О восстановлении изображений по кодам в некоторых вырожденных случаях

Агниашвили П. Г. (Москва, МГУ им. М. В. Ломоносова)

colllapse@mail.ru

Одной из ключевых характеристик изображения является его код. По своему смыслу код должен отражать определенную общность в восприятии изображений. В рассматриваемом дискретно-геометрическом подходе общим признаком является аффинная эквивалентность изображений. Данное направление получило развитие в книге [1], где, в частности, исследуется возможность восстановления изображений по их кодам в двумерном и трехмерном случаях.

В предлагаемой работе исследуется аналогичная проблема в общем случае произвольной конечной размерности. Отдельное внимание уделяется вырожденному случаю изображений, лежащих в двух параллельных гиперплоскостях.

Всюду далее рассматривается пространство $\mathbb{R}^n$, $n \geqslant 2$. Под точкой с индексом $p \in \mathbb{N}$ будем понимать упорядоченную пару $(\mathbf{x}, p)$, где $\mathbf{x} = (x_1, \ldots, x_n)$ — вектор из $\mathbb{R}^n$.

Изображением первого рода в $\mathbb{R}^n$ называется конечное множество индексированных точек, не лежащих в одной гиперплоскости и занумерованных индексами $1, \ldots, k$ в случае k точек, $k \in \mathbb{N}$.

Для изображения первого рода A через $|A|$ будем обозначать число точек в изображении, а через $\mathbb{N}_{|A|} = \{1, \ldots, |A|\}$ — множество индексов у точек изображения. Здесь и далее запись $\mathbb{N}_k$ используется для обозначения начального отрезка натурального ряда.

Два изображения первого рода A_1 и A_2, состоящие из одинакового числа точек ($|A_1| = |A_2|$), называются *a-эквивалентными*, если существует аффинное преобразование, при котором каждая точка из A_1 с индексом $p \in \mathbb{N}_{|A_1|}$ отображается в точку из A_2 с тем же индексом $p \in \mathbb{N}_{|A_2|}$.

Изображением второго рода называется класс всех попарно a-эквивалентных изображений первого рода.

Далее под изображением понимается изображение первого или второго рода. Род изображения не уточняется в тех случаях, когда исследуемые свойства сохраняются при аффинных преобразованиях.

Рассмотрим изображение A. Введем произвольную аффинную систему координат в $\mathbb{R}^n$ и пусть $(x_1^p,\ldots,x_n^p)$ — координаты точки с индексом $p \in \mathbb{N}_{|A|}$ в этой системе координат. Для произвольных индексов $r_1,\ldots,r_{n+1},s_1,\ldots,s_{n+1} \in \mathbb{N}_{|A|}$ определим индексированное число $\mu_{s_1\ldots s_{n+1}}^{r_1\ldots r_{n+1}}$ по формуле:

$$\mu_{s_1\ldots s_{n+1}}^{r_1\ldots r_{n+1}} = \begin{vmatrix} x_1^{r_1} & \cdots & x_n^{r_1} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{r_{n+1}} & \cdots & x_n^{r_{n+1}} & 1 \end{vmatrix} \Bigg/ \begin{vmatrix} x_1^{s_1} & \cdots & x_n^{s_1} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{s_{n+1}} & \cdots & x_n^{s_{n+1}} & 1 \end{vmatrix}$$

В случае равенства знаменателя нулю полагаем, что значение $\mu_{s_1\ldots s_{n+1}}^{r_1\ldots r_{n+1}}$ не определено, и обозначаем такой случай знаком ∞. В случае ненулевых определителей индексированное число является отношением ориентированных объемов n-симплексов на соответствующих точках.

М-кодом изображения A называется множество всех индексированных чисел $\mu\text{-}T_A = \{\mu_{s_1\ldots s_{n+1}}^{r_1\ldots r_{n+1}} | r_i, s_j \in \mathbb{N}_{|A|}\}$.

Симплексным набором индексов для м-кода $\mu\text{-}T_A$ называется набор $S = (i_1,\ldots,i_{n+1})$, для которого $\mu_{\mathbf{i_1},\ldots,\mathbf{i_{n+1}}}^{r_1,\ldots,r_{n+1}} \neq \infty$ при любых $r_i \in \mathbb{N}_{|A|}$. Другими словами, точки с индексами $i_1,\ldots,i_{n+1}$ не лежат в одной гиперплоскости, на что и указывает название набора.

Обозначим через μ_j^s элемент м-кода, у которого нижний набор является симплексным набором $(i_1,\ldots,i_{n+1})$, а верхний набор отличается от симплексного в j-ой позиции: $(i_1,\ldots,i_{j-1},s,i_{j+1},\ldots,i_{n+1})$.

Ключевым подмножеством м-кода $\mu\text{-}T_A$ относительно симплексного набора S называется множество $\{\mu_j^s | s \in \mathbb{N}_{|A|}, j \in \mathbb{N}_{n+1}\}$.

Имеют место следующие соотношения:

$$\mu_1^s + \ldots + \mu_{n+1}^s = 1 \tag{1}$$

$$\mu_{s_1\ldots s_{n+1}}^{r_1\ldots r_{n+1}} = \begin{vmatrix} \mu_1^{r_1} & \cdots & \mu_{n+1}^{r_1} \\ \vdots & \ddots & \vdots \\ \mu_1^{r_{n+1}} & \cdots & \mu_{n+1}^{r_{n+1}} \end{vmatrix} \Bigg/ \begin{vmatrix} \mu_1^{s_1} & \cdots & \mu_{n+1}^{s_1} \\ \vdots & \ddots & \vdots \\ \mu_1^{s_{n+1}} & \cdots & \mu_{n+1}^{s_{n+1}} \end{vmatrix} \tag{2}$$

Данные соотношения используются, в частности, при доказательстве теорем 1 и 2.

Теорема 1. *Между множеством м-кодов и множеством изображений второго рода существует биекция, сопоставляющяя каждому изображению его м-код.*

Кодом изображения A называется множество всех индексированных чисел $T_A = \{\rho^{r_1\dots r_{n+1}}_{s_1\dots s_{n+1}} | r_i, s_j \in \mathbb{N}_{|A|}\}$, где $\rho^{r_1\dots r_{n+1}}_{s_1\dots s_{n+1}} = |\mu^{r_1\dots r_{n+1}}_{s_1\dots s_{n+1}}|$. Данное определение согласуется и обобщает определение кода из [1].

Рассмотрим два изображения A и $\hat{A}$ со следующими свойствами: $|A| = |\hat{A}|$, $S = (i_1, \dots, i_{n+1})$ — общий симплексный набор, и ключевые подмножества м-кодов равны по модулю ($|\mu^s_j| = |\hat{\mu}^s_j|$). Положим $M^s_+ = \{i_j \in S | \mu^s_j = \hat{\mu}^s_j \neq 0\}$ и $M^s_- = \{i_j \in S | \mu^s_j = -\hat{\mu}^s_j \neq 0\}$.

Разделяющим разбиением набора S называется пара множеств (S_1, S_2), для которых выполняется: $S = S_1 \sqcup S_2$ и для любого индекса $s \in \mathbb{N}_{|A|}$ либо $M^s_+ \subseteq S_1, M^s_- \subseteq S_2$, либо $M^s_+ \subseteq S_2, M^s_- \subseteq S_1$.

Здесь и далее $S = S_1 \sqcup S_2$ означает $S = S_1 \cup S_2$ и $S_1 \cap S_2 = \varnothing$.

Теорема 2. *Коды изображений A и $\hat{A}$ с перечисленными свойствами совпадают тогда и только тогда, когда существует разделяющее разбиение симплексного набора S.*

Рассмотрим произвольную аффинную систему координат в $\mathbb{R}^n$ и пусть $M = \{\mathbf{x}^{s_1}, \dots, \mathbf{x}^{s_m}\}$ — некоторое множество точек.

Гранью $<M>$, порожденной множеством M, называется множество $<M> = \{\alpha_1\mathbf{x}^{s_1} + \dots + \alpha_m\mathbf{x}^{s_m} | \sum_{i=1}^m \alpha_i = 1\}$.

Любая грань $<M>$ является аффинным подпространством, то есть сдвигом соответствующего линейного подпространства L на некоторый вектор $\mathbf{x} \in <M>$.

Размерностью $\dim(M)$ грани $<M>$ называется размерность соответствующего линейного подпространства L.

Изображение A называется *допустимым*, если для любых двух непересекающихся граней $<M_1>$ и $<M_2>$, содержащих все точки изображения A, выполняется: $\dim(M_1) + \dim(M_2) = \dim(A) - 1$.

Теорема 3. *Коду соответствует единственное изображение второго рода тогда и только тогда, когда изображение является допустимым.*

Теорема 3 является аналогом утверждений, доказанных в [1] для случаев $\mathbb{R}^2$ и $\mathbb{R}^3$, а также общего результата, полученного Руденко А. Д. для случая пространства произвольной конечной размерности. Согласно этим результатам, коду соответствует единственное изображение с точностью до a-эквивалентности, если оно не лежит в двух параллельных гиперплоскостях. Теорема 3 применима и к такому вырожденному случаю. Следующее утверждение более детально разбирает данный вопрос.

Утверждение. *Пусть* $A = A_1 \sqcup A_2$, *и* $<A_1> \cap <A_2> = \varnothing$. *Изображение* A *допустимо* $\Leftrightarrow \dim(A_1) + \dim(A_2) = \dim(A) - 1$, *и* A_1, A_2 *допустимы.*

Используем полученный результат для классификации допустимых изображений, лежащих в двух параллельных гиперплоскостях, в случаях $\dim(A) = 2$ и $\dim(A) = 3$.

Случай $\dim(A) = 2$. В данном случае изображение лежит на двух параллельных прямых. Все точки расположены на прямой и в точке, не лежащей на этой прямой.

Случай $\dim(A) = 3$. В данном случае изображение лежит на двух параллельных плоскостях. Возможно два варианта:

1) Все точки расположены на плоскости и в точке, не лежащей на этой плоскости. При этом точки на плоскости не могут быть расположены на двух параллельных прямых.
2) Все точки расположены на двух скрещивающихся прямых.

Автор выражает благодарность профессору Козлову Вадиму Никитовичу за постановку задачи и научное руководство, а также Алексееву Дмитрию Владимировичу за ценные замечания.

Список литературы

[1] Козлов В. Н. Элементы математической теории зрительного восприятия. — М.: Изд-во Центра прикладных исследований при механико-математическом факультете МГУ, 2001.

[2] Михалев А. А., Михалев А. В. Начала алгебры: Часть I. — М.: Интернет-университет информационных технологий, 2005.

[3] Клейн Ф. Элементарная математика с точки зрения высшей: Том 2. Геометрия. — М.: Наука, 1987.

Использование метода В. Н. Козлова в образовательном процессе на кафедре МаТИС

Алексеев Д. В. (Москва, МГУ им. М. В. Ломоносова)

dvalex@rambler.ru

В работе рассматривается упрощенный вариант алгоритма В. Н. Козлова для восстановления трехмерных изображений по их плоским проекциям. Также показано использование упрощенного алгоритма В. Н. Козлова в образовательном процессе на кафедре математической теории интеллектуальных систем МГУ им. М. В. Ломоносова.

1. Введение

В работе используются следующие обозначения: точки на плоскости и в пространстве обозначаются прописными буквами: $A, B, C, \ldots$ Их координаты на плоскости и в пространстве обозначаются строчными буквами: (a_x, a_y, a_z), (b_x, b_y), $\ldots$ Прямая, проходящая через точки A и B обозначается (AB), плоскость, проведенная через точки A, B и C обозначается $\pi(ABC)$.

Алгоритм В. Н. Козлова восстановления трехмерного тела по плоским проекциям описан в работе [1]. Суть его такова — даны две проекции трехмерного тела (под телом понимается конечное множество точек в пространстве). Требуется восстановить исходное тело с точностью до аффинной эквивалентности, то есть построить множество точек, которое было бы аффинно эквивалентно исходному.

Краткое изложение алгоритма приводится для основного случая.

Пусть A', B', C', D' и E' — проекции точек A, B, C, D и E на плоскость π_1, а A'', B'', C'', D'' и E'' — на плоскость π_2.

Определение ([1]) Рассмотрим на π_1 и π_2 четверку троек точек $((A'B'C')(A''B''C'')(C'E'D')(C''E''D''))$ такую, что $A'B'C'$ и $A''B''C''$ — треугольники, точки D' и E' лежат вне плоскости треугольника $A'B'C'$ (и, соответственно, точки D'' и E'' лежат вне плоскости треугольника $A''B''C''$) и из троек $C'D'E'$ и $C''D''E''$ хотя бы одна является треугольником. Такую четверку будем называть *пра-*

вильной. Определение того, что точка лежит в плоскости треугольника приведено в [1].

Пусть проекции точек A, B, C, D, E образуют правильную четверку. Тогда, очевидно, существует единственное аффинное отображение, переводящее точки A'' , B'' и C'' в точки A', B', C'. Пусть при этом отображении точки D'' и E'' переходят в точки $\tilde{D}$ и $\tilde{E}$. Пусть F' — точка пересечения прямых $D'E'$ и $\tilde{D}\tilde{E}$. Тогда прямая CF является проекцией прямой $l = \pi(ABC) \cap \pi(CDE)$. Следовательно, можно рассмотреть произвольные (не лежащие в одной плоскости) точки A_0, B_0, C_0 и D_0 в пространстве, построить прямую l_0, соответствующую прямой $C'F'$ и восстановить положение точки E.

2. Упрощенная версия алгоритма В. Н. Козлова

Упрощение алгоритма было предложено в процессе преподавания этой темы на 3 курсе мех.-мат. ф-та. Не ограничивая общности рассуждений можно считать, что точки A', B' и C' обладают координатами $(a'_x, a'_y) = (1, 0)$, $(b'_x, b'_y) = (0, 1)$ и $c'_x, c'_y) = (0, 0)$, Действительно, всегда можно преобразовать эти точки с помощью некоторого аффинного преобразования, или сразу рассматривать аффинную систему координат с началом в точке C' и базисными векторами $C'A'$ и $C'B'$.

Существует единственное аффинное преобразование, переводящее треугольник $A''B''C''$ в треугольник $A'B'C'$. Пусть при этом отображении точки D'' и E'' переходят в точки $\tilde{D}$ и $\tilde{E}$. Пусть F' — точка пересечения прямых $D'E'$ и $\tilde{D}\tilde{E}$. Обозначим $\lambda = \frac{\tilde{E}F'}{\tilde{D}F'}$. Рассмотрим точки в пространстве $A_0(1, 0, 0)$, $B_0(0, 1, 0)$, $C_0(0, 0, 0)$, $D_0(d'_x, d'_y, 1)$ и $E_0(e'_x, e'_y, \lambda)$. Для указанных точек верна следующая

Теорема 1. *Существуют направления проекции, такие, что проекции точек A_0, B_0, C_0, D_0 и E_0 на плоскость $z = 0$ аффинно эквивалентны проекциям A', B', C', D', E' и A'', B'', C'', D'', E''.*

Доказательство. Выберем базис следующим образом: начало координат в точке C', оси x и y проходят через точки A' и B' ось Z перпендикулярна этой плоскости. Очевидно, что проекцией точек на эту плоскость по направлению, параллельному оси Z, являются точки A', B', C', D', E'.

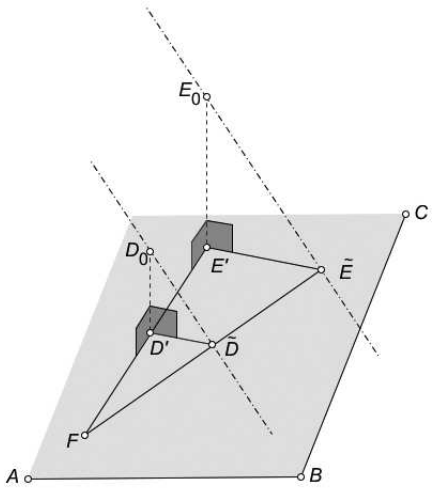


Рис. 1. Восстановление трехмерного изображения.

Рассмотрим проекцию точек A_0, B_0, C_0, D_0, E_0 параллельно прямой $(E_0\tilde{E})$ (см. Рис. 1). Очевидно, проекцией точки E_0 будет точка E. Поскольку точки $D'E'F'$ являются проекциями пространственных точек D, E, F, а отношения отрезков при параллельном проектировании сохраняются, то $\frac{E'F'}{D'F'} = \frac{EF}{DF}$. Аналогично $\frac{E''F''}{D''F''} = \frac{EF}{DF}$, и, следовательно, $\frac{\tilde{E}F'}{\tilde{D}F'} = \frac{EF}{DF}$, поскольку аффинные преобразования сохраняют отношения отрезков на прямой. Из вышеуказанных пропорций вытекает подобие треугольников $\triangle D'\tilde{D}F' \simeq \triangle E'\tilde{E}F'$. Следовательно, отрезки $\tilde{D}D'$ и $\tilde{E}E'$ лежат на параллельных прямых и их отношение равно $\frac{\tilde{E}E'}{\tilde{D}D'} = \frac{\tilde{E}F'}{\tilde{D}F'} = \lambda$. Заметим, что из выбора точек D_0 и E_0 вытекает, что отношение $\frac{E_0E'}{D_0D'} = \lambda$. Следовательно, треугольники $\triangle D_0D'\tilde{D}$ и $\triangle E_0E'\tilde{E}$ подобны. Поскольку сторона E_0E' параллельна D_0D' по построению, а $E'\tilde{E}$ параллельна $D'\tilde{D}$ из подобия треугольников $\triangle D'\tilde{D}F'$ и $\triangle E'\tilde{E}F'$, то стороны $E_0\tilde{E}$ и $D_0\tilde{D}$ тоже будут параллельны. Следовательно, проекцией точки D_0 по направлению $E_0\tilde{E}$ на плоскость $\pi(A'B'C')$ будет точка $\tilde{D}$, что и требовалось доказать.

3. Программа для автоматической генерации задач

Для генерации задач с «хорошими» (то есть целочисленными) ответами была написана программа на языке MATLAB, листинг приводится:

```
function f = generate_probleb()
%% Points  A,B,C are fixed , others -- arbitrary
 A0 =  [ 1; 0], B0 =  [ 0; 1],  C0 =  [ 0; 0],   f=0;
 X0 =  [ 2; 2]; % The pivot point
 D01 = [ 4; 0] , D02 = [ 2; -2]; % Arbitrary points
 factor = -0.5; %  EX:DX ratio
 Transform1 = [  [  -1  3   1  ]  ;  [  1  1  0  ] ];
 Transform2 = [  [   2  -1  -3 ]  ;  [  -1  1  2 ] ];
    % Arbitrary transform matrixes
 E01 = X0 + factor * (D01 - X0), E02 = X0 + factor * (D02
     - X0);
 names =    { 'a';  'b';    'c';    'd';    'e';    'x'};
 points1 =  { A0;   B0;     C0;     D01;    E01;    X0};
 points2 =  { A0;   B0;     C0;     D02;    E02;    X0};
NUM_POINTS = 6;
 points1tr= cell(NUM_POINTS,1), points2tr= cell(
    NUM_POINTS,1);
 for i =1:NUM_POINTS
    points1tr{i} = AffTrans(points1{i}, Transform1);
    points2tr{i} = AffTrans(points2{i}, Transform2);
 end
% Printing problem and answer
 fprintf(1,'Условие:\n\nпроекции на $\\Pi''$:␣\n');
 Disp(1,names, points1tr,NUM_POINTS-1, '''');
 fprintf('\n\nи проекции на $\\Pi''''$:␣\n');
 Disp(1,names, points2tr,NUM_POINTS-1, '''''');
 fprintf(1, '\n==========\nОтветы:\n');
 Disp(1,{names{6}}, {points1tr{6}}, 1, '''');% position X
     '
 Disp(1,{names{6}}, {points2tr{6}}, 1, '''''');% position
     X''
 fprintf(1, '␣\nуравнение прямой $L''$\n');
 PrintLineEquation(1, points1tr{3}, points1tr{6});% line
    C'X'
 fprintf(1, '␣\nуравнение прямой $L''''$\n');
```

```
 PrintLineEquation(1, points2tr{3}, points2tr{6});% line
     C''X''
 z = [ 0; 0; 0; 1; factor; 0];
 fprintf('\n\n Трехмерные координаты \n');
 Disp3(1, names, points1, z, NUM_POINTS - 1, ' ');
 D3=AffTrans(D01,Transform2), E3=AffTrans(E01,Transform2)
     ;
end
% some useful functions (headers only):
function Xt = AffTrans(X, Matr) % Makes affine transform
    with point X
function d = Disp3(hFile, names, coords, z, N, suffix) %
    prints points
function str = number2string(n) % Pretty printer for
    rational numbers
function f = PrintLineEquation(hFile, A, B) % Pretty
    printer for line equations
function [ k, b ] = GetLineEquation(A,B)% line equation
    by 2 points
```

Автор выражает благодарность профессору Вадиму Никитовичу Козлову за полезные замечания и внимание к работе.

Список литературы

[1] Козлов В. Н. Элементы математической теории зрительного восприятия. — М.: Изд-во ЦПИ, 2001. С. 36–52.

[2] Кудрявцев В. Б., Гасанов Э. Э., Подколзин А. С. Введение в теорию интеллектуальных систем. — М.: Изд-во. отд. ф-та ВМиК, МАКС Пресс, 2006. С. 26–32.

Об алгебраизации модели k-гиперпространства СХ-гипертопографов: операции трансформации — развития

Баранович А. Е., Боровиков Д. В., Лакуша Е. Л.
(Москва, Российский государственный гуманитарный университет, Институт информационных наук и технологий безопасности)
barae@rambler.ru, dvborovikov@gmail.com

В работах [1, 2, 4] исследована универсальная модель информационной составляющей сложных систем, параметрически поглощающая известные классы моделей представления декларативных знаний. В качестве абстрактной экспликации модели определено k-гиперпространство СХ-гипертопографов (СХ-$\eta\tau$-графов) [1, 2]. В работе [4] данная модель обобщена до понятия СХ-гипертопосети, моделирующей как декларативные, так и процедурные знания в их сетевой интерпретации.

СХ-$\eta\tau$-граф представляет собой модель статического состояния сложной системы. Для моделирования динамики функционирования системы, наряду с сетевым подходом (нейронные сети, сети Петри и т. п.) в [1] предложено использовать и автоматно-алгебраический подход. Данное направление связано с представлением модели в виде строго формализованной алгебраической системы и опирается на хорошо разработанный механизм алгебраизации множества-носителя (алгебраическая система [6] — множество G («носитель») с заданным на нём набором операций и отношений («сигнатура»), удовлетворяющим некоторой системе аксиом).

В отношении модели классического графа наиболее распространены следующие алгебраические операции (бинарные и унарные) [6]: объединение графов, пересечение графов, удаление вершины, добавление вершины и др. В свою очередь, при рассмотрении операций, связанных с обработкой знаний, наряду с указанными типами операций, возникают и новые, предметно-ориентированные, в частности, операции трансформации/развития, слияния [1], сборки [5] и т. д.

В настоящей работе излагаются результаты алгоритмического синтеза операций трансформации и развития СХ-$\eta\tau$-графов в сиг-

натуре $\Psi^{\Phi}_{G^k_{\eta\tau}x}$, где Φ есть некоторое, вполне определенное, множество *элементарных трансформаций*. Синтезированные операции положены в основу процедур и программных средств автоматизации использования декларативно-процедурных знаний [3].

В работе [1] введены понятия *элементарных трансформаций, трансформации, развития* и *глубины трансформации* графа g, связанные с вполне определенными алгоритмическими преобразованиями g на множестве $G \equiv \{g\}$. *Элементарные трансформации* порождают на G *бинарные алгебраические операции* типа $\varphi(g, g^*) \equiv \overline{g}$, где $g,\ g^*,\ \overline{g} \in G$. Там же ([1]) доказаны утверждения о *порождении* множеством элементарных трансформаций Φ *сигнатуры операций* Ψ^{Φ}_{G} на множестве G и *о существовании* для любых конечномерных графов $g,\ \overline{g} \in G$, по крайней мере, одной операции трансформации $TRF(g) \equiv \bar{g}$, преобразующей исходный (трансформируемый) граф g в граф (результирующий) $\bar{g}$.

Редуцируем задачу алгебраизации классической модели графа на случай СХ-$\eta\tau$-графов.

Определение 1. Элементарная трансформация СХ-$\eta\tau$-графа $g^k_{\eta\tau}x$: $(V^{\mathrm{k}}_{\tau}, E^{\mathrm{k}}_{\eta\tau}, \{P^{V^k_\tau}, P^{E^k_{\eta\tau}}\})$ на $G^k_{\eta\tau}x$ есть *преобразование* типа: *добавить* к $g^k_{\eta\tau}x$ произвольное *гипертопоребро* $e^k_{\eta\tau} \in \tilde{X}^{(k+1)}$ (обозначим его через $\varphi^{+}_{e^k_{\eta\tau}}(g^k_{\eta\tau}x)$); *добавить* к $g^k_{\eta\tau}x$ произвольную *гипертоповершину* $v^k_\tau \in X^{(k)}$ $(\varphi^{+}_{v^k_\tau}(g^k_{\eta\tau}x))$; *исключить* из $g^k_{\eta\tau}x$ произвольное *гипертопоребро* $e^k_{\eta\tau} \in \tilde{X}^{(k+1)}$ $(\varphi^{-}_{e^k_{\eta\tau}}(g^k_{\eta\tau}x))$; *исключить* из $g^k_{\eta\tau}x$ произвольную *гипертоповершину* $v^k_\tau \in X^{(k)}$ $(\varphi^{-}_{v^k_\tau}(g^k_{\eta\tau}x))$; *добавить цвет* $q \in P$ к подмножеству цветов произвольной *гипертоповершины* v^k_τ СХ-$\eta\tau$-графа $g^k_{\eta\tau}x$ $(\varphi^{+}_{qv^k_\tau}(g^k_{\eta\tau}x))$; *исключить цвет* $q \in P$ из подмножества цветов произвольной *гипертоповершины* v^k_τ СХ-$\eta\tau$-графа $g^k_{\eta\tau}x$ $(\varphi^{-}_{qv^k_\tau}(g^k_{\eta\tau}x))$; *добавить цвет* $q \in P$ к подмножеству цветов произвольного *гипертопоребра* $e^k_{\eta\tau}$ СХ-$\eta\tau$-графа $g^k_{\eta\tau}x$ $(\varphi^{+}_{qe^k_{\eta\tau}}(g^k_{\eta\tau}x))$; *исключить цвет* $q \in P$ из подмножества цветов произвольного *гипертопоребра* $e^k_{\eta\tau}$ СХ-$\eta\tau$-графа $g^k_{\eta\tau}x$ $(\varphi^{-}_{qe^k_{\eta\tau}}(g^k_{\eta\tau}x))$; *пустое преобразование* $\varphi_{\varnothing}(g^k_{\eta\tau}x)$.

Элементарные трансформации порождают на $G^k_{\eta\tau}x$ *бинарные алгебраические операции* типа $\varphi(g^k_{\eta\tau}x, {}^*g^k_{\eta\tau}) \equiv \overline{g^k_{\eta\tau}x}$, где $g^k_{\eta\tau}x, \overline{g^k_{\eta\tau}x} \in G^k_{\eta\tau}x$ и ${}^*g^k_{\eta\tau}$ — обобщенное обозначение для вышеуказанных порождающих элементов СХ-$\eta\tau$-графа. Базовым отношением на $G^k_{\eta\tau}x$ при этом являются бинарные отношения *тождества/различия* элементов $G^k_{\eta\tau}x$.

Утверждение 1. *Множество элементарных трансформаций* Φ — $\{\varphi^+_{v^k_\tau}(g^k_{\eta\tau}x),\ \varphi^-_{v^k_\tau}(g^k_{\eta\tau}x),\ \varphi^+_{e^k_{\eta\tau}}(g^k_{\eta\tau}x),\ \varphi^-_{e^k_{\eta\tau}}(g^k_{\eta\tau}x),\ \varphi^+_{qv^k_\tau}(g^k_{\eta\tau}x),\ \varphi^-_{qv^k_\tau}(g^k_{\eta\tau}x),\ \varphi^+_{qe^k_{\eta\tau}}(g^k_{\eta\tau}x),\ \varphi^-_{qe^k_{\eta\tau}}(g^k_{\eta\tau}x),\ \varphi_\varnothing(g^k_{\eta\tau}x)\}$ *порождает сигнатуру* $\Psi^\Phi_{G^k_{\eta\tau}x}$ *на множестве* $G^k_{\eta\tau}x$.

Определение 2. *Трансформация* СХ-$\eta\tau$-графа $TRF(g^k_{\eta\tau}x) \equiv \overline{g^k_{\eta\tau}x}$ на $G^k_{\eta\tau}x$ есть последовательное выполнение некоторой, вполне определенной, совокупности элементарных трансформаций, преобразующих исходный трансформируемый СХ-$\eta\tau$-граф $g^k_{\eta\tau}x$ в результирующий СХ-$\eta\tau$-граф $\overline{g^k_{\eta\tau}x}$.

Определение 3. *Развитие* СХ-$\eta\tau$-графа $RV(g^k_{\eta\tau}x) \equiv \overline{g^k_{\eta\tau}x}$ на $G^k_{\eta\tau}x$ есть трансформация исходного СХ-$\eta\tau$-графа $g^k_{\eta\tau}x$ в СХ-$\eta\tau$-граф $\overline{g^k_{\eta\tau}x}$ при условии $g^k_{\eta\tau}x \subseteq \overline{g^k_{\eta\tau}x}$.

Утверждение 2. *Для любых конечномерных СХ-*$\eta\tau$*-графов* $g^k_{\eta\tau}x$ *и* $\overline{g^k_{\eta\tau}x}$ *множества* $G^k_{\eta\tau}x$ *существует, по крайней мере, одна операция трансформации* $TRF(g^k_{\eta\tau}x) \equiv \overline{g^k_{\eta\tau}x}$*, преобразующая трансформируемый СХ-*$\eta\tau$*-граф* $g^k_{\eta\tau}x$ *в результирующий СХ-*$\eta\tau$*-граф* $\overline{g^k_{\eta\tau}x}$.

Определение 4. *Глубина трансформации* $\gamma\{TRF(g^k_{\eta\tau}x) \equiv \overline{g^k_{\eta\tau}x}\}$ на $G^k_{\eta\tau}x$ есть число элементарных трансформаций в трансформации $TRF(g^k_{\eta\tau}x) \equiv \overline{g^k_{\eta\tau}x}$.

В процессе исследования операции трансформации TRF СХ-$\eta\tau$-графов $g^k_{\eta\tau}x, \overline{g^k_{\eta\tau}x}$, представленных множествами $V^k_\tau, \overline{V^k_\tau}, E^k_{\eta\tau}, \overline{E^k_{\eta\tau}}$ и подмножествами цветов $\{\{p_{v^k_\tau}\}\}$, $\{\{\overline{p_{v^k_\tau}}\}\}$, $\{\{p_{e^k_{\eta\tau}}\}\}$, $\{\{\overline{p_{e^k_{\eta\tau}}}\}\}$, предложена и обоснована теоретико-множественная интерпретация её алгоритмической реализации — ТСХ-процедура.

Утверждение 3. *ТСХ-процедура на конечных СХ-$\eta\tau$-графах всегда сходится за конечное число шагов.*

Утверждение 4. *ТСХ-процедура является процедурой трансформации СХ-$\eta\tau$-графов минимальной глубины.*

Утверждение 5. *Операционная сложность алгоритмической реализации ТСХ-процедуры не превосходит* $T(n) = O\left(m \cdot \log s \cdot \log t\right)$, *где* $m = \max\left\{|V_\tau^k| \cdot |\{\{p_{v_\tau^k}\}\}|,\ |E_{\eta\tau}^k| \cdot |\{\{p_{e_{\eta\tau}^k}\}\}|\right\}$, $s = \max\{|V_\tau^k|, |E_{\eta\tau}^k|\}$, $t = \max\left\{|\{\{p_{v_\tau^k}\}\}|,\ |\overline{\{\{p_{v_\tau^k}\}\}}|, |\{\{p_{e_{\eta\tau}^k}\}\}|,\ |\overline{\{\{p_{e_{\eta\tau}^k}\}\}}|\right\}$.

Для операции развития RV (RCX-процедура) СХ-$\eta\tau$-графов получены аналогичные результаты.

В завершении изложения подчеркнем, что *алгебраическая модель* характеризует статическую картину отношений элементов моделируемой системы. Моделью же *преобразования* статических состояний системы является *операция* [4]. Объединение моделей статического состояния систем (*алгебраическая модель*) и модели изменений состояний модели в текущем настоящем (*алгебра*) порождает, в итоге, динамическую модель *алгебраической системы*. Введение операций на СХ-$\eta\tau$-графах влечет порождение вполне определенной *метаалгебры* («алгебры моделей») или в иной, вышеупомянутой постановке, *метаалгебраической* системы (с вполне определенным множеством-носителем V).

Список литературы

[1] Баранович А. Е. Основы алгебраизации модели: алгоритмическая концепция // Семиотико-хроматические гипертопографы. Введение в аксиоматическую теорию: информационный аспект (прил. 1). — М.: МО РФ, 2003.

[2] Баранович А. Е. K-гиперпространство семиотико-хроматических гипертопографов как универсальная модель представления фактографических знаний // Мат. IX Междунар. конф. «Интеллект. сист. и компьют. науки». Т. 1. Ч. 1. — М.: МГУ, 2006. С. 53–55.

[3] Баранович А. Е., Баранович А. А., Лишин Н. А. Исчисление ценности прагматической информации в интеллектуальной програм-

мной среде «АКСИОН» // Тр. XI национ. конф. по искусственному интеллекту с междунар. участ. Т. 3. — М.: ЛЕНАНД, 2008. С. 364–372.

[4] Баранович А. Е. Семиотико-хроматические гипертопосети: унифицированная модель представления знаний // Открытые семантические технологии проектирования интеллектуальных систем: материалы Междунар. научн.-техн. конф. — Минск: БГУИР, 2011. С. 71–86.

[5] Зайцев Д. В. О сложности сборки и вложения графов / Диссер. на соиск. учен. степ. канд. физ.-мат. наук. — М.: МГУ, мех.-мат. фак-т, 2007.

[6] Капитонова Ю. В., Летичевский А. А., Луцкий Г. М. Лекции по дискретной математике. — СПб.: БХВ-Петербург, 2004.

[7] Яблонский С. В. Введение в дискретную математику: Учебное пособие для вузов / Под ред. В. А. Садовничего. 3-е изд., стер. — М.: Высш. шк., 2002.

О некоторых результатах сравнительного анализа моделей музыкального и вербального текстов

Баранович А. Е., Иглицкая С. М.
(Москва, Российский государственный гуманитарный университет, Институт информационных наук и технологий безопасности)
barae@rambler.ru, sofa.sofa@mail.ru, www.samtcenter.ru

Обширный спектр проблем, связанных с системным анализом и математическим моделированием музыкального текста (МТ), представляет собой весьма мало изученную область. Обзор доступных источников показывает, что существующие методология и инструментарий исследований не позволяют решить целый ряд задач, связанных с характеризацией естественнонаучного базиса МТ, определяемого его принадлежностью к сфере информационной коммуникации вполне определенного подкласса класса антропоморфных интеллектуальных систем (ИС) [6].

Один из возможных подходов к исследованию МТ основывается на использовании (по аналогии) известных (вполне определённым образом модифицированных), методов структурно-алгебраического и семантико-прагматического анализа вербального текста (ВТ). В настоящей работе отражены некоторые результаты исследования МТ так называемого строгого стиля, представленного моделью дискретных сообщений Дж. фон Неймана нулевого и первого приближений, в задаче оценки пропускной способности дискретного канала связи по К. Шеннону. Последующие исследования предполагают введение в разрабатываемый аппарат моделирования МТ информационных моделей семантико-прагматического типа, ориентированных на исследование особенностей семантической музыкальной коммуникации коллектива ИС.

Выдвигалась следующая конструктивная гипотеза: $C_m > C_v$, где C_v и C_m — пропускные способности каналов соответственно вербальной и музыкальной коммуникации, пропорциональные энтропии источника дискретных сообщений. Для вычисления последней используется формула Шеннона: $H(A) = -\sum_{i=1}^{m} P(a_i) \log P(a_i)$; для случая коррелированных символов (условной энтропии): $H(A'/A) =$

$-\sum_{i=1}^{m} P(a_i) \sum_{j=1}^{m} P(a'_j/a_i) \log P(a'_j/a_i)$, где a_i, a_j — символы алфавита A источника сообщений, $|A| = m$, $P(a_i)$ — вероятность появления символа a_i, $P(a'_j/a_i)$ — условная вероятность появления символа a_j после символа a_i [5].

Для представления МТ в виде конечной совокупности конечных последовательностей символов, на основе существующего музыкального направления так называемого строгого стиля (С.С.) (историческое и художественно-стилистическое понятие, относящееся к хоровой полифонической музыке эпохи Ренессанса [7, 8]) был составлен определенный «музыкальный алфавит», генерируемые на базе которого мелодии могут служить моделью композиторских мелодий данного стиля разной степени приближения.

Последовательная семиотическая модель Дж. фон Неймана [1] представляет собой конечные (длины l) последовательности символов алфавита A:
$A : \sigma_i = \sigma_{i_1}, \ldots, \sigma_{i_l}, l \leqslant L \leqslant \infty, \sigma_{i_j} \in A, |A| = z, z \leqslant Z \leqslant \infty$ для $\forall i, j$, $j = \overline{1, l}$, $i = \overline{1, N}$, $N \leqslant \sum_{k=1}^{L} Z^k$

В *модели нулевого приближения* не вводится никаких ограничений на порядок следования символов алфавита; в *модели первого приближения* имеются ограничения на пары подряд идущих символов (запретные биграммы), определяемые бинарной матрицей $B = (\{b_{ij}\}), i, j = \overline{1, Z}$, где $b_{ij} = 1$ для разрешенной, и $b_{ij} = 0$ — для запрещенной пары символов алфавита A.

Количество слов длины, не превышающей l, выражается следующим образом:

$S_0(l) = \sum_{k=1}^{l} Z^k$ в модели нулевого приближения,

$S_1(l) = \sum_{k=2}^{l} \left(\sum_{i,j=1}^{Z} b_{ij}^{(k-1)} \right) + Z$ в модели первого приближения.

Получены следующие числовые характеристики для энтропии МТ (H_m) и ВТ (H_v):

в модели нулевого приближения $H_v \approx 5,04$, $H_m \approx 7,51$;

в модели первого приближения $H_v \approx 4,6398$, $H_m \approx 4,6018$.

Анализ полученных результатов позволяет сформулировать следующие промежуточные выводы:

1. Алфавит МТ обладает по сравнению с ВТ гораздо большей мощностью, однако МТ в силу своей художественной природы обладает и

значительно более строгим набором запретов на сочетания символов. 2. Пропускная способность канала коммуникации зависит от сочетания данных параметров — мощности алфавита и жесткости правил. 3. Поскольку результаты вычислений, проведенные на модели МТ С.С., представляют собой оценку снизу всех аналогичных результатов для подавляющего большинства музыкальных текстов (МТ С.С. обладает минимальным алфавитом и подчиняется максимально строгим правилам, регламентирующим запретные последовательности нот), то уже для одноголосия гипотеза о большей пропускной способности канала музыкальной коммуникации представляется вполне обоснованной. Для двухголосия же примерная оценка в модели нулевого приближения: $H_m = \log_2 6560 \approx 16,01$.

При рассмотрении вопросов построения модели семантики МТ необходимо различать понятия объективной семантики, присущей самой информации, и субъективной (прагматической) семантики, зависящей от воспринимающего субъекта: «$<\ldots>$ *объективная семантика* информации характеризует информационные формы существования материальных систем объективной реальности и взаимосвязана с формой, структурой и организацией материальных систем; $<\ldots>$ *семантика субъективная* (прагматическая) интерпретируется как динамический информационный образ объективной семантики, инициализированный в подсистеме знаний воспринимающей интеллектуальной системы» [3].

Для исследования объективной семантики МТ предлагается использовать известную модель k-гиперпространства СХ-гипертопографов как универсальную абстрактную модель информационной составляющей сложных систем [1, 2]. Моделирование одномерных и многомерных линейных структур (ВТ, одно- и многоголосного МТ, графики) предполагает использование модифицированной модели СХ-гипертопографа, с расширенным множеством-носителем, характеризующим множественность различимых (посредством хроматических атрибутов) экземпляров односортных элементов. Дальнейшая топологизация множества-носителя реализуется согласно классической модели СХ-гипертопографа.

Для моделирования МТ как динамического процесса возможно использование модели СХ-гипертопосетей [4], где в качестве стати-

ческих состояний системы, представленных СХ-гипертопографами, рассматриваются всевозможные вертикальные созвучия, априорно обладающие собственной сложной структурой отношений (особенно для полифонической музыки).

Приоритет в выборе того или иного подхода зависит от стилистических особенностей анализируемого (моделируемого) МТ: если обладающую развитой совокупностью структурных связей сонатную форму целесообразно представить в виде статической модели, то аморфный по форме (то есть характеризуемый слаборазвитыми связями между удаленными элементами), но обладающий строго детерминированными правилами соотношения соседних созвучий МТ С.С. логичным представляется исследовать с применением аппарата моделирования динамических систем.

Список литературы

[1] Баранович А. Е. Структурное метамоделирование телеологических информационных процессов в интеллектуальных системах. — М.: МО РФ, 2002.

[2] Баранович А. Е. Семиотико-хроматические гипертопографы. Введение в аксиоматическую теорию: информационный аспект. — М.: МО РФ, 2003.

[3] Баранович А. Е. Семантические аспекты информационной безопасности: концентрация знаний // Вестник РГГУ. Сер. «Информатика. Защита информации. Математика». — М.: РГГУ, 2011.

[4] Баранович А. Е. Семиотико-хроматические гипертопосети: унифицированная модель представления знаний // Открытые семант. технол. проектир. интеллект. систем (OSTIS-2011): мат. Междунар. научн.-техн. конф. — Минск: БГУИР, 2011. — С. 71–86.

[5] Гуров И. П. Основы теории информации и передачи сигналов: электрон. учеб. по дисципл. «Теория информации и передачи сигналов». — СПб.: СПбГУ-ИТМО, Кафедра компьютерных технологий. http://de.ifmo.ru/bk_netra/page.php?tutindex=11. Дата доступа: 20.04.2011.

[6] Иглицкая С. М. К вопросу структурно-алгебраического и семантико-прагматического анализа музыкального текста // Вестник РГГУ. Сер. «Информатика. Защита информации. Математика». — М.: РГГУ, 2011.

[7] Музыкальная энциклопедия в 6 томах / гл. ред. Ю. В. Келдыш. Т. 5. Симон — Хейлер. — М.: Советская энциклопедия, 1981.

[8] Фраёнов В. П. Учебник полифонии. — М.: Музыка, 1987.

Математическая модель для оптимизации индивидуальной практической работы учащегося

Биштейнов Д. А.
(Смоленск, Смоленский государственный университет)
bisteinoff@gmail.com

Индивидуализация обучения — актуальная задача в нашей стране. В этой статье мы приводим модель представления теоретического материала учебника и практических заданий задачника с целью оптимизации управления индивидуальной практической работой ученика в классе за счет автоматизации выполнения некоторых задач учителя.

Ключевые слова: индивидуализация образовательного процесса, модель, математическое моделирование, графовая модель, система автоматизированного проектирования работы учителя (САПР учителя).

В настоящее время необходимость индивидуализации обучения осознается не только учеными-новаторами, но и государством. В Концепции долгосрочного социально-экономического развития Российской Федерации до 2020 г. указывается: «*Развитие системы общего образования предусматривает* ***индивидуализацию****, ориентацию на практические навыки и фундаментальные умения. . .* ». В современных условиях основным источником образовательного запроса к системе образования становится личность учащегося. Его интересы, потребности, способности, мотивы должны во все большей степени учитываться при проектировании и организации процесса обучения [5]. Интерес представляет изучение не только учебно-воспитательного процесса, но и процесса индивидуального развития личности, в рамках сохраняющейся классно-урочной системы. Для третьей ступени обучения существует нормативно-правовая база для обучения по индивидуальным учебным планам, в профильных классах. Однако попрежнему не дается ответ на вопрос, как обучать в группе, чтобы не сдерживать индивидуальное развитие учащегося. Поставим **задачу** *оптимизации работы учителя, направленной на создание условий, в которых учащийся может выполнять практические задания на уроке в индивидуальном темпе.*

Актуальность реализации планирования учебного процесса в системах автоатизированного проектирования работы (САПР) учителя была рассмотрена в [1]. Представляет интерес САПР учителя «Траектория обучения», которая рассчитывает индивидуальную траекторию обучения класса (группы) или отдельного учащегося [5, с. 71–78].

Предложим математическую модель учебника и задачника, которая может быть положена в основу САПР учителя, позволяющей решить поставленную нами задачу.

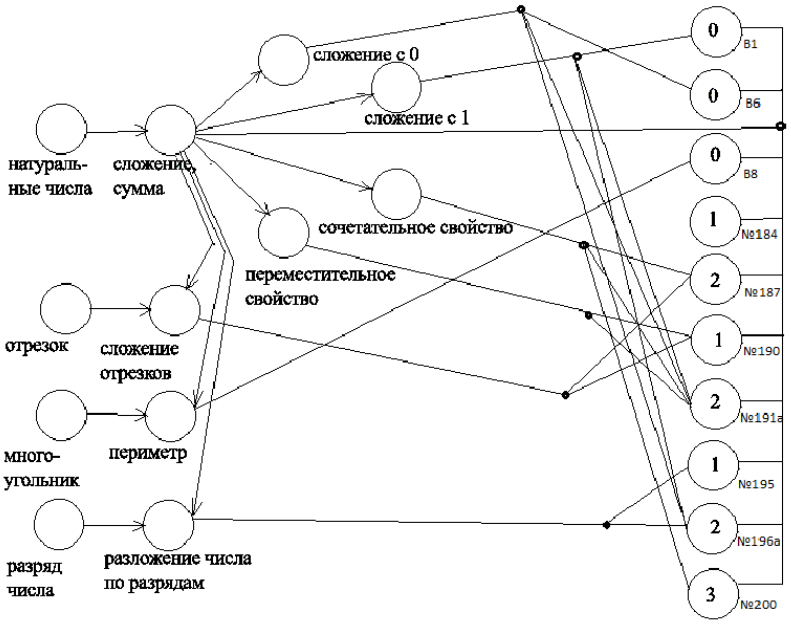
Имеется модель представления учебного материала учебника T в виде последовательности $\{L_i\}$, $i = 1 \dots n$, где L_i — это модель урока, которая представляет собой массив $L_i(C(L_i), Y(L_i), M(L_i), K(L_i))$, где $C(L_i)$ — множество целей урока L_i из множества целей C, $Y(L_i)$ — тип урока L_i из множества типов Y, $M(L_i)$ — множество требуемых материалов для проведения урока L_i из множества материалов M, $K(L_i)$ — граф, моделирующий теоретический материал урока L_i, который мы будем называть *знаниевым графом*. Знаниевый граф с математической точки зрения является графовой моделью цели обучения [5, с. 64–68]. Однако, на наш взгляд, неверно называть его целью обучения, так как он может содержать вершины, ассоциированные с элементами знаний, которые не будут являться обязательными для изучения или вовсе не будут входить в курс изучения предмета в рамках данного УМК.

Знаниевый граф является орграфом [2]. Его вершины ассоциируются с элементами знания по некоторой теме, дуги — с наличием логических и причинно-следственных связей между элементарными компонентами содержания обучения [4, с. 228]. Под теоретическим материалом урока мы будем понимать все элементарные компоненты, которые необходимо узнать учащимся по плану урока и на которые опирается изложение нового материала.

Пусть граф P — это граф со взвешенными вершинами, который мы обобщенно назовем *графом задач* (кроме задач, могут быть упражнения, вопросы, задания и др., а также к задачам иногда может быть целесообразно отнести дополнительный теоретический материал повышенной сложности). Вес вершин графа P означает *сложность* задания. Смоделируем граф $G(V(K) \cup V(P), E(V(K) \cup V(P)))$, где $V(K)$ — множество вершин знаниевого графа K, $V(P)$ — множе-

ство вершин графа задач P, $E(V(K) \cup V(P))$ — множество ребер, содержащее все ребра графа K и все ребра между вершинами графа K и вершинами графа P, которые устанавливаются на основании взаимного соответствия элементов теоретического материала из K задачам из P, для решения которых они необходимы (рис. 1). Отметим, что граф G содержит подграф $G_1 = K(L_i)$ и $G_2(V(G), E(G) - E(K))$. Подграф G_1 ориентированный. Подграф G_2 неориентированный, двудольный.

Отметим, что для ряда УМК может быть целесообразнее построить модель параграфа, темы, раздела. Мы также предполагаем, что сложность задания может быть прямо-пропорциональной степени исхода вершины, ассоциированной с ним.



Приведем пример модели параграфа «Сложение натуральных чисел и его свойства» из учебника 5 класса Н. Я. Виленкина [3]. Задачник содержит более 100 задач для отработки материала этой темы. Кроме того, число задач может быть увеличено за счет других задачников. Для нашей модели мы выбрали 10 задач таким образом,

чтобы все вершины графа задач были смежны хотя бы одной вершине знаниевого графа.

Итак, в данной статье мы предложили теоретическую модель учебника и задачника для системы автоматизированного проектирования работы учителя, потенциально расширяющую функции САПР учителя «Траектория обучения» и позволяющую оптимизировать трудозатраты учителя на организацию индивидуальной практической работы учащихся в группе.

Список литературы

[1] Биштейнов Д. А. Актуальность реализации планирования учебного процесса в системах автоатизированного проектирования работы учителя // Методология и методика информатизации образования в многоступенчатой структуре высшей школы: материалы Всероссийской научно-практической конференции (7–8 декабря 2009 года). — Смоленск: Изд-во СмолГУ, 2009. — С. 4–6.

[2] Кристофидес Н. Теория графов. Алгоритмический подход. — М.: Мир, 1978.

[3] Математика: Учеб. для 5 кл. общеобразоват. учреждений / Н. Я. Виленкин, В. И. Жохов, А. С. Чесноков, С. И. Шварцбурд. — 17-е изд., перераб. — М.: Мнемозина, 2005. — С. 33-41.

[4] Новиков А. М. Развитие отечественного образования. Полемические размышления. — М.: Изд-во «Эгвес», 2005.

[5] Сенькина Г. Е., Емельченков Е. П., Киселева О. М. Методы математического моделирования в обучении. — Смоленск: Смол. гос. ун-т, 2007.

Эволюционные символьные вычисления: методика и инструментарий

Борченинов Я. В., Окуловский Ю. С.
(Екатеринбург, Уральский государственный университет им. А. М. Горького)

yuri.okulovsky@gmail.com

Эволюционные символьные методы (ЭСВ) широко применяются для аппроксимации данных, поиска инвариантов и физических законов, классификации, решении дифференциальных уравнений и других задач. ЭСВ используют генетические алгоритмы для направленного перебора синтаксических конструкций. По конструкции строится выражение, которое оценивается на соответствие исходным данным. Первоначально этот метод был предложен, по всей видимости, Джоном Коза (см., например, [1]) для автоматической генерации программ. Метод был далее адаптирован для других задач: для построения выражений [2], для решения дифференциальных уравнений [3], для восстановления естественных законов из экспериментальных данных [4] и т. д.

Настоящая работа посвящена проработке следующих проблем ЭСВ. Для существующих решений характерна специализация для работы с числами с плавающей точкой. Другие домены (целые числа, нечеткие логические переменные, функции) используются редко, практически не встречается совместное использование доменов (например, логических выражений и чисел), не решены возникающие проблемы с ошибками типизации. Также в исследованиях ЭСВ непропорциональное внимание уделяется замысловатым способам кодирования и решению проблем, которое это кодирование вызывает, хотя, как мы покажем далее, эта проблема решается одним простым и элегантным обобщением. Кроме того, игнорируется опыт систем компьютерной алгебры и дедуктивных преобразований выражений. В лучшем случае, полученное в ходе ЭСВ выражение упрощается на выходе применением какой-либо системы компьютерной алгебры. Нам представляется, что полноценная интеграция дедуктивных преобразований оправдана как с математической, так и с философской

точек зрения. Наконец, мало проработаны метрики оценки выражений, что приводит к появлению сложных, избыточных и неэстетических выражений.

В настоящей работе мы изложим методику проведения ЭСВ, решающую указанные проблемы. Данная методика воплощена в библиотеке ЭСВ на языке C#, которая в данный момент находится в состоянии прототипа. В дальнейшем эта библиотека будет опубликована под лицензией GPL v.3.

Методика использует следующее обобщение генетических алгоритмов. Исторически, при работе со сложными структурами данных, генетический алгоритм кодирует их в виде битовых массивов или строк. Мутация сводится к отрицанию бита или изменению буквы, а скрещивание — к обмену фрагментами массивов. Этот подход приводит к разрушению целостности структуры данных, и вынуждает изобретать методы для ее поддержания. Мы используем обобщенную формулировку генетического алгоритма, которая повторяет классическую, но разрешает обработку произвольных структур данных, для которых определены нульарный оператор генерации, унарный оператор мутации и бинарный оператор скрещивания. Обобщенный генетический алгоритм позволяет выбрать интуитивное, простое кодирование для задачи.

Кодирование выражений выполняется в виде дерева. Узлами дерева являются операции вида $f : D_{i_1} \times \ldots \times D_{i_k} \to D_j$, где D_s — произвольные домены (типы) данных. Листьями дерева являются типизированные константы и переменные. Каждый узел дерева содержит тип данных, возвращаемых выражением, закодированном в соответствующем поддереве.

Операция генерации начального гена для выражения типа D состоит в создании дерева из единственного листа — константы соответствующего типа.

Операции мутации определяются на основе правил. Данный подход заимствован из систем компьютерной алгебры, в которых все свойства операций определяются списками преобразований (таких как $(x + y)z \Rightarrow xz + yx$). В системах компьютерной алгебры эти операции дедуктивны, они изменяют лишь форму выражения, но не кодируемую им функцию. В ЭСВ, мы используем также индуктивные

правила, изменяющую как структуру, так и кодируемую функцию. Все правила сохраняют тип поддерева, чем обеспечивается корректность собираемого выражения.

Приведем примеры индуктивных правил. Правила замены константы вида $c_{\text{bool}} \Rightarrow \neg c$ и $c_{\text{real}} \Rightarrow c(1 + \alpha/2 - \text{rnd}(\alpha))$ позволяют «подгонять» значение константы c. Правила вида $N_{\text{T}} \Rightarrow x_{\text{T}}$ позволяют заменить поддерево N_{T} типа T на переменную того же типа. Операции вводятся правилами ввода: $N_{\text{int}} \Rightarrow N + 0$, $N_{\text{int}} \Rightarrow N \cdot 1$, $N_{\text{int}} \Rightarrow \text{TRUE}?N : 0$, $N_{\text{bool}} \Rightarrow N \wedge \text{TRUE}$, $N_{\text{bool}} \Rightarrow c_{\text{int}} < d_{\text{int}}$ и т. д. Операция также может быть выведена универсальным способом $[f(N_{\text{T}}, \ldots)]_{\text{T}} \Rightarrow N_{\text{T}}$. Приведем примеры дедуктивных правил. Структурные правила используют ассоциативность и коммутативность некоторых операций, и перестраивают дерево в некую каноническую структуру, а также сортируют аргументы коммутативных функций так, чтобы константы оказывались в конце дерева. Правила сокращения выполняют такие замены, как $2 + 2 \Rightarrow 4$ и $0 \cdot x \Rightarrow 0$. Также введены правила преобразований вида $N^2_{\text{real}} \Leftrightarrow N \cdot N$. Применение в левую сторону делает выражение более эстетически привлекательным, в правую — облегчает применение индуктивных правил, делая, например, возможной двухходовую замену $x \cdot x \Rightarrow x^2 \Rightarrow x^{1.9}$. Для поддержки правил в программном коде была реализована система условий и замен фрагментов дерева, по идеологии схожая с применением регулярных выражений к строкам.

Операция скрещивания в простейшем случае сводится к обмену поддеревьями одного типа. Кроме того, возможно сложное скрещивание с введением условного перехода $A, B \Rightarrow TRUE?A : B$, операций полусуммы $A_{\text{real}}, B_{\text{real}} \Rightarrow (A + B)/2$ или геометрического среднего $A_{\text{real}}, B_{\text{real}} \Rightarrow \sqrt{AB}$.

После построения выражения, его необходимо оценить. Базовой метрикой оценки является соответствие, то есть то, насколько точно выражение описывает данные. Для оптимизации вычисления этой оценки, используются инструменты языка C#: лямбда-выражения и Expression Trees [5]. Эти инструменты позволяют скомпилировать для дерева операций лямбда-выражение, вычисление которого значительно быстрее обхода дерева. Лямбда выражение задается для каждого узла, но компилируется только для корня. Для константы

строится выражение $X \mapsto c$ (X — кортеж всех аргументов построенного выражения), для переменной — $X \mapsto X[i]$, для операции f — $X \mapsto f(g_1(X), \ldots, g_k(X)$, где g_i — лямбда-выражения для детей f.

Помимо соответствия, используются и другие метрики. Метрика сложности определяет вычислительную сложность выражения и стремится минимизировать ее, метрика размера приводит к усечению неиспользуемых поддеревьев, метрика эстетичности выбирает наиболее эстетически привлекательные выражения. Данное направление в настоящий момент является предметом дополнительных исследований.

Приведенная методика была протестирована на обычных для ЭСВ задачах аппроксимации, поиска инвариантов и классификации. Создан тестовый набор «стандартных» задач, на котором мы планируем сравнивать различные реализации ЭСВ в дальнейшем.

В заключение кратко сформулируем основные преимущества нашей методики. За счет разрешения использования разных доменов данных, методика обобщает все примеры ЭСВ, известные ранее. Кодирование выражений происходит простым и интуитивным путем. Введение новых операций производится с помощью правил, что позволяет расширять ЭСВ произвольными операциями и доменами. Помимо индуктивных, используются дедуктивные правила. Введены различные метрики оценки решения, что позволяет получить не только верные, но также неизбыточные и эстетически привлекательные решения.

Разработанная методика и инструментарий позволяют проводить дальнейшие исследования ЭСВ в следующих областях: манипулирование частотой применения правил разных типов для получения оптимальных результатов; исследования оптимального веса метрик для получения решений, сбалансированных по разным параметрам; применение ЭСВ к новым областям, в первую очередь к логике и нечеткой математике.

Список литературы

[1] Koza J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. Mit Press, 1992.

[2] Ferreira C. Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Springer, 2006.

[3] Diver D. A. Applications of genetic algorithms to the solution of ordinary differential equations // Journal of Physics A: Mathematical and General. N 26. 1993.

[4] Schmidt M., Lipson H. Distilling Free-Form Natural Laws from Experimental Data // Science. Vol. 324. No. 5923. 2009.

[5] Troelsen A. Pro C# 2010 and the .NET 4 Platform. Apress, 2010.

Приближенное решение задачи о близости в евклидовой метрике

Гасанов Э. Э., Остроухова Е. Н.
(Москва, МГУ им. М. В. Ломоносова)

el_ gasanov@mail.ru

Мы будем использовать терминологию и обозначения из работы [1].

Пусть X — множество запросов с заданным на нем вероятностным пространством $\langle X, \sigma, \mathbf{P}\rangle$, где σ — алгебра подмножеств множества X, $\mathbf{P}$ — вероятностная мера на σ; Y — множество записей (объектов поиска); ρ — бинарное отношение на $X \times Y$, называемое *отношением поиска*. Пятерку $S = \langle X, Y, \rho, \sigma, \mathbf{P}\rangle$ назовем *типом*. Тройку $I = \langle X, V, \rho\rangle$, где V — некоторое конечное подмножество множества Y, в дальнейшем называемое *библиотекой*, будем называть *задачей информационного поиска (ЗИП)*, и будем считать, что ЗИП $I = \langle X, V, \rho\rangle$ содержательно состоит в перечислении для произвольно взятого запроса $x \in X$ всех тех и только тех записей $y \in V$ таких, что $x\rho y$.

Пусть f — одноместный предикат, определенный на X, то есть $f : X \to \{0, 1\}$. Множество $N_f = \{x \in X : f(x) = 1\}$ назовем *характеристическим множеством предиката* f.

Множество $O(y, \rho) = \{x \in X : x\rho y\}$ назовем *тенью записи* $y \in Y$.

Функцию $\chi_{y,\rho} : X \to \{0, 1\}$ такую, что $N_{\chi_{y,\rho}} = O(y, \rho)$ назовем *характеристической функцией записи* y.

Пусть F — множество символов одноместных предикатов, определенных на множестве X, G — множество символов одноместных переключателей, определенных на множестве X. Под переключателем будем понимать функцию, областью значений которой является начальный отрезок натурального ряда. Пару $\mathcal{F} = \langle F, G\rangle$ назовем *базовым множеством*.

Понятие *информационного графа (ИГ) над базовым множеством* $\mathcal{F} = \langle F, G\rangle$ определяется следующим образом. Рассмотрим конечную многополюсную ориентированную сеть. В ней выбираем полюс, который называем корнем. Остальные полюса называются листьями и

им приписываются записи из Y, причем разным листьям могут быть приписаны одинаковые записи. Некоторые вершины сети называются переключательными и им приписаны переключатели из множества G. Ребра, исходящие из этих вершин, нумеруются подряд, начиная с единицы, и называются переключательными ребрами. Остальные ребра сети называем предикатными и приписываем им предикаты из множества F. Таким образом нагруженную многополюсную сеть называем информационным графом над базовым множеством $\mathcal{F} = \langle F, G \rangle$.

Функционирование информационного графа определяется следующим образом. Скажем, что предикатное ребро проводит запрос $x \in X$, если $f(x) = 1$. Переключательное ребро, которому приписан номер n, проводит запрос x, если переключатель, приписанный началу этого ребра, на запросе x принимает значение n. Ориентированная цепь ребер проводит запрос x, если каждое ребро этой цепи проводит x. Запрос $x \in X$ проходит в вершину β ИГ, если существует ориентированная цепочка ребер, ведущая из корня в вершину β и проводящая запрос x. Запись y, приписанная листу α, попадает в ответ ИГ на запрос $x \in X$, если запрос x проходит в лист α. Ответом ИГ U на запрос x назовем множество записей, попавших в ответ ИГ на запрос x, и обозначим его $\mathcal{J}_U(x)$. Функцию $\mathcal{J}_U(x) : X \to 2^Y$ будем считать результатом функционирования ИГ U.

Для ЗИП $I = \langle X, V, \rho \rangle$ обозначим $\mathcal{J}_I(x) = \{y \in V : x \rho y\}$ — ответ задачи I на запрос x, $R(I) = \mathbf{M}_x |\mathcal{J}_I(x)| = \sum\limits_{y \in V} \mathbf{P}(O(y, \rho))$ — средняя длина ответа, или среднее время перечисления ответа.

Пусть нам дана ЗИП $I = \langle X, V, \rho \rangle$. Скажем, что ИГ U решает ЗИП $I = \langle X, V, \rho \rangle$, если $\mathcal{J}_U(x) = \{y \in V : x \rho y\}$. ИГ U, решающий ЗИП I, будем также называть *допустимым* для I. Множество ИГ над базовым множеством $\mathcal{F}$, решающих ЗИП I, будем обозначать $\mathcal{U}(I, \mathcal{F})$.

Пусть β — некоторая вершина ИГ. Предикат, определенный на множестве запросов, который принимает значение 1 на запросе x, если запрос проходит в вершину β, и 0 — в противном случае, назовем *функцией фильтра* вершины β и обозначим $\varphi_\beta(x)$.

Сложностью ИГ U на запросе $x \in X$ назовем число

$$T(U,x) = \sum_{\beta \in \mathcal{P}} \varphi_\beta(x) + \sum_{\beta \in \mathcal{R} \setminus \mathcal{P}} \psi_\beta \cdot \varphi_\beta(x),$$

где ψ_β — количество ребер, исходящих из вершины β, $\mathcal{P}(U)$ — множество вершин переключения ИГ U, $\mathcal{R}(U)$ — множество всех вершин ИГ U.

Скажем, что базовое множество $\mathcal{F}$ — *измеримое*, если каждая функция из $\mathcal{F}$ измерима (относительно σ). Далее везде будем предполагать, что базовое множество измеримо. В этом случае для любого ИГ U над $\mathcal{F}$ функция $T(U,x)$ как функция от x измерима.

Сложностью ИГ U назовем математическое ожидание величины $T(U,x)$, то есть число $T(U) = \mathbf{M}_x T(U,x)$.

Сложностью в худшем случае назовем величину $\widehat{T}(U) = \max\limits_{x \in X} T(U,x)$.

Будем рассматривать следующие сложностные характеристики:

$$T'(U,x) = T(U,x) - |\mathcal{J}_I(x)|,\ T'(U) = \mathbf{M}_x T'(U,x),\ \widehat{T}'(U) = \max_{x \in X} T'(U,x).$$

$T'(U,x)$ характеризует время поиска без перечисления ответа для запроса x.

Объемом $Q(U)$ ИГ U назовем число ребер в графе U.

Пусть $I = \langle X, V, \rho \rangle$ — исходная задача информационного поиска. Рассмотрим вспомогательную задачу $\widetilde{I} = \langle X, V, \widetilde{\rho} \rangle$.

Назовем *шумом пары $\langle I, \widetilde{I} \rangle$ на запросе x* множество $N(x) = \mathcal{J}_{\widetilde{I}}(x) \setminus \mathcal{J}_I(x)$. Назовем *дефицитом пары $\langle I, \widetilde{I} \rangle$ на запросе x* множество $D(x) = \mathcal{J}_I(x) \setminus \mathcal{J}_{\widetilde{I}}(x)$. *Средним шумом пары $\langle I, \widetilde{I} \rangle$* назовем величину $n(I, \widetilde{I}) = \mathbf{M}_x |N(x)| = \sum\limits_{y \in V} \mathbf{P}(O(y, \widetilde{\rho}) \setminus O(y, \rho))$. *Средним дефицитом пары $\langle I, \widetilde{I} \rangle$* назовем величину $d(I, \widetilde{I}) = \mathbf{M}_x |D(x)| = \sum_{y \in V} \mathbf{P}(O(y, \rho) \setminus O(y, \widetilde{\rho}))$.

Величину $m(\alpha) = \frac{\alpha n + (1-\alpha) d}{R(I)}$, зависящую от действительного параметра $\alpha \in [0,1]$, называем *ошибкой пары $\langle I, \widetilde{I} \rangle$ при цене ошибки α*. Величину $\widehat{m} = \sup\limits_{\alpha \in [0,1]} m(\alpha)$ называем *максимальной ошибкой пары $\langle I, \widetilde{I} \rangle$*.

Опишем тип задач поиска, который соответствует двумерной задаче о близости в евклидовой метрике.

Пусть $X = Y = [0,1]^2$ — множества запросов и записей соответственно. Пусть на множестве X задано вероятностное пространство $\langle X, \sigma, \mathbf{P}\rangle$, $\mathbf{P}$ — равномерное распределение на X. Отношение поиска ρ_R определено на $X \times Y$ и задается следующим соотношением: $(x_1, x_2)\rho_R(y_1, y_2) \Longleftrightarrow \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \leqslant R$, где $R \in [0,1]$. Тогда тип $\langle X, Y, \rho_R, \sigma, \mathbf{P}\rangle$ назовем *типом двумерной задачи о близости в евклидовой метрике.*

Если $a \in \mathbb{R}$, то $]a[$ — наименьшее целое, не меньшее, чем a.

Если $\alpha = (\alpha_1, \alpha_2), x = (x_1, x_2) \in \mathbb{R}^2$, то $\alpha x = \alpha_1 x_1 + \alpha_2 x_2$.

Если $\alpha \in \mathbb{R}^2$, $a, b, d \in \mathbb{R}$, $m \in \mathbb{N}$, то обозначим

$$f_{\alpha,d}(x) = \begin{cases} 1, & \text{если } \alpha x \geqslant d, \\ 0, & \text{если } \alpha x < d, \end{cases} \tag{1}$$

$$F_1 = \{f_{\alpha,d}(x) : \alpha \in \mathbb{R}^2, d \in \mathbb{R}\}, \tag{2}$$

$$g_{\alpha,d}(x) = \begin{cases} 1, & \text{если } \alpha x \leqslant d, \\ 2, & \text{если } \alpha x > d, \end{cases} \tag{3}$$

$$G_1 = \{g_{\alpha,d}(x) : \alpha \in \mathbb{R}^2, d \in \mathbb{R}\}, \tag{4}$$

$$g_{\alpha,a,b,m}(x) = \left]\frac{\alpha x - a}{b - a} m\right[, \tag{5}$$

$$G_2 = \{g_{\alpha,a,b,m}(x) : \alpha \in \mathbb{R}^2, a, b \in \mathbb{R}, m \in \mathbb{N}\}, \tag{6}$$

$$\mathcal{F} = \langle F_1 \cup \{1\}, G_1 \cup G_2\rangle. \tag{7}$$

Здесь 1 — предикат тождественная единица.

Будем писать $A(n) \preceq B(n)$, если существует константа $c > 0$, что $A(n) \leqslant cB(n)$, начиная с некоторого номера n_0.

Теорема 1. *Пусть $I = \langle X, V, \rho_R\rangle$ — двумерная задача о близости в евклидовой метрике, $|V| = k$. Базовое множество $\mathcal{F}$ задается соотношениями (1)–(7). Тогда существуют вспомогательные задачи $\widetilde{I}_1 = \langle X, V, \widetilde{\rho}_1\rangle$ и $\widetilde{I}_2 = \langle X, V, \widetilde{\rho}_2\rangle$ и ИГ $U_1 \in \mathcal{U}(\widetilde{I}_1, \mathcal{F})$, $U_2 \in \mathcal{U}(\widetilde{I}_2, \mathcal{F})$ такие, что верны следующие оценки сложности:*

$$\begin{cases} Q(U_1) \preceq q^2 k^{1+\frac{2}{q}} \\ T'(U_1) \preceq q^2 \\ \widehat{T}'(U_1) \preceq q^2 \log_2 k \\ \frac{37}{1000} \leqslant \widehat{m}(I, \widetilde{I}_1) \leqslant \frac{38}{1000} \end{cases} \qquad \begin{cases} Q(U_2) \preceq q k^{1+\frac{1}{q}} \\ T'(U_2) \preceq q \\ \widehat{T}'(U_2) \preceq q \log_2 k \\ \frac{50}{1000} \leqslant \widehat{m}(I, \widetilde{I}_2) \leqslant \frac{51}{1000} \end{cases}$$

где $1 \leqslant q \leqslant \log_2 k$ *— натуральный параметр.*

Список литературы

[1] Гасанов Э. Э., Кудрявцев В. Б. Теория хранения и поиска информации. — М.: ФИЗМАТЛИТ, 2002.

База знаний повышения производительности MPI-приложений

Дергунов А. В. (Нижний Новгород, Нижегородский государственный университет им. Н. И. Лобачевского)

anton.dergunov@gmail.com

Для анализа MPI программ часто используют программные системы, которые осуществляют сбор и визуализацию трассы выполнения программы. Но при использовании таких инструментов пользователь сталкивается с проблемой анализа больших объемов информации. Другой проблемой при использовании средств визуализации является то, что часто встречающиеся ситуации, приводящие к потерям производительности MPI программ, явно не визуализируются, то есть отсутствует база знаний повышения производительности.

Поэтому возникает потребность в средствах, которые бы автоматизировали анализ трассы и подсказали пользователю, как повысить производительность его программы. В данной работе описывается программная система, выполняющая эту задачу.

Схема работы системы представлена на рис. 1. Исходными данными является трасса выполнения MPI приложения, полученная с помощью трассировщика. Она анализируется модулем автоматического анализа трассы. Результат анализа — список выявленных причин недостаточной производительности пользовательского приложения с указанием степени их влияния на общее время работы приложения.

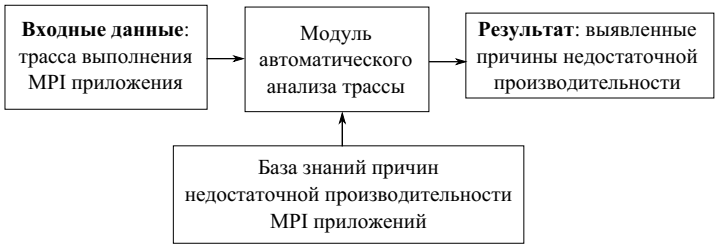


Рис. 1. Схема работы системы.

Ключевым компонетом этой системы является база знаний причин недостаточных производительности MPI приложений, которая состоит из правил, описанных на специальном языке, разработанном в рамках этой системы. Правила используются в системе для следующих целей:

- описание составных событий, которые формируются на основе простых событий, собранных трассировщиком во время работы программы, или других составных событий, и представляют собой высокоуровневые операции программы;
- описание причин недостаточной производительности MPI приложений.

Ниже приведен пример правила, описывающего событие операции приема данных, как состоящее из простых событий, соответствующих вызовам функций MPI_Recv_init и MPI_Start.

```
rule
    event e1
        type function_call_non_blocking_receive_operation
    event e2 type function_call_request_handling
    where
        e1.@function_name eq "MPI_Recv_init" and
        e2.@function_name eq "MPI_Start" and
        e1.request = e2.request
    compose event of type point_to_point_operation
        type = RECEIVE_OPERATION
        function_name = e1.funcion_name
        start_time = e2.start_time
        finish_time = e2.finish_time
        source = e1.source
        dest = e1.dest
        tag = e1.tag
        comm = e1.comm
    delete e1
    delete e2;
```

Одной из причин недостаточной производительности является плохая синхронизация приемов и передач данных в MPI программе. В результате процедура приема данных может простаивать, дожидаясь посылки данных (см. рис. 2). Для решения этой проблемы нужно обеспечить, чтобы сообщения посылались как можно раньше, и таким образом повысить вероятность того, что сообщение прибудет до момента, когда оно потребуется другому процессу. Следующее правило описывает эту ситуацию:

Рис. 2. Поздняя посылка данных.

```
rule
  event e type point_to_point_operation_group
  where
    e.receive_function_name eq "MPI_Recv" and
    e.send_start_time > e.receive_start_time
  create performance observation
    name = "Поздняя посылка данных"
    description = "Операция посылки данных вызывается
      позднее операции приема. Из-за этого блокирующая
      операция приема вынуждена простаивать."
    advice = "Улучшить синхронизацию приемов и передач
      данных, оправляя данные по возможности раньше, или
      использовать неблокирующие операции приема."
    impact = op1.send_start_time - op1.receive_start_time;
```

С помощью приведенного правила осуществляется обнаружение операций двухточечного обмена данными, операция приема данных

в которых блокирующего типа (MPI_Recv) и время начала посылки данных позднее времени начала приема данных. При выполнении перечисленных условий система делает вывод о неэффективной работе программы, связанной с поздней посылкой данных. В правиле указывается подробное описание этой причины недостаточной производительности и совет по ее устранению. Также вычисляется степень влияния на общее время работы программы, как разница между временем начала посылки данных и временем его приема.

База знаний системы состоит из правил, осуществляющих выявление следующих причин недостаточной производительности MPI программ: поздняя посылка данных, поздний прием данных, ранний прием данных при операции «от многих к одному» и т. д. Этот список был составлен на основе анализа литературы по оптимизации MPI программ и стандарта MPI. Более подробно состав базы знаний описан в работе [1].

В работе [1] описан эксперимент по повышению производительности MPI программы с помощью разработанной системы. В результате анализа трассы ее работы на кластере системой были установлены причины недостаточной производительности и выдан совет по изменению программы пользователя для улучшения ее производительности. После внесения изменений общее время работы программы существенно уменьшилось.

Список литературы

[1] Карпенко С. Н., Дергунов А. В. Программные средства повышения производительности MPI-приложений // Супервычисления и математическое моделирование. Труды XII международного семинара. — Саров: ФГУП «РФЯЦ-ВНИИЭФ», 2011. С. 195–202.

Геометрический подход к распознаванию зрительных образов (краткий обзор)

Козлов В. Н. (Москва, МГУ им. М. В. Ломоносова)

Ниже содержательно описывается дискретно-геометрический подход к распознаванию зрительных и основные его результаты. Описание разбито на 5 пунктов.

I) Изображением здесь называется конечное множество точек в евклидовом пространстве. В частности, двумерным или плоским изображением называем конечное множество точек на плоскости. Содержательным обоснованием этому может служить то, что любое реальное (не цветное) изображение можно аппроксимировать изображением из точек, причем в нужной мере можно передать все градации «серого цвета» разной плотностью точек в разных частях изображения. Такое представление не закрывает дорогу и к рассмотрению цветных изображений, поскольку, как известно, цветное изображение можно представить тремя не цветными. Наконец, все, что мы видим, мы видим посредством глаз. Изображение из среды проецируется на сетчатку глаз, что приводит к возбуждению части рецепторных клеток, то есть, в конечном счете, к формированию на сетчатке аналога составленного из точек изображения.

Аналогично двумерному случаю, конечное множество точек в трехмерном пространстве может аппроксимировать с нужной степенью точностью предметы, сцены, и в целом окружающую среду. Наконец, конечные множества точек в четырехмерном пространстве могут представлять трехмерные сцены в динамике. При этом одна из осей интерпретируется как ось времени.

Итак, основной объект рассмотрения — конечные множества точек в евклидовых пространствах. С таким объектом можно уже работать формальными, математическими средствами, то есть доказывать теоремы, делать количественные оценки, и пр. На этих теоремах можно основывать алгоритмы распознавания, базирующиеся уже не на правдоподобных рассуждениях (как при эвристическом подходе), а на точных доказательствах. Это то, что можно назвать доказательным, теоремным уровнем рассмотрения проблем распознавания изображений.

II) Итак, два изображения A и B — это два конечных множества точек, например, две фигуры, в значительной степени повторяющие очертаниями, формой друг друга, и нам это обстоятельство нужно выяснить. Проблема, однако, в том, что априори нам неизвестно, что это за фигуры, более того, неизвестно, где у них правые-левые части, верх-низ, то есть нет ничего, кроме двух множеств точек. Как можно было бы определить схожесть по форме этих фигур? Если бы они были одинаковы, то, очевидно, это можно было бы выяснить наложением их друг на друга (например, движениями, то есть изометрическими преобразованиями) так, чтобы они полностью (поточечно) совпали. Если же фигуры не вполне одинаковы, то надо движениями расположить их одну на другой так, чтобы степень их «несовпадения», «рассогласования» была бы наименьшей из возможных. В этом случае величина этого «рассогласования» была бы количественной характеристикой несовпадения по форме.

Сделаем это требуемое следующим образом. Для каждой точки каждого из изображений A и B в качестве характеристики рассогласования этой точки со сравниваемым изображением возьмем отрезок между нею и ближайшей точкой сравниваемого изображения. Тогда характеризовать степень несовпадения в целом данных двух изображений A и B будем наибольшим из отрезков, взятых по всем точкам обоих изображений. Обозначим длину этого отрезка через $\Delta(A, B)$.

Величина $\Delta(A, B)$ характеризует взаиморасположение конкретных множеств точек A и B. Теперь будем преобразовывать A и B, например, движениями так, чтобы они как можно более совпали, то есть ищем такое взаиморасположение A и B, при котором величина $\Delta(A, B)$ наименьшая из возможных.

Нетрудно видеть, что это задача на поиск экстремума: каждое из возможных положений для A и B характеризуется вполне определенной величиной $\Delta(A, B)$; надо построить множество всех возможных взаиморасположений для A и B, порождаемых некоторыми их преобразованиями (движениями, например). Это даст множество $\{\Delta(A, B)\}$ величин $\Delta(A, B)$. Надо найти минимум на этом множестве (если он существует, конечно).

Проблема в том, что множество $\{\Delta(A, B)\}$ — бесконечное, более точно — континуальное по мощности. Таким оно будет, даже если

взять в качестве преобразований для A и B наиболее простые преобразования — параллельного переноса. Естественно, проблема сохраняется, когда мы расширяем класс преобразований до изометрических — сочетаний параллельных переносов, поворотов и преобразований симметрии относительно прямой. Следующий по сложности класс — преобразования подобия, когда добавляется возможность для A и B произвольно меняться в размерах. Наконец, в случае аффинных преобразований мы добавляем ко всему перечисленному еще и возможность для A и B сжиматься и растягиваться по произвольным направлениям.

Наиболее интересен с практической точки зрения класс аффинных преобразований. Ибо в этом случае мы сравниваем «по форме» два изображения A и B безотносительно к их положению на плоскости, поворотам, изменениям в размерах, сжатиям и растяжениям. Однако, напомним, мы не можем сравнивать A и B непосредственно перебором, попробовав все варианты взаиморасположений, ибо множество $\{\Delta(A, B)\}$ (точнее, его аналог для аффинного случая) — бесконечное.

Неожиданностью здесь было то, что, как теперь доказано, существует конечное подмножество взаиморасположений для A и B, и, соответственно, конечное подмножество на $\{\Delta(A, B)\}$, на котором нужный минимум только и может достигаться. Это конечное множество взаиморасположений A и B может быть построено. Это, в свою очередь, означает, что бесконечный и потому неосуществимый перебор всех возможных вариантов взаиморасположений A и B может быть заменен проверкой лишь конечного, заранее определенного числа их взаиморасположений. Таким образом, нет необходимости пробовать по разному смещать, поворачивать изображения A и B относительно друг друга, менять их размеры, и пр. — можно сразу идти туда, где только и может быть решение.

III) Одна из особенностей зрительной информации — ее огромные объемы [1, 2]. Глаз человека содержит около 130 миллионов светочувствительных элементов (палочек и колбочек). Вместе с тем распознавание изображений осуществляется иногда лишь за доли секунды. Вряд ли это можно сделать, используя целиком всю информацию на сетчатке глаза. Должно существовать некоторое «сито» и для изоб-

ражений из среды, и для визуальной информации из памяти при оперировании с ними. Парадокс, однако, состоит в том, что попытки до собственно распознавания выделить на изображениях «опорные точки», «важные детали» и пр. (как правило, средствами эвристики), есть тоже распознавание, то есть возникает до некоторой степени замкнутый круг. Ясно, что нужно избегать этого.

Зададимся некоторым числом r ($r \geqslant 0$). Выберем на изображении A подмножество A^r точек из A с тем условием, чтобы каждая точка из A находилась бы на расстоянии, не большем r, от какой-либо точки из A^r. Ясно, что A^r можно трактовать как подизображение изображения A, в частном случае A^r может совпадать с A. Изображение A^r можно интерпретировать как покрытие A кругами радиуса r с центрами в точках из A^r, причем такое, что каждая точка исходного изображения попадает хотя бы в один из этих кругов.

Содержательно A^r трактуется как «огрубление» изображения A, устранение на нем излишних деталей и подробностей, поэтому A^r называем эскизом изображения A. При заданном параметре r существует некоторое множество $\{A^r\} = \{A_1^r, \ldots, A_k^r\}$ эскизов, причем исходное изображение A входит в это множество. Однако интерпретировать как «огрубление» A можно, очевидно, лишь те эскизы из A^r,которые состоят из меньшего числа точек, чем исходное изображение A. В частности, в $\{A^r\}$ есть изображение (может быть, не одно) с наименьшим среди $A_1^r, \ldots, A_k^r$ числом точек. Такое изображение назовем остовом изображения A.

При оценке похожести двух изображений достаточно, может быть, оценить их сходство в целом, в главных чертах, без деталей. Такого рода интуитивные соображения неявно предполагают, что вместо собственно изображений будут использованы другие изображения, полученные из исходных устранением излишних деталей, то есть в нашем случае — эскизы. Оказалось возможным теоремным образом связать схожесть между эскизами со схожестью между оригиналами. Это дает возможность, имея дело с эскизами, делать оценки для схожести исходных изображений.

IV) Код изображения A — пара $< M_A, T_A >$, где M_A — множество номеров точек. Множество T_A состоит из чисел с индексами вида $\rho_{mnu,ksp}$. Здесь $\rho_{mnu,ksp}$ получается отношением площадей треуголь-

ников с вершинами в точках изображения с номерами соответственно m, n, u и k, s, p (естественно при условии, что площадь треугольника с вершинами в точках k, s, p не равна нулю). Такие числа ρ строятся для всех пар треугольников. Подчеркнем, что только на этапе формирования числа $\rho_{mnu,ksp}$ мы говорим о треугольниках и о площадях — далее этой информации нет и T_A состоит только из чисел с индексами.

Доказано, что коды двух изображений одинаковы (с точностью до перенумерации точек) тогда и только тогда, когда эти изображения переводятся одно в другое аффинным преобразованием. Тем самым, этот код задает изображение с точностью до аффинных преобразований. Это означает, что по коду изображение можно восстановить по разному, но все варианты восстановления будут отличаться друг от друга только аффинными преобразованиями.

Аналогичным свойством обладает код $< M_A, T_A >$ для точек в трехмерном пространстве (трехмерные изображения или тела). Различие с двумерным случаем состоит только в том, что числу ρ приписаны две четверки индексов m, n, u, v и k, s, p, q и само число получается отношением объемов тетраэдров с вершинами в соответствующих точках. Этот код тоже описывает тело с точностью до аффинных преобразований.

Доказано, что аналогичные коды существуют и для конечных множеств точек в пространствах размерности n, где $n > 3$. Аналогичные коды построены и для проективных преобразований [2].

V) Пусть теперь есть некоторое тело Q (трехмерное изображение) и двумерное изображение S. Вопрос, который нас интересует, может быть поставлен так: как соотносятся Q и S, если известно, что S является проекцией тела Q на некоторую плоскость.

Эта задача является основой для восстановления трехмерного изображения по его плоским проекциям, что в свою очередь есть важнейшая составляющая моделей стереовосприятия для роботов, живых организмов и процедур построения изображений в томографии.

В целом рассмотрения такого рода для стереовосприятия приводят к следующей задаче. Имеются плоские проекции S_1 и S_2 тела Q (проекции на сетчатки). Однако в нашем распоряжении есть только

плоские изображения $\tilde{S}_1$ и $\tilde{S}_2$, полученные из соответственно S_1 и S_2 произвольными аффинными преобразованиями (какими — неизвестно). Кроме того, поточечное соответствие между $\tilde{S}_1$ и $\tilde{S}_2$ не задано. Надо восстановить тело Q.

Ясно, что в такой постановке снимается и вопрос о расстоянии между сетчатками, и о направлении лучей проекции, которыми получены точки на S_1 и S_2.

Оказывается, в такой постановке задача решаема: по $\tilde{S}_1$ и $\tilde{S}_2$ тело Q восстанавливается с точностью до аффинных преобразований, то есть строится тело $\tilde{Q}$, отличающееся от исходного Q некоторым аффинным преобразованием. При этом определяется и поточечное соответствие между $\tilde{S}_1$ и $\tilde{S}_2$. Это не эмпирический результат — соответствующая теорема доказывается.

К настоящему времени есть несколько компьютерных реализаций изложенного подхода.

1) Распознавание произвольных черно-белых фигур: цифр, букв, подписей, иероглифов, рисунков, и пр. На экране «мышкой» рисуются образцы фигур, которые нужно распознавать, они отправляются в память. Распознаваемый объект тоже рисуется на экране. Распознавание состоит в выборе того объекта из памяти, который лучше всего «натягивается» на объект на экране методами, описанными кратко в пунктах III и IV. Это наилучшее «натяжение» выводится на экран и является ответом. Распознавание не зависит от размеров, ориентации, положения на плоскости, сжатий-растяжений и локальных особенностей сравниваемых фигур.

Есть более сложный вариант программы, когда на экране — несколько фигур, возможно пересекающихся, наложенных друг на друга. Образцы из памяти должны сами «найти» на экране те фигуры, на которые они лучше всего «натянутся».

2) Восстановление трехмерных изображений по стереопарам (стереофотографиям). Исходными являются две фотографии объекта или сцены в двух несколько разных ракурсах (обычные стереопары). Методами, изложенными кратко в пункте VI, восстанавливается трехмерное изображение («виртуальное», в памяти компьютера). Это дает возможность получать новые проекции, то есть объект в новом ракурсе (фактически новые изображения объекта). Если это

делать последовательно, то на экране объект или сцена поворачиваются так, как если бы мы перемещались относительно них (конечно, в определенных пределах: восстанавливаются только те точки сцены, которые присутствуют на обеих исходных проекциях).

3) Распознавание мелодий. Мелодии, заносимые в память и распознаваемые — только из «чистых», «синусоидальных» звуков (они сравнительно легко создаются непосредственно на компьютере). Такие мелодии легко «визуализируются» в виде двумерной картинки. Распознаваемая мелодия может отличаться от записанной в память громкостью, темпом, тональностью.

Список литературы

[1] Козлов В. Н. Введение в математическую теорию зрительного восприятия. — М.: Изд-во Центра прикладных исследований при мех.-мат. ф-те МГУ, 2007.

[2] Алексеев Д. В. Кодирование изображений, инвариантное относительно проективных преобразований // Интеллектуальные системы. Т. 13. Вып. 1–4. М., 2009. С. 35–40.

Универсальная модель алгоритмов коллективного разума и ее реализация

Конончук Д. О., Окуловский Ю. С.
(Екатеринбург, Уральский государственный университет им. А. М. Горького)

kononchukdmitry@gmail.com

Алгоритмы коллективного разума (АКР) являются в настоящее время одной из самых динамично развивающихся отраслей алгоритмов искусственного интеллекта. Они основаны на так называемом «интеллекте толпы» — явлении, при котором система из множества слабо интеллектуальных, равноправных и децентрализованных сущностей проявляет свойства интеллектуальности. Примерами таких алгоритмов являются «муравьиные алгоритмы» [1], метод роя частиц [2], искусственные иммунные системы [3], и т. д.

В отличие от других интеллектуальных систем (нейронные сети, экспертные системы, генетические алгоритмы), для АКР до сих пор не существует общеизвестной системы поддержки, решающей то множество сервисных задач, которое возникает при программной эмуляции подобных алгоритмов. В связи с этим, в каждой реализации того или иного АКР приходится заново решать проблемы распараллеливания вычислений, ввода и вывода данных из системы, ее отладки и многие другие. Это усложняет процесс разработки и делает практически невозможной интеграцию различных решений. Кроме того, из-за отсутствия единого фреймворка производительность различных алгоритмов сложнее сравнивать: время работы и потребление памяти разных решений будут в первую очередь зависеть не от принципиальных различий алгоритмов, а от особенностей их реализации. В некоторых случаях в качестве фреймворка могут быть использованы так называемые системы агент-ориентированного моделирования (см. например, [4]). Но, к сожалению, эти системы не поддерживают многих свойств, являющихся существенными для АКР. Кроме того, среди них нет открытых решений на языке C#. Таким образом, для облегчения реализации, интеграции и сравнения различных типов и вариаций АКР требуется создание универсальной системы поддерж-

ки алгоритмов данного типа. Эта система должна подходить для создания на ее базе эмуляций всех общеизвестных АКР и, кроме того, быть удобной для использования, изучения и расширения (поскольку возможно применение ее также и для поддержки других систем, схожих с АКР).

На основании анализа и классификации существующих алгоритмов коллективного разума, была построена их общая модель. Основным элементом нашей модели АКР являются *агенты*. Муравьи в муравьиных алгоритмах и клетки крови в искуственных иммунных системах являются агентами. С точки зрения программиста, агенты представляют собой объекты различных классов, которые поддерживают некоторое количество методов, оформленных в интерфейсе. Помимо агентов, есть также *мир*, который содержит агентов и дополнительную информацию о них. Время в мире дискретизировано тактами, за каждый такт все агенты делают *ходы* последовательно. Также спроектирована система транзакций, позволяющая добиться независимости агентов друг от друга и иллюзии параллельности их работы. Эта система ограничивает свободу программирования логики в действиях агентов, допуская лишь такие действия, как перемещение агентов, их смерть и появление, а также локальное изменение характеристик мира. Указанных действий, однако, достаточно для большинства алгоритмов коллективного разума. Система транзакций также представляет собой эффективный механизм для распараллеливания алгоритмов.

Пространство в мире дискретизировано *ячейками*, в каждой ячейке может содержаться один или множество агентов, в зависимости от настроек алгоритма. Также в каждой ячейке может хранится произвольная структура данных, которая представляет доступную информацию о мире. Эта информация может быть изменена агентами, а также изменяться самопроизвольно, с течением времени. На ячейках задана *топология*, которая определяет для каждой ячейки перечисление соседних с ней ячеек. Это позволяет легко реализовывать локальное зрение агентов: при совершении хода агент, как правило, учитывает состояние лишь несколько соседних ячеек, соответственно, ему достаточно лишь пройти по предоставляемому топологией

перечислению. Разработаны топологии для двумерных и трехмерных сеток, а также топология для произвольного графа.

Агенты имеют следующие возможности взаимодействия друг с другом. Во-первых, они могут обмениваться информацией через структуры данных мира, изменяя их в рамках своих ходов. Такой подход наиболее характерен для муравьиных алгоритмов. Вторым способом взаимодействия является *эфир*, в рамках которого сообщения агентов становятся доступными остальным агентам на следующей итерации. Эфир может быть использован при моделировании иммунных систем при моделирования выброса клетками сигнальных веществ. Третьим способом взаимодействия является произвольное изменение агентами параметров друг друга, вызов методов и т. д. Данный способ взаимодействия несовместим с системой транзакций.

Приведенная модель, с одной стороны, удобна для программной реализации, и, с другой стороны, позволяет легко описывать АКР любого типа. Была создана последовательность прототипов и, затем, полноценная реализация этой модели, предоставляющая множество удобных инструментов для описания АКР. На каждой итерации разработки производились различные типы тестирования, результаты которых использовались для дальнейшего усовершенствования продукта. Система реализована на языке C# 4, под платформу .Net 4.0, совместима с платформой Mono. В ходе разработки активно использовались технологии контрактного и параллельного программирования.

На основании данной модели были реализованы известные алгоритмы из класса муравьиных алгоритмов, искусственных иммунных систем и роя частиц. В настоящее время нами ведутся работе по реализации других алгоритмов коллективного разума на базе разработанной системы. Кроме того, созданная модель применяется для создания интеллектуальных алгоритмов в компьютерных играх, а также для моделирования социума.

Список литературы

[1] Dorigo M. Optimization, Learning and Natural Algorithms / PhD thesis. — Politecnico di Milano, 1992.

[2] Kennedy J., Eberhart R. Particle Swarm Optimization // Proceedings of IEEE International Conference on Neural Networks IV. — 1995. P. 1942–1948.

[3] Dasgupta D. (editor) Artificial Immune Systems and Their Applications. — Berlin: Springer-Verlag, 1999.

[4] Axelrod R. The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. — Princeton: Princeton University Press, 1997.

Интеллектуальная система обучения программированию на REFAL

Котельников С. В. (Ярославль, Ярославский государственный университет им. П. Г. Демидова (ЯрГУ))

svkotelnikov@gmail.com

Актуальной задачей обучения в современных условиях является индивидуализация, которая обеспечивается использованием компьютера. Данная работа представляет собой интеллектуальную систему обучения языку логического программирования REFAL(в объеме REFAL-2).

Целью работы является разработка программной системы обучения языку REFAL, позволяющей предоставлять изучаемый материал, контролировать его освоение, как в части понимания материала, так и в части его практического использования.

В качестве системы обучения, отвечающей поставленным целям, была разработана программная система, реализующая стратегию поэтапного обучения, которая не только организует активное изучение материала, но и контролирует, а так же направляет процесс обучения. Обучение представлено тремя последовательными этапами: на первом этапе происходит изучение теоретического материала и тестирование его усвоения, второй этап обучения представлен выполнением упражнений, требующих осмысленного понимания материала, и последний этап, являющийся основным, представлен применением полученных знаний и навыков для написания полноценных программ на языке REFAL.

Система предоставляет пользователю ресурсы в зависимости от его уровня доступа. Незарегистрированный пользователь получает только возможность просмотра методических материалов и сравнительного состояния обучения студентов. Заметим, что сервер данной интеллектуальной системы предоставляет открытые для общего пользования веб-сервисы, позволяющие получить информацию о состоянии обучения студентов, что может быть использовано для размещения данной информации на сайте в интернете или на каком либо клиентском приложении, что делает обучение прозрачным для препо-

давателей, студентов и заинтересованных лиц. Зарегистрированный пользователь, не являющийся администратором, помимо возможностей незарегистрированного пользователя, также получает возможность прохождения обучения и возможность пользоваться интерпретатором REFAL. Наконец, пользователь с правами администратора имеет возможность изменять параметры системы, редактировать задания, учетные записи пользователей, сессии обучения и т. д. Рассмотрим подробно каждый из этапов обучения, которое предлагается пройти зарегистрированным в системе пользователям.

На первом этапе обучения студенту предоставляется теоретический материал и после его изучения студент переходит к стадии тестирования понятий материала. При этом случайным образом выбираются вопросы теста с закрытыми вариантами ответа, для решения которых требуется указать все правильные ответы. В случае правильного ответа на вопрос теста, количество вопросов теста уменьшается и студент получает возможность вернуться к изучению теории или перейти к ответу на следующий вопрос, а в случае неправильного ответа количество вопросов в тесте увеличивается и студент отсылается к той части теоретического материала, на основе которой был составлен вопрос. При превышении предельного количества ошибок, сеанс обучения заканчивается без сохранения прогресса обучения. При правильных ответах на все предлагаемые вопросы теста теоретической части обучения, происходит переход к следующему этапу обучения — этапу выполнения упражнений требующих осмысленного понимания материала.

На втором этапе обучения также случайным образом студенту предоставляется ряд заданий. Выполнение каждого из таких заданий состоит в написании в окне редактора некоторой части кода, проверяющей осмысленное понимание материала. Задания этого этапа обучения представлены написанием отдельных, синтаксически и семантически правильных, функций на языке REFAL. При неправильном решении задания студенту указывается ошибка, и он вновь отсылается к теоретическому материалу так же, как и на предыдущем этапе обучения. Число упражнений на этом этапе также растет в случае неправильных ответов и при превышении порога неправильных ответов сессия завершается. Только после правильных ответов

на все предлагаемые упражнения данной части обучения студент может перейти к главному этапу обучения.

Последний этап обучения представлен этапом обучения разработке программ на языке REFAL и является основной целью данной обучающей системы. На этапе обучения разработке программ в качестве заданий предлагается писать программы на REFAL по условиям, сформулированным в заданиях. Помимо кода программы предоставляется возможность написания набора тестов к программе, с возможностью их пошагового выполнения. Тесты представляют собой набор ожидаемых результатов выполнения программы для заданных входных значений. Пошаговое выполнение программы является очень важной частью обучения, позволяющей наглядно проследить, как происходит выполнение REFAL-программы и совпадает ли оно с ожидаемым. Данная работа дает возможность применить студенту приобретенные в процессе обучения знания и навыки при написании программ ня языке REFAL. Помимо пользовательских тестов существует «полный» набор тестов прикрепленных к задаче, не отображаемых интерфейсом, и по результату выполнения которых система определяет правильность написанного пользователем программного кода; в случае логической ошибки система выдает соответствующее сообщение, в котором указываются тесты на которых была обнаружена ошибка. В случае превышения предельного количества логических ошибок в задании, задание меняется и количество заданий предлагаемых к решению увеличивается. При превышении предельного количества неправильно решенных заданий сеанс обучения завершается.

Настройка системы обучения должна производиться при помощи выбора значений ее параметров во время опытной эксплуатации. К таким параметрам относятся: исходное количество заданий на каждом этапе обучения, количество заданий на которые уменьшается или возрастает количество предлагаемых заданий на отдельных этапах обучения при правильных и неправильных решениях соответственно, предельное допустимое количество ошибок на каждом из этапов обучения и т. д.

Система реализована на базе платформы Java EE с использованием ORM баз данных, а также различных сторонних Java и Java Script библиотек. Работа пользователей в данной системе осуществ-

ляется через web-интерфейс. Помимо пользовательских интерфейсов система предоставляет ряд веб-сервисов, что позволяет ей взаимодействовать с другими внешними системами и клиентами. Заметим, что в связи с отсутствием подходящих сторонних программных продуктов производящих пошаговую интерпретацию программ REFAL и их анализ, интерпретатор и синтаксический и семантический анализаторы написаны в рамках данной программной системы.

В 2011 г. данная работа была представлена на XIX Международной студенческой конференции-школе-семинаре «Новые информационные технологии», где была удостоена Диплома III степени за лучшую научную работу, представленную на конференции. Автор признателен своему научному руководителю профессору В. С. Рублёву за постоянное внимание к работе и чуткое руководство.

Список литературы

[1] Базисный Рефал и его реализация на вычислительных машинах. — М.: ЦНИПИАСС, 1977.

[2] Климов А. В., Романенко С. А. Система программирования Рефал-2 для ЕС ЭВМ. Описание входного языка. — М.: ИПМ им. М. В. Келдыша АН СССР, 1987.

[3] Романенко С. А. Реализация Рефала-2. — М.: ИПМ им. М. В. Келдыша АН СССР, 1977.

[4] Турчин В. Ф. Феномен науки: кибернетический подход к эволюции. — М.: ЭТС, 2000.

Метод организации работы с данными в прикладных системах распознавания образов

Максимова А. Ю.
(Донецк, Институт прикладной математики и механики НАН Украины)

Maximova.Alexandra@mail.ru

При разработке информационных систем часто возникает задача классификации, которая решается методами теории распознавания образов и анализа данных. В готовых системах распознавания образов (СРО), таких как «Распознавание 1.0» и STATISTICA реализован ряд известных методов, однако без участия специалиста выбрать подходящий и оценить качество полученного решения достаточно сложно [1]. Другим недостатком таких систем является организация работы с данными по средствам работы с файлами следующего формата: каждая строка определяет объект ω из множества всех объектов W в виде вектора значений его признаков. Для задач обучения с учителем также определяется номер класса образов, к которому относиться объект ω. Такая таблица называется выборкой прецедентов. Иногда приходится подготавливать две выборки прецедентов — обучающую и контрольную. При разработке крупных программной системы, внедряемых на производстве, процесс формирования выборок прецедентов необходимо автоматизировать учитывая особенности хранения и работы с данным в конкретной предметной области.

В работе предлагается метод организации работы с данными для системы распознавания образов. Структурная схема работы с данными приводится на рис. 1. На схеме присутствуют три формы представления одной и той же информации.

Изначально данные реального мира представлены в неформализованном виде: знания оператора, показания датчиков либо уже сохраненные файлы практически любых форматов. Второй формой представления данных является БД предметной области, основанная не реляционной модели данных, которая на данном этапе развития информационных систем получила широкое практическое применение. В ней, наряду с актуальными для задачи классификации дан-

ными, сохраняется и вся другая информация предметной области. Третья форма представления данных адаптирована для работы алгоритмов распознавания образов и хранит в виде таблиц коллекции обучающих выборок, так как в теории распознавания образов и анализа данных сформировалась практика представления обучающих выборок в виде стратифицированных таблиц.

Рис. 1. Структурная схема работы с данными.

Программный модуль «Анализатор» является интерфейсом между данными реального мира и БД предметной области. Он является уникальным для каждой конкретной прикладной задачи.

Программный модуль «Интерпретатор» является интерфейсом между данным предметной области и системой распознавания образов. Его основной функцией является предоставление всех необходимых данных для СРО. Данный модуль является универсальным для любой предметной области. Настройка его работы осуществляется с помощью пакета специальных правил, определенных для каждого задания СРО. Правила содержат информацию, позволяющую сформировать структуру классов образов и определить систему информационных признаков на основании БД предметной области. Доступ к объектам БД выполняется по шаблону «имя отношения». «имя атрибута». Также применяются ограничения на значения конкретных

атрибутов. Возможно использование разных правил для формирования обучающих и тестовых выборок.

Основные функции модуля «Интерпретатор»

В системе распознавания образов пользователь дает команду на выполнения определенного задания. Для каждого задания известен метод принятия решения, базирующийся на определенном алгоритме распознавания и способе контроля качества полученного решения. Для каждого метода принятия решения задан свой набор правил, который понимает модуль «Интерпретатор».

Предлагаемая методика организации прикладных систем распознавания образов позволяет создавать контекстно-независимые системы рапознавания образов, которые могут быть интегрированы в существующую информационную систему по средствам настройки модуля «Интерпретатор». Другим достоинством данного подхода является возможность добавлять в программную систему новые типы заданий, не меняя при этом структуру БД предметной области. Следует отметить также поддержку иерархической структуры классов образов и поддержку как пакетного режима, так и on-line режимов обработки данных.

Предлагаемый метод организации работы с данными использован при разработке автоматизированной системы контроля качества нефтепродуктов [2].

Список литературы

[1] Журавлев Ю. И., Рязанов В. В., Сенько О. В. Распознавание. Математические методы. Программная система. Практические применения. — М.: ФАЗИС, 2006.

[2] Козловский В. А., Максимова А. Ю. Нечеткая система распознавания образов для решения задач классификации жидких нефтепродуктов // Научные работы ДонНТУ, серия «Информатика, кибернетика и вычислительная техника». — 2011. № 13 (185). — С. 200–205.

Применение семантического графа для решения текстовых задач

Перпер Е. М. (Москва, МГУ им. М. В. Ломоносова)

e_m_perper@mail.ru

В работе рассматривается метод автоматического решения текстовых задач с помощью преобразований над семантическими графами текста задачи. При этом каждое преобразование соответствует определённому шагу решения задачи человеком.

Введение

Одним из ключевых при решении задачи обработки естественного языка является понятие смысла, которое достаточно трудно чётко определить. Одна из частных проблем обработки естественного языка — автоматическое решение текстовых задач — имеет то преимущество, что понятие «смысл» здесь определить легко. Смысл задачи выражается математическими действиями, которые необходимо произвести для нахождения её решения. Для того, чтобы представить текст задачи в виде чего-то более понятного для машины, чем просто набор слов, следует провести некоторый анализ этого текста. Как правило, выделяют три последовательных этапа анализа: морфологический, синтаксический и семантический. В методе автоматического решения текстовых задач, предложенном в [1] и реализованном А. С. Подколзиным на практике, результатом семантического анализа является набор логически формализованных утверждений, передающий смысл задачи. Обычно же семантический анализ завершается построением семантического графа, выражающего смысловые отношения между частями текста. Если семантические графы предложений задачи построены, то решение задачи можно свести к цепочке действий над этими графами.

Основные понятия и формулировка результатов

Уже упоминавшийся в данной работе семантический анализ — это анализ, который выявляет семантическую структуру предложения,

состоящую из семантических узлов и семантических отношений. Семантический узел — это слово, устойчивое словосочетание (например, «не хватило духа») или жёсткая синтаксическая группа («тридцать три»). Семантическое отношение — это универсальная связь между семантическими узлами, усматриваемая носителем языка в тексте. В системе русско-английского машинного перевода ДИАЛИНГ (см. [2]) семантическое отношение описывается как R(A,B), где A — зависимый член отношения, B — управляющий тип отношения, что означает: «A является R для B». Например, семантическое отношение, соответствующее словосочетанию «сумка из кожи», запишется как «матер(кожа, сумка)», а словосочетанию «жить в глуши» соответствует запись «лок(глушь, жить)». Если считать семантические узлы вершинами графа, а семантические отношения — рёбрами графа, то результат семантического анализа предложения можно представить в виде нагруженного графа, называемого семантическим графом. В данной работе предложена идея решения текстовых задач путём действий над семантическими графами предложений текста задачи. Приведён пример решения простой задачи предложенным методом.

Решение текстовых задач с помощью преобразования семантических графов

В каждой текстовой задаче содержится предложение, из которого мы узнаём, что же требуется от человека, решающего задачу. Вышеуказанное требование выражено местоимением (например, «сколько», «какое» и т. д.) Если в семантическом графе данного предложения (будем называть этот граф графом-вопросом задачи) заменить это местоимение на число, соответствующее ответу задачи, то полученный граф будет семантическим графом предложения, являющегося ответом задачи (будем называть его графом-ответом задачи). Граф-ответ задачи можно получить из графов предложений текста задачи, подвергнув их определённым логически обоснованным изменениям. Фактически, каждый этап решения человеком задачи можно свести к преобразованиям семантических графов предложений текста задачи.

Рассмотрим простую задачу и соответствующие ей семантические графы.

Задача. У Миши 2 яблока, а у Кати — на 5 яблок больше. Сколько яблок у Кати?

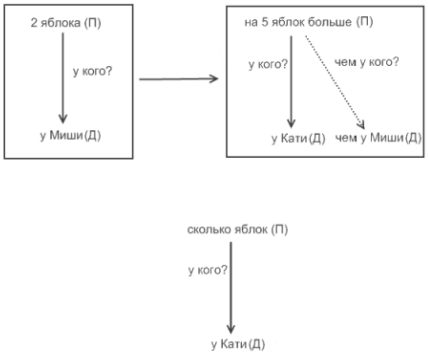


Рис. 1. Семантические графы задачи.

Обратим внимание на последний граф на рис. 1.: почти так и должен выглядеть граф ответа, лишь вместо «сколько» должно находиться конкретное число. На первом шаге решения задачи можно, пользуясь графом «у Миши — 2 яблока», заменить условия «чем у Миши» на «чем 2 яблока».

Это можно было бы записать в виде правила, которое можно было бы применять во всех аналогичных случаях: «[на N единиц объекта A больше, чем у субъекта S]; [M единиц объекта A у субъекта S] $\Longrightarrow$ [на N единиц объекта A больше, чем M единиц объекта A]». Словам «у Миши», вообще говоря, не соответствует какое-либо числовое значение, в то время как словам «2 яблока» — соответствует. Именно поэтому мы осуществляем замену (не противоречащую смыслу). Теперь мы можем произвести необходимые вычисления. Правило можно записать так: «[на N единиц объекта A больше]+[чем M единиц объекта A] = [M+N единиц объекта A]». После применения этого правила граф будет совпадать с графом-вопросом задачи, у которого вместо «сколько» находится число 7. Это и есть граф-ответ. Задача, таким образом, будет решена. Если задать определённый набор правил, соответствующих приёмам, используемых человеком при решении задачи, то задачу можно будет решать автоматически. Это можно устроить, например, так: для конкретного правила проверять, применимо ли оно ли оно к семантическим графам задачи в данный

момент, и если применимо, преобразовывать графы задачи по этому правилу; если же это правило неприменимо, перейти к следующему правилу. Если одновременно применимо несколько правил, можно рассмотреть каждый вариант по отдельности.

Рис. 2. Графы задачи после всех преобразований. Второй сверху — граф-ответ.

Таким образом, если в процессе решения задачи не происходит перехода к решению уравнений, описанный выше метод выглядит вполне приемлемым. Решению же уравнений довольно трудно сопоставить преобразование семантических графов, и для таких случаев требуются специальные методы решения.

Автор выражает благодарность профессору Эльяру Эльдаровичу Гасанову за научное руководство и помощь в подготовке статьи.

Список литературы

[1] Подколзин А. С. Компьютерное моделирование логических процессов. — М.: Физматлит, 2008.

[2] Сокирко А. Семантические словари в автоматической обработке текста (по материалам системы ДИАЛИНГ). http://www.aot.ru/docs/sokirko/sokirko-candid-4.html.

Суммирование по полугрупповой операции для двумерной задачи интервального поиска с фиксированной стороной

Пивоваров А. П. (Москва, МГУ им. М. В. Ломоносова)

pivizz@gmail.com

Двумерная задача интервального поиска с фиксированной стороной рассматривалась ранее, например в [1], однако там рассмотрен её перечислительный вариант. Здесь мы рассматриваем такой вариант этой задачи, в котором каждой точке базы данных приписан некий «ранг» — элемент некоторой коммутативной полугруппы. Задача состоит в том, чтобы для запроса, задающего ограничения сверху и снизу для одной координаты и только ограничение сверху для другой, найти результат полугрупповой операции на рангах всех тех и только тех элементов базы, которые удовлетворяют запросу. В качестве коммутативной полугруппы может, например, выступать, некоторое множество чисел с операцией взятия максимума.

В работе [2] рассматривалась задача, подобная данной, но основанная на двумерной задаче о доминировании. Были получены несколько семейств алгоритмов и с их помощью оценена функциональная сложность соответствующей задачи. Двумерная задача о доминировании является частным случаем двумерной задачи интервального поиска с фиксированной стороной. Таким образом, здесь мы рассматриваем задачу, частный случай которой разобран в [2].

Сформулируем результаты данной работы.

Понятия *вычислительной задачи информационного поиска (ВЗИП)* и *вычислительного информационного графа (ВИГ)* совпадают с предложенными в [3] и используемыми позднее в работе [2].

Пусть $\mathcal{Z} = (Z, \bigoplus)$ — коммутативная полугруппа с нулём. Операцию $\bigoplus$ в дальнейшем будем для простоты именовать *суммой*. Рассмотрим тип вычислительных задач информационного поиска $S^{\mathcal{Z}}_{int2^*} = \langle X_{int2^*}, Y^{\mathcal{Z}}_{int2^*}, Z, \rho_{int2^*}, \xi^{\mathcal{Z}}_{int2^*} \rangle$, где $X_{int2^*} = \{(x'_1, x''_1, x_2) \in [0,1]^2 \colon x'_1 \leqslant x''_1\}$, $Y^{\mathcal{Z}}_{int2^*} = [0,1]^2 \times Z$, отношение поиска ρ_{int2^*} определено таким образом, что для любых $x = (x'_1, x''_1, x_2) \in X_{int2^*}$ и $y = (y_1, y_2, y_*) \in Y^{\mathcal{Z}}_{int2^*}$ соотношение $x \,\rho_{int2^*}\, y$ справедливо тогда и только

тогда, когда одновременно выполнены условия $x'_1 \leqslant y_1 \leqslant x''_1$ и $y_2 \leqslant x_2$. Элементы множества записей Y можно рассматривать как точки на плоскости, которым дополнительно приписаны веса из Z. Функция ответа $\xi^{\mathcal{Z}}_{int2^*} : 2^Y \to Z$ определена на конечных подмножествах множества Y следующим образом (значение на бесконечных подмножествах Y определять не требуется): пусть $V = \{y^1, \ldots, y^k\} \subset Y$ и $y^i = (y^i_1, y^i_2, y^i_{\mathcal{Z}})$, тогда $\xi^{\mathcal{Z}}_{int2^*}(V) = y^1_{\mathcal{Z}} \bigoplus \cdots \bigoplus y^k_{\mathcal{Z}}$. Отдельно определим $\xi^{\mathcal{Z}}_{int2^*}(\varnothing)$ равным нейтральному элементу $\mathcal{Z}$.

Вычислительный информационный граф (ВИГ) для решения задач введённого выше типа $S^{\mathcal{Z}}_{int2^*}$ будем строить над базовым множеством $\mathcal{F}^{\mathcal{Z}}_{int2^*} = \langle F_{int2^*}, G_{int2^*}, H_{\mathcal{Z}}, M_{\mathcal{Z}}, m^{\mathcal{Z}}_0, \sigma_{id}\rangle$. Положим $F_{int2^*} = \{f^i_a \colon a \in [0,1], i \in \{1', 1'', 2\}\}$, где для любого $x = (x'_1, x''_1, x_2) \in X_{int2^*}$ значение $f^{1'}_a(x)$ равно 1 тогда и только тогда, когда $x'_1 \leqslant a$; значение $f^{1''}_a(x)$ равно 1 тогда и только тогда, когда $x''_1 \geqslant a$; значение $f^2_a(x)$ равно 1 тогда и только тогда, когда $x_2 \geqslant a$. Предикат f^2_0, значение которого равно 1 на любом запросе x, будем называть *тождественно единичным предикатом*. Множество переключателей $G_{int2^*} = \{g^i_a \colon a \in [0,1], i \in \{1', 1'', 2\}\}$, где для любого запроса $x = (x'_1, x''_1, x_2) \in X_{int2^*}$ имеет место $g^{1'}_a(x) = \begin{cases} 1, & \text{если } x'_1 > a; \\ 2, & \text{если } x''_1 \leqslant a; \end{cases}$ $g^{1''}_a(x) = \begin{cases} 1, & \text{если } x''_1 < a; \\ 2, & \text{если } x''_1 \geqslant a; \end{cases}$ $g^2_a(x) = \begin{cases} 1, & \text{если } x_2 < a; \\ 2, & \text{если } x_2 \geqslant a. \end{cases}$ Множество состояний ВИГ $M_{\mathcal{Z}}$ совпадает со множеством элементов рассматриваемой полугруппы Z, и начальное состояние $m^{\mathcal{Z}}_0$ — нейтральный элемент $\mathcal{Z}$, множество функций, меняющих состояние $H_{\mathcal{Z}} = \{h_z \colon h_z(b) = b \bigoplus z \ \forall b \in Z\}_{z \in Z}$ и функция выдачи ответа $\sigma_{id}(z) = z$ для любых состояний $z \in Z$.

Пусть U — некоторый ВИГ над базовым множеством $\mathcal{F}$. Пусть $I = \langle X_{int2^*}, V, Z, \rho_{int2^*}, \xi^{\mathcal{Z}}_{int2^*}\rangle \in S^{\mathcal{Z}}_{int2^*}$ — вычислительная задача информационного поиска ($V \subset Y^{\mathcal{Z}}_{int2^*}$). Пусть $\mathcal{U}(I, \mathcal{F})$ — множество ВИГ над базовым множеством $\mathcal{F}$, решающих задачу I. Сложностью задачи I при базовом множестве $\mathcal{F}$ и объёме q назовём число

$$T(I, \mathcal{F}, q) = \min\{T(U) \colon U \in \mathcal{U}(I, \mathcal{F}) \text{ и } Q(U) \leqslant q\}.$$

Здесь и далее под $T(U)$ будем понимать сложность графа U в худшем случае.

Если k — натуральное число, $S = \langle X, Y, Z, \rho, \xi \rangle$ — тип вычислительных задач информационного поиска, то обозначим

$$\mathcal{I}(k, S) = \{I = \langle X, V, Z, \rho, \xi \rangle \in S \colon |V| = k\}.$$

Будем исследовать функцию, характеризующую зависимость сложности задач класса ВЗИП $\mathcal{I}(k, S^{\mathcal{Z}}_{int2^*})$ от объёма памяти, доступной для алгоритма решения:

$$\mathcal{T}(k, S^{\mathcal{Z}}_{int2^*}, \mathcal{F}^{\mathcal{Z}}_{int2^*}, q) = \max_{I \in \mathcal{I}(k, S^{\mathcal{Z}}_{int2^*})} T(I, \mathcal{F}^{\mathcal{Z}}_{int2^*}, q).$$

Теорема 1. *Существуют такие положительные константы c_1, c_2, что для любой коммутативной полугруппы с нулём $\mathcal{Z} = (Z, \bigoplus)$ и любых натуральных $k \geqslant 3$ и $2 \leqslant s < k$ имеет место*

$$\mathcal{T}\left(k, S^{\mathcal{Z}}_{dom2}, \mathcal{F}^{\mathcal{Z}}_{dom2}, c_1 \frac{k \log_2 k}{\log_2 s}\right) \leqslant c_2 \frac{s}{\log_2 s} \log_2 k.$$

Доказательство данного результата мы не будем приводить здесь полностью. Опишем лишь основные идеи, используемые в нём. По сути задача состоит в том, чтобы для некоторых констант c_1, c_2 получить семейство алгоритмов с указанными выше ограничениями на требуемую память и время работы в худшем случае. Для этого предлагается использовать некоторые идеи из [2]. Во-первых для конкретной задачи с фиксированной базой данных V (положим $k = |V|$) и выбранного параметра s, такого что $2 \leqslant s < k$, необходимо построить дерево подбаз T^s_V, каждой вершине которого соответствует некоторая подбаза V. Корню дерева соответствует всё V, а затем из каждой вершины, которой в соответствие поставлено множество V' с $|V'| > s$, то делим V' на s одинаковых частей $V'_1, \ldots, V'_s$ (если $|V'|$ не делится нацело на s, то некоторые части будут содержать на один элемент больше оставшихся). Разбиение следует делать таким образом, чтобы первые координаты записей из множеств с меньшими номерами не превосходили первых координат записей множеств с большими номерами. Затем из V' выпускается s потомков, которым ставятся в соответствие множества $V'_1, \ldots, V'_s$.

По полученному дереву T_V^s строится три как бы параллельные структуры, такие что для каждой вершины v дерева T_V^s каждая из структур содержит по соответствующей вершине. Пусть это будут вершины v_1, v_2, v_3 и вершине v соответствует подбаза $V' \subseteq V$. Тогда прохождение запроса $x = (x'_1, x''_1, x_2)$ в вершину v_1 логически означает поиск в базе V' и при этом про все элементы V' известно, что их первая координата не меньше x'_1; прохождение x в v_2 логически означает поиск в базе V' и при этом про все элементы V' известно, что их первая координата не превосходит x''_1; наконец, прохождение запроса x в v_3 логически означает поиск в базе V' без каких-либо дополнительных знаний о первых координатах записей из V'. Структуры строятся таким образом, что любой запрос идет сначала по одному пути из вершин третьей структуры, а затем разделяется в общем случае и идет по одному пути в первой и по одному пути во второй структуре.

При таком подходе оказывается возможным воспользоваться техникой частичного каскадирования, что позволяет на каждом шаге в определённом смысле запоминать положение запроса по второй координате относительно того множества записей, которое рассматривается в конкретной части создаваемой структуры.

Автор выражает благодарность научному руководителю профессору Э. Э. Гасанову.

Список литературы

[1] McCreight E. M. Priority search trees // SIAM Journal of Computing. 14 (2). 1985. P. 257–276.

[2] Пивоваров А. П. Функциональная сложность задачи подсчёта для двумерной задачи о доминировании // Интеллектуальные системы. Т. 16. Вып. 1–4. С. 147–180. 2012.

[3] Пивоваров А. П. Моделирование вычислительных задач информационного поиска // Интеллектуальные системы. Т. 14. Вып. 1–4. С. 229–250. 2010.

[4] Chazelle B., Guibas L. J. Fractional cascading: I. A Data Structuring Technique // Algorithmica. 1. 1986. P. 133–162.

Моделирование динамических баз данных

Плетнев А. А. (Москва, МГУ им. М. В. Ломоносова)

PletnevOrel@rambler.ru

Функционирование базы данных — это обработка потока запросов типа поиск, вставка и удаление. При этом в результате запросов типа вставка и удаление база данных изменяется, а на запросы типа поиск выдается ответ. Если поток запросов на поиск существенно преобладает над запросами на изменение базы данных, то такие базы данных называются статическими. Для исследования таких баз данных предназначены информационные графы [1]. Если же поток запросов на изменение базы данных сравним с потоком запросов на поиск, то такие базы данных называются динамическими, и моделированию таких баз данных посвящена данная работа.

Предлагаемая модель динамических баз данных построена на взаимодействии конечного детерминированного автомата и информационного графа. Задача автомата перестраивать информационный граф при изменении базы данных, тем самым обрабатывая динамические запросы пользователя. Эту структуру будем называть динамическим информационным графом.

В формальном определении понятия ИГ используются 4 множества: множество запросов X; множество записей Y; *множество F одноместных предикатов*, заданных на множестве X; *множество G одноместных переключателей*, заданных на множестве X (*переключатели* — это функции, область значений которых является начальным отрезком натурального ряда). Понятие *информационного графа* (ИГ) определяется следующим образом. Берется конечная многополюсная ориентированная сеть. В ней выбирается некоторый полюс, который называется корнем. Остальные полюса называются листьями и им приписываются записи из Y, причем разным листьям могут быть приписаны одинаковые записи. Некоторые вершины сети (в том числе могут быть и полюса) называются переключательными и им приписываются переключатели из G. Ребра, исходящие из каждой из переключательной вершин, нумеруются подряд, начиная с 1, и называются переключательными ребрами. Ребра, не являющиеся

переключательными, называются предикатными и им приписываются предикаты из множества F. Таким образом нагруженную многополюсную ориентированную сеть называем ИГ над базовым множеством $\mathcal{F} = \langle F, G\rangle$, где $F = \{f_j, j \in I\}$, $G = \{g_i, i \in J\}$, I, J — конечные множества индексов. Определим три множества функций изменения индексов: $R^1 = \{r_c^1 : (I \cup J \cup Y)^{n_c} \times Y \to I,\ c \in C_1\}$, $R^2 = \{r_c^2 : (I \cup J \cup Y)^{n_c} \times Y \to J,\ c \in C_2\}$, $R^3 = \{r_c^3 : Y^{n_c} \to Y,\ c \in C_3\}$. Введем три множества переменных: $Z = \{z_i\}_{i=1}^{\infty}$, где z_i принимают значения из I, $V = \{v_j\}_{j=1}^{\infty}$, где v_j принимают значения из J, $W = \{w_k\}_{k=1}^{\infty}$, где w_k принимают значения из Y.

Рассмотрим произвольный ИГ U. Заменим каждый индекс предиката на некоторую переменную из Z, каждый индекс переключателя — на переменную из V, а каждую запись — на переменную из W. После этого сопоставления получим нагруженный граф, который назовем простым шаблоном. Если каждый индекс предиката мы заменим на некоторую формулу над множеством переменных Z и множеством функций R^1, каждый индекс переключателя — на формулу над множеством переменных V и множеством функций R^2, а каждую запись — на формулу над множеством переменных W и множеством функций R^3, то полученный нагруженный граф назовем шаблоном.

Будем говорить, что ИГ U и простой шаблон $\mathcal{T}$ согласованы, если они совпадают как графы, и если в ИГ U встречаются одинаковые индексы предикатов (переключателей и записей), то в соответствующих местах шаблона $\mathcal{T}$ находятся одинаковые переменные из Z (V и W). Возникшее соответствие между переменными и индексами назовем интерпретацией данного согласования.

Локальным преобразованием назовем пару $p = (\mathcal{T}_1, \mathcal{T}_2)$, где $\mathcal{T}_1$ — простой шаблон, $\mathcal{T}_2$ — шаблон, в формулах которого встречаются только переменные, входящие в простой шаблон $\mathcal{T}_1$, и возможно еще одна переменная из множества W.

Если ИГ U и простой шаблон $\mathcal{T}_1$ согласованы, то применением локального преобразования $p = (\mathcal{T}_1, \mathcal{T}_2)$ к ИГ U назовем ИГ U', получающийся из шаблона $\mathcal{T}_2$ подстановкой вместо каждой формулы значения данной формулы в интерпретации согласования ИГ U и простого шаблона $\mathcal{T}_1$.

Пример преобразования p показан на рис. 1, где $r^2(v_1, v_2, w) \in R^2; r_1^3(w_1, w_2, w), r_2^3(w_1, w_2, w), r_3^3(w_1, w_2, w) \in R^3$, $a = (w_1, w_2, w)$, $b = (v_1, v_2, w)$, $c = \min(v_1, v_2, w)$, $r^2(b) = (c, \min(\{v_1, v_2, w\} \backslash \{c\}))$, $r_1^3(a) = \min(w_1, w_2, w)$, $r_3^3(a) = \max(w_1, w_2, w)$, $r_2^3(a) = \{w_1, w_2, w\} \backslash \{r_1^3(a), r_3^3(a)\}$.

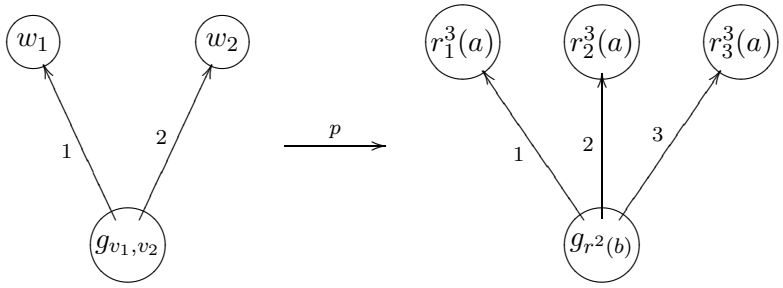


Рис. 1.

Такое преобразование используется при добавлении новой записи в 2–3 деревьях.

Далее будем рассматривать класс информационных графов $\mathcal{K}(N)$, $N \in \mathbb{N}$, таких, что степень инцидентности любой вершины графа не превосходит N. Окрестностью радиуса L некоторой вершины ИГ, будем называть множество вершин ИГ, таких что длина пути до которых от данной не превосходит L, и множество ребер, соединяющих эти вершины, вместе с нагрузками этих вершин и ребер.

Назовем кодом вершины ИГ пару (k_1, k_2), где $k_1 = 0$, если вершина предикатная; $k_1 = 1$, если она переключательная; $k_2 = 0$, если вершина корень; $k_2 = 1$, если вершина листовая; $k_2 = 2$ в остальных случаях. Кодом ребра назовем число, которое равняется 0, если оно предикатное и 1, если переключательное.

Кодом окрестности на запросе x назовем информацию о коде каждой вершины и ребра данной окрестности, а также информацию о значениях всех предикатов и переключателей на запросе x (так же возможно, что код содержит дополнительную информация о пометках на ребрах и вершинах графа).

Пусть $\mathcal{P}$ — некоторое конечное множество локальных преобразований, L, N — натуральные числа, U — ИГ над базовым множеством $\mathcal{F} = \langle F, G \rangle$ из класса $\mathcal{K}(N)$, $\mathcal{A}$ — конечный автомат, входной

алфавит, которого есть множество кодов всевозможных окрестностей радиуса L вершин ИГ из , а выходной алфавит описывает реакции автомата, такие как перемещение автомата по графу, выбор локального преобразования из $\mathcal{P}$, сигнал на завершение работы и, возможно, расстановку или изменение пометок на рассматриваемой окрестности графа. Пару $(\mathcal{A}, U)$ назовем динамическим информационным графом (ДИГ) над $\mathcal{F}$ и $\mathcal{P}$.

Определим *функционирование* ДИГ $(\mathcal{A}, U)$ на запросе. Если запрос есть запрос на поиск, то функционирование ДИГ совпадает с функционированием ИГ U и не задействует автомат $\mathcal{A}$. Если запрос является запросом на вставку или удаление, то функционирование ДИГ происходит следующим образом. В начальный момент текущей вершиной объявляется корень ИГ и считается, что лишних пометок на графе нет. На вход автомата подается код окрестности текущей вершины. Если выход автомата предписывает передвижение по графу, то текущая вершина изменяется. Если выход автомата предписывает некоторое локальное преобразование, то в случае если окрестность текущей вершины согласована с левой частью преобразования, то рассматриваемая окрестность заменяется на результат применения данного локального преобразования. Если окрестность текущей вершины не согласована с левой частью преобразования, то функционирование завершается с ошибкой. Если выход автомата предписывает изменение пометок рассматриваемой окрестности, то эти изменения выполняются. Если выход автомата сигнализирует о завершении работы, то обработка запроса завершается успешно.

Пусть дана задача информационного поиска (ЗИП) $I = \langle X, V, \rho \rangle$. Скажем, что ДИГ решает ЗИП I, если ответ на произвольный запрос x типа поиска, содержит те и только те записи $y \in V$, что $x \rho y$; если функционирование на произвольном запросе типа вставки (удаления) записи $y \in Y$ завершается успешно, результирующий граф не содержит лишних пометок и решает ЗИП $\langle X, V \cup \{y\}, \rho \rangle$ ($\langle X, V \setminus \{y\}, \rho \rangle$).

Если каждому локальному действию сопоставить число, характеризующее его сложность, то можно вводить сложность ДИГ на запросах, и другие сложностные характеристики ДИГ.

Далее можно ставить задачу синтеза, а именно задачу построения для произвольной ЗИП такого ДИГ, который бы решал данную ЗИП и имел при этом как можно меньшую сложность.

Автор благодарит профессора Э. Э. Гасанова за постановку задачи и помощь в работе.

Список литературы

[1] Гасанов Э. Э., Кудрявцев В. Б. Теория хранения и поиска информации. М.: ФИЗМАТЛИТ, 2002.

Компьютерное моделирование логических процессов

Подколзин А. С. (Москва, МГУ им. М. В. Ломоносова)

Мечта человечества об искусственном интеллекте имеет давнюю историю. На сегодняшний день она, увы, остается лишь мечтой. Производительность и память современных вычислительных систем огромны и продолжают быстро увеличиваться. Они относительно дешевы и широко распространены. Однако, среднестатистический стандарт использования этих систем по-прежнему сводится к некоему гибриду телефона, записной книжки, пишущей машинки, телевизора и тренажера для отработки простейших реакций. Понимание мира хотя бы на уровне пятиклассника, с его способностью читать книги и воспринимать смысл изображений, остается далеко за рамками возможностей существующих «интеллектуальных систем». О таких вещах, как научно-техническое творчество в подлинном смысле этого слова, говорить и вовсе не приходится. Автоматизировать удается лишь сравнительно простые и узко специализированные функции естественного интеллекта. Так как данное положение наблюдается уже достаточно давно, есть все основания говорить о наличии некоторого принципиального барьера, возникшего на пути к искусственному интеллекту. Очевидно, этот барьер никак не связан с «физическими» возможностями сегодняшних компьютеров, а связан с острым дефицитом наших знаний о логических процессах, лежащих в основе любой интеллектуальной деятельности. По существу, речь должна идти о создании новой науки, изучающей эти процессы, своего рода «логической динамики».

Нельзя сказать, чтобы попыток начать изучение логических процессов не было вовсе. Прежде всего, они предпринимались в рамках математической логики. Однако, ее попытки создания универсальных эффективных процедур решения задач успехом не увенчались. Из общих соображений возникла лишь общая схема организации перебора. Экспоненциально растущая трудоемкость ограничила сферу применимости этих процедур самыми простыми задачами и даже поставила под сомнение возможность создания искусственного интеллекта вообще. Впрочем, человек вполне эффективно решает задачи, избегая «экспоненциальных переборов», и неудача попытки решить

проблему математическим путем означает лишь заведомую недостаточность умозрительных построений для анализа такого сложного явления, как интеллект.

С другой стороны, практически сразу после появления компьютеров стали создаваться программы для решения «интеллектуальных» задач в самых различных областях. Они имитировали действия эксперта в соответствующей области и получили название экспертных систем. Рассматривались задачи на доказательство теорем в формальной логике, задачи на доказательство геометрических теорем, формальное интегрирование, тригонометрические уравнения, уравнения в целых числах, шахматные задачи, и многое, многое другое. Собственно, сейчас трудно найти такую предметную область, для которой не было бы создано множество экспертных систем. Некоторые из них весьма развиты и успешно могут конкурировать с человеком. Например, для шахмат была создана система, одержавшая победу даже над чемпионом мира. Может быть, некое объединение всех этих систем и является долгожданным искусственным интеллектом? Однако, при ближайшем рассмотрении становится ясно, что богатство здесь иллюзорное. В сколь-нибудь сложных областях развитие экспертной системы останавливалось на простейших задачах, демонстрируя скорее невозможность освоения области современными средствами, чем успешное ее преодоление. В отдельных случаях (в тех же шахматах) удавалось достичь внешне впечатляющих результатов не за счет проникновения в логику действий человека, а лишь методом «грубой силы», используя огромную производительность компьютеров для перебора. Хорошо известны высказывания на этот счет одного из пионеров в области искусственного интеллекта чемпиона мира по шахматам М. М. Ботвинника.

Разумеется, во многих, сравнительно простых с логической точки зрения, предметных областях были созданы эффективные прикладные экспертные системы. Однако, здесь возникло другое негативное явление. Разработчики, находясь под впечатлением достигнутых успехов, экстраполировали принципы организации своих систем далеко за рамки рассматривавшейся предметной области, возводя их в ранг «общей теории» искусственного интеллекта. Естественно, в сколь-нибудь более сложных или просто сильно отличающихся от ис-

ходной предметных областях эти теории никакой пользы не принесли. Здесь прослеживается всё та же линия основанных на недостаточной информации умозрительных построений.

Впрочем, отвлечемся от заведомой слабости какой-то (неважно, малой или большой) части узкоспециализированных экспертных систем. Предположим, что каждая из них в своей области достигла уровня самого опытного эксперта. И в этом случае формальное их объединение никакого искусственного интеллекта не даст. Во-первых, из-за проблем организации взаимодействия. Как известно, знания различных предметных областей тесно взаимосвязаны друг с другом. Нельзя хорошо решать задачи по геометрии, совсем не зная алгебры, нельзя овладеть физикой, не зная математики, и т. д. и т. п. Чтобы научить систему пониманию смысла изображений, нужно заложить в нее примерно столько же знаний о мире и приемов использования этих знаний в рассуждениях, сколько их понадобилось бы для обучения другой системы пониманию естественного языка. Никакой самый умный переключатель между изолированными экспертными системами здесь не поможет, так как решение задач «на стыке» различных областей потребует одновременного их участия. Если бы подобная попытка создания «объединенного» искусственного интеллекта и в самом деле когда-либо была бы предпринята, то она немедленно привела бы к необходимости устранения перегородок между отдельными частями и, фактически, созданию заново некоей «универсальной» экспертной системы.

Но и так мы не получили бы искусственного интеллекта. Экспертная система лишь зафиксировала бы текущий уровень развития наших знаний и умений. Несмотря на свою неоспоримую практическую полезность, она никоим образом не могла бы претендовать на ту роль мощного ускорителя научно-технического прогресса, которую призван сыграть искусственный интеллект. Такая роль однозначно предполагает способность системы к саморазвитию. Простейшая адаптация, связанная с оптимизацией системой каких-то своих параметров, не в счет. Подлинное саморазвитие требует механизмов, создающих новые приемы решения задач на основе имеющихся знаний и пополняющих знания с ориентацией на решение задач. Без этого невозможно решение нестандартных задач, а они в творчестве

составляют подавляющую долю. Интеллектуальной системе просто необходима мощная техника компиляции, выводящая ее архитектуру и программы из идей, порождаемых ею же на языке, максимально приближенном к языку теоретических знаний. Собственно генерация идей происходит на пограничном слое между теоретическими знаниями и практическими приемами, который и должен стать главным объектом изучения.

Здесь мы снова приходим к вопросу о целесообразности создания изолированных друг от друга интеллектуальных систем в различных предметных областях. Конечно, для игры в шахматы, знание, скажем, аналитической геометрии может показаться излишним. Однако, механизмы, порождающие новые приемы на основе теоретических знаний, имеют общелогический характер. Каждая предметная область вносит в копилку таких механизмов, типов приемов и стандартов рассуждений что-то свое. И для получения полной коллекции необходимо изучить как можно более широкий спектр различных предметных областей, чтобы уже впоследствии вернуться «во всеоружии» к какой-то одной области.

Таким образом, путь к искусственному интеллекту лежит через создание экспертной системы, охватывающей достаточно широкий спектр предметных областей и программируемой на уровне «пограничного слоя» между теорией и алгоритмами. Она должна послужить своего рода микроскопом, дающим достаточно богатый фактический материал для изучения общих принципов организации логических процессов: принципов эффективного управления рассуждениями при решении задач, принципов извлечения новых приемов из теорем, принципов автоматического развития теорий, и т. д. Альтернативой является лишь продолжение бесплодных попыток умозрительного угадывания этих принципов.

Данная система вовсе не обязана претендовать на роль скольнибудь сильного решателя. В первую очередь, она должна давать множество примеров, объясняющих, как можно было бы управлять рассуждениями, чтобы решить ту или иную конкретную задачу, не прибегая к непомерно большому перебору. Разумеется, по мере накопления средств она начнет многое делать самостоятельно. Однако, в иных предметных областях возможность полной проработки те-

мы при обучении «вручную» вообще представляется сомнительной, и здесь придется ограничиться накоплением единичных траекторий процессов, которые все же дадут пищу для обобщений. Действительно сильные решатели должны будут появиться впоследствии, когда накопление знаний о логических процессах позволит создать интеллектуальные системы, способные самообучаться. Пока таких систем нет, и речь идет лишь о начале систематического исследования, опирающегося на некий компьютерный «логический микроскоп».

Предлагаемая работа посвящена развитию компьютерной системы, которая могла бы стать основой для универсальной экспертной системы («решателя») указанного выше типа и одновременно «микроскопом» для изучения общих принципов организации логических процессов, включая самообучение. Система основана на применении нового языка программирования ГЕНОЛО́Г, расположенного в точности на пограничном слое между теорией и алгоритмами. Прием на этом языке задается как теорема предметной области, снабженная некоторой алгоритмизирующей разметкой.

Обучение системы предпринималось в широком спектре предметных областей и позволило выявить достаточно богатую коллекцию способов эффективной алгоритмизации теорем. Использование этой коллекции, с одной стороны, существенно ускорило и упростило процесс «ручного» обучения решателя, а с другой стороны — позволило вплотную приблизиться к автоматическому синтезу приемов.

Следует заметить, что не только программирование решателей, но даже традиционное «нелогическое» программирование в конечном счете выводит свои конструкции из теорем. Это означает, что в самообучающихся интеллектуальных системах вообще все программирование неизбежно должно будет проходить через уровень ГЕНОЛО́Га, и он вполне может претендовать на роль «универсального алгоритмического языка будущего». Разумеется, это придает особую значимость развитию компиляторов для языков такого типа.

При развитии ГЕНОЛО́Га и обучении решателя, происходивших одновременно, были рассмотрены такие предметные области, как элементы дискретной математики, алгебра множеств, элементарная алгебра, элементарная геометрия, аналитическая геометрия, линейная алгебра, математический анализ, дифференциальные уравнения,

комплексный анализ, вычисления, теория вероятностей, элементы общей алгебры, ряд разделов элементарных физики и химии, шахматы, текстовые задачи (в двух вариантах — на логическом либо естественном языке), анализ рисунков. Обучающий материал содержал примерно 11000 задач, из которых были извлечены примерно 39000 приемов. В отдельных областях даже такое, сугубо предварительное, обучение позволило выйти на неплохой уровень решения задач средней сложности. В других — удалось проработать лишь небольшое количество примеров, которые, однако, дали возможность существенно скорректировать первоначальные представления и получить сравнительно устойчивую архитектуру для дальнейшего развития.

Предпринимается анализ возможностей самообучения системы. К явлению самообучения следует относиться с определенной осторожностью, так как пока известна лишь одна самообучающаяся в абсолютном смысле интеллектуальная система — это вся человеческая цивилизация в целом. Процессы подобного масштаба далеко выходят за рамки возможностей современных компьютеров. Практический интерес представляет сейчас «ограниченное» самообучение отдельного человека, заключающееся в самостоятельном усвоении по книгам или иным источникам возможно большей суммы знаний и приобретении навыков их практического использования путем тренировки на обучающих примерах. Целью работы с компьютерной системой должно явиться освоение аналогичного режима самообучения: ей сообщается теория и дается поток задач, на котором происходят самостоятельное создание приемов и оптимизация их совместного поведения.

Центральным здесь является создание развитой классификации логических типов приемов, которая позволяла бы по заданной теореме генерировать серию предположительно интересных для задачи приемов («идей»), проверять их ценность и отбирать по окончании решения задачи те приемы, которые оказались результативными. Работа над созданием такой классификации начата, однако она связана с трудоемкой итеративной стандартизацией приемов созданного (в общем, пока относительно «сырого») решателя и оптимизацией его логических режимов.

Архитектура и языки обучения системы описаны в первом томе монографии по компьютерному моделированию логических процессов [1]. Планируется издание последующих томов, в которых будут описаны многообразие приемов решателя, возникших при его обучении, и средства автоматизации синтеза приемов.

Автор выражает искреннюю благодарность В. Б. Кудрявцеву, поддержка которого сделала возможным проведение данного исследования.

Список литературы

[1] Подколзин А. С. Компьютерное моделирование логических процессов. Архитектура и языки решателя задач. М.: Физматлит, 2008.

О нелинейных характеристиках нейронных схем в произвольных базисах

Половников В. С. (Москва, МГУ им. М. В. Ломоносова)

pvser@mail.ru

В данной работе получены следствия некоторых результатов моей кандидатской диссертации, которые могли бы быть полезны при построении нейронных схем из элементов, представляющих собой полную (по операции суперпозиции) систему F кусочно-параллельных функций, содержащих все линейные элементы (сложение, константы из $\mathbb{R}$, умножения на константы из $\mathbb{R}$) и некоторую нелинейную часть.

Нам понадобятся слующие определения, обозначения и утверждения диссертации (нумерация теорем сохранена):
L — множество линейных функций,
PC — множество кусочно-постоянных функций,
$PP = \{f | f = f_c + f_l, f_c \in PC, f_l \in L\}$ — множество кусочно-параллельных функций.
Нейронной схемой Мак-Каллока-Питтса называется схема, построенная с использованием линейных элементов и нелинейной θ-функции Хевисайда.

$$\theta(x) = \begin{cases} 1, & \text{если } x \geqslant 0 \\ 0, & \text{если } x < 0. \end{cases}$$

Теорема 3. *Множество функций, реализуемых нейронными схемами Мак-Каллока-Питтса, совпадает с множеством всех кусочно-параллельных функций. И для любой кусочно-параллельной функции существует нейронная схема Мак-Каллока-Питтса нелинейной глубины не более двух.*

Функция $f(t)$ из $PC(1)$ называется *С-финитной функцией от* 1 *переменной*, если $\exists T_f, c_f \in \mathbb{R}$, $T_f > 0$ такие, что при $t \in (-\infty, -T_f) \cup (T_f, +\infty)$ выполнено $f(t) = c_f$. Обозначим класс С-финитных функций от 1 переменной $\Phi(1)$.

Рассмотрим кусочно-постоянную функцию $f : \mathbb{R}^n \to \mathbb{R}$ и прямую $\mathfrak{N}$ заданную параметрически: $x_1 = a_1 t + b_1, \ldots, x_n = a_n t + b_n$. Функция $g(t) = f(a_1 t + b_1, \ldots, a_n t + b)$ называется *сечением кусочно-постоянной функции f прямой* $\mathfrak{N}$.

Функция $f \in PC$ называется *C-финитной*, если сечение f любой прямой является C-финитной функцией от 1 переменной. Обозначим класс всех C-финитных функций через Φ.

Введем *класс C-финитно-линейных функций*:

$$\Phi L = \{f | f = \phi + l, \phi \in \Phi, l \in L\}.$$

Теорема 4. *F полно в PP по операции суперпозиции тогда и только тогда, когда $F \not\subseteq \Phi L$.*

Пусть $f(x_1, \ldots, x_n)$ — произвольная кусочно-параллельнейная функция в $\mathbb{R}^n$, заданная k гиперплоскостями. За $Z_{MP}(S)$ обозначим число элементов θ в схеме (S, f), то есть *нелинейную сложность* схемы (S, f). Соответственно определим *сложность реализации функции* схемой Мак-Каллока-Питтса

$$Z_{MP}(f) = \min_{S \text{ реализует} f} Z_{MP}(S),$$

где минимум берется по всем нейронным схемам Мак-Каллока-Питтса реализующим f.

Функция Шеннона $Z_{MP}(k) = \max\limits_{f \in PP: f \text{задана } k \text{ гиперплоскостями}} Z_{MP}(f)$.

Теорема 5. *Для функции Шеннона $Z_{MP}(k)$ верна оценка*
1) при $k \leqslant n$:

$$3^k - 1 \leqslant Z_{MP}(k) \leqslant 3^k + 2k.$$

2) при $k > n$:

$$3\sum_{i=0}^{n-1} C_{k-1}^i 2^i + 2^n \sum_{j=n-1}^{k-2} C_j^{n-1} - 1 \leqslant Z_{MP}(k) \leqslant$$
$$\leqslant 3\sum_{i=0}^{n-1} C_{k-1}^i 2^i + 2^n \sum_{j=n-1}^{k-2} C_j^{n-1} + 2k.$$

Из доказательства теоремы 4 вытекает выразимость θ функции через нефинитно-линейную функцию из F, которая согласно теореме

4 обязана содержаться в F ввиду полноты F. Причем схема строится явно со следующими параметрами: нелинейная глубина 4 и нелинейная сложность 4. Для произвольной кусочно-параллельной функции, рассмотрим ее нейронную схему Мак-Каллока-Питтса нелинейной глубины 2 (из теоремы 3). Заменим в ней элемент θ на нейронную схему над F (из теоремы 4). Получаем следущее утверждение:

Следствие 1. *Для любой кусочно-параллельной функции существует нейронная схема над F нелинейной глубины не более* 8.

Далее, аналогично $Z_{MP}(k)$ определим функцию Шеннона $Z_F(k)$ для функций, реализуемых схемами над F. Аналогичной заменой элемента θ на нейронную схему над F, обобщается верхняя оценка функции Шенона из Теоремы 5.

Следствие 2. *Для функции Шеннона $Z_F(k)$ верна оценка*
1) при $k \leqslant n$:

$$Z_{MP}(k) \leqslant 4(3^k + 2k).$$

2) при $k > n$:

$$Z_{MP}(k) \leqslant 4(3\sum_{i=0}^{n-1} C_{k-1}^i 2^i + 2^n \sum_{j=n-1}^{k-2} C_j^{n-1} + 2k).$$

За помощь в научных изысканиях выражаю благодарность к.ф.-м.н. Часовских А. А.

Список литературы

[1] Половников В. С. Об оптимизации структурной реализации нейронных сетей / Дисс. канд. ф.-м. наук. — М., 2007.

[2] Автоматы. Сборник статей / Под ред. К. Э. Шеннона, Дж. Маккарти. Пер. с англ. под ред. А. А. Ляпунова. — М.: Изд. иностр. лит., 1956.

[3] Хайкин С. Нейронные сети: полный курс / 2-е издание. — М.: Вильямс, 2006.

[4] McCulloch W. S., Pitts W. A logical calculus of the ideas immanent in nervous activity // Bulletin of Mathematical Biophysics. Vol. 5. 1943. P. 115–133.

[5] Кудрявцев В. Б. Функциональные системы. — М.: Изд-во МГУ, 1982.

[6] Лупанов О. Б. Асимптотические оценки сложности управляющих систем. — М.: МГУ, 1984.

[7] Кудрявцев В. Б., Алешин С. В., Подколзин А. С. Введение в теорию автоматов. — М.: Наука, 1985.

[8] Rosenblatt F. The perceptron: a probabilistic model for information storage and organization of the brain // Psychological Review. 65. 1958. P. 386–408.

[9] Кофман А. Введение в прикладную комбинаторику. — М.: Наука, 1975.

[10] Гельфонд А. О. Трансцендентные и алгебраические числа / Изд. 2. — М.: КомКнига, 2006.

Моделирование электронного билингвального методического словаря открытого типа как составляющая процесса формирования интеллектуальной информационной системы

Романова А. А. (Смоленск)

Vackhanka88@mail.ru

Одной из задач, решаемых интеллектуальными информационными системами, является обучение. Это подразумевает использование компьютера для обучения какой-либо дисциплине или предмету. Системы обучения диагностируют ошибки при изучении дисциплины с помощью ЭВМ и подсказывают правильные решения. Структура интеллектуальной системы включает три основных блока — базу знаний, решатель и интеллектуальный интерфейс[2]. Таким образом, рассматривая электронный билингвальный словарь открытого типа как базу знаний, его можно отнести к одному из составляющих компонентов интеллектуальных информационных систем, поскольку в данном случае высококвалифицированными специалистами, осуществляющими выявление, исследование и применение знаний, являются одновременно педагогики, учителя-предметники и лингвисты. При построении интеллектуальных систем, основанных на знаниях, используются знания, накопленные экспертами в виде конкретных правил решения тех или иных задач, следовательно, билингвальный методический словарь-справочник, созданный по принципу Википедии, является типичным банком знаний, который впоследствии может быть включен в состав более сложной экспертной системы. Являясь составной частью интеллектуальной информационной системы, такой веб-ресурс может также быть классифицирован как гипертекстовая система.

В свою очередь билингвальное образование в настоящее время становится общезначимым национальным предписанием, причем не только нравственного, но и профессионального характера. Отсюда возникает актуальность проблемы повышения квалификации преподавателя высшей профессиональной школы в системе непрерывного (lifelong) и рекуррентного, возобновляемого (recurrent) образования.

Становится очевидной необходимость разработки двуязычных электронных методических справочников с учетом отдельных специальностей методистов и педагогов, поскольку обучение на билингвальной основе способствует углублению предметной подготовки и расширению сферы межкультурного обучения, совершенствованию общей языковой подготовки и владения иностранным языком в специальных неязыковых предметных целях, а также повышению мотивации в изучении иностранного языка [3].

Одним из главных этапов в разработке любой интеллектуальной информационной системы, в том числе и гипертекстовой, является построение концептуальной модели. В рамках нашего исследования было создано целостное и системное описание используемых знаний, отражающее сущность функционирования проблемной области. В результате концептуализации проблемы проектирования электронного методического словаря открытого типа была получена объектная модель, описывающая структуру предметной области как совокупности взаимосвязанных объектов (схема 1).

Процесс проектирования электронного билингвального методического словаря в данной концептуальной модели представляет собой алгоритм действий, после выполнения которых мы достигнем поставленной цели. Первый шаг алгоритма — постановка проблемы — включает в себя определение типа проблемы, цели и задач [2]. Целью концептуальной модели является повышение уровня знаний и профессиональной языковой компетенции учителей Вузов и педагогов-исследователей.

Следующий этап — это параллельное выполнение двух шагов алгоритма: определение теоретико-методологических основ процесса проектирования, а также выявление и анализ потребностей конечных пользователей к предполагаемому продукту. Параллельное выполнение этих действий обеспечивает их взаимовлияние и взаимообусловленность. При проектировании концептуальной модели нами были взяты за основу следующие методологические подходы: системный, билингвальный, культурологический, рефлексивный, деятельностный. Системный, билингвальный и культурологический подходы позволяют выявить сущностные характеристики, закономерности и принципы интерсоциальной концепции развития межкультурной и

языковой профессиональной компетенции студентов, уровни ее функционирования, построить и обосновать концептуальную модель[4]. Рефлексивный и деятельностный подходы обеспечивают разработку методических основ развития профессиональной языковой компетенции студентов в процессе образования в неязыковом вузе. Данные подходы входят в состав теоретико-методологического компонента концептуальной модели и позволяют положить начало реализации ведущих идей данного исследования — глобализации и модернизации. Актуальность идеи модернизации связана с тем, что полингвальное обучение, являясь средством получения образования, представляет собой также и процесс формирования личности, открытой к взаимодействию с окружающим миром [5]. А идея глобализации ставит цель включения подобного ресурса в состав обучающей экспертной системы, доступной широкой аудитории. Теоретико-методологический компонент концептуальной модели также раскрывает общедидактические принципы реализации указанных подходов: принцип научности, доступности, мотивации и наглядности.

После определения теоретико-методологического компонента и анализа потребительских требований следует этап непосредственного моделирования проекта, пока еще не окончательного оформления идеи, иначе — организационно-методический компонент. Проектирование методического ресурса обуславливает наличие у данного этапа характерных особенностей: методические приемы и условия оформления проекта в некоторую оболочку. Педагогические условия включены в концептуальную модель, так как система не может быть создана и существовать в ином виде, кроме как в комплексе с указанными условиями, иначе мы получим систему с другими характеристиками [4].

Объединяя методические приемы и средства создания, можно выделить 8 этапов в создании электронного билингвального методического словаря открытого типа: 1) анализ литературы, 2) разбиение материала на модули, 3) определение гиперссылок, 4) реализация в электронной форме, 5) разработка компьютерной поддержки, 6) корректировка материала, 7) отбор мультимедийного сопровождения и 8) возможная визуализация материала [1]. Далее алгоритм нашего проектирования предусматривает проведение экспертного анализа с

последующей корректировкой в случае необходимости. И только после этого следует практическая апробация нашего методического словаря открытого типа. Практическая часть эксперимента реализуется в соответствии с разработанной учебной программой при условии активной билингвальной учебной деятельности учащихся и направляющей билингвальной педагогической деятельности учителя.

В качестве результата функционирования предложенной системы мы рассматриваем определенный уровень методической подготовки учащихся, который обеспечивает переход системы на новую, более высокую стадию развития. Оценочно-результативный компонент включает уровни методической готовности к педагогической деятельности учащихся, критерии и показатели, диагностику и методы математической статистики.

Таким образом, спроектированная нами модель обладает всеми признаками системы, а именно:

- целостностью, так как все указанные компоненты взаимосвязаны между собой, выполняют определенную функцию, способствуют достижению планируемого результата, который выражается в высоком уровне билингвальной методической готовности студентов;
- наличием инвариантных компонентов (социальный заказ, государственный образовательный стандарт, цель, теоретико-методологические основы решения проблемы) и вариативных компонентов (средства, механизмы достижения цели);
- открытостью, так как каждый компонент взаимодействует с внешней средой: данная система, с одной стороны, сама испытывает влияние среды, с другой, — оказывает на неё влияние, организуя её в соответствии с целью;
- динамичностью, так как содержание компонентов может меняться в зависимости от социального заказа и предполагает совершенствование в процессуальном плане и в оценке качественной характеристики результата — уровней;
- линейно-возвратным характером, выражающимся в обеспечении оперативной обратной связи, корректирующей недостатки полученного результата [4].

Спроектированная нами модель является концептуальной, так как учитывает основные положения интерсоциальной концепции, закономерности и принципы её построения и направлена на их реализацию.

Список литературы

[1] Зимина О. В. Печатные и электронные учебные издания в современном высшем образовании: Теория, методика, практика. — М.: МЭИ, 2003.

[2] Гаврилова Т. А. Базы знаний интеллектуальных систем. — СПб: Питер, 2000.

[3] Мафтей А. Г. Личностно-ориентированный подход в управлении поликультурной школой / автореферат дисс-и канд. пед. наук. — Смоленск: Приднестровский гос. университет, 2007.

[4] Павлова Л. В. Развитие гуманитарной культуры студентов вуза. — М.: «Академия естествознания», 2010.

[5] Салехова Л. Л. Дидактическая модель билингвального обучения математике в высшей педагогической школе / автореферат дисс-и доктора пед. наук. — Казань, 2007.

Объектная СУБД DIM и пути ее реализации

Рублев В. С., Смирнова Е. А. (Ярославль, Ярославский государственный университет им. П. Г. Демидова)

roublev@mail.ru

Требования современного развития баз данных ставят новые задачи. Последние достижения в научных исследованиях (например, в астрофизике) требуют хранения и обработки колоссальных объемов информации, которые подошли уже к пентабайтам [1]. Потребность в постоянных изменениях не только данных, но и алгоритмов является «неизменной характеристикой современного мира» [2].

В [3] описан новый объектный подход к созданию СУБД, который предполагает не только изменение данных объектов, но и возможность изменения типов объектов, то есть схемы базы данных, названный *динамической информационной моделью*[1] (DIM). В этом подходе мы выделили 6 базовых отношений объектов: *наследования, включения, внутреннего наследования, внутреннего включения, истории и взаимодействия.* Для описания предметной области введено общее определение Дискретной детерминированной модели, объекты которой могут эволюционировать, дана формализация таким моделям (OD-модели) и показана полнота DIM: *любая OD-модель и ее эволюция может быть описана с помощью DIM.* Описанный в [4] язык объектно-динамических запросов ODQL для динамической информационной модели DIM обладает полнотой — *любая группа объектов DIM (вместе с любыми их свойствами) может быть выделена ODQL-запросом* (см. [5]). В [6] описана организация выполнения запросов, при которой трудоемкость оптимальна. Но время выполнения запросов зависит как от организации хранения данных и манипулирования ими, так и от выбора платформы реализации. При этом должны быть учтены вопросы динамического изменения данных. Рассмотрим сначала пути получения такой организации, которые не зависят от выбора платформы для реализации.

Естественный путь организации динамической СУБД может быть определен метауровенем описания объектов, классов объектов, пар-

[1] В алгебре система объектов с введенными отношениями называется *моделью.*

метров и реляционных связей этих сущностей. Метауровень представляет собой реляционную базу данных с набором таблиц, содержащих в себе все нужные значения. При этом выбор в качестве платформы реляционной СУБД может быть сделан позже на основе анализа удачности выбора той или иной платформы для метауровня. В таблицах метауровня информация о классах представлена как об объектах метауровня. Поскольку параметры каждого класса могут меняться, то информация о параметрах классов также представлена в других таблицах, связанных с таблицей классов реляционными связями. Объекты каждого класса организуются в отдельных таблицах, связанных с классами таблицы классов. Подобным образом и значения параметров объектов поместим в отдельные таблицы, связанные с соответствующими таблицами параметров и объектов реляционными связями. Отметим, что каждый класс, параметр, объект имеет свой уникальный идентификатор (IdClass, IdParamter, IdObj) для установления упомянутых связей между таблицами, а также 2 атрибута *Дата рождение* и *Дата смерти*, определяющими период жизни соответствующего данного (при изменении класса, параметра, объекта их объектные идентификаторы также изменяются). На рис. 1 приведена схема организации метауровня DIM.

На этой схеме ClassInheritance, ClassInclusion, ClassInteraction, ClassHistory таблицы связей классов, соответствующих отношениям наследования, включения, взаимодействия и иерархии.Таблица Obj_IdClass объектов класса определяет для каждого объекта пометку Mark выбора объекта в *индекс-выборку класса* (см. [6]), а сами индекс-выборки классов и отношений классов реализуются как внешние индексы этой таблицы и таблиц связей объектов.

Введенный в [4] SQL-подобный язык объектных запросов ODQL выделяет из фразы **WHERE** ограничения **FOR**, накладываемые на данные того или иного класса объектов, и ограничения **LINKS**, накладываемые на связи классов в порядке обхода *схемы слоев* (см. [5]). Алгоритм выполнения запроса состоит из этапов начального определения индекс-выборок классов по фразе **FOR**, коррекции этих индекс-выборок с использованием отношений классов в порядке обратном связям во фразе **LINKS** и получения в качестве результата

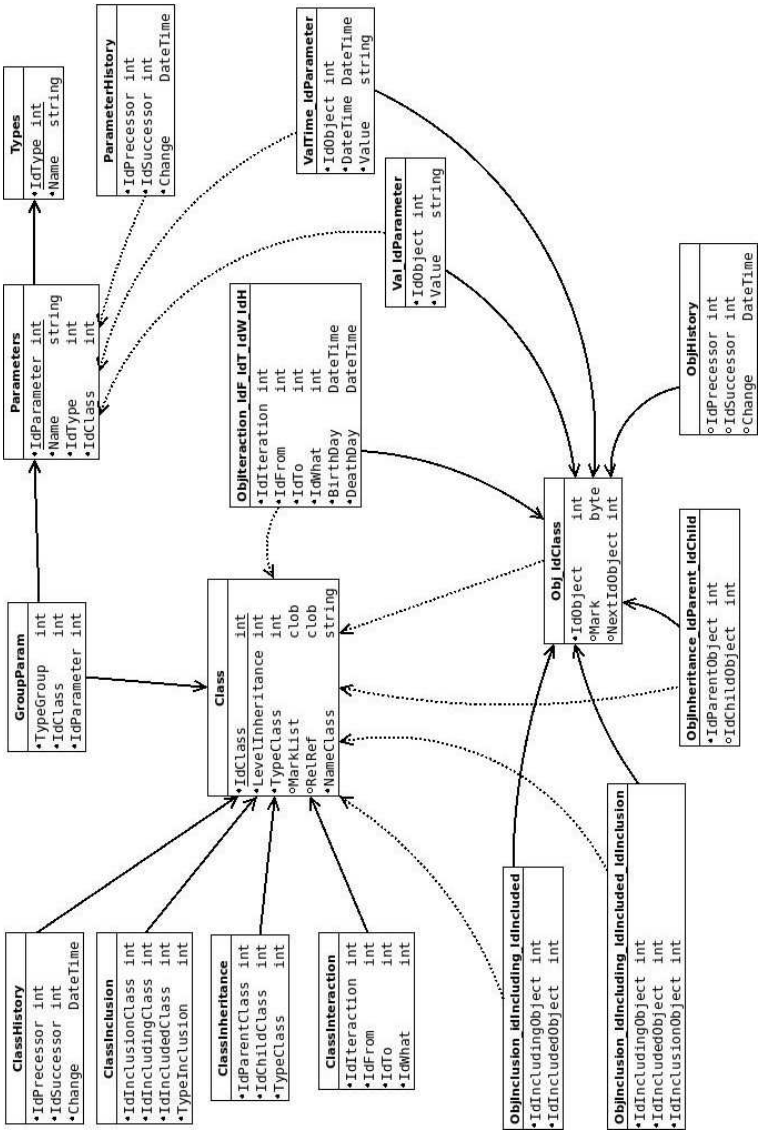


Рис. 1. Схема метауровня.

«деревьев» данных фразы **SELECT** для каждого объекта базового класса (первого во фразе **FROM**).

В качестве платформы метауровня предполагается база данных JAVA. В настоящее время проводится сравнительное исследование быстроты выполнения запросов с использованием вышеуказанного алгоритма и с помощью другого алгоритма обхода схемы слоев, а также сравнение с быстротой выполнения преобразованных ODQL-запросов в последовательности SQL-запросов. Анализ этих исследований должен выявить область использования каждого из подходов, что даст возможность перейти к лучшей реализации DIM.

Список литературы

[1] Gray J., Liu D. T., Nieto-Santisteban M., Szalay A., Dewitt D. J., Heber G. Sientific Data Management in the Coming Deacade // SIGMOD Recjrd. V. 34. No 4. December 2005.

[2] Сивцов А. Шесть компонентов успешных проектов на примере DW/BV // Корпоративные базы данных–2011. Материалы 16-й ежегодной технической конференции. — 2011. — CitForum.ru

[3] Писаренко Д. С., Рублев В. С. Объектная СУБД. Динамическая информационная модель и ее основные концепции // Моделирование и анализ информационных систем. — Т. 16. № 1. 2009. С. 62–91.

[4] Рублев В. С. Язык объектных запросов динамической информационной модели DIM // Моделирование и анализ информационных систем. — Т. 17. № 3. 2010. — С. 144–161.

[5] Рублев В. С. Запросная полнота языка ODQL динамической информационной модели DIM // Ярославский педагогический вестник. Серия «Физико-математические и естественные науки». Вып. 1. — Ярославль, 2011. — С. 69–75.

[6] Рублев В. С. Организация выполнения объектных запросов в динамической информационной модели DIM // Моделирование и анализ информационных систем. — Т. 18. № 2. Ярославль: ЯрГУ, 2011. — С. 39–51.

О кодах конечных множеств точек в евклидовых пространствах

Руденко А. Д. (Москва)

aleksei-rudenko@yandex.ru

Введение

В работах В. Н. Козлова ([1]) при построении геометрического подхода к задаче распознавания изображений вводится код специального вида.

Пусть $A \subset R^2$ — конечное множество. Перенумеруем точки из A так, чтобы их номера были попарно различны. Обозначим через M_A множество этих номеров. Пусть $S(i_0, i_1, i_2)$ — площадь треугольника с вершинами в точках с номерами i_0, i_1, i_2 из M_A.

Введем $\rho(i_0, i_1, i_2; j_0, j_1, j_2) = \frac{S(i_0,i_1,i_2)}{S(j_0,j_1,j_2)}$ ($i_0, i_1, i_2, j_0, j_1, j_2 \in M_A$, порядок номеров в тройках не важен); если $S(j_0, j_1, j_2) = 0$, считаем, что значение $\rho(i_0, i_1, i_2; j_0, j_1, j_2)$ не определено. Обозначим через T_A множество индексированных чисел $\rho(i_0, i_1, i_2; j_0, j_1, j_2)$ для всех таких пар троек. Тогда кодом A назовем пару $< M_A, T_A >$.

Назовем конечные множества $A, B \subseteq R^2$ эквивалентными, если существует такая биекция

$$\psi\colon M_A \to M_B\colon \forall i_0, i_1, i_2, j_0, j_1, j_2 \in M_A\colon$$
$$\rho(i_0, i_1, i_2; j_0, j_1, j_2) = \rho(\psi(i_0), \psi(i_1), \psi(i_2); \psi(j_0), \psi(j_1), \psi(j_2));$$

Оказываются верны следующие утверждения:

Теорема 1. *Если конечные множества $A, B \subset R^2$ аффинно эквивалентны, то они эквивалентны.*

Будем называть конечное множество точек плоскости плоским изображением, если оно не лежит целиком ни на какой паре параллельных прямых.

Теорема 2. *Плоские изображения аффинно эквивалентны тогда и только тогда, когда эквивалентны.*

Аналогичным образом вводится код конечного множества из R^3 (вместо троек точек рассматриваются четверки, вместо треуголь-

ников — тетраэдеры, вместо прямых — плоскости), и аналоги теорем 1 и 2 оказываются верны.

Логично предположить, что похожие построения можно провести и в пространстве R^n произвольной размерности с введенными стандартным образом скалярным произведением и мерой. Здесь рассматривается эта задача.

Понятие кода множества

Через $V(a_0, \ldots, a_n)$ бозначим меру n-мерного симплекса с вершинами в точках $a_0, \ldots, a_n$ из R^n. Пусть $\Phi\colon (R^n)^{2n+2} \to R \cup \{\Delta\}$,

$$\Phi(a_0, \ldots, a_n; b_0, \ldots, b_n) = \begin{cases} \frac{V(a_0,\ldots,a_n)}{V(b_0,\ldots,b_n)}, V(b_0, \ldots, b_n) \neq 0, \\ \Delta, V(b_0, \ldots, b_n) = 0; \end{cases}$$

Определение. Пусть $A, B \subset R^n$ конечны, $g\colon A \to B$ — биекция. Будем говорить, что A эквивалентно B с отображением g, если для любых точек $a'^0, \ldots, a'^n, a''^0, \ldots, a''^n$ из A выполняется

$$\Phi(a'^0, \ldots, a'^n; a''^0, \ldots, a''^n) = \Phi(g(a'^0), \ldots, g(a'^n); g(a''^0), \ldots, g(a''^n));$$

Будем говорить, что коды множеств A и B равны, если существует такая биекция g, что A эквивалентно B с отображением g.

Определение. Конечное множество в R^n, не лежащее целиком ни в какой паре параллельных гиперплоскостей, называется n-мерным изображением.

Для n-мерных изображений оказывается верна теорема 7 — аналог теоремы 2. Далее строится схема её доказательства.

Вспомогательные понятия и утверждения

Определение. Пусть $A \subset R^n$ — конечное множество, $g\colon R^n \to R^n$ — биекция. Назовем пару (A, g) неправильной, если

1) $e^i \in A, g(e^i) = e^i, i = \overline{0, n}$, здесь $e^1, \ldots, e^n$ — векторы стандартного базиса, $e^0 = 0$,
2) $\exists a \in A\colon g(a) \neq a, \sum_{i=1}^n a_i \neq 1$,
3) множества A и $g(A)$ эквивалентны с отображением g.

Далее везде в этом параграфе (A, g) — неправильная пара.

Лемма 1. *Для каждой точки a множества A,* $|a_i| = |g(a)_i|, i = \overline{1,n}$.

Таким образом, положив

$$E_g(a;-1) = \{i \in [1,n] | a_i = -g(a)_i \neq 0\},$$
$$E_g(a;0) = \{i \in [1,n] | a_i = g(a)_i = 0\},$$
$$E_g(a;1) = \{i \in [1,n] | a_i = g(a)_i \neq 0\},$$

получим

$$[1,n] = E_g(a;-1) \oplus E_g(a;0) \oplus E_g(a;1);$$

Лемма 2. *Пусть a — точка множества A,* $g(a) \neq a$. *Тогда либо при* $\alpha = -1$, *либо при* $\alpha = 1$, *множество* $E_g(a;\alpha)$ *не пусто,* $\sum_{i \in E_g(a;\alpha)} a_i = \frac{\alpha+1}{2}$.

Определение. Пусть $l \geqslant 0$ — целое число, (A, g) — неправильная пара, $a^0, \ldots, a^l$ — точки множества A, $\alpha^i \in \{-1, 1\}, i = \overline{0,l}$. Будем говорить, что $[(a^0;\alpha^0), \ldots, (a^l;\alpha^l)]$ — путь по (A, g) длины l из $(a^0;\alpha^0)$ в $(a^l;\alpha^l)$ если либо $l = 0$, либо $l > 0$ и для каждого целого i из $[1, l]$ пересечение множеств $E_g(a^{i-1};\alpha^{i-1})$ и $E_g(a^i;\alpha^i)$ не пусто.

Теорема 3. *Пусть* (A, g) — *неправильная пара. Ни для какой точки a множества A не существует пути по* (A, g) *из* $(a;\alpha)$ *в* $(a;-\alpha)$, $\alpha \in \{-1, 1\}$.

Теорема 4. *Пусть* (A, g) — *неправильная пара,* b, c *суть точки множества A, причем* $b \neq g(b), \sum_{i=1}^{n} b_i \neq 1$. *Пусть также*

$$\beta, \gamma \in \{-1, 1\}, E_g(b;\beta), E_g(c;\gamma) \neq \varnothing,$$
$$\sum_{i \in E_g(b;\beta)} b_i = \frac{\beta+1}{2}, \sum_{i \in E_g(c;\gamma)} c_i \neq \frac{\gamma+1}{2};$$

Тогда не существует пути по (A, g) *из* $(b;\beta)$ *в* $(c;\gamma)$.

Свойства отношения равенства кодов

Теорема 5. *Отношение равенства кодов конечных множеств из* R^n *является отношением эквивалентности.*

Теорема 6. *Если конечное множество* $B \subset R^n$ *переводится в конечное множество* $C \subset R^n$ *аффинным преобразованием* ϕ, B *эквивалентно* C *с отображением* ϕ.

Теорема 7. *n-мерные изображения аффинно эквивалентны тогда и только тогда, когда их коды равны.*

Приведем короткую схему доказательства этой основной теоремы. Пусть коды двух не аффинно эквивалентных конечных множеств $B, C \subset R^n$ равны. Тогда найдётся такая неправильная пара (A, g), что B аффинно эквивалентно A, C аффинно эквивалентно $g(A)$. Возьмем такие $a \in A, \alpha \in \{-1, 1\}$, что

$$a \neq g(a), \sum_{i=1}^{n} a_i \neq 1, E_g(a;\alpha) \neq \varnothing, \sum_{i \in E_g(a;\alpha)} a_i = \frac{\alpha + 1}{2};$$

Пусть E — объединение $E_g(b;\beta)$ по всем $(b;\beta)$, в которые существует путь из $(a;\alpha)$. Тогда из теорем 3, 4 следует, что A лежит в паре гиперплоскостей $\sum_{i \in E} x_i = 0$, $\sum_{i \in E} x_i = 1$.

Выражаю благодарность В. Н. Козлову за научное руководство.

Список литературы

[1] Козлов В. Н. Введение в математическую теорию зрительного восприятия. — М.: Изд-во Центра прикл. иссл. при мех.-мат. ф-те МГУ, 2007.

[2] Кудрявцев В. Б., Гасанов Э. Э., Подколзин А. С. Введение в теорию интеллектуальных систем. — М.: Изд-во ф-та ВМиК МГУ, 2006.

[3] Ильин В. А., Ким Г. Д. Линейная алгебра и аналитическая геометрия. — М.: Проспект, 2008.

Системы оценки и мониторинга сложных процессов и их приложения

Рыжов А. П. (Москва)

ryjov@mail.ru

В работе описывается технология информационного мониторинга сложных процессов, разрабатываемая автором с конца 80-х – начала 90-х годов. Приводится содержательная постановка проблемы информационного мониторинга, описываются технологические и математические аспекты разработки систем информационного мониторинга.

Многие процессы в бизнесе, экономике, политике и других областях, называемых слабо (или плохо) формализуемыми, не возможно представить в виде набора уравнений, автоматов и других математических средств представления и анализа динамических систем, однако специалисты как-то решают задачи оценки состояния процесса и управления им. В общем виде задача заключается в оценке текущего состояния процесса на основе всей доступной информации, построении прогнозов его развития и выработке рекомендаций по управлению исходя из целей, стоящих перед специалистом. В работе приводятся примеры таких задач из политологии, маркетинга, финансового анализа, страхового дела. В качестве примера процессов, не являющихся таковыми, можно привести взаимодействие двух тел или распространение колебаний в однородной среде. Имеются математические модели таких процессов, информация измерима и доступна, результат можно вычислить для любого момента времени.

Будем называть задачу оценки текущего состояния системы (процесса) и построении прогнозов ее развития задачей информационного мониторинга, а человеко-компьютерные системы, обеспечивающие аналитическую поддержку подобного рода информационных задач, системами информационного мониторинга. Основными элементами систем информационного мониторинга являются информационное пространство и аналитик. В работе анализируются свойства информационного пространства: разнородность, фрагментарность, разноуровневость и ненадежность доступной информации, ее противоречивость и изменяемость во времени.

Приводятся архитектурные и технологические особенности компьютерных систем, обеспечивающие обработку такого рода информации [3]. В частности:

- для реализации возможности обработки информации из разнородных источников, в базе данных системы хранятся как сами документы, так и ссылки на них с оценкой содержащейся в них информации, данной экспертом;
- для возможности обработки фрагментарной информации используется модель процесса в виде графа;
- обработка разноуровневой информации достигается за счет предоставления пользователю возможности отнести оценку конкретного информационного материала к разным вершинам модели;
- обработка информации различной степени надежности и обладающей возможной противоречивостью или тенденциозностью достигается за счет использования лингвистических оценок экспертами данной информации;
- изменяемость во времени учитывается фиксацией даты поступления информации при оценке конкретного материала, то есть время является одним из элементов описания объектов системы.

Для эффективного практического применения предложенных технологических решений необходима проработка ряда теоретических проблем, результаты которой приводятся в докладе. Рассматриваются три такие проблемы.

Проблема 1. Можно ли, учитывая некоторые особенности восприятия человеком объектов реального мира и их описания, сформулировать правило выбора оптимального множества значений признаков, по которым описываются эти объекты? Возможны два критерия оптимальности:

Критерий 1. Под оптимальными понимаются такие множества значений, используя которые человек испытывает минимальную неопределенность при описании объектов.

Критерий 2. Если объект описывается некоторым количеством экспертов, то под оптимальными понимаются такие множества значений, которые обеспечивают минимальную степень рассогласования описаний.

Показано [6], что мы можем сформулировать методику выбора оптимального множества значений качественных признаков. Более того, показано, что такая методика является устойчивой, то есть возможные при построении функций принадлежности естественные маленькие ошибки не оказывают существенного влияния на выбор оптимального множества значений. Множества, оптимальные по критериям 1 и 2 совпадают.

Проблема 2. Можно ли определить показатели качества поиска информации в нечетких (лингвистических) базах данных и сформулировать правило выбора такого множества лингвистических значений, использование которого обеспечивало бы максимльные показатели качества поиска информации?

Показано [4], что можно ввести показатели качества поиска информации в нечетких (лингвистических) базах данных и формализовать их. Показано, что возможно сформулировать методику выбора оптимального множества значений качественных признаков, которое обеспечивает максимальные показатели качества поиска информации. Более того, показано, что такая методика является устойчивой, то есть возможные при построении функций принадлежности естественные маленькие ошибки не оказывают существенного влияния на выбор оптимального множества значений.

Проблема 3. Можно ли предложить процедуры выбора операторов агрегирования информации в нечетких иерархических динамических системах, минимизирующих противоречивость модели проблемы/процесса в системах информационного мониторинга?

Можно выделить следующие подходы к решению этой проблемы, базирующиеся на различных интерпретациях операторов агрегирования информации [5]: геометрический, логический и подход на основе обучения, включающий вы себя обучение на основе генетических алгоритмов и обучение на основе нейронных сетей.

В докладе описываются разработанные на базе описанной технологии системы оценки и мониторинга процессов нераспространения

ядерных технологий и материалов [7], системы оценки и мониторинга рисков атеросклеротических заболеваний [1], системы оценки и мониторинга проектов в микроэлектронике [2].

Список литературы

[1] Ахмеджанов Н. М., Жукоцкий А. В., Кудрявцев В. Б., Оганов Р. Г., Расторгуев В. В., Рыжов А. П., Строгалов А. С. Информационный мониторинг в задаче прогнозирования риска развития сердечно-сосудистых заболеваний // Интеллектуальные системы. — Т. 7. Вып. 1–4. 2003. — С. 5–38.

[2] Лебедев А. А., Рыжов А. П. Оценка и мониторинг проектов разработки высокотехнологических изделий микроэлектроники // Известия ТРТУ. Тематический выпуск, ISBN 5-8327-0249-2. — № 8. 2006. — С. 93–99.

[3] Рыжов А. П. Информационный мониторинг сложных процессов: технологические и математические основы // Интеллектуальные системы. — Т. 11. Вып. 1–4. 2008. — С. 101–136.

[4] Рыжов А. П. Модели поиска информации в нечеткой среде. — М.: Изд-во Центра прикладных исследований при мех.-мат. ф-те МГУ, 2004.

[5] Рыжов А. П. Разработка методов агрегирования информации в нечетких иерархических системах с использованием методов мягких вычислений // Интеллектуальные системы. — Т. 6. Вып. 1–4. 2001. — С. 341–364.

[6] Рыжов А. П. Элементы теории нечетких множеств и измерения нечеткости. — М.: Диалог-МГУ, 1998.

[7] Fattah A., Pouchkarev V., Belenki A., Ryjov A., Zadeh L. A. Application of Granularity Computing to Confirm Compliance with Non-Proliferation Treaty // Data Mining, Rough Sets and Granular Computing / Ed. by Tsau Young Lin, Yiyu Y. Yao, L. A. Zadeh. Physica-Verlag Heidelberg, 2002. P. 308–338.

Эффективные реализации строковых словарей для решения задач компьютерной лингвистики

Скатов Д. С. (Нижний Новгород, Нижегородский государственный университет им. Н. И. Лобачевского)

ds@dictum.ru

Введение

В настоящем исследовании нас будут интересовать реализации словарей для хранения строк и ассоциированных с ними значений. Рассматривается 5 библиотек на C/C++ из открытых репозиториев, две авторских реализации на основе trie-дерева и хэширования, а также эскизная реализация алгоритмов из [6].

Для отобранных реализаций выполнены эксперименты по индексации $3 \cdot 10^6$ запросов к ПС Яндекс. Результаты позволяют решить инженерную задачу: (1) выбрать одну из существующих реализаций, либо (2) принять целесообразность создания собственной. Приведена оценка результатов и перспективы дальнейшего развития представленных авторских наработок.

1. Свойства словарей

Дан алфавит A и $K \subseteq A^+$, $|K| < \infty$, $\forall w \in K \; \exists val\,(w) \in \mathbb{Z}$. На символах A задано отношение непосредственного следования $succ$ (a, b), $a, b \in A$, индуцирующее лексикографический порядок на A^+: $v \prec w$, $v, w \in A^+$.

Интересующие нас функции словаря:

1) По $x \in A^+$ узнать: $x \in K$ и $val\,(x)$, если $true$ (поиск по ключу);
2) По $pr = a_1 a_2 \ldots a_{|pr|}$ найти все $y = pr \cdot a_{|pr|+1} \ldots a_{|y|} \in K$, $a_j \in A$, $j = \overline{1, |y|}$ (префиксный поиск);
3) По такому же pr, для которого есть аналогичный $y \in K$, найти ближайший справа $pr' \succ pr$, так что $\exists\, y' = pr' \cdot a_{|pr'|+1} \ldots a_{|y'|} \in K$ (итерируемость).

В (1) ключ можно интерпретировать как цепочку байтов, что допускает любую известную реализацию (хэш-таблицы, бинарные деревья и пр.). Функция (2) реализуема на словаре, допускающем выборку диапазона. Действительно, для $pr = a_1 \dots a_n$ можно взять правого соседа $pr' \succ pr$ (для $A = \{a, b, c\}$, $pr = abc \Rightarrow pr' = abaa$) и извлечь диапазон $[pr, pr')$. Реализация (3) достижима представлением словаря графом переходов. Он может быть затем расширен до конечного автомата (КА), размечающего неиндексированный текст вхождением ключей (напр., по схеме Ахо-Корасик).

2. Исследуемые реализации

Trie-дерево. Из 5 реализаций были отобраны [1] (A) и [2] (B) как обнаружившие меньше сбоев. К сравнению была добавлена авторская реализация (C) trie, применяемая в промышленных приложениях (напр. [5]), со следующими свойствами: хранение набора значений по каждому ключу, компактная сериализация, десериализация прямым отображением файла в память. Все реализации итерируемы.

Judy-массив. В 2004-м году была предложена trie-подобная реализация словаря [3] с операцией (2), оптимизированная технически: использованы десятки приёмов сжатия для уменьшения числа кэш-промахов, но результирующие алгоритмы обработки весьма сложны. Для тестирования была выбрана не [3], а более простая для повторного использования и несериализуемая [4] (D).

DAWG-граф. DAWG (E) является сжатой формой trie-дерева. «Прямые» алгоритмы его построения, основанные на минимизации trie как КА, трудно применимы из-за расхода памяти. Улучшение заключается в онлайн-минимизации, но требует, чтобы последовательность входных терминов была отсортирована. Современные результаты дают более сложный алгоритм, но с допуском произвольного порядка. Известная реализация [8] трудно используема повторно и даёт сбои при количестве ключей, большем $2 \cdot 10^6$. Для экспериментов была построена эскизная реализация алгоритмов из [6].

Исправление опечаток состоит в том, чтобы по заданному $\tilde{w} \in A^+$ извлечь из словаря правильных слов K варианты его исправления: $Corr_\varepsilon(\tilde{w}) = \{w \in K : \rho(w, \tilde{w}) < \varepsilon\}$. Операциям (вставка, заме-

на, транспозиция) назначаются веса, а расстояние $\rho(w, \tilde{w})$ вычислимо алгоритмом Левенштейна-Дамерау. Его сложность $O\left(|w|^2\right)$ можно улучшить методом ветвей и границ на словаре для K с функцией (3). Большее улучшение, до $O(|w|)$, достижимо, если при фиксированном ε на основании весовых матриц получить объединение шаров

$$Corr_{\varepsilon}^{-1}(K) = \cup_{w \in K} \{\tilde{w} | \rho(\tilde{w}, w) < \varepsilon\}$$

с дальнейшим сохранением в словарь. От него требуется только (2), но из-за огромного объёма данных хранение рационально в DAWG. Алгоритм построения DAWG не должен зависеть от порядка терминов (в силу сложности генерации шаров).

Хэширование. Предлагается гибридная реализация (F) хэш-словаря с возможностью (2). Он представлен матрицей из $\max\{|w|\} \times |A|$ слотов, в каждом хранятся записи для ключей и префиксов. Выбор слота определяется последней буквой термина. Используется конкатенация значений хэш-функций из [7]. Другие, известные, реализации не рассматривались в силу отсутствия (2).

SQLite DB, `std::map`**.** В классе `std::map` (G) на основе красно-чёрного бинарного дерева функция (2) реализуема `lower_bound(pr)` и `upper_bound(pr')`. `std::hash_map` не тестировалась, но, по опыту автора, в среднем она слабее trie и хэш-схем. В SQLite (H) выбор диапазона осуществляется по первичному ключу, с `synchronous=OFF` и `:memory:` (сохранение базы в память вместо файла).

3. Эксперимент и результаты

(A) и (B) из открытых репозиториев одинаковым образом не справились с задачей, начиная с числа запросов в словаре, превышающего $3 \cdot 10^5$.

Авторская (C) и judy (D) одинаково производительны, причём (D) не сериализуемо, а (C) требует больше памяти для построения. Хэш-схема (F), несмотря на тривиальную реализацию слотов на `std::map`, показывает лучшую в сравнении с ним производительность. Дальнейшие улучшения могут быть достигнуты применением техник из (D) в (C) и оптимизацией хранения в (F).

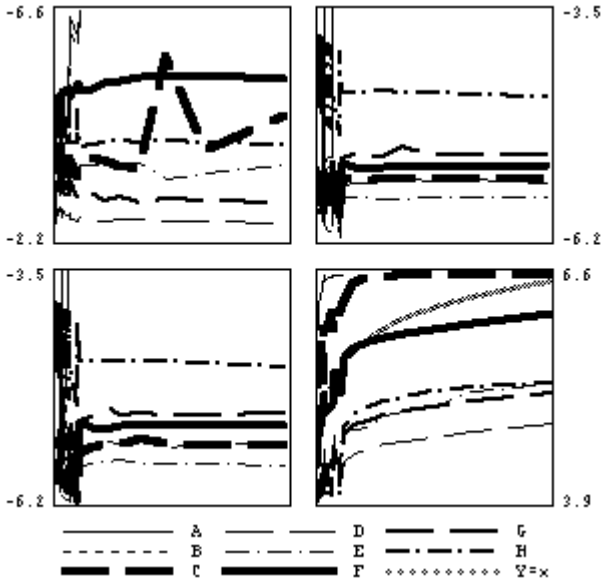


Рис. 1. Замеры (Intel Core Quad Q8200, Visual C++ 2010 x64) в шкале $\log_{10} x$. Даны оценки среднего времени для операций с одним запросом при уже имеющихся в словаре (в количестве от 1000 до 3 млн.): верхний левый (ВЛ) — добавление, ВП — успешный поиск, НЛ — неуспешный. НП — макс. объём памяти в Мб в сеансе добавления.

(H) (несмотря на `:memory:`) не является производительной реализацией, однако (G) может стать хорошим вариантом на начальных этапах — несмотря на доступность, его нельзя рассматривать в качестве худшей границы для оценок производительности. Лидером стала эскизная реализация DAWG, построенная на базе [6]. Предполагается дальнейшее улучшение этого кода с использованием приёмов из (C) и (D) и реализация алгоритма, не зависящего от порядка входных терминов.

Список литературы

[1] `http://opensource.jdkoftinoff.com/jdks`.

[2] `http://c-algorithms.sourceforge.net`.

[3] http://judy.sourceforge.net.

[4] judyarray.googlecode.com.

[5] Скатов Д. С., Ливерко С. В., Вдовина Н. А., Окатьев В. В. Язык описания правил в системе лексического анализа ЕЯ текстов DictaScope Tokenizer // Труды Международной конференции Диалог'2010. — М.: Наука, 2010.

[6] Carrasco R. C., Forcada M. L. Incremental construction and maintenance of minimal finite-state automata // Computational Linguistics. 28 (2). 2002. P. 207–216.

[7] http://www.burtleburtle.net/bob/hash.

[8] www.pg.gda.pl/~jandac/fsa.html.

Критерий сводимости задачи о предотвращении столкновений к задаче о прокалывании

Снегова Е. А. (Москва, МГУ им. М. В. Ломоносова)

lenasnegova@gmail.com

В работе исследуется задача о поиске движущихся объектов, которые могут столкнуться с движущимся объектом-запросом, где под столкновением понимается нахождение объектов в опасной близости.

Опишем задачу о предотвращении столкновений, имеющую несколько параметров:

1) параметр $\rho \in (0, \frac{1}{2})$, обозначающий *расстояние опасной близости* по Манхэттену,
2) аналитическая на всей области определения строго возрастающая функция $f : \mathbb{R}^+ \to [-\rho, 1+\rho]$, такая что $f(0) = -\rho$, называемая *законом движения объектов*.
3) аналитическая на всей области определения строго возрастающая функция $f_q : \mathbb{R}^+ \to [-\rho, 1+\rho]$, такая что $f_q(0) = -\rho$, называемая *законом движения запроса*,
4) множество объектов $V(t)$, находящихся в момент времени t в области $[-\rho, 1+\rho] \times [0, 1]$.

Четверку $(f, f_q, \rho, V(t))$ назовем *задачей о предотвращении столкновений*.

Предполагаем, что объект $o_i = (t_i, y_i)$ появляется на границе прямоугольника $[-\rho, 1+\rho] \times [0, 1]$ в момент времени t_i в точке с координатами $(-\rho, y_i)$ и движется по закону движения f параллельно оси x, то есть в момент $t \in [t_i, t_i + \tau_{max}]$ объект o_i находится внутри прямоугольника $[-\rho, 1+\rho] \times [0, 1]$ в точке с координатами $(f(t - t_i), y_i)$.

Запросом q назовем пару (t_q, x), где $t_q \in \mathbb{R}$, а $x \in [0, 1]$.

Предполагаем, что запрос $q = (t_q, x)$ появляются на границе прямоугольника $[0, 1] \times [-\rho, 1+\rho]$ в момент времени t_q в точке с координатами $(x, -\rho)$ и движется по закону движения f_q параллельно оси y, то есть в момент $t \in [t_q, t_q + \tau^q_{max}]$ запрос q находится внутри прямоугольника $[0, 1] \times [-\rho, 1+\rho]$ в точке с координатами $(x, f_q(t - t_q))$.

Скажем, что объект $o_i = (t_i, y_i)$ и запрос $q = (t_q, x)$ *сталкиваются*, если существует момент времени $t \in \mathbb{R}$, такой что

$$|f(t - t_i) - x| + |y_i - f_q(t - t_q)| \leqslant \rho.$$

В задаче требуется для произвольного запроса, поступившего в момент времени t, перечислить все объекты из библиотеки $V(t)$, с которыми он столкнется в процессе своего движения.

Основной характеристикой алгоритма решения этой задачи является сложность поиска, измеряемая в операциях вычисления значений некоторых функций, принятых за элементарные. Поскольку библиотека динамически меняется со временем, то важными характеристиками являются также сложности вставки и удаления объектов в БД. Еще одной характеристикой является объем памяти, требуемый алгоритму для хранения структур данных.

Одним из способов эффективно решить задачу о предотвращении столкновений может быть сведение данной задачи к уже известным одномерным задачам. В этой работе в качестве такой задачи рассматриваются задача о прокалывании.

Задача о прокалывании имеет эффективное статическое решение — логарифмическое время поиска и линейный объем памяти [1], а в динамическом случае [2] оптимальное решение имеет сложность поиска порядка $\sqrt{n}$, сложность вставки/удаления порядка $\log^2 n$, и использует память порядка n, где n есть число объектов в области наблюдения.

В [3] рассматривался случай фиксированных скоростей объектов, то есть $f(t) = vt$, а $f_q(t) = v_q t$, в [4] рассматривался случай, когда $f'(t + t') - f'_q(t) \leqslant 0$ для любого $t \in [0, \tau^q_{max}]$ и любого t', такого, что $t + t' \in [0, \tau_{max}]$. В обоих случаях задача решалась путем сведения задачи о предотвращении к задаче одномерного интервального поиска, которая имеет логарифмическую относительного общего числа объектов в библиотеке сложность поиска вставки и удаления, и линейный объем памяти. В [5], [6] приводились критерии сводимости задачи о предотвращении столкновений к задаче одномерного интервального поиска и к задаче о прокалывании, соответственно, для случая произвольных (не обязательно аналитических) законов движения объек-

тов и запросов. В [7] приводился критерий сводимости к одномерным задачам для случая, когда законы движения — многочлены.

Задачей о прокалывании назовем пару $(\mathbb{R}, Z)$, где библиотека Z есть конечное множество всех интервалов с концами из $\mathbb{R}$, а $\mathbb{R}$ есть множество всех действительных чисел. Содержательно эта задача состоит в том, чтобы для произвольного запроса $p \in \mathbb{R}$ перечислить все те и только те отрезки из Z, которые содержат p.

Ответ на запрос $p \in \mathbb{R}$ при библиотеке Z в задаче о прокалывании есть множество $J(p, Z) = \{z \in Z : p \in z\}$.

Будем говорить, что задача о предотвращении столкновений $(f, f_q, \rho, V(t))$ *сводится* с задаче о прокалывании, если существуют такие отображения $\varphi, \varphi_1, \varphi_2 : \mathbb{R} \times [0,1] \to \mathbb{R}$, что для любой библиотеки $V(t)$, любого запроса $q = (t_q, x)$ и любого объекта $o \in V$ верно

$$o \in J(f, f_q, \rho, V, q) \Leftrightarrow [\varphi_1(o), \varphi_2(o)] \in J(\varphi(q), Z),$$

где $Z = \{[\varphi_1(o), \varphi_2(o)] : o \in V\}$.

Теорема 1. *Задача о предотвращении столкновений* $(f, f_q, \rho, V(t))$ *сводится к задаче о прокалывании тогда и только тогда, когда выполнено хотя бы одно из следующих двух условий*

Условие 1.

1) *Если* $(x, y) \in [\rho, 1+\rho] \times [0,1] : f_q^{-1}(y) - f^{-1}(x) \leqslant 0$, *то* $(f_q^{-1}(y))' \leqslant (f^{-1}(x))'$;
2) *Если* $F_L(x, 0) \leqslant 0$, *то* $\rho \in \arg\min\limits_{\xi \in [0,\rho]} F_L(x, 0, \xi)$;
3) *Если* $F_R(x, 1) \leqslant 0$, *то* $-\rho \in \arg\max\limits_{\xi \in [-\rho,0]} F_L(x, 1, \xi)$;
4) *Если* $(x, y) \in [0, 2\rho] \times [0,1] : F_R(x, y) \leqslant 0$, *то* $-\rho \in \arg\max\limits_{\xi \in [-\rho,0]} F_R(x, y, \xi)$;
5) *Если* $(x, y) \in [0, \rho] \times [0,1] : F_L(x, y) \leqslant 0$, *то* $\rho \in \arg\min\limits_{\xi \in [0,\rho]} F_L(x, y, \xi)$;

Условие 2. f — *линейный многочлен.*

Автор благодарит своего научного руководителя профессора Гасанова Э. Э. за постановку задачи и помощь в работе.

Список литературы

[1] Arge L., de Berg M., Haverkort H. J., Yi K. The Priority R-Tree: A Practically Efficient and Worst-Case-Optimal R-Tree // Symp. of the ACM Special Interest Group on Management of Data (SIGMOD). — Paris, 2004. P. 347–358.

[2] Arge L., Vitter J. S. Optimal dynamic interval management in external memory // Foundations of Computer Science. — 1996.

[3] Скиба Е. А. Логарифмическое решение задачи об опасной близости // Интеллектуальные системы. — 2007. Т. 11. Вып. 1–4. — С. 693–719.

[4] Снегова Е. А. Случай задачи об опасной близости, сводящийся одномерному интервальному поиску // Интеллектуальные системы. — Т. 13. Вып. 1–4. 2009. — С. 97–118.

[5] Снегова Е. А. Критерий сводимости задачи об опасной близости одномерному интервальному поиску // Дискретная математика. В печати.

[6] Снегова Е. А. Критерий сводимости задачи об опасной близости к одномерной задаче о прокалывании // Интеллектуальные системы. — Т. 15. Вып. 1–4. 2011. — С. 241–264.

[7] Снегова Е. А. Критерий сводимости задачи об опасной близости к одномерным задачам для полиномиальных законов движения // Интеллектуальные системы. — Т. 16. Вып. 1–4. 2012. — С. 181–201.

Анализ формы изображений, заданных с погрешностью

Чуличков А. И., Демин Д. С., Копит Т. А., Цыбульская Н. Д.
(Москва, МГУ им. М. В. Ломоносова, физический факультет)
achulichkov@gmail.com

Форма изображения

Под изображением понимается числовая функция $f(\cdot)$, заданная на поле зрения $X \subset R^2$, множество всех изображений образует полное линейное нормированное пространство $\mathcal{R}$. Значение $f(x)$ называется яркостью изображения $f(\cdot)$ в точке x поля зрения.

Изображения одного и того же объекта или сцены, полученные при различных условиях, могут различаться радикально. Заключенная в них «часть» информации, не меняющаяся при изменении условий их регистрации, называется формой изображения [1]. Форма может быть построена, если задана операция сравнения по форме: определим класс $\mathcal{F}$ функций, преобразующих яркость изображения при изменении условий его регистрации: пусть $f \in \mathcal{R}$ — изображение сцены, тогда для любого $F \in \mathcal{F}$ изображение $F * f \in \mathcal{R}$: $F * f(x) = F(f(x))$, $x \in X$, есть изображение той же сцены, полученное при некоторых условиях наблюдения. Примерами класса $\mathcal{F}$ могут быть все (измеримые) функции, монотонные функции, полиномы степени не выше заданной и т. п.

Определение. Изображение $g \in \mathcal{R}$ не сложнее по форме изображения $f \in \mathcal{R}$, если существует функция $F \in \mathcal{F}$, такая, что $g = F * f$.

Определение. Множество $V_f \subset \mathcal{R}$ всех изображений, которые не сложнее по форме, чем изображение f, называется формой изображения f.

Если V_f выпукло и замкнуто, то задача наилучшего приближения

$$\|\xi - \xi_f\| = \inf\{\|\xi - g\| \,|\, g \in V_f\}$$

разрешима для любого $\xi \in \mathcal{R}$, а если ее решение единственно, то оно определяет проекцию ξ на V_f, $\xi_f = P_f\xi$, оператор проецирования P_f

в этом случае связан с V_f взаимно однозначно и также называется формой изображения f.

Задачи, решаемые методами морфологического анализа изображений

Пусть проектор $P_f : \mathcal{R} \to \mathcal{R}$ существует. В терминах P_f решается ряд важных задач анализа изображений, в частности — следующие [1].

- Пусть требуется определить, сравнимо ли предъявленное изображение ξ по форме с f. Ответ на этот вопрос положителен, если и только если $\xi = P_f\xi$.
- Пусть имеется набор форм $V_1, \ldots, V_M$, и требуется определить, к какой форме из заданных наиболее близко (по форме) предъявленное изображение ξ. Ответ: к форме с номером k_*, где $\|\xi - P_{k_*}\xi\| \leqslant \|\xi - P_{k_*}\xi\|$, $k = 1, \ldots, M$. Если таких номеров несколько, то ответ неоднозначен.
- Пусть требуется указать, чем отличается предъявленное изображение ξ от f по форме. Ответ дает изображение $\xi - P_f\xi$.

Для решения этих задач априори требуется задать форму как множество V (или проектор P).

Однако на практике часто информация о форме содержится в изображении сцены, заданном с погрешностью или в наборе изображений одной и той же сцены, полученных при различных и неконтролируемых условиях наблюдения и искаженных шумом. В работе рассматриваются морфологические методы, основанные на форме, оцененной из неточно заданных изображений сцены.

Сравнение по форме изображений, заданных с погрешностью

Пусть заданы изображения $\xi, \eta \in \mathcal{R}$, полученные по схеме

$$\xi = f + \nu, \quad \eta = g + \mu,$$

где неискаженные шумом ν и μ соответственно изображения f и g ненаблюдаемы. Требуется по заданным ξ и η определить, сравнимо ли g по форме с f, то есть найдется ли такая функция $F \in \mathbf{F}$, что $g = F * f$, если $\|\nu\| \leqslant \varepsilon$, $\|\mu\| \leqslant \varepsilon$.

Пусть $\mathcal{F}_m$ — класс монотонных преобразований $F(\cdot)$, то есть таких, что из $x \leqslant y$ следует $F(x) \leqslant F(y)$, и изображения заданы в конечном числе точек своими значениями $(f_1, \ldots, f_N)$, так, что пространство $\mathcal{R}$ является N-мерным линейным нормированным пространством с нормой $\|f\| = \max\limits_{i=1,\ldots,N}\{|f_i|\}$. Тогда задача сравнения изображений ξ и η по форме сводится к проверке неравенства

$$\inf_{j(\cdot)} \inf_{f_1 \leqslant f_2 \leqslant \ldots \leqslant f_N; g_1 \leqslant g_2 \leqslant \ldots \leqslant g_N} \{Q_j(f, g)\} \leqslant \varepsilon,$$

где

$$Q_j(f, g) = \max_{i=1,\ldots,N} \left\{ \max \left(\left| f_i - \xi_{j(i)} \right|, \left| g_i - \eta_{j(i)} \right| \right) \right\},$$

а $j(\cdot)$ — биекция множества $\{1, 2, \ldots, N\}$ на себя. Метод и алгоритм решения этой задачи приведены в работах [1–3].

Оценка формы, заданной упорядочением яркости точек поля зрения

В евклидовом пространстве всех изображений $\mathcal{R}_N = \{f = (f_1, f_2, \ldots, f_N) : f_1, f_2, \ldots, f_N \in (-\infty, \infty)\}$ замыкание класса изображений $V_0 = \{f : f_1 < f_2 < \ldots < f_N\}$ является замкнутым выпуклым конусом и играет важную роль в приложениях.

Пусть $g = (g_1, \ldots, g_N) \in \mathcal{R}_N$ есть изображение некоторой сцены, и $j(\cdot)$ — биекция множества $\{1, 2, \ldots, N\}$ на себя, такая, что $(g_{j(1)}, g_{j(2)}, \ldots, g_N) \in V_0$, однако упорядочение $j(\cdot)$ неизвестно, и его требуется установить по наблюдению последовательности изображений $\{\xi_j = F_j * f + \nu_j, j = 1, 2, \ldots\}$, в которой $F_j(\cdot)$, $j = 1, 2, \ldots$ — монотонно возрастающие функции, а $\nu_j \in \mathcal{R}_N$, $j = 1, 2, \ldots$, независимы в совокупности, обладают независимыми в совокупности координатами, причем $\|\nu\| \leqslant \delta$ с вероятностью 1.

При указанных условиях справедлива следующая теорема [4].

Теорема 1. *Упорядоченность координат вектора $g \in R_N$ определяется с вероятностью единица по конечному числу наблюдений $\xi_1, \ldots, \xi_n \in R_N$.*

В этой же работе [4] предложен алгоритм упорядочения координат и получены оценки вероятности ошибочного результата упорядочения. Эти результаты получены по аналогии с упорядочением значений возможностей, предложенных Ю.П.Пытьевым в работе [5].

Форма как линейное подпространство

Другим важным примером формы изображения является конечномерное линейное подпространство евклидова пространства $\mathcal{R}_N$ всех изображенний. Это подпространство формально может быть задано как пространство $\mathcal{R}(A)$ значений линейного оператора $A \in \mathcal{R}_n \to \mathcal{R}_N$, восстановленного эмпирически из наблюдений изображений $\xi_1, \ldots, \xi_m \in /R_N$ сцены в условиях регистрации, заданных вектором параметров $g \in \mathcal{R}_n$, принимающим в каждой тестовой ситуации с номером j известное значение g_j, $j = 1, \ldots, m$, по схеме

$$\xi_j = Ag_j + \nu_j, \quad , j = 1, \ldots, m,$$

где ν_j — погрешность регистрации изображения в ситуации с номером j, $j = 1, \ldots, m$. Методы оценки пространства значений линейных операторов по данным тестовых измерений, выполненных с погрешностью, описаны в работах [6,7].

Работа выполнена при поддержке гранта РФФИ 11–07–00338.

Список литературы

[1] Пытьев Ю. П., Чуличков А. И. Методы морфологического анализа изображений. — М.: ФИЗМАТЛИТ, 2010.

[2] Демин Д. С., Чуличков А. И. Сравнение формы нескольких сигналов, порожденных нелинейным монотонным преобразованием из неизвестного прообраза, и оценивание параметров их формы // Интеллектуализация обработки информации. Сб. докладов. — М.: МАКС Пресс, 2010. — С. 407–409.

[3] Куличков С. Н., Чуличков А. И., Демин Д. С. Морфологический анализ инфразвуковых сигналов в акустике. — М.: Новый Акрополь, 2010.

[4] Цыбульская Н. Д., Чуличков А. И. Эмпирическое упорядочение яркости пикселей изображения, задающее его форму // Математические методы распознавания образов: 15-я Всероссийская конференция. Сб. докладов. — М.: МАКС Пресс, 2011. — С. 444–447.

[5] Пытьев Ю. П. Математические методы и алгоритмы эмпирического восстановления стохастических и нечетких моделей // Интеллектуальные системы. — Т. 11. Вып. 1–4. 2007. — С. 277–327.

[6] Чуличков А. И., Цыбульская Н. Д., Шахбазов С. Ю. Классификация сигналов по форме, модель которой определена эмпирически // Вестник Московского университета. Сер. 3. Физика. Астрономия. — № 5. 2010. — С. 9–13.

[7] Копит Т. А., Чуличков А. И., Устинин Д. М. Интерпретация экспериментальных данных на основе кусочно-линейной аппроксимации модели измерений // Вестник Московского университета. Сер. 3. Физика. Астрономия. — № 5. 2010. — С. 3–8.

Разработка модели реализации предметно-ориентированной интерактивной сети

Ширяева И. А. (Смоленск, Смоленский государственный университет)

ir.shiryaeva@gmail.com

Анализ современного состояния организации образовательного процесса, изучение дистанционных образовательных технологий, понятия «интерактивность», а также выявление основных аспектов построения интерактивной образовательной сети позволяет сформулировать представление о модели организации обучения на базе предметно-ориентированной интерактивной сети образовательного учреждения.

Разработанная модель организации обучения на базе предметно-ориентированной интерактивной сети в качестве методологической основы имеет системный подход, рассматривающий объект изучения как целостное образование, учитывающее и приводящее в скоординированное действие все факторы и условия, существенно влияющие на него [2].

Предметно-ориентированная интерактивная сеть должна подчиняться действию следующих принципов: доступности, интерактивности, вариативности, системности и последовательности, целесообразности применения новых информационных технологий, сознательности и активности, наглядности.

Рассматриваемая модель должна отвечать требованиям нормативности, преемственности, целостности, связанности компонентов модели, адаптивности.

В процессе изучения дистанционного обучения выявлены функции предметно-ориентированной интерактивной сети, среди которых: создание единого информационного образовательного пространства ОУ, внесение инновационных изменений в образовательный процесс, повышение уровня интерактивного взаимодействия субъектов образовательного процесса, адаптивность к индивидуальным потребностям субъектов образовательного процесса и т. д.

Разработанная модель реализации предметно-ориентированной интерактивной сети (рис. 1) включает в себя технический компонент и компонент реализации.

Рис. 1 Модель реализации предметно-ориентированной интерактивной сети

Рис. 1.

В логике построения предметно-ориентированной интерактивной сети технический компонент модели заключается в выборе программной оболочки, технологии дистанционного обучения, создании базы электронных образовательных ресурсов.

Компонент реализации состоит из следующих блоков: выделены нормативный, потребностно-мотивационный, содержательный, технологический и контролирующий.

Нормативный блок включает в себя правовые, теоретические и методологические основания обучения учащихся на базе предметно-ориентированной интерактивной сети, гарантирует обучение в соответствии с требованиями ГОС, а также охватывает специфические требования, предъявляемые к выпускнику образовательного учреждения как современной личности, способной жить и развиваться в информационном обществе.

Потребностно-мотивационный блок модели является основным условием, побуждающим к внедрению и применению в деятельности образовательного учреждения предметно-ориентированной интерактивной сети, содержит следующие определяющие критерии: анализ существующей ситуации в образовательном учреждении, непосредственно внедрение, а также условия стимулирования деятельности пользователей.

Следующим важным критерием является непосредственно внедрение предметно-ориентированной интерактивной сети на базе определенной ранее платформы дистанционного обучения, заключающееся в непосредственно инсталляции программного продукта, его интеграции с приложениями внешних разработчиков уже используемыми в образовательном учреждении и заполнении базы данных всей необходимой информацией.

Осознание необходимости внесения изменений в образовательный процесс недостаточно для качественного использования интерактивной сети. Отсюда следует критерий стимулирования деятельности, относящийся к преподавательскому составу (учет учебной нагрузки преподавателя, организация консультаций, учет времени проведенного преподавателем в интерактивной сети и т. д.), учащимся (создание ситуаций успеха, поощрение и т. д.), родителям (создание ситуаций успеха, организация консультаций) [1].

Содержательный блок представлен теми видами образовательного контента, которые заключают в себе платформа, построенная на ее основе интерактивная сеть и дистанционный курс: интерактив-

ные модули знаний по предмету, адаптивные тестирующие оболочки, средства интерактивной коммуникации.

Модуль знаний — основная единица дистанционного обучающего курса. При разработке модулей знаний необходимо учитывать не только то, что каждый модуль должен давать совершенно определенную самостоятельную порцию знаний по теме урока, но и сформировать необходимые умения. Отсюда следует необходимость в интерактивности, которая заключается в своевременном отслеживании достижений учащегося, взаимодействии с преподавателем или самим модулем знаний, который в случае отсутствия преподавателя в интерактивной сети должен выполнять ряд его функций.

Для соответствия требованиям объективности в оценке качества знаний учащихся обучающий курс интерактивной сети должен иметь адаптивные тестирующие оболочки, позволяющие придать большую индивидуальность дистанционному обучению.

Средства интерактивной коммуникации — обязательный элемент интерактивного дистанционного курса. Для обеспечения интерактивности в курсах дистанционного обучения целесообразно использовать форумы, чаты, службы коротких сообщений, аудио- и видеоконференции, электронную почту.

Технологический блок модели призван обеспечить процесс обучения методами, приемами, средствами и формами, способствующими наиболее полной организации дистанционного обучения, и включает в себя два блока: способы организации обучения на базе предметно-ориентированной интерактивной сети и определение средств интерактивной коммуникации.

Контролирующий блок модели направлен на мониторинг качества обучения, организованного на базе предметно-ориентированной интерактивной сети с использованием как встроенных в оболочку протоколов формирования отчетов, так и эмпирических методов исследования. Качественный мониторинг необходим на каждом этапе реализации предметно-ориентированной интерактивной сети, поэтому контролирующий блок относится к каждому рассмотренному блоку модели.

Таким образом, разработанная нами и рассмотренная модель реализации предметно-ориентированной интерактивной сети позволяет

подойти к внедрению и использованию дистанционного обучения как целостному процессу, в ходе которого осуществляется целенаправленное интерактивное взаимодействие педагога, учащегося и дистанционного курса.

Список литературы

[1] Галченкова И. С. Алгоритм стратегии внедрения дистанционного обучения // Известия Смоленского государственного университета. — № 2 (6). 2009. — С. 271–280.

[2] Кушнер Ю. З. Методология и методы педагогического исследования / Учебно-методическое пособие. — Могилев: Могилевский государственный университет им. А. А. Кулешова, 2001.