подобрать обучающую выборку, критерии и методы обучения. Поддерживаются наиболее известные нейросетевые парадигмы: многослойные сети, сети Кохонена, самоорганизующиеся структуры и др.

К недостаткам нейропакета NeuroSolutions можно отнести то, что в пакет включено недостаточное количество критериев обучения, если же требуется применять нестандартные критерии обучения, пользователь сталкивается с тем, что ему необходимо создавать внешние модули с последующим подключением к пакету.

В пакете реализовано мало методов обучения: алгоритм обратного распространения, сети Кохонена, самоорганизующиеся структуры. Это наиболее популярные методы, но их явно недостаточно. Для использования других методов обучения пользователю потребуется создать их самому.

В итоге можно сделать вывод, что нейропакет NeuroSolutions имеет множество плюсов и некоторые недостатки и обладает гибкостью и универсальностью.

2.2. NeuralWorks Professional

Нейропакет NeuralWorks Professional так же, как и пакет NeuroSolutions, является мощным средством для моделирования искусственных нейронных сетей. В отличие от пакета NeuroSolutions в NeuralWorks Professional основной упор сделан на применение стандартных нейронных парадигм и алгоритмов обучения, и в этом данный пакет превосходит все остальные. В нем реализованы 28 стандартных нейронных парадигм, используемых при решении прикладных задач. Нейропакет содержит также большое число алгоритмов обучения нейронной сети. Дополнительно поставляемый модуль UDND (User Define Neural Dynamics) позволяет пользователю создавать свои собственные нейронные структуры и работать с ними средствами нейропакета.

Так же, как и NeuroSolutions, NeuralWorks Professional имеет грамотно организованную систему визуализации данных. Можно просмотреть структуру нейронной сети, изменение весовых коэффициентов в процессе обучения, изменение ошибки обучения, а также корреляцию весов нейронной сети при обучении. Последнее является уникальной возможностью, предоставляемой только пакетом NeuralWorks Professional, и весьма полезной при анализе поведения нейронной сети при обучении и работе.

Так же, как и NeuroSolutions, NeuralWorks Professional представляет собой открытую систему, в которую можно интегрировать внешние программные модули, написанные пользователями. Пакет имеет встроенный генератор кода, поддерживающий компилятор Microsoft Visual C++.

Способ представления информации несколько отличается от NeuroSolutions, хотя больших качественных отличий между ними нет. Следует обратить внимание на окно корреляции весовых коэффициентов в левом верхнем углу экрана.

К недостаткам пакета можно отнести то, что он не позволяет пользователям создавать свои нейросети, алгоритмы обучения и даже изменять критерии обучения. Все эти возможности можно реализовать в отдельном модуле UDND. Вторым недостатком является отсутствие мощной системы предварительной обработки данных.

2.3. Process Advisor

Нейропакет Process Advisor специально создавался для решения задач управления динамическими процессами (в частности, технологическими). Однако разработчикам удалось создать программу, которая вполне может считаться нейропакетом.

В данном нейропакете реализована только многослойная нейронная сеть прямого распространения, обучаемая с помощью модифицированного алгоритма обратного распространения ошибки (back-propagation). Поскольку изначально предполагалось, что нейропакет предназначен для решения задач управления, т.е. динамических задач, то в него были введены возможности работы с динамическими процессами.

В частности, в Process Advisor возможна работа с входными сигналами как с функциями времени, а не дискретным набором точек. Такую возможность, помимо Process Advisor, предоставляет только нейропакет NeuroSolutions. Кроме того, нейропакет Process Advisor позволяет осуществлять управление внешними аппаратными контроллерами, подключаемыми к компьютеру. В основном только эти две особенности делают нейропакет Process Advisor примечательным.

Недостатки данного пакета налицо. Нет возможности изменять архитектуру сети, обучающие алгоритмы, критерии обучения. В инструментарии пакета есть только многослойная нейронная сеть, и модифицированный алгоритм обратного распространения ошибки, у которого можно изменять очень ограниченное число параметров. Так же практически отсутствует необходимая система предварительной обработки данных.

2.4. NeuroShell 2

Нейропакет NeuroShell 2 сильно проигрывает по сравнению с NeuroSolutions и NeuralWorks и может считаться универсальным с некоторой натяжкой.

Кроме недостаточно продуманного интерфейса, нейропакет NeuroShell 2 имеет и недостаточно продуманную систему визуализации данных: контролировать можно многие параметры, но в разных режимах работы нейропакета. Из-за отсутствия единого интегрального контроля данных в процессе обучения или работы нейронной сети часто приходится переключаться из одного режима в другой, что отнимает много времени и весьма неудобно в использовании.

К особенностям нейропакета следует отнести жестко реализованную последовательность действий при работе с нейронной сетью. Так, невозможно определить структуру нейронной сети до того, как заданы входные данные. С одной стороны, это очень удобно, особенно для начинающих пользователей, поскольку сразу становится ясно, что и в какой последовательности следует делать. С другой стороны, более опытного пользователя такая жесткая зафиксированная последовательность действий утомляет. Для того, чтобы внести в нейронную есть небольшое изменение, приходится выполнять всю цепочку действий. Таким образом, можно сказать, что пакет NeuroShell 2 удобен для начинающих пользователей.

В NeuroShell 2 реализована достаточно мощная система обмена данными с другими приложениями. Данный нейропакет позволяет читать данные, представленные в текстовом бинарном виде, а также в наиболее популярных финансовых форматах Mala Stock и Dow Jones. Нейропакет имеет также генератор исходного кода на языках Visual C и Visual Basic

Подводя итог, следует отметить главные недостатки. В пакете нет возможности создания собственной архитектуры нейронной сети, нельзя использовать специфичные алгоритмы обучения и изменять критерии обучения нейронной сети.

2.5. Особенности среды Significo

Среда Significo является попыткой создания универсального и гибкого инструментария для работы с нейронными сетями (НС) и различными алгоритмами обработки данных. Ключевыми особенностями среды являются следующие аспекты: среда позволяет создавать системы нейронных сетей, поддерживать 3 уровня пользователей, предусмотреть как использование различных НС и алгоритмов их обучения, так и различных специализированных алгоритмов обработки данных. Рассмотрим более детально эти три основных аспекта.

Возможность организации в среде системы нейронных сетей, содержащей любой элемент среды, очень важна. Практика применения нейронных сетей показывает, что для решения довольно сложных задач требуется применение некой совокупности нейронных сетей в связке с различными специфичными алгоритмами обработки данных. Это позволяет получать не только лучшие результаты, но и верифицировать их. Возможно структурировать решаемую задачу, в результате чего полученное решение можно использовать в будущем, изменяя только часть решения. Также данная методика позволяет интерпретировать результат в логически понятной форме. Затраты ресурсов как временных, так и вычислительных, значительно снижаются, так как приходится обучать несколько нейросетей небольших размеров вместо одной большой сети. Для решения каждой части задачи могут применяться наиболее подходящие элементы среды.

Поддержка трёх уровней пользователей позволяет организовать оптимальный процесс работы исследовательской группы или другого научного коллектива. При данной постановке процесса каждая целевая группа пользователей выполняет свои специфические функции. Предложены три группы с условными названиями: программисты, аналитики, специалисты. Функции трёх целевых групп наиболее удобно рассмотреть на предполагаемом процессе разработки системы нейронных сетей.

- Специалистами составляется техническое задание, включающее общее описание решаемой ими задачи, модель исследуемого объекта, описание входных данных, описание требуемого результата.
- Аналитик или группа аналитиков, изучив техническое задание, составляет спецификацию на создаваемую систему, которая должна включать выбор и обоснование применяемых для решения задачи методов и алгоритмов с учётом таких аспектов, как наличие применяемых методов в среде и др., требуемые затраты ресурсов, спецификации на отсутствующие в среде элементы (для программи-

стов). Далее спецификация передаётся группе специалистов, и, если у них не возникает никаких замечаний, аналитики передают всю требуемую документацию программистам. Получив недостающие элементы среды, они составляют из элементов систему, настраивают её, тестируют и составляют документацию для пользователей. После чего система нейронных сетей готова к использованию специалистами.

- Программисты, изучив полученную документацию, составляют программную спецификацию на требующиеся аналитикам элементы среды. Далее реализуют элементы, интегрируют их в среду, тестируют и составляют документацию по реализованным элементам.

Как видно из краткого описания технологического процесса, данная поддержка направлена на структуризацию задач, выполняемых в процессе разработки той или иной системы нейронных сетей. Предполагается, что разные этапы различной направленности выполняют участники 3-х целевых групп. Это позволяет выполнять работу в кратчайшие сроки, не отвлекая специалистов от своих специфичных задач. Круг пользователей среды при таком подходе может быть неограниченным.

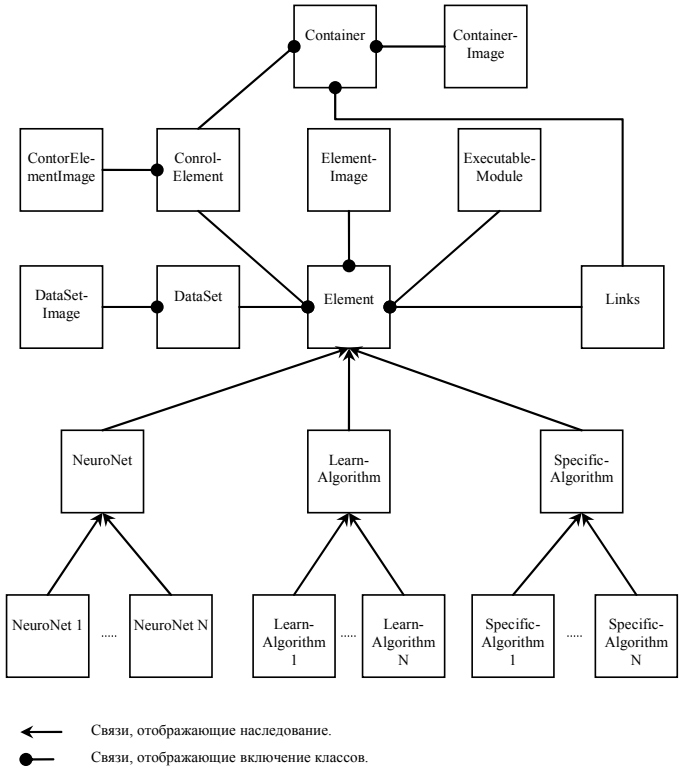
Использование различных алгоритмов обработки данных может значительно расширить область применения среды. Представляется возможным организация мощной системы предварительной обработки данных. Это наиболее важный этап, так как, в основном, любая задача, решаемая с помощью нейросетевых технологий, решается именно на этом этапе. Ведь задача нейросетевого моделирования — найти статистически достоверные зависимости между входными и выходными переменными. Единственным источником информации для статистического моделирования являются примеры из обучающей выборки. С другой стороны, в большинстве случаев при решении комплексных задач данные стекаются со множества нейронных сетей, а далее с помощью детерминированных методов интерпретируются в окончательный результат. Таким образом, возможность использования и дополнения среды различными алгоритмами обработки данных трудно переоценить.

Подводя итоги, хотелось бы отметить, что при использовании среды представляется возможным не только накапливать реализованные в среде методы и алгоритмы, но и осуществлять накопление наработок, решений по различным задачам. Это позволит применять полученные результаты в дальнейшем и распространять их.

3. КРАТКОЕ ОПИСАНИЕ СРЕДЫ

Среду Significo, в соответствии со сравнительным анализом относительно других приложений нейросетевого моделирования, можно классифицировать как универсальный нейросетевой пакет, но с учётом всех особенностей Significo наиболее точным определением является термин «среда нейросетевого моделирования». Для более детального описания среды будут рассмотрены следующие вопросы: базовая архитектура среды, схема взаимодействий среды, подробное описание функционалов для каждой из трёх пользовательских групп.

3.1. Базовая архитектура среды



Функциональное назначение представленных в схеме классов дано в таблице.

Наименование класса	Назначение
ControlElement	Реализует логическую модель групп параметров для управления элементом среды или контейнером. Содержит в себе название параметра, его описание, характеристики.
ControlElementImage	Реализует визуализацию ControlElement. Позволяет пользователю производить манипуляции над ControlElement.
DataSet	Реализует унифицированный набор данных среды, позволяет производить простейшие манипуляции с данными.
DataSetImage	Реализует образ набора данных и позволяет пользователю производить манипуляции с объектом DataSet.
Links	Класс, с помощью которого реализуется взаимодействие между элементами среды и контейнерами.
ExecutableModule	Класс, реализующий логику работы элемента.
Element	Класс, реализующий элемент среды, включает функциональность всех вышеперечисленных классов. Логически его можно представить как совокупность обвязки среды и логическое ядро, которое взаимодействует со средой через интерфейсную оболочку.
ElementImage	Реализует образ элемента среды, включает функции манипуляции элементом, заложенные во всей интерфейсной оболочке элемента среды.
Container	Класс, предназначенный для объединения элементов создаваемой пользователем системы в группы. Данная функция позволяет упрощать графическое представление сложных систем. Вторая функция контейнера — это группировать параметры управления содержащихся в системе элементов.
ContainerImage	Реализует образ контейнера и позволяет им манипулировать в рамках его интерфейса.
NeuroNet	Реализует на базе класса Element класс нейронной сети со всей специфической функциональностью.

LearnAlgorithm	Реализует на базе класса Element класс алгоритма обучения нейронной сети.
SpecificAlgorithm	Реализует на базе класса Element класс алгоритма обработки данных. Это может быть как алгоритм предварительной обработки данных, так и какой-либо специфический алгоритм.
NeuroNet1..NeuroNetN, LearnAlgorithm1..LearnAlgorithmN, SpecificAlgorithm1..SpecificAlgorithmN	Данные классы наследуются от 3-х вышеприведённых классов. Реализуют конкретные нейросетевые архитектуры, алгоритмы обучения и алгоритмы

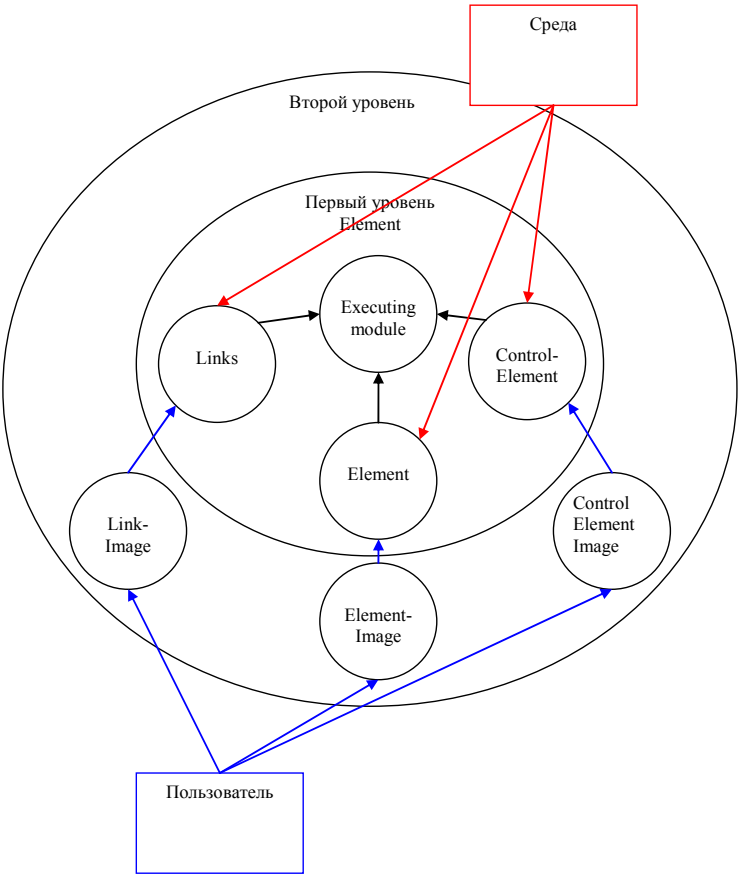
3.2. Схема взаимодействий среды

Центральным элементом среды является класс Element. У данного класса есть две основных области взаимодействия: область взаимодействия с пользователем и область взаимодействия с другими элементами среды.

Область взаимодействия с пользователем организована на классах ContorlElementImage, DataSetImage, ContainerImage, ElementImage. Взаимодействие пользователя с элементом производится по цепочке: событие образа, интерфейс класса образа, и замыкает цепочку интерфейс класса Element.

Область взаимодействия между элементами среды осуществляется через одну интерфейсную оболочку.

Таким образом, у элемента среды имеются две интерфейсных оболочки: одна из них участвует при любом взаимодействии, вторая — только лишь при взаимодействии с пользователем.


Среда
Второй уровень
Первый уровень
Element
Executing
module
Links
Control-
Element
Element
Link-
Image
Control
Element
Image
Element-
Image
Пользователь

3.3. Описание пользовательских групп

В среде предусмотрено три группы пользователей. Они поддерживаются путём сокрытия тех частей интерфейса, которые не участвуют в технологических процессах той или иной группы. Если всё же пользователю из какой-либо группы требуется использовать какую-либо функцию, которая не отображается в интерфейсе, то он может легко получить доступ к ней. Разделение интерфейса на 3 группы пользователей обусловлено стремлением к максимальному удобству использования среды. Весь процесс работы со средой также разбит на три уровня: уровень пользователя (специалиста), уровень аналитика и уровень программиста. Рассмотрим более подробно процесс разработки с помощью среды и наиболее точно обозначим функции, выполняемые той или иной группой пользователей.

Группа аналитиков — это основная группа, участвующая в разработке нейросетевых систем среды. Главным направлением данной группы является создание систем нейронных сетей, разработка интерфейса для специалистов, написание документации по проекту. Таким образом, аналитик должен не только ориентироваться в нейросетевых технологиях, но и знать принципы составления интерфейсных XML файлов.

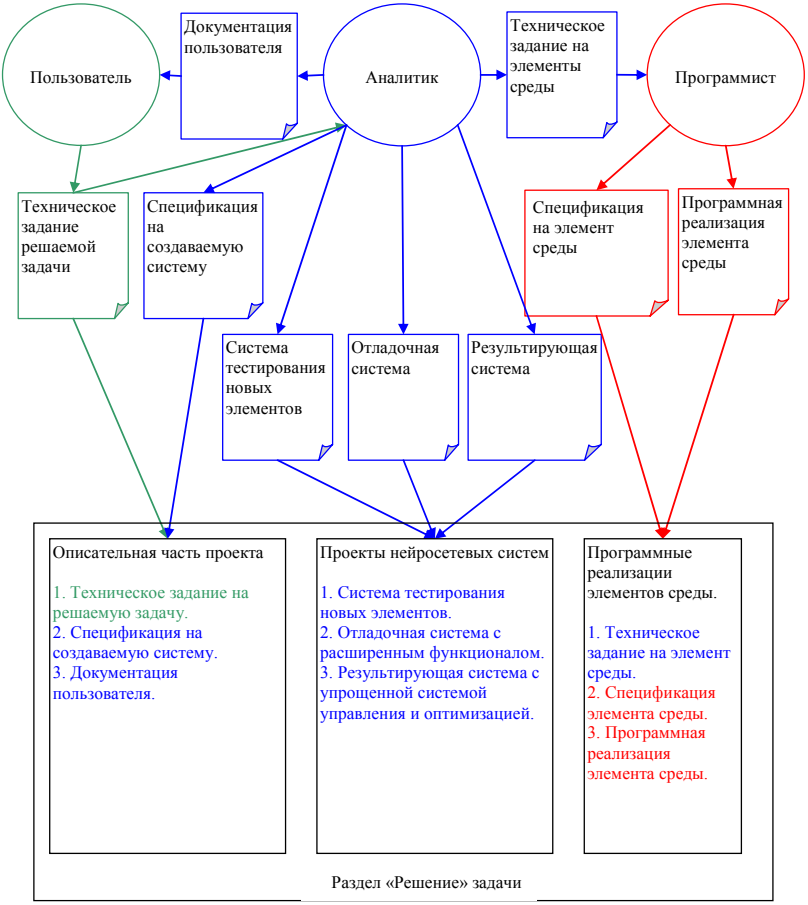
При работе со средой группа программистов не является обязательной, программисты требуются лишь в том случае, если инструментария среды недостаточно для решения поставленной задачи. В противном случае обязанностью программиста будет программная реализация требующегося элемента среды на любом из языков семейства .NET, а также составление интерфейсных XML-файлов.

Группа специалистов представляет собой основную группу пользователей. Для неё использование полученного аналитиками решения должно быть предельно простым. Предполагается, что эта группа пользователей наиболее узкоспециализированная и имеет лишь поверхностные представления о работе среды и нейросетевых технологиях.

Процесс разработки в среде состоит в общем случае из трёх частей. В среде вводятся понятия «Решение» и «Проект». «Решение» должно содержать в себе документацию по решаемой задаче: техническое задание, спецификацию проекта, описание системы, полученной в результате, пользовательскую документацию. Также раздел «Решение» должен включать в себя проекты, созданные в рамках решаемой задачи. Например, при решении задачи потребовалось реализовывать отсутствующие в среде элементы. В этом случае в раздел «Решение» должны быть добавлены проекты тести-

рования добавленных элементов, отладочные системы, используемые аналитиками при решении задачи, и проект с оптимизированной системой для решения поставленной задачи и созданными для пользователей удобными и упрощенными интерфейсами системы.

Весь технологический процесс представлен на следующей схеме.



СПИСОК ЛИТЕРАТУРЫ

1. Горбань А.Н., Дунин-Барковский В.Л., Кирдин А.Н. и др. Нейроинформатика. — Новосибирск: Наука. Сиб. отд-ние РАН, 1998. — 296 с.
2. Уоссермен Ф. Нейрокомпьютерная техника, теория и практика. — Norwood, 1992. — 118 с.
3. Новиков И.С. Обзор нейросетевого программного обеспечения. — 2001. — <http://vlasov.iu4.bmstu.ru/>
4. Нейронные сети. — 2004. — <http://ole-u.cefb.ru/>
5. Загоруйко Н.Г. Прикладные методы анализа данных и знаний. — Новосибирск: Изд-во Ин-та математики, 1999. — 270 с.
6. Дюк В.А. Data Mining — интеллектуальный анализ данных. — 2003

А. П. Стасенко[1]

ОБЗОР ПОТОКОВЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ[2]

ВВЕДЕНИЕ

Язык программирования называется потоковым, если он подходит для описания программ для потоковых архитектур, которые на машинном уровне описывают вычисления в виде графа потока данных. Поэтому потоковый язык должен удовлетворять следующим критериям.

1. Возможность извлечения зависимостей по данным из текста программы, что достигается при соблюдении принципа локальности действия (locality of effect) или прозрачности имён ссылок (referential transparency), олицетворяющих значения, а не ячейки памяти.
2. Последовательность вычислений определяется только готовностью данных, что обеспечивается отсутствием побочных эффектов (side-effect) вычислений или их математической правильностью.

Потоковые языки программирования обычно являются представителями более широкого класса «чистых» (без побочных эффектов) функциональных языков, задающих вычисления путём описания применений функций к их аргументам. Функциональные языки, основанные на λ-исчислении Черча, обладают большей «выразительной мощностью», чем императивные языки программирования, основанные на машинных языках. Выразительная мощь функциональных языков заключается в легкости описания операций над сложными объектами, такими как сами функции, и неявно задаваемом машинно-независимом параллелизме, ограниченном лишь зависимостями по данным.

В связи с тем, что функциональные языки не фиксируют порядок вычислений, для них актуально понятие стратегии вычисления, определяющей порядок применения функций. Строгая стратегия (strict, eager, call-by-value), заключающаяся в применении функции уже к вычисленным аргументам, может приводить к зацикливанию вычислений, если зацикливается даже какая-то его часть, не влияющая на общий результат. Ленивая страте-

[1] astasenko@gmail.com

[2] Работа выполнена при финансовой поддержке Министерства образования РФ (научная программа «Университеты России», грант № УР.04.01.201).

гия (lazy, non-strict, call-by-need), откладывающая вычисления пока это возможно, свободна от этого недостатка, что позволяет включать в язык потенциально бесконечные структуры данных, но сильнее ограничивает параллелизм. Возможны и другие (lenient) смешанные стратегии вычисления.

Принцип прозрачности имён ссылок меняет семантику оператора присваивания на определение связи между именем и значением. Свойство однократного присваивания (single-assignment), не являющееся обязательным для потоковых языков, запрещает переопределение одного имени в пределах одной области. Также принцип локальности действия приводит к необходимости обрабатывать ошибочные ситуации с помощью специально выделенных для этой цели ошибочных значений.

Вначале свойства потоковых языков в значительной степени определялись особенностями существующих потоковых архитектур, первые из которых были статическими и не допускали повторное использование потокового кода, в том числе рекурсии. Данные ограничения были в дальнейшем преодолены в динамических потоковых архитектурах, которые разрешали одновременное продвижение множества наборов разметок по дугам потокового графа.

В настоящее время потоковые языки потеряли свою изначальную привязанность к потоковым архитектурам и в основном рассматриваются из-за своей выразительной мощности. Далее более подробно будут рассмотрены типичные представители потоковых языков, такие как языки Lucid, Id, Val, Post, Sisal и Пифагор.

1. LUCID

Нестрогий язык Lucid, разработанный в 1976 г. [1] для динамической потоковой архитектуры, был одним из первых потоковых языков. В своём изначальном виде язык Lucid представлял собой расширение двумя новыми операторами *next* и *fby* одного из первых функциональных языков ISWIN (If You See What I Mean). Изначально язык Lucid предназначался для верификации программ.

В языке Lucid все значения, даже константные, являются потоками, причем числовые константы обозначают потоки, составленные из их повторений. Обычные арифметические операции выполняются для потоков покомпонентно. Пусть есть потоки $X = (x_0, x_1, \ldots, x_n, \ldots)$ и $Y = (y_0, y_1, \ldots, y_n, \ldots)$, тогда *next* $X = (x_1, x_2, \ldots, x_{n+1}, \ldots)$ и X *fby* $Y = (x_0, y_0, y_1, \ldots, y_{n-1}, \ldots)$. Потоки можно определять рекурсивно. Потоки языка Lucid являются конечными,

если в качестве их хвоста использовать специальное значение *eod* (End Of Data).

В процессе развития в языке Lucid произошла смена базовых операций *fby* и *next* на операции запроса (querying) текущего индекса и операции, индексирующей (navigating) поток X потоком Y. Новые операции являются фундаментальными для содержательного (intensional) программирования, основанного на интенсиональной логике.

Современная версия языка Lucid называется Indexical Lucid и поддерживает многомерные потоки и операции над их отдельными именованными размерностями. Также язык Lucid был расширен за счет операций над самими размерностями и функциями.

В дальнейшем язык Lucid был расширен в нескольких направлениях. Разновидности языка использовались для спецификации трёхмерных электронных таблиц, систем реального времени с помощью разновидности Lustre и реагирующих (reactive) систем с помощью Lucid Synchrone, взаимодействий агентов с помощью AIPL (Agent Intensional Programming Language). Для реализации этих и других диалектов языка Lucid была разработана единая среда GIPSY (General Intensional Programming System).

Одно из интересных расширений языка Lucid, разработанное в калифорнийской компании SRI International, получило название Granular Lucid (GLU) [2] и предназначалось для описания крупнозернистого параллелизма, где последовательные части задавались на языке Си. Программа на языке GLU состоит из текста программы языка Lucid с объявлениями типов и заголовками функций и языка Си, тела которых находятся в отдельном файле. Допустимы любые объявления типов языка Си, так как их использование ограничено аргументами и результатами импортируемых функций языка Си. Тем самым от языка Lucid язык GLU, по сути, отличается более гранулированными элементарными операциями.

2. ID (IRVINE DATAFLOW)

Язык параллельного программирования общего назначения Id, первое упоминание о котором датируется 1978 г. [3], был в основном спроектирован в Массачусетском технологическом институте (MIT), где была разработана версия Id Nouveau [4]. Последняя рассматриваемая ниже ревизия языка Id 90.1 [5] датируется 1991 г.

Язык Id всё ещё остаётся исследовательским языком, развивающимся в направлении лучшего выражения недетерминированных вычислений, вво-

да-вывода и управления ресурсами. Для языка Id существует функционирующий компилятор для потоковой архитектуры с реализацией передачи маркера Массачусетского технологического института и вычислительной системы Monsoon фирмы Motorola.

Язык Id предназначается для программирования потоковых и других параллельных архитектур и содержит три синтаксически разделяемых слоя.

1. Основная часть языка, являющаяся чисто функциональным языком с нестрогой семантикой вычисления.
2. Расширение с помощью I-структур, сохраняющее детерминизм основной части, но теряющее гарантируемую прозрачность ссылок (referential transparency).
3. Расширение с помощью M-структур и ввода-вывода, которое может приводить к недетерминированным вычислениям.

Функциональная часть языка Id поддерживает функции высшего порядка, средства для явного задания отложенных вычислений, сопоставление с образцом и явно выделенные циклические выражения. Также в языке используются статическая полиморфная система типов в стиле Милнера с перегрузкой, определяемые пользователем типы, списки и массивы с развитыми средствами их генерации (comprehension).

Язык Id поддерживает переменные типов для обозначения зависимостей между компонентами полиморфных агрегатных типов. Допускается определение алгебраических типов (или несвязных объединений), которые позволяют задавать многие типы, такие как булевский тип, встроенные в других языка программирования. Языком Id также поддерживаются абстрактные типы данных.

Компоненты структуры данных языка Id могут иметь функциональную семантику, семантику I-структуры или семантику M-структуры. В случае функциональной семантики компоненты создаются и инициализируются одновременно, тогда как компоненты I-структуры и M-структуры создаются пустыми и инициализируются позже.

Значение I-структуры может инициализироваться только один раз и, в случае повторной инициализации, возникает исключение, делающее всю программу несостоятельной. Читать значение I-структуры можно произвольное число раз, и попытка прочитать значение пустой I-структуры откладывается до её инициализации.

Значение M-структуры может меняться произвольное число раз, но изменение возможно только, если M-структура является пустой. Попытка изменения непустой M-структуры приводит к исключению, делающему всю программу несостоятельной. Чтение непустого значения M-структуры

делает её снова пустой, откладывая все другие, возможно, параллельные чтения этого значения, произвольное из которых будет вновь возобновлено при последующем присваивании значения этой М-структуры.

Типичное применение М-структур заключается в использовании несколькими параллельными вычислениями одного неразделяемого ресурса. Каждое вычисление читает значение ресурса, задаваемого М-структурой, работает с ним и записывает его обратно. Семантика доступа М-структуры гарантирует эксклюзивный доступ к ресурсу.

В целях машинно-зависимых оптимизаций в языке Id можно ограничивать степень параллелизма циклов и других конструкций языка, исполняемых параллельно, явным образом, путем указания максимального количества одновременно исполняемых итераций. Эти ограничения могут стать причиной зависания вычислений с обратными (циклическими) зависимостями, которые не запрещаются языком.

3. VAL

Язык Val (Value Algorithmic Language) был разработан в 1979 г. [6] в Массачусетском технологическом институте (MIT) и предназначался для статической потоковой архитектуры, что привело к отсутствию типичных для динамической потоковой архитектуры возможностей, таких как рекурсия. В то же время язык не содержал машинно-зависимых элементов, включая встроенный ввод и выход. Являясь небольшим исследовательским языком, язык Val демонстрирует возможность использования языка программирования, принадлежащего высокому уровню, в качестве потокового.

В отличие от большинства других функциональных языков, язык статически типизирован (применяется структурная эквивалентность) разветвленной системой типов и синтаксически напоминает язык программирования Pascal. В частности, в языке содержится «оператор присваивания», который семантически является однократным связыванием (single-assignment) значения с именем. Также язык Val в явном виде содержит несколько форм циклических выражений, одна из которых может использоваться для задания явного параллелизма. Функции языка Val, как и другие выражения, могут возвращать несколько значений.

Каждый тип языка Val дополнительно содержит одно выделенное ошибочное значение, что гарантирует независимость результата от порядка вычисления. Стратегия вычислений языка Val в первую очередь определяется соображениями эффективности и простоты, сочетая ленивые и строгие

вычисления. Ввиду отсутствия в языке побочных эффектов и наличия выделенных ошибочных значений, язык Val хорошо подходит для строгих вычислений. Однако в некоторых конструкциях, таких как выражения выбора и циклы с условием, где сначала вычисляется управляющее выражение, используется ленивая стратегия для уменьшения ненужных вычислений. Параметры функции в языке Val всегда вычисляются до её вызова.

Смешанная стратегия вычислений нарушает эквивалентность отдельных преобразований. Например, в общем случае нельзя заменить выражение выбора эквивалентным вызовом функции, так как выражению выбора может потребоваться только часть используемых в нём значений, а вызов функции должен вычислить их все. Также возможность языка Val обнаруживать и исправлять ошибочные значения, в отличие, например, от функционального языка FP, сохраняющего $\perp$ (ошибочное состояние), приводит к некорректности некоторых преобразований программ, свойственных функциональным языкам, таких как перенос функционального вызова на ветви условного выражения.

4. POST

Экспериментальный язык Post появился в 1981 г. в качестве одного из результатов кандидатской диссертации [7]. Язык Post более точно, чем языки Val и Sisal, отражает особенности потоковых архитектур, но его программы сложнее для понимания. В языке Post предложено несколько новых идей развития потоковых языков:

- возможность влиять на используемую стратегию вычисления;
- смешанно-синхронные структуры данных, являющиеся частично синхронными, а частично — асинхронными;
- взаимодействие между вычислениями для прекращения ненужных вычислений, что нехарактерно для чистых функциональных языков.

Язык Post не был полностью реализован за исключением прообраза компилятора, преобразующего программный текст в потоковый граф и, затем в инструкции динамической потоковой машины со специфическими особенностями, которая тоже так и не была создана.

5. SISAL

Язык Sisal (Streams and Iterations in a Single Assignment Language) является развитием языка Val, по большей части разработанным в Ливерморской национальной лаборатории имени Лоренца (LLNL). Первая основная версия языка Sisal 1.2 была зафиксирована в 1985 г. [8]. Для этой версии разработаны оптимизирующие компиляторы под различные машинные архитектуры. Последующие «официальные» версии языка Sisal 2.0 [9] и Sisal 90 [10, 11], разработанные в 1991 и 1995 гг. соответственно, так и не были реализованы. Для языка Sisal 90 не было создано даже строгой спецификации. Языки Sisal 1.2, Sisal 2.0 и Sisal 90 синтаксически не совместимы между собой. В то же время язык Sisal 90 находится значительно ближе, чем язык Sisal 2.0, к языку Sisal 1.2.

Язык Sisal является универсальным языком общего назначения, но с ярко выраженной научной направленностью, позиционирующийся как замена языка Fortran. Синтаксически язык Sisal, как и язык Val, напоминает язык Pascal. Язык Sisal, в отличие от языка Val, разрешает использование рекурсии, так как был разработан для более прогрессивных динамических потоковых архитектур.

Уже в версии Sisal 1.2 в язык были введены тип потока для облегчения организации машинно-независимого ввода-вывода и поддержка модульности на уровне импорта-экспорта отдельных функций. Версия Sisal 2.0 расширяла язык функциями высшего порядка, множествами типов, более полной поддержкой модульности, алгебраическими матрицами и покомпонентными операциями над массивами и потоками.

Версия Sisal 90 содержала все нововведения версии Sisal 2.0 за исключением типа алгебраических матриц. Покомпонентные операции над массивами и потоками в языке Sisal 90 были позаимствованы из языка Fortran 90. В языке Sisal 90 появилась возможность задавать пользовательские редукции и константы времени исполнения, не нарушающие принцип математической чистоты функций в пределах одного запуска программы.

Следующий концептуальный виток развития языка Sisal 90 был совершен в версии Sisal 3.0 [12], разработанной в Институте систем информатики (ИСИ) имени А. П. Ершова СО РАН. Нововведения языка Sisal 3.0 заключаются в возможности задавать отдельные части программы на императивном языке Си, расширенной поддержке модульности программ, возможности их препроцессинга и аннотирования для упрощения оптимизирующих преобразований.

Для последней версии языка Sisal 3.1, находящейся в разработке в ИСИ, впервые со времён языка Sisal 90 был строго формализован синтаксис его функциональной части. Функциональная часть языка была усилена за счёт возможности переопределения операций и перегрузки вызовов функций и редукций. Синтаксис и семантика некоторых конструкций языка Sisal 90 были изменены для повышения читаемости и упрощения разбора в существующем трансляторе переднего плана языка Sisal 3.1.

6. ПИФАГОР

Язык программирования Пифагор [13], разработанный в Красноярском Государственном Техническом Университете (КГТУ) в 1995 г., предназначен для разработки переносимых параллельных программ, управление вычислениями в которых осуществляется по готовности данных. Язык Пифагор ориентирован на непосредственное описание информационного графа, что привело к синтаксису языка, несколько отличающемуся от общепринятого синтаксиса. В частности, не поддерживаются инфиксные бинарные операции, что снижает читаемость программы. Ввиду экспериментальной природы языка Пифагор, в нем отсутствуют конструкции, повышающие технологичность разработки программ, такие как модульное построение программ, раздельная трансляция и согласование со стандартными библиотеками.

Существующие на данный момент версии языка Пифагор и исполнительной системы поддерживают только динамическую типизацию. Язык поддерживает простые типы и три вида списков, которые могут быть вложены друг в друга. Списки бывают последовательными, параллельными и задержанными. Каждый тип языка дополнительно содержит несколько различных значений, обозначающих разные виды ошибок вычисления. Альтернативные вычисления реализуются через механизм задержанных списков. Особенности модели вычислений и синтаксиса языка Пифагор накладывают отпечаток на методы и стиль программирования. Отсутствие операторов цикла позволяет писать потоковые программы без синхронизации перед входом в циклический фрагмент, но в то же время приводит к необходимости использования рекурсии.

Динамическая типизация позволяет получать интересные эффекты, расширяющие возможности языка. В частности, сочетание механизма перегрузки функций и динамических типов, определяемых пользователем, обеспечивает написание эволюционно расширяемых параллельных программ. В

языке Пифагор реализован механизм перегрузки функций, позволяющий гибко и безболезненно расширять уже разработанную программу. Так как в языке отсутствует строгая типизация, все функции с одинаковыми именами становятся неразличимы (имеют одинаковую сигнатуру). Поэтому вместо выбора одной из перегруженных функций осуществляется их одновременное выполнение. Результат возвращается в виде параллельного списка.

ЗАКЛЮЧЕНИЕ

Рассмотрены общие определяющие качества потоковых языков программирования и характерные особенности некоторых распространенных и экспериментальных потоковых языков.

СПИСОК ЛИТЕРАТУРЫ

1. **Ashcroft E. A. and Wadge W. W.** Lucid: A formal system for writing and proving programs // SIAM J. on Computing. — 1976. — Vol. 5, No. 3. — P. 336–354.
2. **Jagannathan R. and Faustini A. A.** The GLU programming language. — Menlo Park, CA, 1990. — (Tech. Rep. / SRI International, Computer Science Laboratory; SRI-CSL-90-11).
3. **Arvind, Gostelow K. P. and Plouffe W.** An asynchronous programming language and computing machine. — Irvine, CA, 1978. — (Tech. Rep. / Univ. of California, Department of Information and Computer Science; 114a).
4. **Arvind, Nikhil R. S. and Pingali K. K.** Id Nouveau reference manual. — Cambridge, MA, 1986. — 64 p. — (Tech. Rep. / Massachusetts Institute of Technology, Laboratory for Computer Science, Computation Structures Group; Memo-265).
5. **Nikhil R. S.** Id language reference manual (version 90.1). — Cambridge, MA, 1991. — 54 p. — (Tech. Rep. / Massachusetts Institute of Technology, Laboratory for Computer Science, Computation Structures Group; Memo-284-2).
6. **McGraw J. R.** Val language, description and analysis. — Livermore, CA, 1980. — 51 p. — (Tech. Rep. / Lawrence Livermore National Laboratory; UCRL-83251, Rev. 1).
7. **Ravishankar C. V.** Post: a language for dataflow programming. — Ph.D. thesis. — Madison: University of Wisconsin, Computer Sciences Department, 1987. — 213 p.
8. **McGraw J. R.** Sisal: Streams and iterations in a single assignment language, Language Reference Manual, Version 1.2. / McGraw J. R., Skedzielewski S. K., Allan S. J., Oldehoeft R. R., Glauert J., Kirkham C., Noyce B. and Thomas R. — Livermore, CA, 1985. — (Tech. Rep. / Lawrence Livermore National Laboratory; M-146, Rev. 1).

9. **Böhm A. P. W.** The Sisal 2.0 reference manual / Böhm A. P. W., Cann D. C., Feo J. T. and Oldehoeft R. R. — Livermore, CA, 1991. — 128 p. — (Tech. Rep. / Lawrence Livermore National Laboratory; UCRL-MA-109098).
10. **Feo J. T.** Sisal 90 user's guide / Feo J. T., Miller P. J., Skedzielewski S. K. and Denton S. M. — Livermore, CA: Lawrence Livermore National Laboratory, Draft 0.96, 1995. — 80 p.
11. **Бирюкова Ю. В**. Sisal 90: Руководство для пользователя. — Новосибирск, 2000. — 84 с. — (Препр. / РАН. Сиб. Отд-е. ИСИ; № 72).
12. **Касьянов В. Н., Бирюкова Ю. В., Евстигнеев В. А.** Функциональный язык Sisal // Поддержка супервычислений и интернет-ориентированные технологии. — Новосибирск: ИСИ СО РАН, 2001. — С. 54–67.
13. **Legalov A. I., Kazakov F. A., Kuzmin D. A.** Description of parallel-functional programming language // Advances in Modeling & Analysis, Series A. — Brno, Czech Republic: AMSE Press, 1995. — Vol. 28, No. 3. — P. 1–17.

А. С. Тараскина

НЕЧЕТКАЯ КЛАСТЕРИЗАЦИЯ ПО МОДИФИЦИРОВАННОМУ МЕТОДУ *C*-СРЕДНИХ И ЕЕ ПРИМЕНЕНИЕ ДЛЯ ОБРАБОТКИ МИКРОЧИПОВЫХ ДАННЫХ

ВВЕДЕНИЕ

Во многих областях биомедицинских исследований экспрессию генов изучают с помощью ДНК-микрочипов [1]. Для анализа растущего объёма данных, полученных с помощью этой технологии, кластеризация становится практически необходимой [2].

Методы кластеризации [3, 4] делятся на иерархические и итерационные (методы разбиений).

Иерархические алгоритмы связаны с построением дендрограмм. В агломеративных алгоритмах перед началом кластеризации все объекты считаются отдельными кластерами, которые в ходе алгоритма объединяются. Вначале выбирается пара ближайших кластеров, которые объединяются в один кластер. В результате количество кластеров уменьшается на 1. Процедура повторяется, пока все классы не объединятся. На любом этапе объединение можно прервать, получив нужное число кластеров. Однако процедура иерархического кластерного анализа хороша для малого числа объектов и не годится для данных большого объёма из-за трудоемкости агломеративного алгоритма и слишком больших размеров дендрограмм.

В итерационных алгоритмах данные сразу разбиваются на несколько кластеров, число которых оценивается исходя из условий. Далее элементы перемещаются между кластерами так, чтобы был оптимизирован некоторый критерий, например, минимизируется изменчивость внутри кластеров [5].

Целью данной работы явилась разработка на основе нечеткого алгоритма *c*-средних нового алгоритма кластеризации, находящего близкое к оптимальному решение задачи кластеризации данных микрочипов.

1. АЛГОРИТМ НЕЧЁТКИХ *C*-СРЕДНИХ

Исходной информацией для кластеризации является матрица наблюдений $l \times n$

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ & & \dots & \\ x_{l1} & x_{l2} & \dots & x_{\ln} \end{bmatrix},$$

где l — число объектов, n — число признаков (наблюдений) для каждого объекта [6, 7].

Задача кластеризации состоит в разбиении множества объектов на группы (кластеры) «похожих» между собой объектов. В n-мерном метрическом пространстве признаков мерой «сходства» двух объектов будем считать расстояние между ними.

В данной работе применяется метод нечёткой кластеризации, позволяющий каждому объекту принадлежать с различной степенью нескольким или всем кластерам одновременно. Число кластеров c считается заранее известным.

Кластерная структура задаётся матрицей принадлежности ($c \times l$ матрица):

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1l} \\ m_{21} & m_{22} & \dots & m_{2l} \\ & & \dots & \\ m_{c1} & m_{c2} & \dots & m_{cl} \end{bmatrix},$$

где m_{ij} — степень принадлежности j-го элемента i-му кластеру.

Отметим, что матрица принадлежности должна удовлетворять следующим условиям:

0) $m_{ij} \in [0,1], i = \overline{1,c}, j = \overline{1,l}$,

1) $\sum_{i=1}^{c} m_{ij} = 1, j = \overline{1,l}$, т.е. каждый объект должен быть распределён между всеми кластерами,

2) $0 < \sum_{j=1}^{l} m_{ij} < l, i = \overline{1,c}$, т.е. ни один кластер не должен быть пустым или содержать все элементы.

Для оценки качества разбиения используется критерий разброса, показывающий сумму расстояний от объектов до центров кластеров с соответствующими степенями принадлежности:

$$J = \sum_{i=1}^{c} \sum_{j=1}^{l} (m_{ij})^w d(v_i, x_j) \text{ , где}$$

$d(v_i, x_j)$ — Евклидово расстояние между *j*-м объектом $x_j = (x_{j1}, x_{j2}, ..., x_{jn})$ и *i*-м центром кластера $v_i = (v_{i1}, v_{i2}, ..., v_{in})$,

$w \in (1, \infty)$ — экспоненциальный вес, определяющий нечёткость, размытость кластеров,

$V = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ & & \dots & \\ x_{c1} & x_{c2} & \dots & v_{cn} \end{bmatrix}$ — $c \times n$ матрица координат центров кластеров, элементы которой вычисляются по формуле $v_{ik} = \dfrac{\sum_{j=1}^{l} (m_{ij})^w x_{jk}}{\sum_{j=1}^{l} (m_{ij})^w}, k = \overline{1,n}$ (v).

Задачей является нахождение матрицы *M*, минимизирующей критерий *J*. Для этого используется алгоритм нечётких *c*-средних, в основе которого лежит метод множителей Лагранжа. Он позволяет найти локальный оптимум, поэтому для различных запусков могут получиться разные результаты.

На первом шаге матрица принадлежностей *M*, удовлетворяющая условиям 0)–2), генерируется случайным образом. Далее запускается итерационный процесс вычисления центров кластеров и пересчёта элементов матрицы степеней принадлежности:

$$m_{ij} = \frac{1}{(d_{ij})^{\frac{2}{w-1}} \sum_{k=1}^{c} \frac{1}{(d_{kj})^{\frac{2}{w-1}}}} \text{ при } d_{ij} > 0 \text{ и } m_{kj} = \begin{cases} 1, k = i \\ 0, k \neq i \end{cases} \text{при } d_{ij} = 0,$$

где $d_{ij} = d(v_i, x_j)$ для $i = \overline{1,c}, j = \overline{1,l}$.

Вычисления продолжаются до тех пор, пока изменение матрицы M, характеризующееся величиной $\left\| M - M^* \right\|^2$, где M^* — матрица на предыдущей итерации, не станет меньше заранее заданного параметра остановки ε.

Сходимость алгоритма нечётких c-средних доказана в [8].

Остановимся на выборе значения w — экспоненциального веса. Чем больше это значение, тем матрица принадлежности более размазанная и при $w \to \infty$ элементы примут вид $m_{ij} = \frac{1}{c}$, что является плохим решением, т.к. все объекты с одинаковой степенью распределены по всем кластерам. Теоретически обоснованного правила выбора веса пока не существует, и обычно устанавливают $w = 2$.

2. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Локальный минимум, полученный с помощью алгоритма нечётких c-средних, зачастую отличается от глобального минимума. Поиск глобального минимума функционала J не осуществим ввиду большого объема вычислений, но существуют алгоритмы, получающие решение, близкое к глобальному минимуму.

Нами был использован генетический алгоритм (ГА), основанный на генетических процессах биологических организмов: биологические популяции развиваются в течение нескольких поколений, подчиняясь законам естественного отбора и по принципу «выживает наиболее приспособленный». Программа также может развивать соответствующим образом закодированные решения, выбирая из них наиболее подходящее. Обычно ГА дают хорошие результаты для задач оптимизации многопараметрических функций, а именно такую задачу мы и решаем. Однако, как и другие методы эволюционных вычислений, они не гарантирует обнаружения глобального решения за полиномиальное время. ГА не гарантируют и того, что глобальное решение будет найдено, но они хороши для поиска «достаточно хорошего» решения задачи «достаточно быстро».

ГА работает с популяцией — совокупностью особей, которые представляют собой возможные решения данной задачи. Каждая особь оценивается степенью её приспособленности, что соответствует тому, насколько «хорошо» данное решение задачи. Наиболее приспособленные особи могут скрещиваться и производить потомство. В результате получаются новые особи, сочетающие в себе «хорошие» характеристики, полученные от родителей. Возможность скрещивания менее приспособленных особей меньше, поэтому признаки, которыми они обладали, будут элиминироваться из популяции в процессе эволюции. Итак, из поколения в поколение хорошие характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных особей приводит к тому, что исследуются наиболее перспективные участки пространства поиска. В конечном итоге, популяция будет сходиться к оптимальному решению задачи. Также существует возможность мутации особи, когда часть её характеристик случайным образом изменяется. Благодаря этому можно выйти из состояния локального оптимума, получить новое возможное решение.

3. ОПИСАНИЕ ПРОГРАММЫ

3.1. Данные

Для обработки могут использоваться два типа данных.

1. Микрочипы (microarray).

Данные, полученные в результате экспериментов с микрочипами, можно представить в виде матрицы X наблюдений, где в строках будут располагаться различные гены, а в столбцах — их уровни экспрессии в различных экспериментах.

В качестве расстояния между генами берётся Евклидово расстояние в n-мерном метрическом пространстве. Координаты центров кластеров находятся по формулам (v).

Если данные нормализованы (нулевой средний уровень экспрессии для каждого гена и единичное среднеквадратичное отклонение), то в результате кластеризации получаются группы генов со сходным профилем экспрессии. В противном случае в один кластер попадают гены с близкими значениями экспрессии на протяжении всех экспериментов.

2. Матрицы расстояний.

Полученные некоторым образом матрицы расстояний между объектами можно использовать для кластеризации этих объектов. В этом случае в качестве исходных данных имеется симметричная матрица для системы из l объектов:

$$D = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1l} \\ d_{21} & d_{22} & \dots & \\ & & \dots & \\ d_{l1} & d_{l2} & \dots & d_{ll} \end{bmatrix}, \text{ где } d_{ii} = 0, d_{ij} = d_{ji}, i, j = \overline{1,l}.$$

Естественно, что в качестве расстояний берутся элементы этих матриц. Непосредственные наблюдения являются «скрытыми». Центры кластеров в этом случае совпадают с некоторыми из заданных объектов. Координаты по методу *c*-средних не вычисляются, а новым центром *j*-го кластера объявляется *k*-я вершина, минимизирующая сумму $\sum_{i=0}^{l} m_{ji} d_{ki} \quad (v_d)$.

3.2. Реализация

В программе используется комбинация описанных выше алгоритмов (*c*-средних и генетического). В качестве члена популяции для микрочипов берётся массив координат центров кластеров, а для матриц расстояний — массив номеров элементов, выбранных в качестве центров.

Шаг 1. Случайным образом создаётся начальная популяция с заданным числом особей *n*.

Для этого генерируются матрицы принадлежностей, а по ним определяются соответствующие особи (формулы (v) и (v_d)).

Шаг 2. К каждой особи применяется метод *c*-средних, пока изменения на каждой итерации не станут меньше заданного параметра.

Шаг 3. Выбирается некоторое количество «элитных» особей с наименьшими значениями критерия.

Шаг 4. Производится скрещивание.

Методом рулетки (roulette-wheel selection) из популяции выбирается пара особей. Колесо рулетки содержит по одному сектору для каждого члена популяции. Размер сектора пропорционален соответствующей приспособленности, т.е. обратно пропорционален значению критерия. При таком от-

боре члены популяции с более высокой приспособленностью с большей вероятностью будут выбираться чаще, чем особи с низкой приспособленностью. После отбора для каждой пары с некоторой вероятностью происходит двухточечный кроссовер [9]. Случайным образом выбирается первая точка — целое число от 0 до $c \cdot p_{cross}$, где c — число кластеров, а p_{cross} — процент признаков, который потомок должен получить от одного из родителей. Вторая точка отстоит от первой на $c(1-p_{cross})$ позиций. Обе родительские структуры разделяются в этих точках. Затем соответствующие центральные сегменты меняются местами и вновь объединяются с концевыми. Получаются два генотипа потомков. Кроссовер может не произойти, тогда на следующую стадию переходят неизмененные особи. Элитные особи также переходят в новое поколение без изменений. Число пар рассчитывается так, чтобы в новом поколении было то же количество особей n.

Для наших типов данных сегмент соответствует некоторому числу подряд идущих строк в матрице координат центров или элементов массива номеров. Таким образом, при кроссовере частично изменяются центры кластеров, определяемые данной особью.

Шаг 5. Мутация.

Все особи, полученные на предыдущем шаге, за исключением элитных, подвергаются мутации. С некоторой вероятностью случайное число элементов особи меняется на произвольные (разумеется, в границах, определяемых условиями).

Шаг 6. Снова с помощью c-средних обрабатываются новые и мутировавшие особи.

Шаг 7. Из получившейся популяции элиминируются одинаковые организмы и вновь выбираются элитные.

Шаг 8. Переход на 4 шаг. Число переходов, т.е. жизненных циклов популяции, задаётся заранее.

Шаг 9. Наиболее приспособленная особь объявляется искомым решением задачи.

4. ДОПОЛНИТЕЛЬНЫЕ ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ.

4.1. Выбор параметра w

В работе [10] установлено, что значение $w = 2$ не подходит для данных, полученных с микрочипов. В нашей программе используются эксперимен-

тально установленные формулы для вычисления более подходящего значения, приведённые в вышеупомянутой работе.

Как было уже отмечено, при больших значениях w степени принадлежности становятся близки к $\frac{1}{c}$. Можно таким образом оценить граничное значение w_{ub}. Очевидно, значения m_{ij} зависят от расстояний между элементами и центрами кластеров. Центры кластеров близки к некоторым элементам (генам), поэтому можно предположить, что существует взаимосвязь результатов нечёткой кластеризации и коэффициента вариации cv для множества $Y_w = \left\{ d(x_i, x_j) \right\}^{\frac{2}{w-1}}, i \neq j = \overline{1,l}$, где $cv = \frac{\sigma(Y_w)}{\overline{Y_w}}$. По результатам экспериментов для нахождения w_{ub} было предложено уравнение $cv(Y_w) \approx 0.03n$, где n — размерность данных. Численное решение этого уравнения находится методом дихотомии.

В итоге, значение параметра выбирается следующим образом:

$$w = 1 + w_0, w_0 = \begin{cases} 1, w_{ub} \geq 10 \\ \frac{w_{ub}}{10}, w_{ub} < 10 \end{cases}.$$

4.2. Силуэт

Для оценки качества кластеризации можно использовать величину силуэта [11]. Допустим, ген x_i лежит в кластере C_r. При нечёткой кластеризации номер кластера определяется по максимальному значению степени принадлежности. Вычисляются значения $a(x_i) = \frac{1}{|C_r|} \sum_{x_j \in C_r} d(x_i, x_j)$ и $b(x_i) = \min \left\{ \frac{1}{|C_s|} \sum_{x_j \in C_s} d(x_i, x_j), r \neq s = \overline{1,c} \right\}$. Силуэт гена определяется как $s(x_i) = \frac{a(x_i) - b(x_i)}{\max(a(x_i), b(x_i))}$. Значение силуэта лежит в интервале $[-1; 1]$, если оно отрицательно, то ген считается плохо кластеризованным.

4.3. Входные данные и результаты

Данные для кластеризации считываются из файла, выбранного пользователем. Тип данных (микрочипы, матрица расстояний) определяется самой программой.

Столбцы отделены друг от друга символом табуляции. Файл не должен содержать пропущенных значений.

Микрочипы.

Первая строка содержит названия всех столбцов. Каждая последующая – названия генов (один или более столбцов) и значения экспрессии для всех экспериментов.

Матрица расстояний.

Первая строка: заголовок (строка без символов табуляции) и названия столбцов. Последующие строки: название, совпадающее с названием соответствующего столбца, и данные. Матрица значений симметрична, на диагонали – нули.

Программа также может работать с матрицей сходства, преобразуя её в матрицу расстояний.

Параметры алгоритма.

Общие:

– число кластеров,

– параметр остановки, определяющий точность вычислений,

– экспоненциальный вес.

Метод *c*-средних:

– число итераций — запусков программы со случайными начальными данными с выбором наилучшего результата.

Генетический алгоритм:

– число особей в популяции,

– число жизненных циклов популяции,

– число элитных особей,

– частота мутаций,

– вероятность кроссовера в процессе скрещивания,

– процентное соотношение признаков в кроссовере.

Результаты кластеризации частично выводятся в окне программы в виде списка элементов по кластерам со степенью принадлежности выше порогового значения, которое можно изменять.

Сохранённый файл с результатами содержит:

– параметры алгоритма,

– список генов по кластерам со степенью принадлежности выше $\frac{1}{c}$,

– матрицу принадлежностей,

– координаты центров кластеров,

– значения силуэтов, если они были вычислены пользователем.

5. ТЕСТОВЫЙ ПРИМЕР И СРАВНИТЕЛЬНЫЙ АНАЛИЗ РЕЗУЛЬТАТОВ

Работа алгоритма была проверена на наборах данных, полученных в экспериментах по изучению клеточного цикла, которые можно найти на сайте [12].

Для кластеризации взяты нормализованные значения экспрессии для генов, участвующих в регуляции клеточного цикла, которые измерялись с периодичностью в 1 час. Мы провели разбиение генов на 5 кластеров по числу стадий клеточного цикла. Для каждого гена соответствующая стадия, а значит и кластер, были предсказаны с помощью алгоритма иерархической кластеризации [13]. Если предположить, что подобное предсказанное распределение точно, то отношение максимального числа генов одной стадии, попавших в один кластер к общему числу генов данной стадии, характеризует точность кластеризации с помощью нашего алгоритма. Результаты для двух наборов данных (47 и 27 измерений для каждого гена) представлены ниже:

Набор данных	Стадии	Номера кластеров					Отношение
		1	2	3	4	5	
1	G1/S	13	0	0	186	4	0.916256
	S	134	1	0	74	0	0.641148
	G2	33	128	51	0	20	0.551724
	G2/M	3	99	64	0	102	0.380597
	M/G1	0	8	2	1	175	0.94086
2	G1/S	6	1	0	118	23	0.797297

	S	1	12	0	29	114	0.730769
	G2	19	70	39	4	29	0.434783
	G2/M	68	80	58	2	3	0.379147
	M/G1	6	1	0	118	23	0.723684

Видно, что для некоторых стадий результаты согласуются с достаточно высокой точностью, а для некоторых – нет. Одной из причин может быть сходство профилей экспрессии у генов близких стадий (G2, G2/M). Также вполне вероятно, что предварительное распределение иерархическим алгоритмом отличается от истинного.

Ещё одно сравнение мы провели для генов, стадии активности которых были определены и описаны в различных работах по изучению клеточного цикла методами, отличными от кластеризации. Результаты представлены в таблице.

Набор данных	Стадии	Номера кластеров				Отношение
		1	2	3	4	
1	G1/S + G1	16	0	2	0	0.888889
	S	7	17	0	0	0.708333
	G2	0	0	1	5	0.833333
	G2/M	0	1	12	10	0.521739
2	G1/S + G1	0	12	2	2	0.75
	S	1	5	14	0	0.7
	G2	5	0	0	1	0.833333
	G2/M	2	0	0	13	0.866667

ЗАКЛЮЧЕНИЕ

Нечёткая кластеризация по методу *c*-средних — это удобный подход для выделения генов, тесно связанных с заданными кластерами. Применяя его в комбинации с генетическим алгоритмом, можно найти решение задачи кластеризации, близкое к оптимальному.

Нами была написана программа для кластеризации, в которой реализованы вышеизложенные методы. Дополнительно в программе присутствует возможность автоматического определения значения параметра размытости кластеров, подходящего для конкретного типа данных, и оценки качества кластеризации.

Также с помощью нашей программы можно осуществить разбиение объектов на группы, зная только попарные расстояния между ними, не задумываясь о координатном представлении этих объектов.

Программа реализована в среде Microsoft Visual Studio и доступна по адресу: http://biorainbow.com/fuzzyclustering/

СПИСОК ЛИТЕРАТУРЫ

1. Lockhart D. J. et al. Expression monitoring by hybridization to high-density oligonucleotide arrays // Nat. Biotechnol. — 1996. — Vol. 14. — P. 1675–1680.
2. Golub T.R. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring // Science. — 1999. — Vol. 286 (5439) — P. 531–537.
3. Anderberg, M. R. Cluster Analysis for Applications. Academic Press, New York, NY, 1976
4. Hartigan J. Clustering Algorithms. Wiley, New York, NY, 1975.
5. http://www.statsoft.ru/home/textbook/modules/stcluan.html
6. Штовба С.Д. Введение в теорию нечетких множеств и нечеткую логику, гл.12.
7. Höppner F., Klawonn F., Kruse R., Runkler T. Fuzzy Cluster Analysis, Wiley,1999.
8. Bezdek, J.C. Pattern Recognition With Fuzzy Objective Functional Algorithms. Plenum Press, New York, 1981.
9. Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Mass., 1989.
10. Dembele D., Kastner P. C-means method for clustering microarray data // Bioinformatics. — 2003. — Vol. 19(8). — P. 973–980.
11. Rousseeuw J.P. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis // J. Comp. Appl. Math. — 1987. — Vol. 20 — P. 53–65.
12. http://genome-www.stanford.edu/Human-CellCycle/Hela/
13. Whitfield M. L. et al. Identification of Genes Periodically Expressed in the Human Cell Cycle and Their Expression in Tumors // Mol. Biol. Cell. — 2002. — Vol. 13 — P. 1977–2000.

Д.В. Шкурко

ОТКАЗОУСТОЙЧИВОСТЬ В РАСПРЕДЕЛЕННЫХ СЕТЯХ: ПРОБЛЕМЫ КОНСЕНСУСА

1. ПРОБЛЕМЫ СОГЛАШЕНИЯ

Распределенной системой будем считать некоторое множество активных компонентов системы — *процессов*, взаимодействующих посредством компонентов связи. Отказоустойчивость распределенной системы определяется возможностью функционирования системы, несмотря на отказы ограниченного числа ее компонентов. Используемые для этого алгоритмы связаны с взаимодействием различных ее частей. Одной из фундаментальных проблем является соглашение о значении каких-либо входных данных. Например, управление базой данных в распределенных системах требует достижения соглашения относительно совершения (commit) или отбрасывания (abort) транзакции. Другим примером служит определение значения, считываемого датчиками. Еще один образец применения дают алгоритмы синхронизации часов, если каждый процесс не имеет доступа к централизованной службе синхронизации времени.

Естественным решением проблемы является следующий сценарий. Процессы сообщают о своем предложении и затем выбирают значение, встречающееся чаще всего. Этот метод хорошо работает, если нет ошибок, но не приемлем, если процессы могут сообщать разные данные различным процессам и влиять таким образом на правильные процессы, заставляя их принимать разные решения.

Простая постановка задачи состоит в соглашении о значении одного бита. Предположим, что имеется распределенная система, состоящая из фиксированного множества процессов, часть которых или изначально некорректны или могут давать сбой во время исполнения алгоритма. Каждый процесс i имеет входную бинарную переменную x_i. Задача *консенсуса* состоит в достижении соглашения относительно некоторого значения y. Более точно, требуется, чтобы все корректные процессы i когда-нибудь закончили исполнение алгоритма со значением переменной $y_i = y$. Это требование называется требованием *соглашения*.

Значение y в общем случае будет зависеть от начальных значений x_i, в противном случае имеется тривиальное решение: каждый процесс

присваивает $y_i = 0$ и останавливается. Рассматриваются несколько требований зависимости y от значений x_i в порядке усиления.

- Нетривиальность. Для любого $y \in \{0, 1\}$ должно существовать допустимое исполнение, ведущее к соглашению с результатом y. Допустимые исполнения могут иметь ограничения на количество ошибок, на тип системы (синхронная или асинхронная) и т.п.
- Слабое соглашение. Если $x_i = x \in \{0, 1\}$ для всех i, то $y = x$ при условии, что во время исполнения не было ошибок.
- Сильное соглашение. Если $x_i = x \in \{0, 1\}$ для всех корректных i, то $y = x$.

Требования зависимости называют еще требованием *обоснованности (validity).*

Имеются две другие проблемы, тесно связанные с алгоритмами консенсуса и которые широко рассматривались в литературе. Первая проблема, называемая проблемой *соглашения византийских генералов* (коротко *проблемой генералов*) или проблемой *отказоустойчивой рассылки*, предполагает наличие выделенного процесса ("генерала" или "отправителя"), который пытается разослать входное значение x всем остальным. Как и раньше все корректные процессы должны достичь соглашения при одном условии зависимости результата от x.

- Слабая зависимость: $y = x$, если во время выполнения алгоритма не было ошибок.
- Сильная зависимость: $y = x$, если "генерал" работает корректно.

В дальнейшем, если не оговорено обратное, используется строгая зависимость.

Вторая проблема будет рассмотрена в разд. 3. Эта проблема рассматривалась в ранней работе [1] и легко сводится к проблеме соглашения генералов [2]. Поэтому эта проблема практически не встречается в более поздних работах.

2. МОДЕЛЬ ВЫЧИСЛЕНИЙ

Решения задач соглашений, которые могут быть получены, зависят существенно от предположений, сделанных относительно моделей вычислений и взаимодействий и относительно допустимых ошибок/сбоев, устойчивость к которым требуется от решения. Далее предполагается, что число процессов фиксировано и равно N. Протокол называется *t-устойчивым,* если он корректно работает при условии, что не более t процессов дают сбои в течение работы алгоритма (процесс также может

давать сбой, просто не участвуя в протоколе).

Рассматриваются несколько моделей ошибок.

Остановка с сигналом о сбое (fail-stop). Процесс прекращает работу и сообщает о своем сбое другим процессам [3]. В этой модели можно точно определить, сломался ли процесс или нет. Эта модель предоставляет самые удобные для программирования предположения относительно сбоев в системе [4].

Остановка (crash) происходит, если процессор преждевременно прекращает всю активность. Вплоть до остановки процессор работает корректно, а после остановки он полностью останавливает работу. Не предполагается, что процессор восстанавливается после остановки и включается в работу системы.

Потеря части сообщений (omission). Потери сообщений могут происходить как при отсылке, так и при отправлении сообщений [5].

Ошибка согласования времени (timing fault). Происходит, когда процесс заканчивает задачу раньше или позже указанного срока. Такие ошибки существенны в задаче синхронного старта, тесно связанной с задачей соглашения [6, 7].

Наиболее серьезными ошибками являются *византийские ошибки*, при этом не делается никаких предположений относительно поведения некорректного процесса. Например, он может посылать лишние или противоречивые сообщения, не работать какое-то время и т.д. Протокол, который позволяет обойти такие ошибки, называется *византийским протоколом.*

Чтобы показать устойчивость к произвольным ошибкам, требуется рассмотреть все возможные поведения процессора, включая случаи, когда некорректный процесс пытается помешать исполнению протокола. Это может показаться очень сильным требованием, однако в отсутствии точного описания ошибок принятие консервативного подхода является подходящим решением.

Предполагается, что система передачи сообщений полностью отказоустойчива и только процессы могут давать сбои. Далее предполагается, что каждый получатель может безошибочно определить отправителя сообщения и все сообщения доставляются без изменений и ошибок. Если не оговорено обратное, предполагается, что процессы взаимодействуют попарно и любые два процесса могут взаимодействовать непосредственно.

Конечно, в реальных системах система сообщений, так же как и

процессы, могут давать сбои. Поэтому сбой канала обмена сообщениями обычно приписывается одному из процессов на одном из концов канала. Таким образом, t-устойчивый протокол позволяет обойти до t ошибок процессов или каналов связи.

Делались попытки более точно характеризовать ошибки в каналах связи. Например, более точное описание отказоустойчивости в случае ошибок-остановок и в случае потерь сообщений с разделением ошибок процессов и ошибок каналов связи есть в [8]. Введение ошибок, связанных с потерей сообщений [5], тоже было сделано с целью, чтобы более аккуратно описывать ошибки при передаче сообщений.

Очень важным предположением о поведении системы является возможность обнаружения ошибки процесса: было отослано сообщение или произошел сбой при отсылке сообщения. В этом случае получатель имеет очень важное знание о некорректности отправителя. В модели с аккуратными часами и ограничением на время доставки сообщения это достигается посредством тайм-аутов (ср. [9]). Таким же образом ошибки при отсылке автоматически обнаруживаются в синхронной модели, в которой вычисления и обмен сообщениями происходит по шагам: каждый шаг состоит из рассылки сообщений процессом, получения всех сообщений, адресованных процессу и отосланных на текущем шаге, и локальных вычислений. Однако, обнаружение невозможно в полностью асинхронной модели, в которой не делается никаких предположений об относительной скорости процессов и о времени доставки сообщений, так как в этом случае нельзя отличить медленно работающий процесс от остановившегося. Этот момент во многом определяет разрешимость задач соглашения.

Далее, если не оговорено обратное, будут использоваться термины *синхронная* и *асинхронная* системы для различения этих двух крайних случаев. Предполагается, что в синхронной модели посылка сообщений происходит раньше, чем прием сообщений, и поэтому содержимое и адресаты рассылаемых сообщений не зависят от содержимого сообщений, полученного в текущем раунде.

Другим существенным предположением является наличие подписей. Предполагается, что автор подписанного сообщения может быть определен любым процессом, независимо от того, получено ли сообщение от автора (или опосредованно) и независимо от действий некорректных процессов (если автор корректный). Другими словами, если корректный процесс A получил сообщение от B, подписанное корректным процес-

сом C, то A может быть уверен в том, то C действительно отсылал сообщение и B не мог подделать. Наличие подписей также во многом определяет разрешимость задач соглашения.

Цифровые подписи или простой контроль четности могут быть использованы в качестве подписей в зависимости от приложения: достаточно, чтобы некорректный процесс не мог сгенерировать сообщение корректного. В случае ошибок остановок или потерь сообщений, очевидно, механизм подписей не нужен.

С точки зрения применения для оценки протоколов соглашения используются различные оценки сложности алгоритмов. Среди важных характеристик считаются *время*, необходимое для достижения соглашения, и *количество и длина сообщений*, отсылаемых всеми корректными процессами во время исполнения протокола. Все эти характеристики в общем случае зависят от того, какие сбои происходит и когда.

В синхронной системе время измеряется в количестве раундов обмена сообщениями. Предполагается, что в течение одного раунда каждый процесс имеет возможность обменяться сообщениями с любым другим. Относительно пересылаемых сообщений интересуются также общим числом пересланных битов и количеством пересланных подписей (в случае протоколов с подписями).

3. СВЯЗЬ МЕЖДУ ПРОБЛЕМАМИ СОГЛАШЕНИЯ

Кроме задачи соглашения генералов рассматривалась задача *взаимной согласованности (interactive consistency)*. Эта проблема состоит в построении общего вектора $\mathbf{y}$ и для нее рассматривались следующие требования к зависимости компонент вектора от начальных значений x_i.

- Слабая зависимость: для всех j $\mathbf{y}_j = x_j$, если во время исполнения алгоритма не было ошибок.
- Сильная зависимость: для всех корректных процессов j $\mathbf{y}_j = x_j$.

Видно, что проблема соглашения генералов является специальным случаем проблемы взаимной согласованности, в которой значение только одного процесса представляет интерес, т.е. решение задачи взаимной согласованности дает решение задачи соглашения генералов. С другой стороны N параллельных исполнений протокола задачи соглашения генералов дают протокол для задачи взаимной согласованности.

Задачу консенсуса сложнее напрямую сравнивать с другими задачами соглашения. С одной стороны, она может быть решена с помо-

щью задачи взаимной согласованности выбором самого часто встречающегося значения в результирующем векторе. При условии, что число некорректных процессов меньше $N/2$ получается решение сильного консенсуса, в противном случае получается решение слабого консенсуса. С другой стороны, для сведения задачи соглашения генералов к задаче консенсуса требуется дополнительный раунд на рассылку начального значения выделенным процессом. Несложно показать, что построенный таким образом протокол решает задачу соглашения генералов. Требование соглашения выполняется тривиально, а требование зависимости получается с помощью следующего замечания. Если генерал корректный, то в начале протокола консенсуса все корректные процессы будут иметь одинаковое входное значение и по требованию зависимости для задачи консенсуса все корректные генералы в качестве финального значения примут начальное значение.

Многие алгоритмы соглашения генералов имеют в качестве первого шага рассылку начального значения и имеют в качестве встроенного алгоритм консенсуса. Однако встроенный алгоритм не всегда решает полную задачу консенсуса. Например, решение задачи консенсуса из [10] использует тот факт, что наличие разных входных значений у всех процессов после рассылки выделенным процессом означает некорректность выделенного процесса и задача консенсуса решается с ограничением в $t-1$ для числа некорректных процессов. В результате модификация алгоритма соглашения для решения задачи консенсуса требует даже большего числа раундов: $2t+4$ вместо $2t+3$.

Похожие соображения работают и для слабых версий соглашений. Очевидно, что сильные версии дают решения слабых версий задачи, но нет явной схемы преобразования решения слабого соглашения к решению сильного соглашения.

Более того, для “приблизительного” соглашения в слабом случае решение существует, а для сильного нет [11]. См. также разд. 4.

4. ПРИБЛИЗИТЕЛЬНЫЕ И НЕТОЧНЫЕ СОГЛАШЕНИЯ

Кроме задач точного консенсуса, рассматривались некоторые задачи приблизительного и неточного консенсуса. Предполагается, что каждый процесс i имеет входное действительное значение x_i.

Один из вариантов проблемы консенсуса позволяет показать, что слабая зависимость накладывает существенно меньше ограничений по сравнению с сильной зависимостью. Требование соглашения состоит в

том, чтобы в конце работы алгоритма все получили значение y_i, удовлетворяющие условию $|y_i - y_j| < \epsilon$ для любых корректных процессов i и j. Рассматриваются следующие требования зависимости.

- Сильная зависимость. Если все корректные процессы имеют одинаковое входное значение y, то все корректные процессы примут значение y.
- Слабая зависимость. Если все процессы имеют одинаковое входное значение y и во время исполнения протокола не происходит ошибок, то все процессы процессы примут значение y.

При условии, что множество входных значений ограничено, в работах [11, 2] показано, что проблема со слабой зависимостью разрешима при $N \leq 3t$, а с сильной зависимостью нет.

Другая более содержательная проблема консенсуса рассматривалась в работе [12]. Требование зависимости в этой проблеме усилено: выходное значение процесса должно лежать между входными значениями корректных процессов. Решение этой задачи основано на обмене текущими значениями, отбрасывании крайних значений (t максимальных и t минимальных) и вычислении некоторой статистики (усреднения). На основе результатов [12] в [13] построен алгоритм синхронизации часов. Требование приблизительности консенсуса позволило построить решения в полностью асинхронных системах при условии $N > 5t$.

В работе [14] описано более слабая проблема неточного соглашения для применения в алгоритме синхронизации часов. Предполагается, что каждый процесс i имеет входное действительное значение x_i, удовлетворяющее требованию начальной точности $\max_{i,j \in G} |x_i - x_j| \leq \delta$ и начальной аккуратности $\max_{i \in G} |x_i - \bar{x}|$. G — множество корректных процессов. Требуется улучшить начальную точность (в применении к синхронизации часов — ограничить разброс локальных часов).

Невозможность решения задачи неточного и приблизительного соглашения при $N \leq 3t$ есть в работе [15].

Теорема 1. *Задачи приблизительного и неточного соглашения имеют решения без использования подписей тогда и только тогда, когда* $N > 3t$.

5. РАЗРЕШИМОСТЬ ЗАДАЧ СОГЛАШЕНИЯ

5.1. Синхронные системы

Базовой проблемой при построении алгоритмов соглашения является вопрос о существовании решения. По соображениям, изложенным в разд. 3, t-устойчивые решения проблем консенсуса и взаимной согласованности имеют решения тогда и только тогда, когда есть t-устойчивое решение задачи соглашения генералов. Поэтому можно ограничиться рассмотрением только задачи соглашения генералов.

В синхронном случае в [1, 2] показано, что ограничений для t нет.

Теорема 2. *В синхронных системах с использованием подписей можно построить протокол для решения t-устойчивой сильной (слабой) задач соглашения генералов при $t \leq N$.*

На рис. 1 представлено решение с использованием подписей, оно проще и эффективнее первого решения [1, 2]. Впоследствии это решение было улучшено в [16, 17].

Доказательство корректности алгоритма достаточно простое. Если G корректный, то все корректные процессы могут извлечь только одно значение m, подписанное G. Если процесс извлек первое или второе значение во время раунда $i < t + 1$, то все корректные процессы тоже извлекут это значение в раунде $i + 1$. Наконец, если корректный процесс извлек второе значение в раунде $t + 1$, то среди сообщений, вызвавших его сделать это, есть сообщение от корректного процесса q, и этот процесс дает возможность извлечь это значение всем остальным корректным процессам в раунде $i \leq t + 1$.

Замечание. Видно, что если процесс G дал сбой и послал сообщение m не всем процессам в первом раунде, то этот сбой не отразится на конечном решении процессов, и процессы считают, то G некорректный, только если он отсылает противоречивые сообщения.

Без использования подписей решение существует при достаточно большой избыточности системы.

Теорема 3. *В синхронных системах можно построить протокол без использования подписей для решения t-устойчивой сильной (слабой) задач соглашения генералов тогда и только тогда, когда $N > 3t$.*

Невозможность для сильного соглашения при $N \leq 3t$ была получена в [1, 2], а для слабого соглашения — в [11]. Решение для $N > 3t$ было

получено в [1, 2].

На рис. 2 приведено описание алгоритма из [1, 2] в той форме, которая была представлена в работе [19] и потом многократно использовалась в более поздних работах [20, 21, 22, 23, 24, 25]. Доказательство корректности алгоритма опирается на следующие два факта.

1. Если процесс p корректный, то при переразметке дерева узел σp сохраняет свое значение, которое (вследствие корректности p) одно и то же в узлах σp всех корректных процессов.
2. Если большинство наследников узла при переразметке получили одно и то же значение в деревьях всех корректных процесов, то и сам узел получит одно и то же значение в деревьях всех корректных процесов.

Замечание. Построенный алгоритм легко модифицируется в алгоритм консенсуса. В этом случае корень дерева хранит входное значение процесса и имеет пустую метку, построение дерева продолжается $t + 1$ раунд. Правила переразметки те же самые.

5.2. Асинхронные системы

В полностью асинхронном случае решения не существует. Более того, даже существенное усиление ограничений на поведение некорректных процессов не позволяет получить решение [26].

Теорема 4. *В полностью асинхронном случае не существует протокола для решения задачи консенсуса с одним некорректным процессом и с допустимыми ошибками, являющимися остановками. Результат верен, если предполагать нетривиальность зависимости от входных данных.*

Идея доказательства теоремы состоит в следующем.

1. Изначально каждый процесс потенциально может принять любое решение, как 0, так и 1.
2. Чтобы выбрать окончательное решение процесс должен получать сообщения от других процессов, при этом полученные сообщения могут как склонять процесс к принятию решения, так и оставлять его дальше перед дилеммой.
3. Показано, что если процесс еще не склонился ни к какому решению, то существуют приемы таких сообщений, которые не помогут принять решение. Таким образом, возможно бесконечное

Предусловие: Выделенный процесс G имеет значение m, которое он должен разослать остальным
Постусловие: Все процессы достигают соглашения генералов
/* Запись (q, m) означает сообщение m подписанное процессом q */
/* Первый раунд */
1 Процесс G рассылает сообщение (G, m)
2 Остальные процессы получают сообщение и извлекают из него m.
3 $temp \leftarrow m$
/* 2 - $(t+1)$ раунды (процесс (q, m)) */
4 **for** $i \leftarrow 2$ **to** $t+1$ **do**
5 **if** *temp первое или второе извлеченное значение, подписанное* G **then**
6 разослать $(q, (G, temp))$ и разослать доказательство (i сообщений $(p, (G, m))$ или (G, m))
7 **end**
8 **if** *Если получены сообщения* $(p, (G, m))$ *или* (G, m) *от* i *разных процессов в предыдущих раундах* **then**
9 извлечь m
10 $temp \leftarrow m$
11 **end**
12 **end**
/* Решение */
13 **if** *Если извлечено только одно значение* m **then**
14 принять m
15 **else**
16 G некорректный
17 **end**

Алгоритм 1: Алгоритм соглашения с использованием подписей [18]

Предусловие: Выделенный процесс G имеет значение m, которое он должен разослать остальным

Постусловие: Все процессы достигают соглашения генералов

`/* Алгоритм состоит в построении помеченного дерева с последующей переразметкой. Узлы дерева соответствуют последовательностям` σ `имен процесов без повторений */`

`/* Первый раунд */`

1 G сохраняет в корне G дерева значение m и рассылает значение m остальным процессам. Остальные процессы получают сообщение m от G и тоже сохраняют в корне дерева G значение m

`/* 2 -` $(t+1)$ `раунды (процесс` (q, m)`) */`

2 **for** $i \leftarrow 2$ **to** $t+1$ **do**

3 Процесс q рассылает значения, сохраненные в листьях σ построенного дерева

4 Процесс q получает сообщения от других процессов и достраивает дерево: значение для узла σ, полученное от процесса $p \notin \sigma$, сохраняется в новом листе σp

5 **end**

`/* Решение. */`

6 Значения в листьях дерева сохраняются, а во внутренних узлах заменяются рекурсивно от листьев к корню. В качестве нового значения для внутреннего узла выбирается значение, которое встречается в более чем половине наследников (или просто выбирается 0, если 0 и 1 поровну).

7 В качестве финального значения берется значение в корне дерева.

Алгоритм 2: Алгоритм соглашения без использования подписей [19]

откладывание приема ключевых сообщений за счет приема сообщений, которые не влияют на прогресс всего алгоритма.

В работе [27] был предпринят тщатетльный разбор доказательства из [26]. Были разобраны следующие предположения относительно системы:

- процессы синхронные или асинхронные, т.е. гарантирована или нет скорость исполнения;
- доставка сообщений синхронная или асинхронная, т.е. гарантировано или нет время доставки;
- порядок получения сообщений произвольный или в порядке отправления;
- отсылка сообщений только одному процессу или широковещательная рассылка;
- атомарная обработка сообщений (получить–обработать–отправить) или неатомарная.

Были выделены 4 минимальных разрешимых случая консенсуса с ошибками-остановками:

1) синхронное отправление сообщений и синхронные процессы (стандартные предположения в синхронной модели);
2) синхронные процессы и получение сообщений в порядке отправления;
3) широковещательная рассылка и получение сообщений в порядке отправления;
4) синхронная доставка сообщений, широковещательная рассылка и атомарная обработка сообщений.

Новые разрешимые случаи не получили большого внимания исследователей, известно только о работе [28], посвященной разработке последнего разрешимого случая.

В работе [29] рассматривались другие ослабления синхронности системы. Были исследованы случаи частичной синхронности доставки сообщений и частичной синхронности работы процессов. Под частичной синхронностью подразумевается следующее. Предполагается, что есть оценка скорости передачи сообщений или скорости работы процессов, но либо она верна только начиная с некоторого момента, либо она не известна процессам и не должна быть явно использована в алгоритме. Для всех вариантов частичной синхронизации можно построить алгоритм, но при этом необходима дополнительная избыточность, например, для

остановок при частично синхронных обменах требуется $N > 2t$ а для произвольных ошибок с использованием подписей требуется $N > 3t$ (в полностью синхронном случае системе достаточно в обоих случаях $N > t$).

6. РАСШИРЕНИЕ МНОЖЕСТВА ВХОДНЫХ ЗНАЧЕНИЙ

В предыдущих главах рассматривались проблемы соглашения генералов и проблемы консенесуса для случая, когда входные значения лежат в двухэлементном множестве $\{0, 1\}$. Довольно часто, однако, требуется достичь консенсуса относительно значения из произвольного конечного множества. Наиболее простое решение состоит в соглашении относительно $\lceil \log V \rceil$ битов, где $V = |M|$ мощность множества входных значений, этот метод сохраняет число раундов, но увеличивает пропорционально количество сообщений.

Другой метод состоит в введении двух дополнительных раундов [30]. Этот метод работает при условии $N > 3t$ и не требует использования подписей. В первом руанде процессы обмениваются входными значениями m_i. Если процесс уверен, что большинство корректных процессов имеют одно и то же входное значение и некорректные процессы не могут привести к противоречивому решению другой корректный процесс (т.е. получено больше $\lfloor (N+t)/2 \rfloor$ одинаковых значений), то процесс отсылает во втором раунде это значение n_i, в противном случае отсылается значение по умолчанию n'. Во втором раунде корректный процесс, который получил сообщение n_i от корректного процесса (т.е. получил больше t сообщений n_i), использует в качестве входного значения для бинарного алгоритма значение 1, иначе 0. В конце работы алгоритма бинарного соглашения, если результат равен 0, принимается значение по умолчанию, иначе значение n_i. Доказательство достаточно простое. Принципиальными моментами являются

- требование, чтобы корректные процессы не принимали противоречивых решений в первом раунде;
- возможность при одинаковых входных данных принимать одинаковое (не n') значение после второго раунда.

Расширение множества входных значений имеет важные последствия для разрешимости проблемы консенсуса. В [25] показано, что если требовать, чтобы результат консенсуса был входным значением какого-либо корректного процесса, то необходимо, чтобы число процессов удо-

влетворяло неравенству $N > \max(3t, Vt)$, где $V = |M|$ мощность множества входных значений.

7. СОГЛАШЕНИЯ В СЕТЯХ С ПРОИЗВОЛЬНОЙ ТОПОЛОГИЕЙ

В [2] показано, что соглашение генералов при использовании подписей достигается, если подграф, индуцированный корректными процессами, связанный. В той же работе приведен пример класса графов, в которых можно построить задачи соглашения без использования подписей. В [31] дано описание полного класса графов, в которых разрешима задача соглашения без подписей.

Теорема 5. *Рассмотрим синхронную сеть с графом, имеющим связность по ребрам, равную k. Пусть граф состоит из N вершин-процессов, t из которых могут быть некорректными. Тогда задача соглашения генералов имеет решение тогда и только тогда, когда $N > 3t$ и $k > 2t$.*

Поскольку в теореме 5 требуется достаточно большая связность сети, чего сложно достичь на практике, то в [32] было определено ослабление требования соглашения: общее решение должно быть достигнуто для всех корректных процессов, за исключением некоторого числа X. Такое соглашение называется X-соглашением. Среди работ, связанных с построением X-соглашений, можно отметить [32, 33, 34, 35].

8. СЛОЖНОСТЬ ПРОТОКОЛОВ

8.1. Верхние оценки

t-устойчивый алгоритм из [1, 2] требует $t + 1$ раундов и требует обмена экспоненциальным от t числом битов. Первым алгоритмом с полиномиальным числом отсылаемых процессом битов был алгоритм из [36], улучшенный в [37, 10, 38]

Теорема 6 ([38]). *Пусть $N > 3t$. Существует протокол для решения задачи соглашения генералов без использования подписей, в котором пересылается $O(nt + t^3 \log t)$ битов и который работает $2t + 1$ раунда.*

Какое-то время даже считалось, что нельзя получить алгоритм с полиномиальным числом передаваемых битов и работающий меньше $2t$ раундов. Однако, в работах [39, 40, 19, 20, 41, 23, 24] были построены алгоритмы, которые сначала позволили сократить число раундов до $t+t/d$

при числе пересылаемых битов $O(n^d)$, а потом сократили число пересылаемых битов до полиномиального и число раундов до $t+1$ за счет небольшого уменьшения отказоустойчивости ($N > (3+\epsilon)t$). Наилучший результат представлен в работе [21].

Теорема 7. *Существует алгоритм решения задачи консенсуса (соглашения), без использования подписей с требованием сильной зависимости и оптимальной отказоустойчивостью ($N > 3t$), полиномиальным числом пересылаемых битов и работающий в течение $t+1$ раунда*

С использованием подписей и считая количество сообщений вместо количества пересылаемых битов получаем следующий результат.

Теорема 8.

1. *Существует t-устойчивый протокол для решения задачи соглашения с использованием подписей, в котором пересылается $O(nt)$ сообщений и который работает $t+1$ раунд.*
2. *Существует t-устойчивый протокол для решения задачи соглашения с использованием подписей, в котором пересылается $O(n+t^2)$ сообщений и который работает $O(t)$ раундов.*

Первая часть теоремы доказана в [18], а вторая часть доказана в [16].

С практической точки зрения эти оценки не очень хорошие, особенно оценка в $t+1$ раундов. В общем случае эта оценка не улучшается при условии, что число ошибок было t. Однако, в [42] был рассмотрен вопрос о раннем окончании работы алгоритма при условии, что число ошибок в действительности было $f < t$. Оказывается, что ответ зависит от того, требуется синхронизация при окончании или нет.

Пусть процесс *останавливается в течение r раундов,* если он корректный, не отсылает и не принимает сообщений после r раунда и выбирает выходное значение до $r+1$ раунда. Он *останавливается в r раунде*, если он останавливается в течение r раундов, но не останавливается в течение $r-1$ раунда. Протокол соглашения останавливается, когда останавливаются все корректные процессы. Если все процессы останавливаются в одном раунде, то соглашение называется *немедленным*, в противном случае оно называется *достижимым.* Таким образом, немедленное соглашение помимо принятия решения включает в себя еше и синхронизацию при принятии решения.

Теорема 9 ([43]). *Пусть $N > 3t$. Тогда существует t-устойчивый алгоритм без использования подписей, который решает задачу соглашения и решает достижимое соглашение в течение* $\min(2t+3, 2f+5)$ *раундов, где $f \leq t$ действительное число ошибок.*

Позднее этот результат был улучшен в уже упомянутых работах [39, 19, 20, 41, 23, 24, 21] и в [22] до алгоритмов с полиномиальным числом пересылаемых битов и числом раундов $\min(t+1, f+2)$.

Еще одним направлением улучшения алгоритмов являются алгоритмы с размером сообщения в 1 бит [44, 33, 34].

Теорема 10 ([45]). *Существует алгоритм решения задачи соглашения $N = O(t \log t)$, работающий $t+1$ раунд, требующий пересылки $O(Nt)$ 1-битовых сообщений.*

8.2. Нижние оценки

В работе [26] показано, что в случае t некорректных процессов решить задачу взаимной согласованности меньше чем за $t+1$ раунд невозможно, даже если считать, что ошибки являются простыми остановками. В [26] предполагается, что $N > 2t$, и поэтому доказательство работает только для случаев без использования подписей. Позднее этот результат был усилен в разных направлениях: в [36, 46] он был обобщен на случаи с использованием подписей, в [47] результат был обобщен на проблему слабого консенсуса, в [42] результат был обобщен на случай достижимого и немедленного соглашений.

Теорема 11.

1. *Любой t-устойчивый алгоритм для слабого консенсуса работает в худшем случае $t+1$ раунд.*
2. *Любой t-устойчивый алгоритм для задачи соглашения генералов работает в худшем случае $t+1$ раунд.*
3. *Любой t-устойчивый алгоритм для задачи немедленного соглашения генералов работает в худшем случае $t+1$ раунд, даже если никаких ошибок в действительности не было.*
4. *Любой t-устойчивый алгоритм для задачи достижимого соглашения генералов работает в худшем случае $\min(t+1, f+2)$ раунд, если в действительности есть $f \leq t$ некорректных процессов.*

В работе [16] доказаны следующие соотношения для числа пересылаемых сообщений:

Теорема 12.

1. *Общее число собщений и подписей в любом t-устойчивом протоколе соглашения генералов равно $O(nt)$.*
2. *Общее число сообщений в любом t-устойчивом протоколе соглашения генералов равно $O(n + t^2)$.*

Так как каждое сообщение может содержать более одной подписи, то вторая часть теоремы дает оценку для алгоритмов с использованием подписей, а первая часть дает нижнюю оценку для алгоритмов без использования подписей.

9. РАНДОМИЗИРОВАННЫЕ И ВЕРОЯТНОСТНЫЕ АЛГОРИТМЫ

Как показано в [26], детерминированного решения для консенсуса не существует. Как решение этой проблемы были предложены рандомизированные (использующие подбрасывание монеты) [48] и вероятностные (предполагающие ограничения на произвольность поведения асинхронной системы).

В качестве примера предположений относительно поведения системы в асинхронном случае можно привести предположения из [49]. Предполагается, что алгоритм организован в виде раундов (с тем ограничением, что гарантировано получение сообщений только $N-t$ процессов). Далее считается, что вероятность $R(q,p,t)$ получения сообщения процессом q от процесса p в течение раунда t ограничена снизу ненулевым числом $R(q,p,t) > \epsilon$. Еще одно предположение состоит в независимости получения сообщений от разных процессов. Следствием этих допущений является то, что вероятность получения сообщений от одного и того же множества процессов в последовательных раундах ненеулевая, т.е. поведение некорректных процессов не является полностью произвольным.

С точки зрения разрешимости проблемы с помощью рандомизированных и вероятностных алгоритмов в [49] доказана следующая теорема.

Теорема 13. *Для разрешимости консенсуса в асинхронном случае необходимо, чтобы в случае ошибок остановок было выполнено неравенство $N > 2t$ и в случае произвольных ошибок — $N > 3t$. Для вероятностных алгоритмов выполнение этих оценок является достаточным условием разрешимости.*

В работе [50] показано, что в случае произвольных ошибок выполнение оценки теоремы 13 является достаточным условием разрешимости с помощью рандомизированных алгоритмов.

Другим преимуществом рандомизированных алгоритмов является возможность получать соглашения быстрее, чем за $t+1$ раунд. В ра-

ботах [51, 30, 52] построены рандомизированные алгоритмы, которые заканчивают работу в среднем за постоянное число раундов, не зависящее от t.

10. ЗАКЛЮЧЕНИЕ

Активное изучение проблемы соглашения ведется до сих пор, и данный обзор включает небольшую часть результатов из данной области. В частности, описание рандомизированных алгоритмов достаточно краткое и не включает некоторые более поздние, но важные результаты из этой области. Совсем не затронуты исследования, связанные с модальными логиками "знаний". В расширенной версии предполагается описание работ по этим вопросам с акцентом на исследования по применению алгоритмов соглашения на практике.

СПИСОК ЛИТЕРАТУРЫ

1. **Lamport L.** Reaching agreement in the presence of faults // JACM. — 1980. — Vol. 27, N 2. — P. 228–234.
2. **Lamport L.** The byzantine generals problem // ACM Transactions on Programming Languages and Systems. — 1982. — Vol. 4, N 3. — P. 382–401.
3. **Schlichting R.** Fail-stop processors: an approach to designing fault-tolerant computing systems // ACM Transactions on Computing Systems. — 1983. — Vol. 1, N 3. — P. 222–238.
4. **Schneider F. B.** Synchronization in distributed programs // ACM Transactions on Programming Languages and Systems. — 1982. — Vol. 4, N 2. — P. 179–195.
5. **Perry K. J.** Distributed agreement in the presence of processor and communication faults: Tech. Rep. 84-610 / K. J. Perry: Cornell University, 1984.
6. **Burns, J. E.** The byzantine firing squad problem / J. E. Burns, N. A. Lynch.
7. The distributed firing squad problem / B. A. Coan, D. Dolev, C. Dwork, L. Stockmeyer // ACM Symposium on the Theory of Computing. — 1985. — P. 335–345.
8. **Hadzilacos V.** Connectivity requirement for byzantine agreement under restricted types of failures // Distributed Computing. — 1987. — Vol. 2, N 2. — P. 95–103.
9. **Lamport L.** Using time instead of timeouts for fault-tolerant distributed systems // ACM Transactions on Programming Languages and Systems. — 1984. — Vol. 6, N 22. — P. 254–280.
10. An efficient algorithm for byzantine agreement without authentication / D. Dolev, M. J. Fischer, R. Fowler, H. R. Strong // Information and Control. — 1982. — Vol. 52, N 3. — P. 257–274.
11. **Lamport L.** The weak general problems // JACM. — 1983. — Vol. 30, N 3. — P. 668–676.
12. Reaching approximate agreement in the presence of faults / D. Dolev, N. A. Lynch, S. S. Pinter et al. // JACM. — 1986. — Vol. 33, N 3. — P. 499–516.
13. **Lynch N. A.** A new fault-tolerant algorithm for clock synchronization // ACM Symposium on Principles of Distributed Computing. — 1984. — P. 75–88.

14. **Schneider S. M. F.** Inexact agreement: accuracy, precision, and graceful degradation // ACM Symposium on Principles of Distributed Computing. — 1985. — P. 237–249.
15. **Fischer M. J.** Easy impossibility proofs for distributed consensus problems // ACM Symposium on Principles of Distributed Computing. — 1985. — P. 59–70.
16. **Dolev D.** Bounds on information exchange for byzantine agreement // JACM. — 1985. — Vol. 32, N 1. — P. 191–204.
17. **Perry K. J.** An authenticated byzantine generals algorithm with early stopping: Tech. Rep. TR-84-620 / K. J. Perry, S. Toueg: Cornell University, 1984.
18. **Dolev D.** Authenticated algorithms for byzantine agreement // SIAM J. on Computing. — 1983. — Vol. 12, N 4. — P. 656–666.
19. Shifting gears: changing algorithms on the fly to expedite byzantine agreement / A. Bar-Noy, D. Dolev, C. Dwork, H. R. Strong // Information and Computation. — 1990. — Vol. 97, N 2. — P. 203–233.
20. **Moses Y.** Coordinated traversal: $(t + 1)$-round byzantine agreement in polynomial time // Annual Symposium on Foundations of Computer Science. — 1988. — P. 246–255.
21. **Garay J. A., Moses Y.** Fully polynomial byzantine agreement for n $>$ 3t processors in t+1 rounds. — SIAM J. of Computing. — 1998. — Vol. 27(1).
22. **Berman P., Garay J. A., Perry K. J.** Optimal early stopping in distributed consensus // Internat. Workshop on Distributed Algorithms. — 1992. — P. 221–237.
23. **Berman P., Garay J. A.** Cloture votes: $n/4$-resilient distributed consensus in $t+1$ rounds // Mathematical Systems Theory. — 1993. — Vol. 26. — P. 3–19.
24. **Berman P., Garay J. A.** Efficient distributed consensus with $n = (3 + \epsilon)t$ processors // Lect. Notes Comput. Sci. — 1991. — Vol. 579. — P. 129–142.
25. **Neiger G.** Distributed consensus revisited: Tech. Rep. GIT-CS-93/45 / G. Neiger: Georgia Institute of Technology, 1993.
26. **Fischer M. J., Lynch N. A., Paterson M. S.** Impossibility of distributed consensus with one faulty process // JACM. — 1985. — Vol. 32, N 2. — Pp. 374–382.
27. **Dolev D., Dwork C., Stockmeyer L.** On the minimal synchronism needed for distributed consensus // JACM. — 1987. — Vol. 34, N 1. — P. 77–97.
28. **Attiya G., Dolev D., Gil J.** Asynchronous byzantine consensus // ACM Symposium on Principles of Distributed Computing. — 1984. — P. 119–133.
29. **Dwork C., Lynch N. A., Stockmeyer L.** Consensus in the presence of partial synchrony // JACM. — 1988. — Vol. 35, N 2. — P. 288–323.
30. **Perry K. J.** Randomized byzantine agreement: Tech. Rep. TR-84-595 / K. J. Perry: Cornell University, 1984.
31. **Dolev D.** The byzantine generals strike again.: Tech. Rep. STAN-CS-81-846 / D. Dolev: 1981.
32. Fault tolerance in the network of bounded degree / C. Dwork, D. Peleg, N. Pippenger, E. Upfal // Annual ACM Symposium on Theory of Computing. — 1986. — P. 370–379.
33. **Berman P., Garay J. A.** Asymptotocally optimal distributed consensus // Lect. Notes Comput. Sci. — 1989. — Vol. 372. — P. 80–94.
34. **Berman P.** Fast consensus in networks of bounded degree // Dsitributed Computing. — 1993. — Vol. 7. — P. 67–73.
35. **Upfal E.** Tolerating linear number of faults in networks of bounded degree // Annual ACM Symposium on Principles of Distributed Computing. — 1992. — P. 83–89.

36. **Dolev D., Strong H. R.** Polynomial algorithm for multiple processor agreement// ACM symposium on Theory of computing. — 1982. — P. 401–407.
37. **Fischer M. J., Fowler R., Lynch N. A.** A simple and efficient byzantine generals algorithm // IEEE Symposium on Reliability in Distributed Software and Database Systems, IEEE CS 2, Wiederhold(ed), Pittsburgh PA, 1982.
38. **Srikanth, T. K.** Simulating authenticated broadcasts to derive simple fault-tolerant algorithms: Tech. Rep. TR-84-623 / T. K. Srikanth, S. Toueg: Cornell University, 1984.
39. **Coan B. A.** A communication-efficient canonical form for fault-tolerant distributed protocols // Annual ACM Symposium on Principles of Distributed Computing. — 1986. — P. 63–72.
40. **Coan B. A., Welch J. L.** Modular construction of nearly optimal byzantine agreement protocols // Annual ACM Symposium on Principles of Distributed Computing. — 1989. — P. 295–305.
41. **Berman P., Garay J. A., Perry K. J.** Toward optimal distributed consensus // Annual Symposium on Foundations of Computer Science. — 1989. — P. 410–415.
42. **Dolev D., Reischuk R., Strong H. R.** Early stopping in byzantine agreement // JACM. — 1990. — Vol. 37, N 4. — P. 720–741.
43. **Perry, K. J.** Fast distributed agreement: Tech. Rep. TR-84-621 / K. J. Perry, T. K. Srikanth, S. Toueg: Cornell University, 1984.
44. **Bar-Noy A., Dolev D.** Families of consensus algorithms // AWOC. — 1988. — P. 380–390.
45. **Coan B. A., Welch J. L.** Modular construction of an efficient 1-bit byzantine agreement protocol // **Mathematical Systems Theory**. — 1993. — Vol. 26. — P. 131–154.
46. **DeMillo R. D., Lynch N. A., Merritt M.** Cryptographic protocols // ACM Symposium on Theory of Computing. — 1982. — P. 383–400.
47. **Fischer, M. J.** Byzantine generals and transaction commit protocol: Tech. Rep. Opus 62 / M. J. Fischer, L. Lamport: SRI inc, 1982.
48. **Ben-Or M.** Another advantage of free choice: completely asynchronous agreement protocols // ACM symposium on Principles of distributed computing. — 1983. — P. 27–30.
49. **Bracha G., Toueg S.** Asynchronous consensus and broadcast protocols // JACM. — 1985. — Vol. 32, N 4. — P. 824–840.
50. **Bracha G.** An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol // ACM symposium on Principles of distributed computing. — 1984. — P. 154–162.
51. **Rabin M.** Randomized byzantine generals // Symposium on Foundations of Computer Science. — 1983. — P. 403–409.
52. **Toueg S.** Randomized byzantine agreement // ACM Symposium on Principles of Distributed Computing. — 1984. — P. 163–178.
53. **Fischer M. J.** The consensus problem in unreliable distributed systems (a brief survey): Tech. Rep. YALEU/DCS/TR-273 / M. J. Fischer: Yale University, 1983.

С.В. Юрьев

УНИВЕРСАЛЬНАЯ СИСТЕМА ПОСТРОЕНИЯ И АДМИНИСТРИРОВАНИЯ ЛАБОРАТОРНЫХ ВЕБ-САЙТОВ

ВВЕДЕНИЕ

Раньше основная часть ресурсов в сети строилась с использованием статических страниц, написанных на HTML. Эти страницы редактировались в каком-нибудь текстовом или HTML-редакторе, после чего закачивались на хостинг-сервер (чаще всего по ftp).

Однако со временем в таком способе администрирования обнаружились серьезные недостатки. По мере увеличения количества страниц утрачивалась прозрачность структуры сайта, все труднее становилось находить и редактировать нужные страницы. Возрастала вероятность появления так называемых «мертвых ссылок», указывающих на несуществующие более страницы. Кроме того, назрела необходимость использования различных интерактивных элементов, форм регистрации пользователей, поиска, каталогов и т.д. Тогда и стали появляться первые «бэкофисы»[1] — системы администрирования веб-ресурсов.

Первые системы создавались под конкретные проекты. Основным их достоинством была высокая степень индивидуализации, однако они также обладали рядом серьезных недостатков. В них, как правило, намертво вшивался дизайн сайта, а возможности администрирования были ограничены некоторым набором программных функций, не поддающимся самостоятельному масштабированию и расширению возможностей без участия его разработчиков. Чтобы добавлять новые функции или тиражировать уже существующие, владельцам сайта снова приходилось привлекать команду программистов, что было крайне нецелесообразно с точки зрения временных и материальных затрат.

Именно тогда назрела необходимость в универсальных CMS[2] — системах управления содержанием ресурса. К концу 90-х гг. именно такие сис-

[1] От англ. back office — вспомогательный офис

[2] От англ. content management system — в разных источниках переводится как система управления содержанием или система управления контентом

темы стали новым этапом в индустрии веб-разработок. Появился рынок CMS для малых, средних и крупных веб-порталов.

В данный момент на российском рынке представлены более шестидесяти широко известных профессиональных систем управления [2,3] и еще сотни менее раскрученных пакетов. Их цена колеблется в среднем от $1000 до $3000, они ориентированны на малый и средний бизнес. Различные enterprise-решения от таких гигантов как Microsoft и IBM, стоящие сотни тысяч долларов, подходят лишь для промышленных гигантов и потому в поле зрения нашего исследования не попадают.

Основной задачей этих систем является разделение содержания сайта и его представления, а также предоставление пользователю удобных инструментов для публикации и изменения содержания при условии, что навыки пользователя не превосходят умения работать с Интернетом (точнее браузером) и офисными приложениями, такими как Microsoft Word и т.п. При этом представление содержания чаще всего ограничено форматом HTML. Само содержание имеет неоднородную структуру. Помимо обычных текстовых документов пользователи сайта работают с такими объектами, как ленты новостей, форумы и т.п. CMS предоставляют пользователям набор инструментов для работы с наиболее типичными объектами содержания, что вполне достаточно для поддержки большинства сайтов. Однако для поддержки лабораторных сайтов предоставляемых инструментов зачастую недостаточно, поскольку в лабораториях работают с информацией, структура которой зачастую кардинально отличается от структуры объектов, которые представлены в системах. Публиковать эту информацию на сайте с помощью таких систем крайне неудобно.

Целью работы являлась разработка CMS, предоставляющей возможность удобной работы с различной информацией, нетипичной для среднестатистических сайтов. Кроме того, система должна иметь возможность публиковать информацию не только в HTML, но и в других форматах, чтобы осуществлять её автоматическую обработку. При этом вновь разрабатываемая CMS не должна потерять тех преимуществ, которыми обладают уже существующие системы.

В результате проделанной работы была разработана CMS, удовлетворяющая всем поставленным задачам. Она позволяет более гибко работать с информацией, публиковать ее в различных форматах. При этом выгодные свойства других CMS не только не утрачиваются, но и реализуются еще более удачно. В частности, задача отделения информации от представления решается без наложения всяческих ограничений на последующую изменяемость структуры и дизайна сайта, чего нельзя сказать о других систе-

мах, которые все же такие ограничения накладывают. На базе разработанной CMS создается веб-сайт лаборатории. С его помощью система была проверена на соответствие условиям, описанным в постановке задачи.

Текст работы построен следующим образом. В разделе «Обзор предметной области» рассказывается о том, что такое CMS и какие задачи она решает. Также в ней описывается общая схема работы большинства CMS. Далее следует постановка задачи, выделяется ряд требований к разрабатываемой системе. Требования разделяются на общие для всех CMS и специфичные для конкретной системы ввиду того, что она должна быть оптимизирована для создания лабораторных сайтов. В разделе «Описание решения задачи» вначале дается сжатое изложение решения, его удовлетворение требованиям, рассмотренным в постановке задачи, кратко описываются преимущества предложенных механизмов работы по сравнению с существовавшими ранее. После этого следует подробное описание решения, рассматриваются все объекты системы и их взаимодействие между собой и с администраторами сайта. Рассматриваются инструменты, предоставляемые пользователю для управления сайтом. В разделе «Механизм генерации готовой страницы раздела» подробно описывается процесс сборки страниц, соответствующих разделам, запрашиваемым посетителем сайта. Заканчивается глава описанием функций, входящих в ядро системы, содержащее всю логику работы сайта. В заключение описываются результаты проведенной работы, их практическое применение и направления дальнейшей работы.

1.ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Понятие CMS систем

Система управления контентом [1], иначе называемая CMS, — это программное обеспечение, которое позволяет публиковать и изменять опубликованную на сайте информацию самостоятельно, без привлечения разработчиков сайта. Подразумевается, что от пользователей такой системы не требуется специальных знаний технологий, отличающихся от обычно используемых в офисных процессах (текстовый редактор, Интернет и т.п.). При этом не следует считать, что такая система не требует обучения персонала, но это обучение касается порядка работы в системе, а не изучения новых технологий.

Большинство CMS можно разделить на среду разработки, т.е. инфраструктурную систему, обеспечивающую функциональность и хранение ин-

формации, и среду редактирования, интерфейс работы с пользователем. В большинстве современных CMS среда разработки базируется на той или иной СУБД, может включать сервера приложений, а среда редактирования имеет веб-интерфейс и допускает использование стандартных офисных пакетов редактирования документов (текстовые редакторы, электронные таблицы, средства создания презентаций, почтовые системы и т.п.). При этом вся функциональность, сложность разработки и администрирования сосредоточены в среде разработки, а пользовательские свойства — в среде редактирования. Обобщенная типичная схема работы сайта, использующего CMS, представлена на рис. 1.

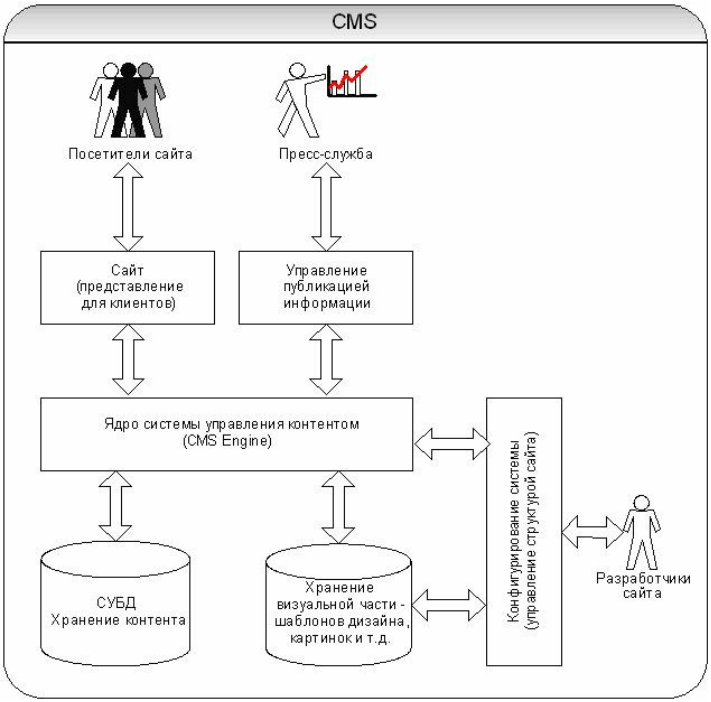


Рис. 1. Схема работы сайта с использованием CMS

В системе присутствуют два хранилища. В первом (обычно реляционная СУБД) хранятся все данные, которые публикуются на сайте. Во втором (обычно файловая система) хранятся элементы представления — шаблоны,

графические изображения и т.д. Кроме внешнего представления сайта, каким его видят все пользователи, есть как минимум два специализированных рабочих места. Первое рабочее место — для разработчиков сайта. С его помощью они задают структуру сайта, структуру содержания, определяют внешний вид сайта, настраивают шаблоны дизайна. Этот инструментарий обычно не полностью автоматизирован. Для настройки сайта разработчики частично работают через средства CMS, часть информации размещается напрямую (т.е. идет работа с файлами на сервере и с СУБД). Второе рабочее место — для владельцев сайта. Оно позволяет сотрудникам компании самостоятельно размещать информацию на сайте, без участия разработчиков. Менеджеры заказчика работают только через специализированное рабочее место. Прямого доступа к файлам и СУБД они не имеют.

Использование CMS предоставляет следующие преимущества.

1. Оперативное обновление информации — информацию публикует человек (сотрудник компании), владеющий информацией, без дополнительных посредников в виде технических специалистов.
2. Возможность реализации дополнительных сервисов, таких как форумы, голосования и т.д. Они зачастую требуют интерактивного взаимодействия с пользователем.
3. Уменьшение сроков и затрат на разработку сайта — наиболее востребованная функциональность уже реализована в CMS и может быть сразу использована.
4. Повышение качества разработки — использование автоматизированных средств разработки позволяет сохранять прозрачность структуры сайта и избегать большинства ошибок и неудобств, сопутствующих работе с сайтами, не использующими CMS.
5. Упрощение дальнейших модификаций — CMS позволяют разделить и независимо редактировать данные и их представление.
6. Возможность совмещения работы нескольких человек над одним сайтом с разделением их прав и обязанностей.

Чтобы обеспечить данные преимущества, CMS должна решить следующие основные задачи.

1. Публикация информации человеком, не знакомым с технологиями разработки сайтов.
2. Разделение данных и их представления.
3. Организация работы нескольких человек над сайтом.

2. ПОСТАНОВКА ЗАДАЧИ

Поскольку областью использования разрабатываемой системы являются именно лабораторные сайты, то система должна быть способна реализовывать работу с динамическими информационными объектами, нетипичными для среднестатистических сайтов и систем. Например, если в лаборатории идет разработка некоторого программного проекта, то могут возникнуть нижеперечисленные задачи:

1. Публикация результатов работы в Интернете. При этом должна быть возможна публикация в различных форматах, отличных от традиционного HTML. Это дает возможность их точной автоматической обработки другими приложениями.
2. Предоставление сторонним лицам возможности воспользоваться программной функциональностью разрабатываемого проекта. Иными словами, предоставить возможность использования некоторого приложения через Интернет.

Система должна предоставлять инструменты решения этих задач. Кроме того, система должна обладать следующими свойствами.

1. Универсальность — система должна представлять собой универсальное средство для занесения и обновления информации, которое можно было бы применить к сайтам и разделам с различной визуальной и логической структурами.
2. Настройки безопасности, разграничение прав пользователей на редактирование сайта.
3. Возможность формирования динамической структуры сайта — структура сайта должна быть легко изменяемой, прозрачной и четко выраженной, что, в свою очередь, подразумевает следующие аспекты реализации:
   - гибкость и прозрачность архитектуры — наличие возможности беспрепятственно отслеживать и изменять структуру сайта, данных;
   - возможность беспрепятственного редактирования дизайна;
   - возможность внедрения приложений независимых разработчиков;
   - модульная структура подключения объектов функционала;
   - разделение данных и их представления.
4. Возможность хранения различных видов информации: не только одностраничных документов, но и динамических информационных

блоков, таких как ленты новостей и другие приложения как типичные, так и индивидуальные.

5. Простота в использовании. Для использования системы вебмастеру должно быть достаточно знания офисных приложений. Здесь важно понимать различие между веб-разработчиком и вебмастером, поскольку первый отвечает за установку сайта, его начальную настройку, подключение дополнительных модулей, управление дизайном, а второй — за добавление и изменение информации как статической, так и динамической.

3. ОПИСАНИЕ РЕШЕНИЯ ЗАДАЧИ

3.1. Краткое описание решения

Сайт, построенный на вновь разработанной системе, можно условно разбить на несколько глобальных объектов.

1. Дерево разделов — иерархически упорядоченное множество разделов сайта, каждый из которых имеет одного родителя и сколько угодно дочерних разделов. Исключение составляет корневой раздел, у которого нет родителя.
2. Множество пользователей/администраторов — людей, обладающих различными правами на просмотр и редактирование содержания и структуры сайта.
3. Множество информационных пакетов — объектов, несущих некоторую информацию, которую нужно разместить на сайте. Такими объектами могут быть, например, текстовые документы, ленты новостей и т.п. Каждый пакет имеет свой тип, которому соответствует ровно одно приложение, позволяющее его редактировать.
4. Множество шаблонов отображения — объектов, несущих информацию о том, как должна информация отображаться на сайте.
5. Множество приложений — программ, отвечающих за обработку содержания информационных пакетов. Число этих приложений равно числу типов информационных пакетов, поскольку одно приложение отвечает за один тип. Множество приложений не является фиксированным. В него можно добавлять новые приложения в соответствии с нуждами проекта.

Все эти объекты взаимосвязаны и образуют основное ядро сайта, обеспечивающее ему основу для работы и дальнейшего развития. Информационные пакеты размещаются в разделах сайта, которые отображаются посе-

тителям в соответствии с шаблонами. Посетители могут путешествовать по сайту, перемещаясь по дереву разделов и изучая информацию, содержащуюся в информационных пакетах. Администраторы могут изменять дерево разделов в соответствии со своими нуждами, размещать в них информацию в виде информационных пакетов. Публиковать новую информацию в пакетах и редактировать уже существующую администраторы могут при помощи приложений, отвечающих за работу с этими пакетами. Также администраторы могут управлять отображением информации на сайте при помощи шаблонов.

Разработанная система решает поставленные задачи следующим образом (рис. 2).

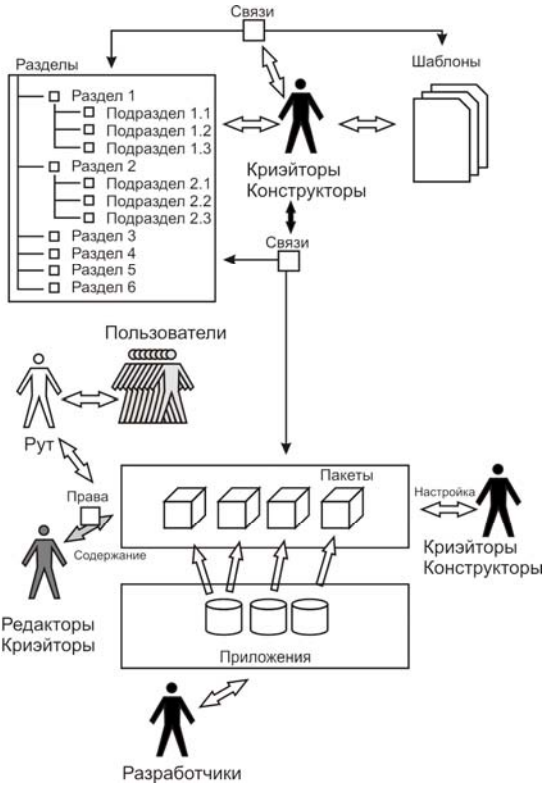


Рис. 2. Схема работы создаваемой CMS

- Публикация содержания сайта возможна в любом формате, основанном на технологии XML, поскольку внутри CMS все информационные объекты и документы, составляющие содержание сайта, описываются с помощью языка, также основанного на XML, а перед тем как быть отправленным пользователю любой документ претерпевает XSL-трансформацию в нужный формат. Фактически эта трансформация играет роль шаблона отображения информации. Это также позволяет публиковать документы не только для просмотра пользователем, но и для автоматической обработки документов другими приложениями.
- Гибкость и прозрачность архитектуры реализована за счет построения явного дерева разделов, хранящегося в базе данных. В обычных CMS структура сайта ассоциируется с расположением файлов, соответствующих разделам на жестком диске. Это делает структуру сайта менее прозрачной, поскольку явная иерархическая связь между разделами отсутствует.
- Система универсальна — за счет реализации независимой работы с разделами, информационными пакетами и шаблонами отображения, о которых речь пойдет ниже. Она позволяет реализовывать сайты с самой разнообразной структурой.
- Права пользователей на редактирование сайта строго разграничены за счет контроля прав пользователей на доступ к разделам и информационным пакетам.
- Работа с дизайном и вообще с форматами представления информации на сайте производится за счет шаблонов представления, которые можно беспрепятственно изменять, независимо от структуры содержания сайта.
- В систему можно внедрять приложения, реализующие работу с нетипичной информацией, как статической, так и динамической, с помощью механизма информационных пакетов, которые являются своего рода интерфейсом взаимодействия между ядром системы и другими приложениями. Благодаря этому механизму функциональность внедряемых приложений можно свободно тиражировать, например, создавать неограниченное количество лент новостей на сайте.
- Данные отделены от представления, поскольку всю работу с данными осуществляют подключенные приложения, а за представление отвечают шаблоны отображения.

- Разнообразие информационных объектов, работа с нетипичной информацией реализуется также при помощи специальных приложений, подключаемых к системе через информационные пакеты.
- Пользователю для публикации информации на сайте не нужно обладать широкими знаниями в области Интернет-технологий, поскольку вся работа, затрагивающая эти технологии, либо автоматизирована, либо проводится один раз разработчиком сайта в начале его работы.

Далее мы рассмотрим работу всех объектов системы более подробно.

3.2. Дерево разделов

Представляет собой множество объектов, у каждого из которых есть родитель, для которого он, в свою очередь, является дочерним объектом. Таким образом, у каждого объекта-раздела может быть только один родитель и любое количество дочерних разделов.

К каждому разделу можно привязать любое количество информационных пакетов, речь о которых пойдет ниже. Также к разделу привязывается шаблон отображения из базы шаблонов, который используется при генерации готовой страницы. Более подробно речь о шаблонах также пойдет ниже.

3.3. Множество пользователей и администраторов

Множество пользователей и администраторов — это база данных, содержащая следующие данные: id, логин, пароль, тип пользователя, информация о пользователе, его права. Пользователи могут быть шести типов.

1. *Рут* — может редактировать дерево и свойства разделов, базу пользователей, пакетов, приложений, шаблонов, а также содержание информационных пакетов. Вообще говоря, это пользователь, которому доступен полный контроль над сайтом. Хотя система не запрещает наличия нескольких пользователей такого типа, все же в идеале Рут в системе должен быть один.
2. *Криэйтор*[3] — может редактировать дерево и свойства разделов, базу пакетов, шаблонов, а также содержание информационных пакетов. Иными словами, этому пользователю доступен весь инструментарий системы, за исключением работы с пользователями.

[3] От англ. Creator — создатель.

3. *Конструктор* — может редактировать только дерево и свойства разделов, подключать уже существующие шаблоны и пакеты, а также создавать новые. Фактически Конструктор может полностью редактировать структуру сайта, однако работа с информационной частью ему недоступна. Такую роль в системе можно при необходимости выделять разработчику, перед которым поставлена задача изменения структуры сайта.
4. *Редактор* — может просматривать и редактировать содержание информационных пакетов в зависимости от прав доступа к ним. Эту роль можно выделять веб-мастерам, которые отвечают за публикацию информации в тех или иных информационных пакетах.
5. *Обычный пользователь* — может просматривать различные разделы сайта в зависимости от прав доступа к ним.
6. *Гость, или незарегистрированный*, — может лишь просматривать содержание открытых для всех разделов.

Права на доступ пользователей четвертой и пятой категории к разделам и пакетам сайта редактируются Рутом. Именно эти пользователи составляют основное множество людей, работающих с сайтом (исключая гостей, права которых минимальны, но которые при желании самостоятельно могут стать пользователями пятой категории, зарегистрировавшись на сайте). Рут может разрешать/запрещать просмотр/редактирование пользователями информации на сайте, а также менять статус пользователя в системе. Однако Рут не может разрешить пользователю пятой категории редактировать сайт. Для этого сначала необходимо присвоить ему статус редактора (4-я категория). Работа с пользователями сайта также не требует знания дополнительных технологий и может быть освоена обычным «офисным» работником.

Каждому *редактору* назначаются права на работу с пакетами и разделами. Редактор может иметь право на просмотр некоторого раздела и на редактирование конкретного пакета в конкретном разделе, например, ленты новостей только на главной странице. Необходимость такого определения прав пользователя обусловлена тем, что содержание одних и тех же пакетов в разных разделах может отличаться. Например, информация, содержащаяся в информационном пакете типа «обычный текст»[4], — своя для каждого раздела.

[4] Пакет этого типа соответствует приложению под названием «Менеджер Текстов», о котором речь пойдет ниже.

Каждому *обычному пользователю* можно разрешить/запретить просматривать тот или иной раздел. При этом, если просмотр данного раздела запрещен, то запрещен просмотр всех его потомков. Если же просмотр раздела разрешен, то права на просмотр дочерних разделов определяются отдельно.

Права пользователей остальных категорий определяются самими категориями.

3.4. Множество информационных пакетов

Множество информационных пакетов представляет собой систему обработки и хранения всей информации, отображаемой на сайте. Информация хранится в виде информационных пакетов. Каждый такой пакет представляет собой объект, свойства которого определяются приложением, за него отвечающим. Это может быть некоторый набор текстов, лента новостей, меню разделов, некоторая база данных и т.п. Каждый такой пакет зависит лишь от своего приложения, которое предоставляет пользователю полный набор средств для его просмотра и редактирования. Чтобы отобразить пакет на сайте, его необходимо привязать к некоторому разделу.

Любой пакет — это интерфейс взаимодействия ядра системы и приложения, обеспечивающего некоторую функциональность. Через него приложение предоставляет ядру доступ к объектам пакета, их свойствам и методам их обработки.

Редактирование содержания пакетов — наиболее часто выполняемая на сайте операция. Именно с помощью нее на сайте происходят публикация и изменение документов и прочего контента, как статического, так и динамического. Также на сайте доступна настройка некоторых свойств пакетов, например, количество новостей, отображаемых на одной странице для пакета-ленты новостей. Работа с пакетами в системе напоминает работу с файлами. Их можно удобно рассортировать по виртуальным папкам, что повысит удобство их использования.

3.5. Множество шаблонов отображения

Множество шаблонов отображения представляет собой набор шаблонов, использующихся для описаний структуры, вида и дизайна конечного документа, отсылаемого пользователю. Каждый такой шаблон представляет собой заранее подготовленный XSLT-документ, преобразующий внутренний XML-документ, соответствующий некоторому разделу, во внешний

документ некоторого формата (XHTML или другой формат, созданный на основе технологии XML).

При отображении на сайте именно этот документ отсылается пользователю. Ниже будет описан механизм генерации системой внутреннего XML-документа и его преобразования во внешний.

Из сказанного выше следует, что редактирование шаблонов требует знания технологий XML/XSLT. Поэтому подразумевается, что работа с шаблонами осуществляется разработчиком сайта при его установке. Множество шаблонов достаточно создать один раз вначале использования сайта. Дальнейшее же изменение сайта, в том числе изменение структуры и масштабирование функционала, не требует редактирования шаблонов. В последующем может возникнуть потребность в изменении шаблонов лишь при редизайне сайта, добавлении новых приложений и интеграции нового формата данных, основанного на XML, для последующего отображения документов на сайте в этом формате. Однако все эти задачи нетривиальны и решаются только командой разработчиков, и потому требование знания технологий XML/XSLT вполне оправданно.

Работа с шаблонами также напоминает работу с файлами. Для большего удобства также присутствует работа с директориями.

3.6. Набор приложений

Набор приложений — это набор классов, отвечающих за логическую структуру и содержание информационных пакетов, его редактирование и отображение. Таким образом, эти классы отвечают фактически за все содержимое сайта. Подключение приложений к системе — действие нетривиальное, и потому осуществляется только разработчиком сайта. После того как приложение было подключено, необходимо во все шаблоны занести информацию о том, как будут отображаться объекты и методы, создаваемые данным приложением. Рассмотрим работу приложений на примере некоторых из них.

Менеджер новостей

Это приложение, позволяющее создавать неограниченное количество лент новостей на сайте. Каждая лента — это информационный пакет, который можно привязать к одному или нескольким разделам. Лента новостей имеет одинаковое содержание во всех разделах, к которым она подключена.

Каждая новость имеет дату публикации, заголовок, фабулу (иначе предисловие, тезис или краткое содержание) и содержание. Все эти данные отображаются на сайте в зависимости шаблона отображения.

Пакет предоставляет следующие методы для пользователей.

1. Просмотр списка последних n новостей (где n задается в настройках пакета).
2. Поскольку все новости пакета разбиты на страницы по n новостей в каждой, то возможен просмотр какой-то из страниц.
3. Подробный просмотр конкретной новости с возможностью возвращения к списку новостей.

Кроме того, для редакторов существуют такие методы, как:

- создание новой новости,
- редактирование существующей новости,
- удаление существующей новости.

Менеджер голосований

Это приложение, позволяющее создавать неограниченное количество лент голосований на сайте. Каждая лента — это информационный пакет, который можно привязать к одному или нескольким разделам. Каждая лента содержит максимум одно активное голосование и неограниченное количество завершенных. Каждый пользователь имеет право проголосовать один раз. Ленты во всех разделах имеют одинаковое содержание.

Каждый объект голосования имеет дату публикации, заголовок, описание и варианты ответа.

Пакет предоставляет следующие методы для пользователей.

1. Просмотр последнего активного голосования.
2. Голосование с выбором одного из вариантов ответа.
3. Просмотр архива голосований.
4. Просмотр результатов активного голосования без выбора вариантов ответа.

Редакторам также предоставляются:

- добавление/удаление голосования,
- блокировка голосования — перевод из статуса активного в статус завершенного.

Менеджер меню

Это приложение позволяет создавать различные меню навигации по сайту. Меню может быть одного из следующих типов:

1. *Меню разделов первого уровня* — отображает ссылки на все верхние разделы.
2. *Меню «братьев»* — отображает ссылки на всех детей раздела, приходящегося родительским для текущего раздела.
3. *Меню «родителей»* — отображает ссылки на братьев родительского раздела и на сам родительский раздел.
4. *Меню «детей»* — отображает ссылки на детей данного раздела.
5. *Цепочка навигации* — отображает полный путь до раздела, начиная с раздела первого уровня.

Содержание меню полностью зависит от раздела, в котором оно отображается и от типа меню. Оно генерируется автоматически по запросу на основе дерева разделов и не может быть отредактировано пользователем.

Менеджер текстов

Это приложение отвечает за вывод простых текстов. Его пакеты не имеют никаких дополнительных свойств. Однако через текстовые информационные пакеты можно очень удобно вставлять тексты в разделы. Каждый «текстовый» информационный пакет сопоставляет каждому разделу из тех, к которым он подключен, отдельный текстовый документ. Таким образом, в разных разделах этот пакет имеет разное содержание.

Менеджер семинаров

Это приложение, разработанное для администрирования базы данных семинаров. Эта база данных содержит в себе информацию о заседаниях семинаров. Семинар представляет собой информационный пакет. Объектами являются заседания семинаров. Каждое заседание обладает следующими свойствами: номер заседания, дата и время проведения, тема заседания, докладчик, содержание/описание.

Пакет предоставляет пользователям следующие методы.

1. Просмотр списка заседаний.
2. Подробный просмотр выбранного заседания.
3. Сортировка заседаний по номерам, датам, докладчикам.

Редакторам, в свою очередь, предоставляются методы добавления, удаления и редактирования свойств заседания.

3.7. Структура классов-приложений и их взаимодействие с ядром системы

Любое приложение разбивается на две независимые части — внутренний редактор и внешнее отображение содержимого пакетов. Они представлены в двух разных классах, подключенных соответственно к интерфейсу администрирования и программе, отображающей сайт. Редактор взаимодействует с интерфейсом администрирования, который, в свою очередь, обращается к соответствующему приложению при необходимости отредактировать свойства и содержание объектов пакета. Редактор хранит всю информацию о содержании и свойствах объектов в базе данных, содержащей одну или несколько таблиц, структура которых целиком определяется свойствами приложения, как было показано выше на примере реализованных приложений. Программа вывода соответствующего приложения использует эту же базу данных для вывода содержания пакетов. Её запускает основная программа отображения, чтобы затем вставить возвращаемый приложением код в генерируемый внутренний XML-документ.

В случае необходимости приложению через интерфейс пакета передается XML-документ, содержащий некоторые параметры. Приложение распоряжается этим документом по своему усмотрению. Подробнее об это речь пойдет ниже.

3.8. Встроенный текстовый редактор

Для расширения возможностей редактирования в ядро встроен текстовый редактор, который может быть использован приложениями при редактировании текстового содержания информационных пакетов для получения форматированного текста. В случае необходимости его использования, приложение обращается к специальному модулю, содержащемуся в ядре, и тот возвращает DHTML-код, содержащий все элементы управления и рабочие функции, написанные на JavaScript.

Помимо работы с текстом, его копирования, вырезки и вставки, этот редактор также позволяет моделировать таблицы, редактировать их основные свойства, после чего вставлять их в документ и в последующем заполнять их ячейки. Кроме того, он позволяет вставлять картинки из базы картинок, менять их размер и свойства, вставлять некоторые специальные символы, недоступные с клавиатуры, например € — евро. Также он умеет создавать нумерованные и ненумерованные списки, менять цвет и стиль написания текста, его шрифт и размер. Еще он умеет вставлять ссылки. В случае необходимости он также позволяет редактировать получившийся HTML-код.

3.9. Механизм генерации готовой страницы раздела

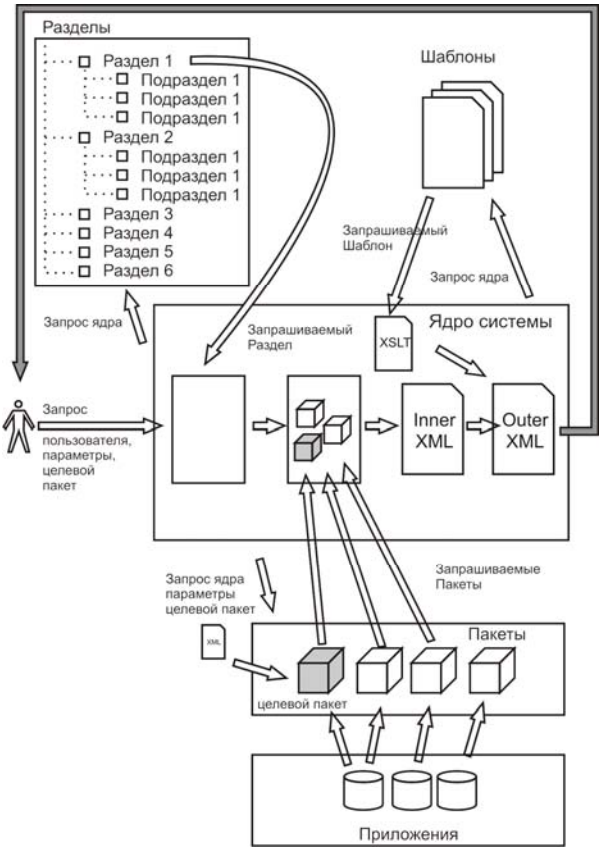


Рис 3. Механизм создания страницы сайта по запросу пользователя

Схема механизма генерации готовой страницы раздела представлена на рис. 3. Когда ядро системы получает запрос от пользователя на отображение некоторого раздела, она запрашивает свойства раздела из базы данных. Раздел обладает следующими свойствами:

- название,
- положение раздела в дереве разделов,
- шаблон отображения,

- привязанные пакеты.

После этого ядро запрашивает информацию о пакетах и соответственно узнает их названия и соответствующие приложения, отвечающие за их работу. После этого ядро обращается к этому приложению с запросом на генерацию содержимого пакета. Для каждого пакета, соответствующему приложению, передается несколько общих для всех пакетов параметров, таких как id раздела, id пакета. Также один из пакетов при необходимости помечается как *целевой пакет*. Для него ядро может передать приложению ряд индивидуальных параметров в виде XML-документа. Эти параметры индивидуальны для каждого приложения. Например, **Менеджеру Новостей** может быть передана команда на подробное отображение новости вместо общего списка новостей и соответственно id этой новости. Все остальные пакеты никаких дополнительных параметров не получают и отображаются, исходя из настроек по умолчанию.

В ответ ядро получает от приложения XML-документ, содержащий некоторый набор объектов данного пакета и набор действий, которые можно совершать над этими объектами и самим пакетом. Эти действия содержат те самые индивидуальные параметры, которые впоследствии передаются целевому пакету в случае, если это действие выполняется.

```
<package id="1" title="Новости" package_index="1" package_type="news">
    <item type="new" id="29" mode="brief">
        <str name="title">Заголовок новости 2</str>
        <datetime name="date">14.05.2005 13:45:00</datetime>
        <text name="fabula">
            Фабула 2, то есть краткое описание основной мысли текста.
        </text>
        <text name="content">
            Основной текст 2 идет здесь. Он может быть очень большим и
            длинным.
        </text>
        <package_action caption="Подробнее" value="view_item"
         partition_id="1" package_id="1">
                <parameters><![CDATA[
                    <item_id>29</item_id>
                    ]]>
                </parameters>
        </package_action>
    </item>
    <item type="new" id="28" mode="brief">
        <str name="title">Заголовок новости</str>
        <datetime name="date">14.05.2005 13:40:00</datetime>
        <text name="fabula">
```

```
                Фабула, то есть краткое описание основной мысли текста.
            </text>
            <text name="content">
                Основной текст идет здесь. Он может быть очень большим
                и длинным.
            </text>
            <package_action caption="Подробнее" value="view_item"
             partition_id="1" package_id="1">
                    <parameters><![CDATA[
                            <item_id>28</item_id>
                    ]]></parameters>
            </package_action>
        </item>

        <package_action caption="1" value="view_page" partition_id="1"
        package_id="1">
                <parameters><![CDATA[
                <page_number>1</page_number>
                ]]></parameters>
        </package_action>

        <package_action caption="2" value="view_page" partition_id="1"
        package_id="1">
                <parameters><![CDATA[
                        <page_number>2</page_number>
                    ]]></parameters>
        </package_action>
    </package>
```

Получив все документы от соответствующих приложений, ядро образует из них XML-документ, соответствующий вызываемому разделу.

```
    <document title="Главная" id="1">
        <package ...>
          ...
        </package>
        <package ...>
          ...
        </package>
    </document>
```

Перед отправкой пользователю этот документ претерпевает трансформацию, согласно XSLT-шаблона, привязанного к разделу. В итоге получа-

ется новый документ в некотором формате на основе XML (XHTML, SVG и т.п.), который отсылается обратно пользователю.

3.10. Функционалы, входящие в ядро системы

1. **Менеджер разделов** — специальная программа, входящая в состав ядра, отвечает за добавление, редактирование, удаление разделов, позволяет также устанавливать связи с шаблонами и пакетами через ячейки.
2. **Менеджер пользователей** — программа, входящая в состав ядра системы. Отвечает за редактирование базы пользователей, позволяет менять их свойства, добавлять, удалять, а также назначать права доступа к просмотру и редактированию разделов и пакетов.
3. **Менеджер пакетов** — отвечает за информационные пакеты, их создание, удаление и свойства. С его помощью осуществляется редактирование свойств пакетов через взаимодействие с соответствующими приложениями.
4. **Менеджер содержания** — отвечает за работу с информацией, содержащейся на сайте, осуществляет доступ к содержимому подключенных информационных пакетов через соответствующие разделы. Для редактирования содержимого пакетов он передает управление программе-приложению.
5. **Менеджер файлов и картинок** — отвечает за закачку файлов и картинок на сервер для последующего их использования. Закачанные файлы можно выкладывать для скачивания пользователями. Картинки можно вставлять в документы через HTML-редактор, а также использовать в других областях, определяемых информационными пакетами.
6. **Менеджер шаблонов отображения** — отвечает за работу с шаблонами, их добавление, редактирование и удаление.

ЗАКЛЮЧЕНИЕ

В результате проведенной работы были изучены несколько существующих систем и методы их работы, а также была разработана CMS, удовлетворяющая, поставленной задаче.

- Система универсальна и позволяет создавать сайты с документами различных форматов, отличных от традиционного HTML, любым дизайном, любой структурой и любым информационным наполнением.

- Пользователи системы имеют различный статус и доступ к редактированию настроек, структуры и содержания сайта.
- В силу древовидного иерархического строения структура сайта прозрачна и легко изменяема, изменения не влияют ни на дизайн, ни на содержание сайта;
- Дизайн сайта легко изменяем и настраиваем и не влияет, в свою очередь, на структуру и информационное наполнение сайта.
- Благодаря разработанному интерфейсу взаимодействия между программами-приложениями и ядром системы через информационные пакеты, существует возможность модульного подключения дополнительных объектов/функционала в виде классов-приложений.
- Возможность хранения различных нетривиальных видов информации осуществляется за счет подключаемых приложений и встроенного текстового редактора.
- Для редактирования информации на сайте пользователю необходимо иметь лишь базовые знания о работе с офисными приложениями. Для разработки сайта необходимо также знание технологий XML, XSLT.

Среди преимуществ системы перед другими, описанными ранее, можно выделить следующие.

- Если в ранее описанных системах достигалось лишь отделение данных от их отображения, то во вновь построенной — достигнуто разделение структуры, содержания и отображения сайта. Это позволяет более гибко работать с каждым из этих трех аспектов, не опасаясь нарушить остальные. Например, можно изменять дерево разделов, независимо от содержания и дизайна, или менять дизайн, независимо от содержания и структуры.
- Структура сайта стала более прозрачной за счет использования явного дерева разделов.
- Отображение данных возможно не только в формате традиционного HTML, но и в любом другом формате, основанном на технологии XML, что делает возможным использование сайта не только людьми, но и различными программами.

На базе разработанной системы создается веб-сайт лаборатории конструирования и оптимизации программ ИСИ. С его помощью система была проверена на работоспособность и соответствие условиям, описанным в постановке задачи. С целью практического обоснования возможности подключения дополнительного нетипичного функционала был написан специализированный модуль для работы с базой данных семинаров.

В дальнейшем планируется исследование проблем реализации поиска, сбора статистики и алгоритмов кэширования страниц сайта для ускорения его работы, а также организации документооборота с участием пользователей внутри системы.

СПИСОК ЛИТЕРАТУРЫ

1. **Заостровцев Н. В.** Выбираем систему управления контентом для небольшого предприятия // Elashkin Research 2004. — 29с. — http://business-site.ru/articles/cmsnp.pdf
2. **Косяков И.** Стратегия выбора системы управления сайтом: сравнение систем по формальным параметрам // Информационный проект Business-Site.ru. — http://business-site.ru/articles/wsms_strat.htm
3. **Терехов А.** Сравнение Контент-Менеджеров // Информационный проект cmslist.ru. — http://cmslist.ru/?ext=content&lang=1&pid=170
4. Bitrix-управление сайтом: руководство пользователя / М: Битрикс, 2004 — 108с. — http://www.bitrixsoft.ru/download/BSM_Expert_UserGuide.pdf
5. Bitrix-управление сайтом: руководство по интеграции / М: Битрикс, 2004 — 40с. — http://www.bitrixsoft.ru/download/BSM_IntegrationGuide.pdf
6. Saitistika: руководство пользователя / М: Individ Company, 2003 — 155с. — http://www.saitistika.ru/files/articles/2002/SaitistikaUserGuide.pdf
7. Q-Publishing: руководство пользователя / М: Quantum Art, 2002 — 83с. — http://www.quantumart.ru/doc/Q-Publishing%20User%20Guide%20Russian%20Revised.pdf
8. Q-Publishing: руководство разработчика / М: Quantum Art, 2003 — 92с. — http://www.quantumart.ru/doc/site_developer_tutorial_rus.pdf

СОДЕРЖАНИЕ

Предисловие редактора 5
Арапбаев Р.Н., Осмонов Р.А. Анализ зависимостей по данным для многомерных массивов на базе модифицированного λ-теста 7
Батура Т.В., Мурзин Ф.А. Обработка поисковых запросов на естественном языке с помощью REFAL-подобных конструкций 24
Добрынин А.А., Мельников Л.С., Вальтер Х., Шрейер Й. Число косых полиэдральных графов с малым числом вершин 34
Дунаев А.А., Валеев Т.Ф., Тарасов Е.А. Исследование методов организации визуальной обратной связи в аппаратно-программном комплексе «Бослаб» 42
Дунаев А.А. Исследовательская система для анализа текстов на естественном языке 55
Касьянов В.Н. Музеи и Интернет 67
Касьянова Е.В. Адаптивная система поддержки дистанционного обучения программированию 85
Козырева А.В. Определение координат мобильного устройства в пространстве на основе изображений, получаемых от его видеокамеры 113
Козырева А.В. О некоторых способах калибровки видеокамеры 132
Мельников Л. С., Петренко И. В. Путевые разбиения в неориентированных графах 142
Несговорова Г.П. Современные информационно-коммуникационные и цифровые технологии в сохранении культурного и научного наследия и развитии музейного дела 153
Осмонов Р.А., Штокало Д.Н. Преобразования циклов, основанные на несингулярных матрицах 162
Серебренников А.Л. Обзор возможностей среды Significo на примере решения прикладной задачи 177
Серебренников А.Л. Сравнительный анализ нейросетевых пакетов и место среды Sig-nifico среди них. Краткое описание среды 192
Стасенко А.П. Обзор потоковых языков программирования 207
Тараскина А.С. Нечеткая кластеризация по модифицированному методу с-средних и ее применение для обработки микрочиповых данных. 217
Шкурко Д.В. Отказоустойчивость в распределенных сетях: проблемы консенсуса 229
Юрьев С.В. Универсальная система построения и администрирования лабораторных веб-сайтов 249

CONTENTS

Preface 5

Arapbaev R.N., Osmonov R.A. Data dependences analysis for multidimensional arrays on the basis of a modified λ-test 7

Batura T.V., Murzin F.A. Processing of a query in a natural language with the help of REFAL-like constructions 24

Dobrynin A.A., Mel'nikov L.S., Schreyer J., Walther H. The number of oblique polyhedral graphs with a small number of vertices 34

Dunaev A.A., Valeev T.F., Tarasov E.A. Research of the methods of organization of a visual feedback in a hardware-software complex "BOSLAB" 42

Dunaev A.A. A research system for analysis of texts in a natural language .. 55

Kasyanov V.N. Museums and Internet 67

Kasyanova E.V. An adaptive system of support for distant education in programming 85

Kozyreva A.V. Detecting the mobile device position by analyzing images made by its camera 113

Kozyreva A.V. Some methods of video camera calibration 132

Mel'nikov L.S., Petrenko I.V. Path partitions in non-oriented graphs 142

Nesgovorova G.P. Modern information, communication, and digital technologies in conservation of the cultural and scientific heritage and in the development of the museum activity 153

Osmonov R.A., Shtokalo D.N. Loop transformations based on nonsingular matrix 162

Serebrennikov A.L. A review of capabilities of the Significo environment by the example of deciding an applied problem 177

Serebrennikov A.L. Benchmark analysis of neural network packages and the place of the Significo environment among them. Short description of the environment 192

Stasenko A.P. A review of dataflow programming languages 207

Taraskina A.S. Fuzzy clusterization by the modified method of c-average and its application to microchip data processing 217

Shkurko D.V. Fault-tolerance of distributed systems: the consensus problem . 229

Yur'ev S.V. Universal system for building and management of laboratorial web-sites 249

УДК 519.68 + 681.3.06

Анализ зависимостей по данным для многомерных массивов на базе модифицированного λ-теста / Арапбаев Р.Н., Осмонов Р.А. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 7–23.

При распараллеливании циклов одной из важных проблем является выявление зависимости по данным. Тесты на зависимость должны определять, существуют ли целочисленные решения системы линейных диофантовых уравнений, полученных при выявлении зависимостей, удовлетворяющие ограничениям границ циклов. В настоящей работе предлагается новый модифицированный вариант λ-теста, в котором λ-тест интегрирован с точным IR-тестом, благодаря чему при анализе зависимостей многомерных массивов были получены более точные результаты. Практическим результатом данной работы является тест, который может быть использован в блоке анализа зависимостей по данным в проектируемой системе быстрого прототипирования компилятора. — Библиогр.: 9 назв.

Data dependences analysis for multidimensional arrays on the basis of a modified λ-test / Arapbaev R.N., Osmonov R.A. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 7–23.

One of the important problems in loop parallelization is a revelation of data dependences. Tests for dependence should find whether there exist integer solutions of a system of linear Diophantine equations obtained at revealing the dependences that satisfy the constraints on the loop bounds. In this work a new modified variant of the λ-test is presented, in which and λ-test is integrated with the exact IR-test. This allowed us to obtain more exact results in the analysis of dependences in multidimensional arrays. A practical result of this work is a test, which can be used in the block of the data dependences analysis in the system of compiler fast prototyping which is now under development. — Refs: 9 titles.

УДК 519.68 + 681.3.06

Обработка поисковых запросов на естественном языке с помощью REFAL-подобных конструкций / Батура Т.В., Мурзин Ф.А. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 24–33.

В статье кратко обосновывается возможность применения модифицированных конструкций языка символьных преобразований REFAL для формирования древообразного представления предложений на естественном языке и схем “вопрос—ответ”, а также описан алгоритм использования их в поисковых системах. — Библиогр.: 2 назв.

Processing of a query in a natural language with the help of REFAL-like constructions / Batura T.V., Murzin F.A. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 24–33.

The paper briefly considers the possibility of application of modified constructions of the language of symbolic transformations REFAL to formation of a tree-like representation of a sentence in a natural language and a “question—answer” scheme. The algorithm of using them in search systems is described. — Refs: 2 titles.

УДК 519.68 + 681.3.06

Число косых полиэдральных графов с малым числом вершин / Добрынин А. А., Мельников Л. С., Вальтер Х., Шрейер Й. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 34–41.

Рассматриваются полиэдральные графы (графы полиэдров), т. е. плоские 3-связные графы. Грань размера k полиэдрального графа имеет тип $\langle a_1, a_2, ..., a_k \rangle$, если инцидентные этой грани вершины, обходимые в циклическом порядке, имеют степени $a_1, a_2, ..., a_k$, и этот набор является лексикографически минимальным среди всех подобных наборов. Если в полиэдральном графе все грани имеют разные типы, то такой граф называется косым (oblique). Для полиэдральных графов с числом вершин не более 12 найдены количества косых графов, в том числе с дополнительными свойствами. — Библиогр.: 5 назв.

The number of oblique polyhedral graphs with a small number of vertices / Dobrynin A.A., Mel'nikov L.S., Schreyer J., Walther H. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 34–41.

The paper considers polyhedral graphs (graphs of polyhedron or planar 3-connected graphs). A face α of size k of a polyhedral graph is of type $\langle a_1, a_2, ..., a_k \rangle$ if the vertices incident with α in a cyclic order have degrees $a_1, a_2, ..., a_k$ and this sequence is lexicographically minimal. A polyhedral graph is oblique if it has no two faces of the same type. The number of oblique polyhedral graphs with up to 12 vertices is found. Some additional properties of such graphs are also examined. — Refs: 5 titles.

УДК 519.68 + 681.3.06

Исследование методов организации визуальной обратной связи в аппаратно-программном комплексе "Бослаб" / Дунаев А.А., Валеев Т.Ф., Тарасов Е.А. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 42–54.

В статье описываются алгоритмы и программное обеспечение для поддержки исследований в физиологии и процесса биотренинга для медицинских и других целей. — Библиогр.: 3 назв.

Research of the methods of organization of a visual feedback in a hardware-software complex "BOSLAB" / Dunaev A.A., Valeev T.F., Tarasov E.A. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 42–54.

The paper presents the algorithms and software support for research in the field of physiology and for the process of biotraining (biological feedback) for medical and other purposes. — Refs: 3 titles.

УДК 519.68 + 681.3.06

Исследовательская система для анализа текстов на естественном языке / Дунаев А.А. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 55–67.

Статья посвящена разработке исследовательской системы для анализа текстов на естественном языке. Эта система позволит проводить лингвистические исследования и тестировать различные алгоритмы обработки текстов. — Библиогр.: 6 назв.

A research system for analysis of texts in a natural language / Dunaev A.A. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 55–67.

The article is devoted to the development of a research system for analysis of texts in a natural language which is intended to carry out linguistic research and to test various algorithms of text processing. — Refs: 6 titles.

УДК 519.68 + 681.3.06

Музеи и Интернет / Касьянов В.Н. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 67–84.

С появлением сети Интернет и развитием сетевых технологий музеи и другие учреждения культурного наследия переосмысливают свои задачи и возможности. Все большее число музеев принимает решение поддерживать свой веб-сайт (цифровой музей), чтобы расширить предоставление полезной информации о себе и привлечь новых пользователей.

Статья содержит описание некоторых новых возможностей, связанных с представлением музеев в сети Интернет. Описываются основные свойства музейных сайтов и виртуальных музеев. Излагаются подходы к унификации доступа и интеграции музейных информационных ресурсов. Рассматриваются методы адаптивной гипермедиа. Исследуются возможности открытых виртуальных музеев. Дается краткое описание работ по созданию открытого виртуального адаптивного музея по истории информатики в Сибири. — Библиогр.: 43 назв.

Museums and Internet / Kasyanov V.N. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 67–84.

With the advent of the digital age and the Web, museums and cultural heritage institutions are rethinking their roles. More and more museums make a decision to maintain their websites (digital museums) in order to provide useful information and attract new visitors.

The paper describes some new possibilities related to presentation of museums in the Web. The main properties of the museum sites and virtual museums are also described. Some approaches to access unification and integration of information resources are presented. The methods of adaptive hypermedia are considered. The possibilities of open virtual museums are studied. A short description of an open virtual adaptive museum of informatics history in Siberia is given. — Refs: 43 titles.

УДК 519.68 + 681.3.06

Адаптивная система поддержки дистанционного обучения программированию / Касьянова Е.В. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 85–112.

Рассматривается проект адаптивной системы WAPE, предназначенный для поддержки дистанционного обучения программирования. — Библиогр.: 11 назв.

An adaptive system of support for distant education in programming / Kasyanova E.V. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 85–112.

The paper presents a project of an adaptive system WAPE aimed at support for distant education in programming. — Refs: 11 titles.

УДК 519.68 + 681.3.06

Определение координат мобильного устройства в пространстве на основе изображений, получаемых от его видеокамеры / Козырева А.В. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 113–131.

Данная работа анализирует алгоритмы разработки программного инструмента для мобильных устройств, который позволит определять их положение в пространстве относительно предыдущего положения на основе снимков, сделанных их видеокамерой. — Библиогр.: 9 назв.

Detecting the mobile device position by analyzing images made by its camera / Kozyreva A.V. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 113–131.

This work presents analysis of algorithms constructed for the development of an application for a mobile device which could detect its position with respect to the previous one by analyzing images made by its video camera. — Refs: 9 titles.

УДК 519.68 + 681.3.06

О некоторых способах калибровки видеокамеры / Козырева А.В. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 132–141.

Данная работа посвящена процессу калибровки видеокамеры. Дано описание некоторых способов калибровки. — Библиогр.: 12 назв.

Some methods of video camera calibration / Kozyreva A.V. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 132–141.

The paper considers a process of video camera calibration. Some calibration techniques are described. — Refs: 12 titles.

УДК 519.68 + 681.3.06

Путевые разбиения в неориентированных графах / Мельников Л.С., Петренко И.В. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 142–152.

Количество вершин в наиболее длинном простом пути графа G обозначается $\tau(G)$. Подмножество S множества вершин $V(G)$ называется P_{n+1}-*свободным множеством* в G, если $\tau(G[S]) \leq n$. P_{n+1}-свободное множество максимального порядка в графе G называется *максимальным* P_{n+1}-*свободным множеством* графа G.

Известна гипотеза о том, что для любых G-графа и $n < \frac{\tau(G)}{2}$ существует P_{n+1}-свободное множество M в G, такое что $\tau(G - M) \leq \tau(G) - n$.

В настоящей работе приводится доказательство этой гипотезы для $n \leq 8$. — Библиогр.: 21 назв.

Path partitions in non-oriented graphs / Mel'nikov L.S., Petrenko I.V. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 142–152.

The number of vertices in the longest simple path of a graph G is denoted by $\tau(G)$. A subset S of the vertex set $V(G)$ is called P_{n+1}-*free set in the graph* G,

if $\tau(G[S]) \leq n$. P_{n+1}-free set of maximum cardinality in the graph G is called *maximal P_{n+1}-free set* of the graph G.

There is a well known hypothesis that for any graph G and $n < \frac{\tau(G)}{2}$ there exists a P_{n+1}-free set M such that $\tau(G - M) \leq \tau(G) - n$.

In this work we had proved this hypothesis for $n \leq 8$. — Refs: 21 titles.

УДК 519.68 + 681.3.06

Современные информационно-коммуникационные и цифровые технологии в сохранении культурного и научного наследия и развитии музейного дела / Несговорова Г.П. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 153–161.

Рассматриваются информационно-коммуникационные и цифровые технологии в музейном деле и их роль в повышении образовательного и культурного уровней широких слоев населения. Дается обзор проекта MINERVA PLUS, реализуемого в Евросоюзе и в России и посвященного оцифровке национального культурного и научного наследия. — Библиогр.: 5 назв.

Modern information, communication, and digital technologies in conservation of the cultural and scientific heritage and in the development of the museum activity / Nesgovorova G.P. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 153–161.

The article describes a block of high-level reducing optimizations for Sisal 3.0 compiler. The reducing optimizing transformations are transformations which do not decrease any qualities of a program. The block performs the following high-level optimizations: Common Subexpression Elimination, Constant Folding, Constant Propagation, IF-transformations, Dead Code Elimination, etc. — Refs: 5 titles.

УДК 519.68 + 681.3.06

Преобразования циклов, основанные на несингулярных матрицах / Осмонов Р. А., Штокало Д. Н. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 162–176.

В статье описан алгоритм, который преобразует гнездо цикла под действием несингулярного преобразования. Алгоритм в первую очередь подсчитывает границы цикла, используя метод исключения Фурье—Моцкина, а затем уточняет эти границы и вычисляет шаги цикла, используя метод, основанный на теории нормальной формы Эрмита. Алгоритм хорошо работает с унимодулярными преобразованиями, которые рассматриваются как частный случай. — Библиогр.: 8 назв.

Loop transformations based on nonsingular matrix / Osmonov R.A., Shtokalo D.N. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 162–176.

The paper presents an algorithm that rewrites a loop nest under any nonsingular loop transformation. This algorithm, first, calculates the loop bounds using the Fourier–Motzkin elimination method and, next, adjusts these loop bounds and calculates the loop strides using a method based on the theory of Hermite normal form. The algorithm works nicely with unimodular transformations being treated as a special case. — Refs.: 8 titles.

УДК 519.68 + 681.3.06

Обзор возможностей среды Significo на примере решения прикладной задачи / Серебренников А.Л. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 177–191.

Приведено решение задачи, направленной на определение параметров крови с помощью спектра, отражённого от поверхности тела. Процесс решения состоит из пяти этапов: описание физических процессов, проходящих при отражении света, и создание физической модели, анализ спектров, предварительная обработка данных, выбор наиболее подходящей архитектуры сети и алгоритма её обучения. В процессе описания этапов решения задачи рассматриваются особенности среды Significo. — Библиогр.: 7 назв.

A review of capabilities of the Significo environment by the example of deciding an applied problem / Serebrennikov A.L. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 177–191.

This paper describes the process of deciding a problem related to determination of blood parameters using the spectrum reflected from the body surface. The decision is found in four stages: description of physical processes taking place in light reflection and creation of a physical model, analysis of spectrums, data preprocessing, the choice of the most appropriate network architecture and of the algorithm of its learning. Some specific features of the Significo environment are also considered. — Refs: 7 titles.

УДК 519.68 + 681.3.06

Сравнительный анализ нейросетевых пакетов и место среды Significo среди них. Краткое описание среды / Серебренников А.Л. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 192–206.

Статья состоит из двух частей: в первой определяется место среды Significo среди множества современных нейропакетов, вторая часть содержит ее описание. Первая часть представляет общее описание приложений нейросетевого моделирования и сравнительный анализ нейропакетов (таких как NeuroSolutions, NeuralWorks, Process Advisor, NeuroShell 2 и др). Вторая часть дает описание базовой архитектуры среды и схемы взаимодействий ее элементов, а также определяет пользовательские группы, которые поддерживаются средой Significo. — Библиогр.: 6 назв.

Benchmark analysis of neural network packages and the place of the Significo environment among them. Short description of the environment / Serebrennikov A.L. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 192–206.

This paper consists of two parts: the place of the Significo environment among other modern neural packages is shown in the first part, and the second part presents its description. The first part generally describes the applications of neural network modeling and gives the benchmark analysis of a variety of neural packages like NeuroSolutions, NeuralWorks, Process Advisor, NeuroShell 2, etc. The second part presents the base architecture of the Significo environment and the interaction scheme of its elements, as well as its user groups. — Refs: 6 titles.

УДК 519.68 + 681.3.06

Обзор потоковых языков программирования / Стасенко А.П. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 207–216.

Статья рассматривает общие определяющие качества потоковых языков программирования и достаточно кратко описывает характерные особенности некоторых распространенных и экспериментальных потоковых языков, таких как Lucid, Id, Val, Post, Sisal и Пифагор. Представлены все существующие версии потокового языка Sisal, включая находящиеся в разработке, и их основные отличия друг от друга. — Библиогр.: 13 назв.

A review of dataflow programming languages / Stasenko A.P. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 207–216.

The paper considers the general defining qualities of dataflow programming languages and briefly describes the specific features of some widespread and experimental dataflow languages, such as Lucid, Id, Val, Post, Sisal and Pythagoras. All versions of the Sisal language, including those being under development, and their basic distinctions are presented. — Refs: 13 titles.

УДК 519.68 + 681.3.06

Нечеткая кластеризация по модифицированному методу c-средних и ее применение для обработки микрочиповых данных / Тараскина А.С. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 217–228.

Экспрессия генов в биомедицинских исследованиях изучается с помощью ДНК-микрочипов. Для анализа растущего объема данных, полученных по этой технологии, используется кластеризация. Рассматриваются методы кластеризации, которые делятся на иерархические и итерационные. Цель работы - разработка на основе нечеткого алгоритма c-средних нового алгоритма кластеризации, находящего близкое к оптимальному решение задачи кластеризации. — Библиогр.: 13 назв.

Fuzzy clusterization by the modified method of c-average and its application to microchip data processing / Taraskina A.S. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 217–228.

In biomedical research, gene expression is studied with the help of DNA-microchips and, to analyze the increasing data amount obtained by this technology, clusterization is used. The paper considers the methods of clusterization, which are divided to hierarchical and iteration ones, and is aimed at the development of a new clusterization algorithm that can find a solution of the clusterization problem close to optimal. — Refs: 13 titles.

УДК 519.68 + 681.3.06

Отказоустойчивость в распределенных сетях: проблемы консенсуса / Шкурко Д.В. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 229–248.

В статье представлен обзор исследований, связанных с проблемами соглашения. Описаны варианты постановки задач и приведены оценки качества алгоритмов. — Библиогр.: 53 назв.

Fault-tolerance of distributed systems: the consensus problem / Shkurko D.V. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 229–248.

The paper presents a review of investigations related to the consensus problems: variants of the problem statements and results related to the quality of solutions: time-, bit- and message-complexities. — Refs: 53 titles.

УДК 519.68 + 681.3.06

Универсальная система построения и администрирования лабораторных вебсайтов / Юрьев С.В. // Проблемы интеллектуализации и качества систем информатики. — Новосибирск, 2006. — С. 249–270.

Объектом исследования являлись системы управления содержанием сайтов иначе называемые CMS (от англ. Content management system). Целью работы ставилась разработка CMS оптимизированной для работы с лабораторными сайтами. В ходе работы были изучены существующие CMS, принципы их работы, требования, предъявляемые к таким системам, а также методы их оценки. После этого требования были проанализированы и дополнены собственными требованиями, возникшими в контексте того, что новая система должна служить базой для особого вида сайтов — лабораторных. Для этих сайтов характерна работа с информацией, нетипичной для среднестатистических сайтов, на которые рассчитаны существующие CMS. На следующем этапе работ была разработана CMS, удовлетворяющая вновь выдвинутым требованиям, а также обладающая рядом преимуществ по нескольким уже существовавшим требованиям по сравнению с другими CMS. На базе новой CMS был создан сайт Лаборатории конструирования и оптимизации программ ИСИ, на котором была проверена работа системы. — Библиогр.: 8 назв.

Universal system for building and management of laboratorial web-sites / Yur'ev S.V. // Problems of intellectualization and quality of informatics systems. — Novosibirsk, 2006. — P. 249–270.

The subject of research are web site content management systems also known as CMS. Aim of work is development of CMS, that is optimized for laboratorial web site management. Several existing CMS were examined with principles of their work, demands, made to them, and methods of their evaluation during the research. These demands were examined and expanded with demands for laboratorial web sites. Distinctive for these sites is the management of information that is atypical for average statistical sites, for which the existing CMS were made. The next stage of work was the development of new CMS, satisfying all demands and with number of advantages in comparison with other CMS. On the basis of the new CMS the Program Construction and Optimization Laboratory site was created. This site was used for examination of CMS functioning. — Refs: 8 titles.

ПРОБЛЕМЫ ИНТЕЛЛЕКТУАЛИЗАЦИИ И КАЧЕСТВА СИСТЕМ ИНФОРМАТИКИ

Под редакцией
проф. Виктора Николаевича Касьянова

Рукопись поступила в редакцию 15. 02. 2006
Ответственный за выпуск Г. П. Несговорова
Редактор З. В. Скок

Подписано в печать 19. 12. 2006
Формат бумаги 60 × 84 1/16 Объем 16,1 уч.-изд.л., 17,6 п.л.
Тираж 75 экз.

Центр оперативной печати "Оригинал 2", г. Бердск, 49-а, оф. 7
тел./факс 8 (241) 5 38 77