IT-ТЕХНОЛОГИИ:
ТЕОРИЯ И ПРАКТИКА

Материалы семинара

Владикавказ 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное
образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ГОРНО-МЕТАЛЛУРГИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ)»

Кафедра автоматизированной обработки информации

IT-ТЕХНОЛОГИИ:

ТЕОРИЯ И ПРАКТИКА

Материалы семинара

Владикавказ 2018

УДК 004
ББК 73
А36

А36 **IT-технологии: теория и практика:** Материалы семинара / Коллектив авторов; Северо-Кавказский горно-металлургический институт (государственный технологический университет). – Владикавказ: Северо-Кавказский горно-металлургический институт (государственный технологический университет). Изд-во «Терек», 2018. – 64 с.

ISBN 978-5-9500070-2-6

В сборник вошли лучшие доклады, представленные на ежегодном декабрьском семинаре «ИТ-технологии: теория и практика» в 2017 году. В них подведены итоги и отражены результаты по ряду основных направлений исследований, осуществлявшихся преподавателями и студентами кафедры автоматизированной обработки информации СКГМИ (ГТУ) в 2017 году. В рамках этих направлений были предложены новые технологии принятия решений, получили развитие современные методы экстремального программирования, параллельной обработки данных и обработки изображений.

Сборник материалов семинара зарегистрирован в РИНЦ и размещен на сайте Научной электронной библиотеки elibrary.ru.

Договор на размещение материалов в РИНЦ №2224-12/2017К (непериодические издания).

УДК 004
ББК 73

Научное издание

IT-ТЕХНОЛОГИИ: ТЕОРИЯ И ПРАКТИКА

Материалы семинара

Редактор: *Иванченко Н. К.*
Компьютерная верстка: *Цищук Т. С.*

Подписано в печать 10.05.2018. Формат 60x84 $^{1}/_{16}$. Бумага "Снегурочка". Гарнитура «Таймс». Печать на ризографе. Усл. п.л. 3,78. Уч.-изд. л. 2,27. Тираж 50 экз. Заказ № .
Северо-Кавказский горно-металлургический институт (государственный технологический университет)
Отпечатано в отделе оперативной полиграфии СКГМИ (ГТУ).
362021, Владикавказ, ул. Николаева, 44

ISBN 978-5-9500070-2-6

УДК 519.156

Гроппен В. О., Датиев А. А., Пановская К. В.

ЭКСПЕРИМЕНТАЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ КОМПОЗИТНЫХ ВЕРСИЙ МЕТОДОВ НЕЯВНОГО ПЕРЕБОРА ПРИМЕНИТЕЛЬНО К ЗАДАЧЕ О РАНЦЕ

Экспериментально исследуется эффективность композитных модификаций динамического программирования и методов типа ветвей и границ применительно к задаче о ранце. Все эти алгоритмы совмещают технологии вычисления оценок, присущие методам типа ветвей и границ и принципы отсечения «бесперспективных» планов динамического программирования. Экспериментально продемонстрирована высокая эффективность предложенных подходов, причем показано, что превосходство композитных версий алгоритмов неявного перебора над традиционными реализациями этих методов возрастает с ростом размерности задачи о ранце.

Ключевые слова: *глобально оптимальное решение, булевы переменные, композитные алгоритмы, динамическое программирование, методы типа ветвей и границ, экспериментальные исследования, задача о ранце.*

1. Введение

Методы неявного перебора являются одним из основных инструментов решения прикладных задач, сводимых к математическим моделям с дискретно меняющимися переменными [1–3]. Ниже приведены экспериментальные результаты сравнительного анализа эффективности предложенных в [4] композитных версий этих методов применительно к задаче о ранце [5] по сравнению с «классическими» алгоритмами, гарантирующими глобально оптимальные решения задач такого рода. Далее к процедурам такого рода отнесены методы типа ветвей и границ и динамическое программирование [6–10], а под композитными алгоритмами, в соответствии с [4], понимаются процеду-

ры, в которых единовременно присутствуют компоненты этих двух технологий. При этом процедура ветвления в композитной версии методов типа ветвей и границ, осуществляющей поиск с возвратом, реализована введением в базис (выведением из базиса) на каждой итерации единовременно $h = 2$ переменных, что сопровождается перебором и вычислением оценок 2^h планов, а реализация динамического программирования и методов типа ветвей и границ, осуществляющих фронтальный спуск по дереву ветвлений, базировалась на $h = 1$. Применительно ко всем исследованным композитным версиям методов неявного перебора технология отсечения «бесперспективных» планов, используемая в динамическом программировании [9], была дополнена учетом оценок, генерируемых методами типа ветвей и границ. При этом ниже не анализировалась эффективность попыток сократить время счета методами неявного перебора за счет распараллеливания вычислений [11–14]. В отличие от такого рода подходов, сокращение времени поиска оптимальных решений композитными версиями методов неявного перебора достигается за счет сокращения числа анализируемых и хранимых в памяти планов, что позволяет надеяться на сокращение не только времени счета, но и объема используемой оперативной памяти компьютера. При этом ниже не рассматриваются методы Гомори, использующие целочисленность переменных для отсечения заведомо неперспективных направлений поиска из-за их медленной сходимости [17; 18].

Схема проведения экспериментов по проверке сравнительной эффективности всех вышеназванных алгоритмов была следующей:

1. Размерность решавшихся задач "n" менялась в диапазоне 3÷32.
2. При каждой фиксированной размерности решалось десять различных задач вида (1), коэффициенты которых генерировались с помощью генератора случайных чисел. В памяти фиксировались среднеарифметические величины времени счета T и объема M использованной оперативной памяти (последнее – только применительно к методам типа ветвей и границ, осуществляющим фронтальный спуск по дереву ветвлений) применительно к каждой величине "n".
3. Эффективность предложенных подходов отображалась отношениями одноименных характеристик их «классической» и композитной версии.

Далее представлены результаты экспериментальной проверки сравнительной эффективности классических и композитных процедур поиска глобально оптимальных решений задачи о ранце, в рамках которых применяются следующие обозначения, допущения и определения.

2. Обозначения, допущения и определения

Ниже рассматриваются задачи о ранце с вектором Z булевых переменных вида:

$$\begin{cases} F = \sum_i C_i z_i \to \max; \\ \sum_i b_i z_i \le a \ ; \\ \forall i : z_i = 1,0, \end{cases} \tag{1}$$

где $Z = \{z_1, z_2, \ldots, z_n\}$; $\forall i$, C_i, b_i, a – константы, причем далее полагаем, что все константы неотрицательны.

Далее верхняя оценка величины F при условии, что в базис введено подмножество $Z' \in Z$ переменных, значениям которых соответствует вершина x_k дерева поиска, ниже определяется выражением:

$$\Delta(x_k) = \Delta(Z') = \begin{cases} \sum_{i \in I_1(x_k)} C_i z_i + \sum_{j \in I(Z) \setminus I_1(x_k)} C_j \ , \text{если} \sum_{i \in I_1(x_k)} b_i z_i \le a; \\ -\infty, \text{в противном случае}, \end{cases} \tag{2}$$

где $I_1(x_k)$ – обозначает множество индексов введенных в базис булевых переменных, значения которых определяются положением $x_k \in X$ на дереве поиска;

$I(Z)$ – множество индексов всех булевых переменных решаемой задачи.

Аналогично определяется нижняя оценка $\delta(I_1)$ величины F задачи (1), отвечающая вершине x_k дерева поиска:

$$\delta(x_k) = \begin{cases} \sum_{i \in I_1(x_k)} C_i z_i, \text{если} \sum_{i \in I_1(x_k)} b_i z_i \le a; \\ -\infty, \text{если} \sum_{i \in I_1(x_k)} b_i z_i > a. \end{cases} \tag{3}$$

Еще одной характеристикой вершины x_k дерева поиска является ресурс $\mu(x_k)$:

$$\mu(x_k) = a - \sum_{i \in I_1(x_k)} b_i z_i. \tag{4}$$

Таким образом, каждой вершине x_k дерева поиска, построенного композитной версией любого алгоритма, реализующего неявный пе-

ребор векторов переменных задачи (1), можно поставить в соответствие вектор $V(x_k) = \{\Delta(x_k), \delta(x_k), \mu(x_k)\}$, опираясь на который можно сформулировать правило отсечения подмножеств «неперспективных» векторов переменных. Так, если на дереве поиска существуют две вершины x_k и x_p, такие, что справедливо хотя бы одно из приводимых ниже условий а, б:

$$\text{а) } \Delta(x_k) < \delta(x_p); \tag{5}$$

или

$$\text{б) } \begin{cases} I(x_k) = I(x_p); \\ \mu(x_k) \le \mu(x_p); \\ \delta(x_k) \le \mu(x_p), \end{cases} \tag{6}$$

то подмножество векторов переменных, отвечающих вершине x_k, можно исключить из дальнейшего рассмотрения.

Далее, применительно к методам неявного перебора, осуществляющим поиск решения на множестве частичных планов, используются следующие определения [19]:

1. «, отвечает «куст» с высотой h. Таким «кустом» с корнем в вершине $x_k \in X$, является ориентированный подграф [20], обладающий следующими свойствами (см. рис. 1):

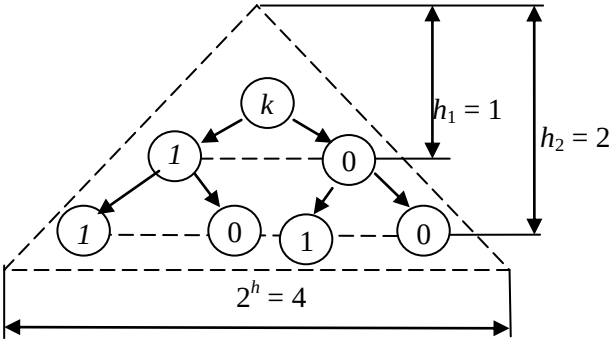

Рис. 1. Примеры кустов с корнем в $x_k \in X$, для случаев $h < 3$.

- в каждую вершину куста, за исключением корневой, входит только одна дуга;
- из каждой вершины куста, за исключением множества висячих вершин, в задачах с булевыми переменными исходят две дуги.

Таким образом, каждый куст обладает следующими свойствами:

- из корневой вершины куста $x_k \in X$, в каждую его висячую вершину существует только один путь, число дуг которого равно h;
- число вершин куста в задачах с булевыми переменными равно $2^{h+1} - 1$, число висячих вершин равно 2^h, число дуг куста определяется выражением: $2(2^h - 1)$.

Ветвлением из заданной вершины $x_k \in X$, которой соответствует подмножество введенных в базис булевых переменных $I(x_k)$, является построение «куста» с корнем в этой вершине, причем каждой из 2^h висячих вершин этого куста соответствует $\left| I(x_k) \right| + h$ введенных в базис переменных.

3. Методы типа ветвей и границ

Ниже рассмотрены традиционные и композитные версии двух типов методов такого рода: к первому отнесены методы типа ветвей и границ, осуществляющие фронтальный спуск по дереву ветвлений, а ко второму – методы типа ветвей и границ, осуществляющие «поиск с возвратом» [15; 16]. В первом случае первый же полный план, соответствующий наилучшей оценке фронта висячих вершин, является глобально оптимальным решением, во втором – даже после получения глобально оптимального решения алгоритм продолжает поиск, чтобы убедиться в этом.

3.1. Методы типа ветвей и границ, осуществляющие фронтальный спуск по дереву ветвлений

Содержательно работа такого рода алгоритма ниже описана последовательным построением дерева ветвлений $G(X, U)$

Алгоритм 1

1. На множестве висячих вершин $X_1 \subseteq X$ построенной части дерева ветвлений $G(X, U)$ выбирается вершина x_j с наилучшей оценкой. Если это осуществляется на первой итерации, то такой вершиной *a priori* считается корневая вершина дерева.

2. Если выбранной вершине отвечает равенство $I_1 = I$, то перейти к последнему шагу, в противном случае – к следующему шагу.

3. Ветвление осуществляется из выбранной на шаге 1 последней итерации вершины x_j одновременным введением в базис $h \geq 1$ пере-

менных и генерацией 2^h планов, что соответствует построению «куста», содержащего 2^h висячих вершин и аналогичного приведенным выше на рис. 1. Новое множество висячих вершин дерева вновь обозначаем X_1.

4. Вычисляются оценки, отвечающие висячим вершинам построенного на предыдущем шаге «куста». Для этого можно воспользоваться выражением (3). Перейти к шагу 1.

5. Алгоритм закончен. Вектор переменных, соответствующий выбранной вершине, является оптимальным.

Применительно к этому алгоритму на рис. 2 и 3 соответственно приведены экспериментальные зависимости среднего времени счета T_1 и объема используемой оперативной памяти M_1 от размерности решаемых задач "n".

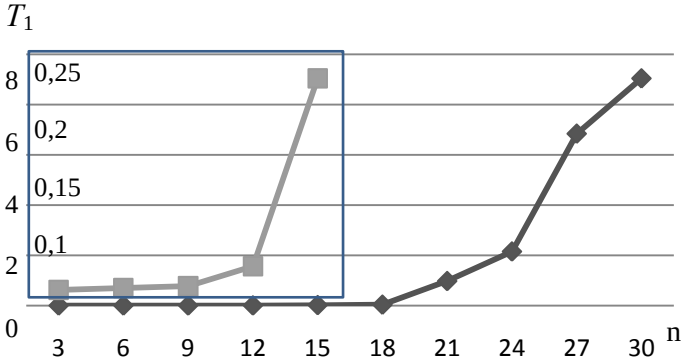

Рис. 2. Экспериментальная зависимость среднего времени T_1 поиска решения задачи (1) алгоритмом 1 от числа переменных "n" (в секундах)

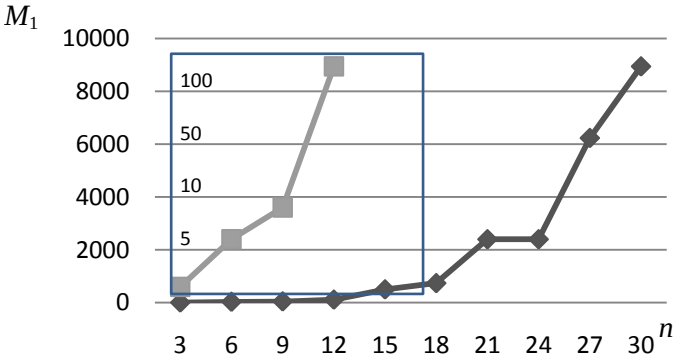

Рис. 3. Зависимость объема используемой алгоритмом 1 памяти от числа переменных "n"

Композитная версия этого алгоритма отличается от приведенного выше описанием шагов 4 и 5, причем последним шагом становится шестой шаг. Реализация этих шагов имеет вид (используется штриховая нумерация шагов):

4’. Применительно к каждой висячей вершине x_j построенного на предыдущем шаге «куста» с помощью (2)–(4) вычисляются её характеристики $\Delta(x_j)$, $\delta(x_j)$ и $\mu(x_j)$.

5’. С помощью условий (5) и (6) на множестве висячих вершин построенной части дерева ветвлений удаляются неперспективные вершины.

6’. Алгоритм закончен. Вектор переменных, соответствующий выбранной вершине, является оптимальным.

Применительно к этому алгоритму на рис. 4 и 5 соответственно приведены экспериментальные зависимости среднего времени счета $T_1^{'}$ и объема используемой оперативной памяти $M_1^{'}$ от размерности решаемых задач "n".

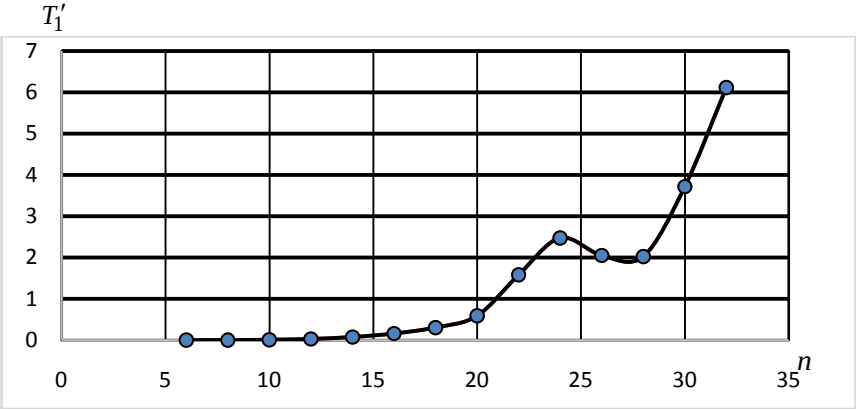

Рис. 4. Экспериментальная зависимость среднего времени $T_1^{'}$ поиска решения задачи (1) композитной версией алгоритма 1 от числа переменных "n" (в секундах)

Отношения $\eta_1 = T_1(n) / T_1^{'}(n)$ и $\eta_2 = M_1(n) / M_1^{'}(n)$, характеризующие сравнительную эффективность этих алгоритмов приведены ниже на рисунках *6* и *7* соответственно. На рис. 6 показан увеличенный в десять раз выигрыш во времени счета, демонстрируемый композитной версией алгоритма *1*, а на рис. 7 – экспериментально полученная экономия объема использованной тем же алгоритмом оперативной памяти.

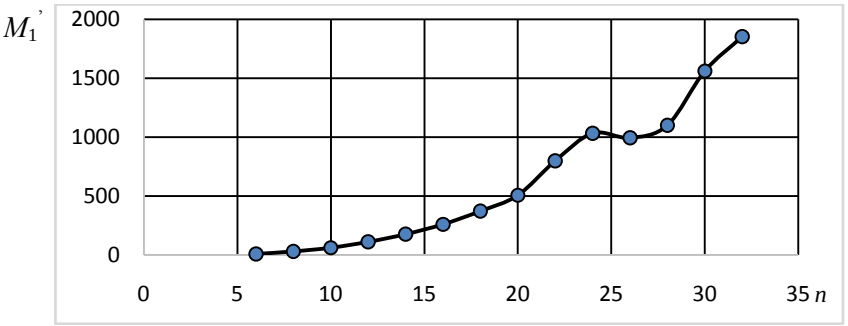

Рис. 5. Зависимость средней величины используемой оперативной памяти M_1' при поиске решения задачи (*1*) композитной версией алгоритма *1* от числа переменных "*n*" (в килобайтах)

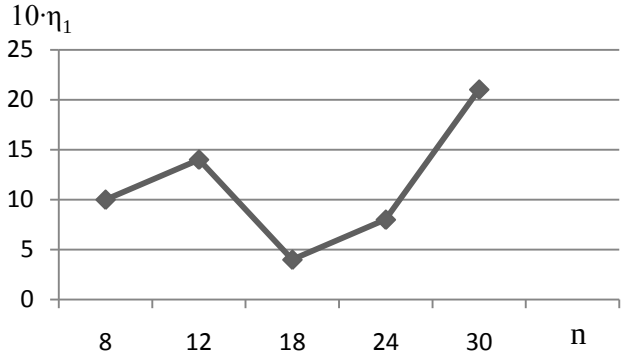

Рис. 6. Зависимость отношения $\eta_1(n)$ от числа переменных n задачи о ранце

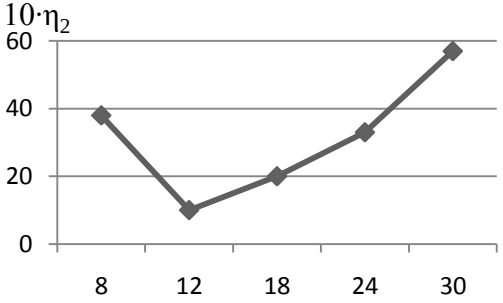

Рис. 7. Зависимость величины $10 \cdot \eta_2$ от числа переменных *n* задачи о ранце

3.2. **Методы типа ветвей и границ, осуществляющие на дереве ветвлений поиск с возвратом**

Далее приводится описание метода типа ветвей и границ, отличающееся от приведенных выше тем, что в памяти компьютера в течение всего времени поиска решения единовременно сохраняются только два числа: одно из них равно оценке Δ, отвечающей текущей вершине дерева ветвлений, другое – найденному на предыдущих итерациях значению целевой функции F [15; 16].

Алгоритм 2

1. Величине F присваивается значение, равное $-\infty$.
2. $i = 1$
3. Переменной z_i присваивается значение, равное единице.
4. Вычисляется оценка $\Delta(z_i)$:

$$\Delta(z_i) = \begin{cases} \sum_{j \le i} C_j z_j + \sum_{k > i} C_k, \text{ если } \sum_{j \le i} b_j z_j \le a; \\ -\infty, \text{ в противном случае,} \end{cases}$$

5. Если $\Delta(z_i) > F$, то перейти к шагу 6, в противном случае – к шагу 9.
6. Если $i = n$, то перейти к следующему шагу, если же $i < n$, то перейти к шагу 12.
7. Рекорду F присвоить значение, равное $\Delta(z_i)$; новые значения F и соответствующего вектора Z сохраняются в памяти, прежние удаляются.
8. Если $z_i = 1$, то перейти к шагу 9, в противном случае – к шагу 10.
9. Переменной z_i присвоить значение, равное нулю и перейти к шагу 4.
10. Если $i = 1$, то перейти к шагу 13, в противном случае – к шагу 11.
11. Величину i уменьшить на единицу и перейти к шагу 8.
12. Величину i увеличить на единицу и перейти к шагу 3.
13. Конец алгоритма. Хранящиеся в памяти значения F и Z, оптимальны.

Экспериментальная зависимость времени решения задачи о ранце алгоритмом *2* от числа переменных приведена ниже на рис. 8. Очевидно, что в силу вышеупомянутой специфики этого алгоритма, зависимость объема использованной памяти от размерности задачи описывается константой и не является актуальной.

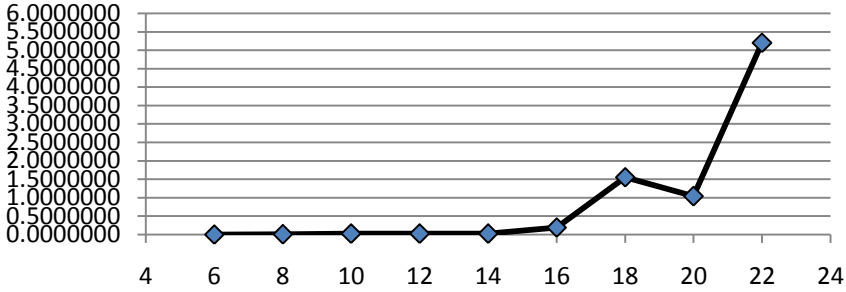

Рис. 8. Экспериментальная зависимость времени T_2 поиска решения алгоритмом 2 в секундах от числа переменных задачи о ранце

Приведенная далее композитная версия алгоритма типа ветвей и границ, осуществляющего на дереве ветвлений поиск с возвратом решения задачи о ранце, отличается от приведенного выше алгоритма *2* следующими компонентами:

- на каждой итерации в базис вводятся или из базиса выводятся $h = 2$ переменных;
- вершины x_k и x_q, удовлетворяющие условиям (6), выбираются на текущей итерации на множестве висячих вершин построенного куста;
- ветвление осуществляется на подмножестве не вычеркнутых и не помеченных вершин последнего построенного куста.

Пошаговое описание этого алгоритма приведено ниже.

Алгоритм 3

1. Величине F присваивается значение, равное $-\infty$.
2. $i = 1$
3. Сочетанию $(2i - 1)$-й и $2i$-й булевых переменных присваиваются четыре различных значения {1,1}; {1,0}; {0,1}; {0,0}, соответствующие висячим вершинам построенного куста, содержащего $h = 2$ ярусов.
4. Каждой висячей вершине x_j построенного куста ставится в соответствие вектор $R(x_j) = \{r_1(x_j), r_2(x_j), r_3(x_j)\}$, первые две компоненты которого вычисляются на основании выражений (3) и (4) соответственно, а $r_3(x_j)$ определяется по формуле (5).
5. Если среди вершин построенного куста существуют две вершины x_k и x_q, удовлетворяющие условиям (6), то вершина x_k вычеркивается.

6. На множестве не вычеркнутых и не помеченных вершин выбирается такая вершина x_j, у которой первая компонента вектора R имеет наибольшее значение. Выбранная вершина помечается. Если таковых нет, то перейти к шагу 12.

7. Если $r_1(x_j) > F$, то перейти к шагу 8, в противном случае – к шагу 11.

8. Если выполняется неравенство (5), то перейти к шагу 9, в противном случае – к шагу 11.

9. Если $i = 0.5 \cdot n$, то перейти к шагу 10, если же $i < 0.5 \cdot n$, то – к шагу 13.

10. Рекорду F присвоить значение, равное $r_1(x_j)$; новые значения F и соответствующего вектора Z сохраняются в памяти, прежние удаляются.

11. Если на множестве висячих вершин построенного куста есть не вычеркнутые и не помеченные вершины, то перейти к шагу *6*, в противном случае – к шагу *12*.

12. Величину i уменьшить на единицу. Если $i = 0$, то перейти к шагу 14, в противном случае – к шагу *11*.

13. Величину i увеличить на единицу и перейти к шагу 3.

14. Конец алгоритма. Хранящиеся в памяти значения F и Z, оптимальны.

Так как алгоритмы такого рода не критичны к объему использованной оперативной памяти [15; 16], ниже на рис. 9 и 10 соответственно приведены графики, отображающие зависимость среднего времени (в секундах) поиска
решения задачи о ранце алгоритмами *2* и *3* от числа переменных "*n*".

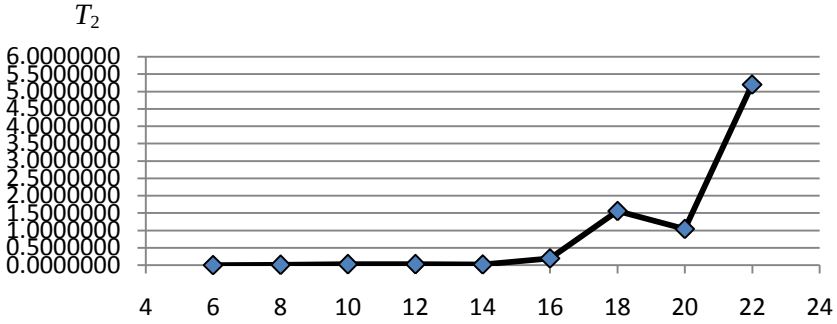

Рис. 9. Экспериментальная зависимость времени T_2 поиска решения алгоритмом *2* в секундах от числа переменных задачи о ранце

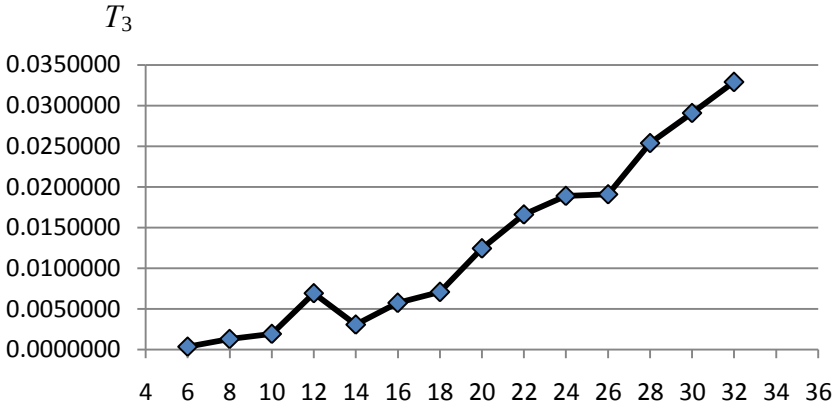

Рис. 10. Экспериментальная зависимость времени T_3 поиска решения алгоритмом *3* в секундах от числа переменных задачи о ранце

Сравнительная эффективность этих алгоритмов ниже отображается зависимостью $\eta_3(n) = T_2(n) / T_3(n)$ на рис. 11, а сравнительная эффективность рассмотренных выше двух композитных алгоритмов отображается зависимостью $\eta_4(n) = T_{1к}(n) / T_3(n)$ на рис. 12.

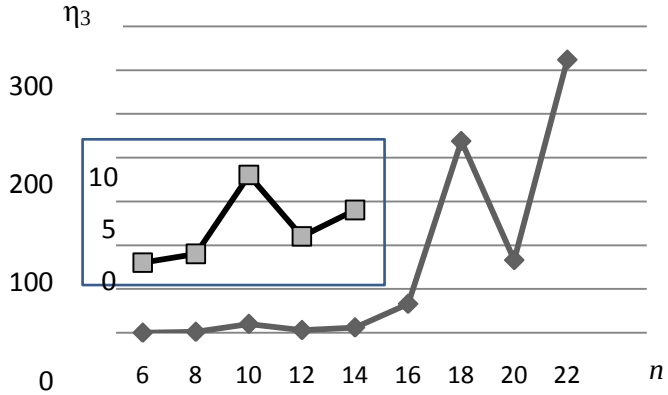

Рис. 11. Зависимость отношения $\eta_3 = T_2 / T_3$ от размерности задачи о ранце n

4. Динамическое программирование

Как и в рассмотренных выше методах типа ветвей и границ, содержательно решение экстремальной задачи с булевыми переменными с помощью динамического программирования может быть описано

последовательным, ярус за ярусом, построением сети $G_4(X_4,U_4)$ применительно к задаче (1). Ниже приводится агрегированное описание процедуры такого рода.

Алгоритм 4

1. На множестве висячих и не помеченных вершин $X' \subseteq X_4$ построенной части дерева ветвлений $G_4(X_4,U_4)$ выбирается вершина x_j. Если таковой вершины нет, то перейти к шагу 4. Если выбор осуществляется на первой итерации, то такой вершиной *a priori* считается корневая вершина дерева.

2. Осуществляется ветвление из выбранной на шаге 1 последней итерации вершины x_j и с использованием (3) и (4) вычисляются компоненты $\delta(x_k)$ и $\mu(x_k)$ каждого вектора $R(x_k)$, отвечающего каждой висячей вершине x_k построенного «куста» (величина $h = 1$).

3. Все вершины куста, построенного на шаге 2 последней итерации, помечаются, перейти к шагу 1.

4. Убираются все пометки подмножества висячих вершин дерева ветвлений.

5. Если на множестве висячих вершин существует вершина x_k, для которой на том же множестве вершин существует вершина x_q такая, что справедлива система (6), то вершина x_k вычеркивается.

6. Если число введенных в базис переменных висячих вершин дерева ветвлений равно числу переменных решаемой задачи, то перейти к следующему шагу, в противном случае перейти к шагу 1.

7. На множестве не вычеркнутых висячих вершин выбирается вершина x_k с наибольшей $\delta(x_k)$ компонентой соответствующего ей вектора $R(x_k)$.

8. Алгоритм закончен. Вектор переменных Z, соответствующий выбранной на шаге 7 вершине x_k, является оптимальным.

Ниже на рис. 12 представлена экспериментальная зависимость среднего времени T_4 поиска решения задачи (*1*) алгоритмом *4* от числа переменных n этой задачи.

Композитная версия алгоритма 4 отличается реализацией только шагов 2 и 5, которые далее приводятся с штриховой индексацией:

2’. Осуществляется ветвление из выбранной на шаге 1 последней итерации вершины x_j и с использованием (2), (3) и (4) вычисляются три компоненты $\Delta(x_k)$, $\delta(x_k)$ и $\mu(x_k)$ каждого вектора $R(x_k)$, отвечающего каждой висячей вершине x_k построенного «куста» (величина $h = 1$).

5’. Если на множестве висячих вершин существует вершина x_k, для которой на том же множестве вершин существует вершина x_q та-

кая, что справедливо хотя бы одно из условий (5) и (6), то вершина x_k вычеркивается.

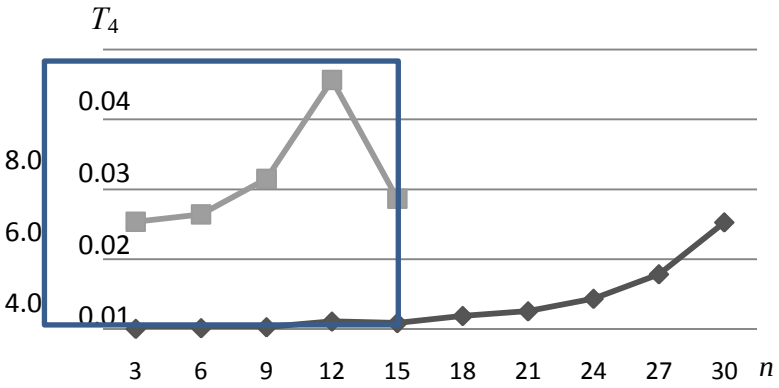

Рис. 12. Экспериментальная зависимость среднего времени поиска решения задачи (1) алгоритмом 4 от размерности задачи

Ниже на рис. 13 представлена экспериментальная зависимость среднего времени $T_4^{'}$ (в секундах) поиска решения задачи (1) композитной версией алгоритма *4* от числа переменных n этой задачи.

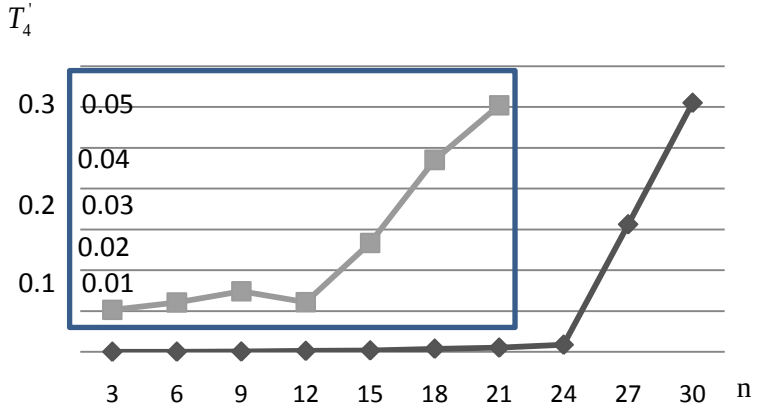

Рис. 13. Экспериментальная зависимость среднего времени $T_4^{'}$ поиска решения задачи о ранце композитной версией алгоритма 4 от числа переменных *n* в секундах

Величина выигрыша во времени поиска решения композитной версией динамического программирования по сравнению его тради-

ционной реализацией определяется отношением $\eta_4 = T_4(n) / T_4'(n)$, зависимость которого от размерности задачи о ранце представлена ниже на рис. 14.

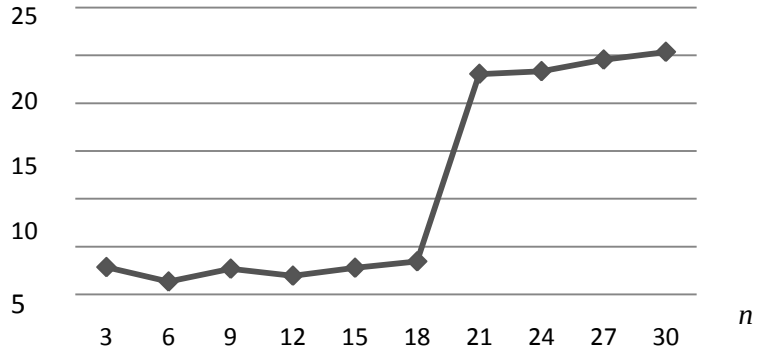

Рис. 14. Выигрыш η_4 во времени поиска решения композитной версией динамического программирования по сравнению его традиционной реализацией применительно к задаче о ранце, как функция числа переменных этой задачи

5. Заключение

Экспериментальный анализ эффективности предложенных в [1] композитных модификаций методов неявного перебора применительно к задаче о ранце подтвердил их превосходство над традиционными процедурами такого рода [6–8, 10]. К экспериментально подтвержденным достоинствам композитных алгоритмов можно также отнести рост их эффективности с ростом размерности решаемых задач (рис. 9, 10 и 13), что позволило более чем на 25 % расширить рекомендованный в [10] диапазон применения методов типа ветвей и границ.

Литература

1. *Nowozin, Sebastian; Lampert, Christoph H.* Structured Learning and Prediction in Computer Vision. Foundations and Trends in Computer Graphics and Vision. 2011, 6 (3–4), pp. 185–365.

2. *Achterberg, T.; T. Koch; A. Martin.* Branching rules revisited. Operations Research, 2005, 33 (1), pp. 42–54.

3. *Mehlhorn, Kurt*; *Sanders, Peter* (2008). Algorithms and Data Structures: The Basic Toolbox. 2008, Springer. P. 249.

4. *Гроппен В. О.* Композитные алгоритмы поиска глобально оптимальных решений экстремальных задач с булевыми переменными. Материалы семинара “IT- технологии: теория и практика”. Владикавказ: СКГМИ (ГТУ). Изд-во "Терек", 2017. С. 21–38.

5. *Caccetta, L. & Kulanoot, A.* Computational Aspects of Hard Knapsack Problems. Nonlinear Analysis. 2001, 47. Pp. 5547–5558.

6. *Land, A. H. & Doig, A. G.* An automatic method of solving discrete programming problems // Econometrica. 1960. Vol. 28, № 3. Pp. 497–520.

7. *Little, John D. C.; Murty, Katta G.; Sweeney, Dura W.; Karel, Caroline (1963).* An algorithm for the traveling salesman problem. Operations Research, 1963, 11 (6), pp. 972–989.

8. *Balas, Egon; Toth, Paolo (1983).* Branch and bound methods for the traveling salesman problem. Report in the Carnegie Mellon University, Graduate School of Industrial Administration, 1983.

9. *Bellman, R.* The Theory of Dynamic Programming. The RAND Corporation, 1954.

10. *Корбут А. А., Финкельштейн Ю. Ю.* Дискретное программирование. М.: Наука, 1969. 368 с.

11. *Casti J., Richardson M. & Larson R.* Dynamic programming and parallel computers. *JOTA*, Vol. 12, № 4, 1973. Pp. 423–438.

12. *Brochard, L.* Efficiency of some parallel numerical algorithms on distributed systems. *Parallel Computing*, 1989, Vol. 12, № 1. Pp. 21–44.

13. *Groppen V. O.* Efficiensy of parallel computations in solving extreme combinatorial problems. First World Conference on Parallel Computing in Engineering and Engineering Education. UNESCO, Paris, France, October 8–12, 1990. P. 127–131.

14. *Bader, David A.; Hart, William E.; Phillips, Cynthia A.* Parallel Algorithm Design for Branch and Bound. In Greenberg, H. J. Tutorials on Emerging Methodologies and Applications in Operations Research. Kluwer Academic Press, 2004.

15. *Donald E. Knuth.* The Art of Computer Programming. Addison-Wesley, 1968.

16. *Rossi, Francesca; Beek, Peter Van; Walsh, Toby.* Constraint Satisfaction: An Emerging Paradigm. Handbook of Constraint Programming. Foundations of Artificial Intelligence. Amsterdam: Elsevier, 2006. P. 14.

17. *Padberg, M. & Rinaldi, G. A* Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. Siam Review, 1991. Pp. 60–100.

18. *John E., Mitchell (2002).* Branch-and-Cut Algorithms for Combinatorial Optimization Problems. Handbook of Applied Optimization, 2002. Pp. 65–77.

19. *Асанов М. О. В. А., Баранский, В. В.* Расин. Дискретная математика: графы, матроиды, алгоритмы: учеб. Пособие. 2-е изд., испр. и доп. СПб.; М.; Краснодар: Лань, 2010. 362 с.

20. *Зыков А. А*. Теория конечных графов. Том I, «Наука», М., 1969.

EXPERIMENTAL ANALYSIS OF THE EFFECTIVENESS OF COMPOSITE VERSIONS OF METHODS OF IMPLICIT SEARCH FOR THE PROBLEM OF KNAPSACK

Groppen V.O., Datiev A.A., Panovskaya K.V.

The paper contains experimental analysis of efficiency of composite algorithms aimed at solving of extreme problems with Boolean variables providing global optimal solution. These algorithms combine bound computing used by branch and bound algorithms together with the cut off "having no prospects" vectors of variables technique used in dynamic programming. A posteriori is shown that these algorithms' efficiency excels efficiency of the parent procedures solving extreme problems with Boolean variables and this difference grows with the increase of complexity of problems solved.

***Keywords*:** *globaly optimal solution, Boolean variables, composite algorithms, dynamic programming, branch and bound procedures, experimental verification, knapsack problem.*

Сведения об авторах

Гроппен В. О.,
д.т.н., проф., зав. кафедрой "Автоматизированная обработка информации"; директор НИИ теоретической и прикладной информатики, Северо-Кавказский горно-металлургический институт (государственный технологический университет), Владикавказ, Россия.
Тел. +7(8672)40-71-07, e-mail: groppen@mail.ru

V. O. Groppen,
Professor, Head of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Russia, Vladikavkaz. Tel.: +7(8672)40-71-07, e-mail: groppen@mail.ru

Датиев А. А.,
магистр, аспирант кафедры "Автоматизированная обработка информации", программист НИИ теоретической и прикладной информатики. Северо-Кавказский горно-металлургический институт (государственный технологический университет), Владикавказ, Россия.
Тел. +7(8672)40-71-08, e-mail: atsamaz.datieff@yandex.ru
A. A. Datiev,
Master of Computer Science, Graduate student of dept. «Automated information processing». *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Russia, Vladikavkaz.
Tel.:+7(8672)40-71-08, e-mail: atsamaz.datieff@yandex.ru

Пановская К. В.,
бакалавр, магистрант кафедры "Автоматизированная обработка информации". Северо-Кавказский горно-металлургический институт (государственный технологический университет). Владикавказ, Россия.
Тел.+79194228863, e-mail: ksenija15reg@yandex.ru
K. V. Panovskaya,
Bachelor of Computer Science, Graduate student of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Russia, Vladikavkaz.
Tel.:+79194228863, e-mail: ksenija15reg@yandex.ru

УДК 681.343.001

Гроппен В. О.,
Будаева А. А.

ПОВЫШЕНИЕ БЫСТРОДЕЙСТВИЯ МЕТОДОВ РЕШЕНИЯ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ С БУЛЕВЫМИ ПЕРЕМЕННЫМИ С ПОМОЩЬЮ ЭТАЛОНОВ

Предлагается новый подход к формулировке и решению многокритериальных задач с булевыми переменными на основе модифицированного метода свертки критериев методом эталонов. Предлагаемый в работе модифицированный алгоритм позволяет уйти от полного перебора, сократив тем самым объемы задействованных для решения задачи ресурсов процессорного времени и оперативной. Важность полученных результатов обусловлена широкой областью применения дискретных булевых моделей.

Ключевые слова: *многокритериальные, дискретные, метод эталонов, булевы, переменные, быстродействие, свертка.*

1. Введение

Преимуществом развиваемого в [1–10] подхода к решению многокритериальных экстремальных задач с булевыми переменными, использующим эталоны для свертки критериев, по сравнению с широко распространенными методами взвешенной суммы критериев или их лексикографического упорядочения [11–13], является отсутствие необходимости в дополнительной информации о важности критериев, отсутствующей в оригинальной постановке задачи. В то же время этот подход обладает существенным недостатком: его реализация применительно к решению конкретной многокритериальной задачи с дискретными переменными связана с использованием существенных машинных ресурсов. Последнее объясняется неоднократным применением комбинаторных алгоритмов, гарантирующих глобально оптимальное решение, как на этапе поиска эталонов, так и после свертки критериев, в ходе решения полученной однокритериальной задачи [4; 8]. Ниже предлагается модификация этого подхода, позволяющая существенно сократить затраты машинных ресурсов. При этом ниже используются следующие обозначения и определения.

2. Обозначения, определения и допущения

$\boldsymbol{X}$ – вектор булевых переменных, где:

$$\begin{cases} \boldsymbol{X} = \{x_1, x_2, \ldots, x_n\}; \\ x_i = 1{,}0; \;\; i = 1, 2, \ldots, n. \end{cases}$$

$F_j(\boldsymbol{X})$ – j-й критерий: $F_j(\boldsymbol{X}) = \sum_{i=1}^{i=n} c_{i,j}\, x_i$; $j = 1, 2, \ldots, m$.

$D_k(\boldsymbol{X})$ – k-е ограничение: $D_k(\boldsymbol{X}) = \sum_{i=1}^{i=n} d_{i,k} x_i \le h_k$; $k = 1, 2, \ldots, g$.

$B_{j,q}$ – полученная q-м методом величина, лучше которой значение j-го критерия $F_j(X)$ не может быть получено: для любого сочетания значений булевых переменных вектора X справедливо:

$$\forall q : B_j \succ B_{j_q} \succ F_j(X).$$

$t(B_{j,q})$ – время вычисления оценки критерия $F_j(\boldsymbol{X})$ q-м методом;

$t_q(F_j(\boldsymbol{X}))$ – среднее время получения глобально оптимального решения однокритериальной задачи с вектором булевых переменных X и целевой функцией $F_j(\boldsymbol{X})$ q-м методом.

Далее используются следующие допущения:

1) Если оценки *j*-й целевой функции некоторой задачи $B_{j,1}$ и $B_{j,2}$ получены одним из методов неявного перебора и симплекс методом соответственно, причем в последнем случае ограничения целочисленности отброшены, то

$$t(B_{j,1}) >> t(B_{j,2}). \quad (1)$$

2) Если q соответствует одному из методов неявного перебора, то применительно к любой фиксированной многокритериальной задаче с вектором булевых переменных X справедлива система:

$$\begin{cases} \forall j \neq p : t_q(F_j(X)) = t_q(F_p(X)) = \tau_q; \\ \forall j : t(B_{j,q}) = t_q(F_j(X)), \end{cases} \quad (2)$$

где τ_q – некоторая константа, зависящая от размерности решаемой задачи, от параметров используемого компьютера и от специфики q-й процедуры.

3. Формальная постановка задачи

В общем случае формальная постановка многокритериальной задачи с булевыми переменными может быть представлена в виде:

$$\begin{cases} \forall j : F_j(X) \to \max(\min); \\ \forall k : D_k(X) Q_k h_k; \\ \forall x_i \in X : x_i = 1,0; \\ Q_k \in \{\leq; =; \neq; \geq\}. \end{cases} \quad (3)$$

В [2–4] для определения величин B_j предлагается искать глобально оптимальные решения m задач с булевыми переменными (каждую – для «своего» значения j) методами типа ветвей и границ либо с помощью динамического программирования [14; 15]:

$$\begin{cases} B_j = F_j(X) \to \max(\min); \\ \forall k : D_k(X) Q_k h_k; \\ \forall x_i \in X : x_i = 1,0; \\ Q_k \in \{\leq; =; \neq; \geq\}. \end{cases} \quad (4)$$

В [2] показано, что оптимальный вектор переменных приведенной ниже однокритериальной задачи (5) одновременно является Парето-оптимальным решением задачи (3):

$$\begin{cases} \sum\limits_{j} [B_j - F_j(X)]^2 \to \min; \\ \forall k : D_k(X) Q_k h_k; \\ \forall x_i \in X : x_i = 1{,}0; \\ Q_k \in \{\leq; =; \neq; \geq\}. \end{cases} \quad (5)$$

Иными словами поиск решения многокритериальной задачи с булевыми переменными вида (1) заменяется поиском такого вектора переменных X, которому в m-мерном евклидовом пространстве критериев отвечает точка и вектор $F = \{F_1, F_2, \ldots, F_m\}$, находящиеся на минимальном расстоянии от точки $B = \{B_1, B_2, \ldots, B_m\}$. С учетом сделанных выше допущений (2) легко показать, что время T_q поиска решения задачи (3) предложенной в [2; 4] технологией, в рамках которой используется q-й метод неявного перебора, равно:

$$T_q = [m + signum\,(m - 1)]\,\tau_q. \quad (6)$$

4. Минимизация времени поиска решения

Минимизация времени поиска решения, опирающаяся на (6), сводится к выбору наиболее эффективного метода неявного перебора, время поиска решения которого

$$T = \min_q T_q. \quad (7)$$

Ниже предлагается подход к сокращению времени поиска решения задачи (3), основанный на двух принципах:

- отказ от целочисленности переменных при вычислении оценок B_j;
- применение композитных алгоритмов поиска глобально оптимальных решений задач с булевыми переменными [16].

Отказ от целочисленности переменных при вычислении оценок B_j означает, что (4) преобразуется в задачу линейного программирования вида:

$$\begin{cases} B_j = F_j(X) \to \max(\min); \\ \forall k : D_k(X) Q_k h_k; \\ \forall x_i \in X : 0 \le x_i \le 1; \\ Q_k \in \{\le; =; \ne; \ge\}. \end{cases} \quad (8)$$

В связи с допущением (1) временем, затраченным на решение (8) симплекс методом, можно пренебречь, что позволяет заменить (6) равенством:

$$T_q = \tau_q. \quad (9)$$

В результате учитываются лишь затраты времени, связанные с решением системы (5), для чего предлагается воспользоваться композитными алгоритмами [16]. В этой работе эффективность композитных версий трех алгоритмов – метода типа ветвей и границ, осуществлявшего фронтальный спуск по дереву ветвлений (q = 1); метода типа ветвей и границ, осуществлявшего поиск с возвратом (q = 2); динамического программирования (q = 3) – оценивалась выигрышем во времени счета η_q по сравнению с традиционной версией того же алгоритма. Ниже приведены полиномы второго порядка, отображающие в первом приближении зависимости этой величины от размерности решаемых задач при условии, что число переменных определяется приведенным выше диапазоном:

$$\eta_1 = 25{,}93 - 2{,}2332 \cdot |X| + 6{,}78 \cdot |X|^2; \quad (10)$$

$$\eta_2 = 106{,}088 - 26{,}2 \cdot |X| + 1{,}56 \cdot |X|^2; \quad (11)$$

$$\eta_3 = 6{,}167 - 0{,}233 \cdot |X| + 0{,}3114 \cdot |X|^2; \quad (12)$$

С учетом (10)– (12) равенство (9) можно преобразовать к виду:

$$T_q = \tau_q / \eta_q. \quad (13)$$

Сравнивая (6) и (13), можно оценить границы выигрыша η во времени решения задачи (3) в соответствии с предлагаемой выше методикой:

$$\min_q \eta_q \le \eta \le \max_q \left[m + signum\,(m-1)\right] \cdot \eta_q. \quad (14)$$

5. **Заключение**

Предложенный выше подход к решению многокритериальных экстремальных задач с булевыми переменными, использующий эталоны для свертки критериев, позволяет существенно сократить время поиска решения этих задач.

Литература

1. *Гроппен В. О.* Принципы решения многокритериальных задач с помощью эталонов / Труды XII Всероссийской научно-методической конференции "Телематика". Санкт Петербург, 6–9 июня 2005. СПб., 2005. Том 1. Стр. 125–128.

2. *Гроппен В. О*. Решение задач многокритериальной оптимизации и ранжирования объектов методом эталонов // Телекоммуникации и информатизация образования. Москва. 2006.: № 2 (33), март–апрель, С. 14–31.

3. *Гроппен В. О.* Принципы принятия решений с помощью эталонов // Автоматика и телемеханика. 2006. № 4. С. 167–184.

4. *Groppen V. O.* New Solution Principle for Multi-criteria Problems Based on Comparison Standards: Models, Algorithms, Applications. Program and Abstracts Eurogen 2007. Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems. 11–13 June 2007 Jyvaskyla, Finland. Pp. 100–101.

5. *Вагин В. С., Гроппен В. О., Позднякова Т. А., Будаева А. А.* Многокритериальное ранжирование объектов методом эталонов как инструмент оптимального управления // Устойчивое развитие горных территорий. 2010. № 1. С. 47–56.

6. *Будаева А. А., Гроппен В. О.* Выбор оптимальной технологии ранжирования // Устойчивое развитие горных территорий. 2014. Том 6. № 3 (21). С. 3–7.

7. *Шевченко В. А*. Прогнозирование успеваемости студентов на основе методов кластерного анализа // Вестник ХНАДУ. 2015. Вып. 68. С. 15–18.

8. *Будаева А. А*. Использование метода эталонов для решения задач дискретной многокритериальной оптимизации // Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. 2015. Т. 15. № 1. С. 22–27.

9. *Томаев М. Х., Кисиев В. П.* Многокритериальный подход к оптимальной декомпозиции пользовательского программного кода /

IT-технологии: теория и практика: Материалы семинара. Владикавказ: СКГМИ (ГТУ). Изд-во «Терек», 2017. С. 56–67.

10. *Будаева А. А.* Использование метода эталонов для повышения качества группировок в задачах таксономии / IT-технологии: развитие и приложения: Владикавказ: СКГМИ (ГТУ), Изд-во "Терек", 2016. С. 3–9.

11. *Миркин Б. Г.* Проблема группового выбора. М.: Наука, 1974. 256 с.

12. *Пужаев А. В.* Управленческие решения. СПб.: МБИ, 2004. 152 с.

13. *Шелобаев С. И.* Математические методы и модели в экономике, финансах, бизнесе. М.: ЮНИТИ-ДАНА, 2000. 367 с.

14. *Land A. H. & Doig A. G.* An automatic method of solving discrete programming problems // Econometrica. Vol. 28, No. 3. 1960. Pp. 497–520.

15. *Bellman R.* The Theory of Dynamic Programming. The RAND Corporation, 1954.

16. *Гроппен В. О.* Композитные алгоритмы поиска глобально оптимальных решений экстремальных задач с булевыми переменными / IT-технологии: развитие и приложения: Материалы семинара. Владикавказ: СКГМИ(ГТУ), изд-во "Терек", 2016. С. 21–38.

INCREASING SOLVING SPEED OF THE MULTICRITERIA PROBLEM WITH BOOLEAN VARIABLES USING METHOD OF STANDARDS

Groppen V. O., Budaeva A. A.

A new approach to the formulation and solution of multicriteria problems with Boolean variables based on the modified method of convolution of criteria by the method of standards is proposed. The modified algorithm proposed in this work allows you to avoid a full search, thereby reducing the amount of CPU time and operational resources used to solve the problem. The importance of the results obtained is due to the wide range of applications of discrete Boolean models.

Keywords: *multicriterial, discrete, method of standards, Boolean, variables, speed, convolution.*

Сведения об авторах

Гроппен В. О.,
д.т.н., проф., зав. кафедрой "Автоматизированная обработка информации"; директор НИИ теоретической и прикладной информатики, Северо-Кавказский горно-металлургический институт (государственный технологический университет), Владикавказ, Россия.
Тел. +7(8672)40-71-07, e-mail: groppen@mail.ru

V. O. Groppen,
Professor, Head of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Russia, Vladikavkaz.
Tel.: +7(8672)40-71-07, e-mail: groppen@mail.ru

Будаева А. А.,
к.т.н., доцент кафедры "Автоматизированная обработка информации", программист НИИ теоретической и прикладной информатики. Владикавказ, Россия, Северо-Кавказский горно-металлургический институт (государственный технологический университет).
Тел.: +7672407518. E-mail: budalina@yandex.ru

A. A. Budaeva,
Ph.D., assistant professor of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia,
Tel.: +7672407518, e-mail: budalina@yandex.ru

УДК 681.343.001

Будаева А. А.

ОБ ОДНОМ ПОДХОДЕ К ПРОГНОЗИРОВАНИЮ УСПЕВАЕМОСТИ СТУДЕНТОВ

Предлагается процедура прогнозирования персональной успеваемости студентов. В основе процедуры прогнозирования лежит поиск студентов-аналогов на базе методов таксономии и эталонов. Персональный прогноз эффективности дальнейшего обучения, базируется на прецедентах. Приводятся формальные постановки этой задачи, условия экспериментальной проверки предложенного подхода и экспериментально полученная точность прогноза, подтверждающая высокую эффективность предложенного подхода.

***Ключевые слова**: персональная успеваемость, прогнозирование, аналитические модели, аналоги, расстояние, оптимальная таксономия, эталон.*

1. Введение

В связи с информационным развитием общества и научно-техническим ростом подготовка квалифицированных специалистов приобретает важное значение. Прогнозирование успеваемости студентов высших учебных заведений может существенно помочь в решении данного вопроса. Благодаря предвидению успеваемости каждого студента в начале обучения возможно правильно скоординировать весь ход дальнейшего обучения.

В настоящее время существуют сотни методов прогнозирования: корреляционный анализ, регрессионный анализ, таксономия (кластерный анализ), факторный анализ и др. Рассмотрением сущности прогнозирования в области обучения занимались Б. С. Гершунский, В. И. Загвязинский, А. Ф. Присяжная, Р. В. Майер и др. [1–5].

В ходе анализа публикаций сделан вывод, что для достоверного прогнозирования успеваемости студентов больше всего подходят методы таксономии, поскольку они позволяют:

- рассматривать множество исходных данных практически произвольной природы;
- производить разбиение объектов не по одному параметру, а по целому набору признаков, что дает возможность анализа результатов учебной деятельности студентов за весь период обучения;

- осуществлять прогноз дальнейшего обучения студента, основываясь на прецедентах.

2. Обозначение и допущения

Ниже используются следующие обозначения:

$F_j(\vec{Z})$ – j-й целевой критерий;

B_j – величина, соответствующая наилучшему значению j-го критерия $F_j(\vec{X})$;

$\vec{B}$ – вектор наилучших критериев $\vec{B} = \{B_1, B_2, \ldots, B_k\}$;

c_i – центр i-го таксона;

$r(c_i, p)$ – расстояние между центром i-го таксона и p-м объектом;

$z(i, j)$ – булева переменная, равная единице, если j-й объект принадлежит i-му таксону, и равная нулю если не принадлежит,

n – число таксонов;

S – множество всех студентов $S = \{s_1, s_2, \ldots s_n\}$;

T – множество таксонов студентов;

f – таксон аналогов выбранного студента;

$P(i,k)$ – прогноз k-й характеристики для i-го анализируемого студента;

$X(j,k)$ –k-ая характеристика j-го аналога, где $j \in T_f$;

m_i –количество аналогов i-го анализируемого студента.

3. Процедура прогнозирования успеваемости студентов на базе методов таксономии и эталонов

В основе процедуры прогнозирования успеваемости лежит поиск студентов-аналогов выбранного студента, отстоящих от него в пространстве признаков на минимальном расстоянии.

Прогноз дальнейшего обучения базируется на прецедентах (т.е. на сравнении результатов обучения студента за пройденный этап с результатами других студентов, обучающихся на старших курсах, либо закончивших обучение).

Очевидно, что качество прогнозирования, зависит от объема выборки и от степени близости студентов и их аналогов. Поэтому с целью повышения достоверности прогноза были приняты следующие ***допущения***:

1. Сравниваются студенты одного направления
2. Прогнозирование проводится персонально по каждому студенту.
3. Выбранный студент сравнивается только со студентами, закончившими обучение или обучающимися на старших курсах
4. Поиск аналогов осуществляется в евклидовом пространстве.
5. В качестве аналогов рассматриваются студенты, попавшие в ближайший таксон.

Таким образом, на основании принятых допущений выявляются основные **этапы** в решении поставленной прикладной задачи.

1. *Таксономии студентов и выбор таксона аналогов*

Суть этапа состоит в выборе математической модели поиска оптимальной таксономии студентов старших курсов и студентов, закончивших обучение. Оптимальной таксономией будем считать таксономию, в которой достигаются наилучшие значения выбранных целевых критериев, таких как:

- максимальная схожесть объектов с центром таксонов;
- максимальный объем объектов внутри таксонов;
- максимальная удаленность объектов разных таксонов друг от друга.

Эти критерии позволяют построить таксономию, в которой в таксон попадают максимально схожие объекты

В качестве таксона аналогов студента выбирается таксон, до центра которого расстояние минимально.

Формальная постановка задачи построения оптимальной таксономии представлена системой (1):

$$\begin{cases} F_1\left(\vec{Z}\right) = \max\limits_{i} \dfrac{1}{N_i} \sum\limits_{p} \sum\limits_{q} r(p,q) \to \min \\ F_2\left(\vec{Z}\right) = \sum\limits_{t} \sum\limits_{p} r(c_i, p) \cdot z(p,i) \to \min \\ F_3\left(\vec{Z}\right) = \min\limits_{i,\, j \neq i} \min\limits_{p,\, q \neq p} r(p,q) \cdot z(p,i) \cdot z(q,j) \to \max \\ \forall p : \sum\limits_{i=1}^{n} z(p,i) = 1 \\ \forall p, \forall i : z(p,i) = 1{,}0. \end{cases} \qquad (1)$$

Задача (1) относится к классу задач дискретной многокритериальной оптимизации, одним из способов решения которой является сведение ее к однокритериальной (система (2)) посредством метода эталонов [6–9].

$$\begin{cases} F_{\text{sup}}\left(\vec{B}\right) = \sum_{i=1}^{3}\left[B_i - F_i\left(\vec{Z}\right)\right]^2 \to \min \\ \forall p : \sum_{i=1}^{n} z(p,i) = 1 \\ \forall p, \forall i : z(p,i) = 1{,}0. \end{cases} \quad (2)$$

2. Прогнозирование успеваемости

Целью этапа является прогнозирование успеваемости выбранного студента на основании анализа данных успеваемости найденных аналогов. Для выделения подмножества аналогов, а также для непосредственного прогноза успеваемости используется формальная постановка (3):

$$\begin{cases} \forall h \in S : \exists f \in T : r\left(c_f, h\right) = \min\limits_i r\left(c_i, h\right) \\ \forall i, \forall k : P_{i,k} = \dfrac{1}{m_i} \sum\limits_{j \in T_f} X\left(j,k\right). \end{cases} \quad (3)$$

В качестве прогнозной оценки по дисциплине используется среднее значение оценок студентов-аналогов.

Вследствие того что студенты-аналоги выбираются как из выпускников, так и из студентов старших курсов, то невозможно корректно спрогнозировать успеваемость за весь период обучения. В этой связи имеет смысл осуществлять прогноз успеваемости на ближайший семестр.

Экспериментальные исследования предложенной процедуры прогнозирования успеваемости проводились на кафедре автоматизированной обработки информации СКГМИ (ГТУ) на оценках студентов, обучавшихся по программе бакалавриата направления «Информатика и вычислительная техника» в 2016 году, информация о которых содержалась в базе данных АСУ СКГМИ (ГТУ). Всего в эксперименте оценивались результаты обучения 307 студентов, в том числе: 124 обучающихся и 183 выпускника. Для оценки достоверности прогноза выполнялось прогнозирование уже известных результатов обучения студентов. Средняя ошибка прогноза составила 0,55 баллов.

4. Заключение

Представлена процедура прогнозирования персональной успеваемости студентов на базе методов таксономии и эталонов. Процедура базируется на поиске оптимальных группировок, что приводит к более качественному подбору студентов-аналогов и, как следствие, обеспечивает более высокую точность прогноза.Предложенная процедура представляется перспективной, однако для получения более качественного прогноза необходимо увеличить и оптимизировать количество исходных данных.

Литература

1. *Гершунский Б. С.* Прогностические методы в педагогике. К.: Вища школа, 1979. 240 с.

2. *Загвязинский В. И.* Педагогическое предвидение. М.: Знание, 1987. 77 с.

3. *Присяжная А. Ф.* Прогнозирование как функция педагога (от будущего учителя до профессионала): монография. Челябинск: Образование, 2006. 306 с.

4. *Майер Р. В.* Классификация учебных фактов методом кластерного анализа // Проблемы учебного физического эксперимента: сб. науч. и метод. работ. 1998. Вып. 5. С. 12–19.

5. *Шевченко В. А*. Прогнозирование успеваемости студентов на основе методов кластерного анализа // Вестник ХНАДУ. 2015. Вып. 68. С. 15–18.

6. *Будаева А. А.* Использование метода эталонов для решения задач дискретной многокритериальной оптимизации // Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. 2015. Т. 15, № 1. С. 22–27.

7. *Томаев М. Х., Кисиев В. П.* Многокритериальный подход к оптимальной декомпозиции пользовательского программного кода / IT-технологии: теория и практика: Материалы семинара. Владикавказ: СКГМИ (ГТУ). Изд-во «Терек», 2017, С. 56–67

8. *Будаева А. А*. Использование метода эталонов для повышения качества группировок в задачах таксономии / IT-технологии: развитие и приложения: Сборник докладов. Владикавказ: СКГМИ (ГТУ), Изд-во "Терек", 2016. С. 3–9.

9. *Будаева А. А., Гроппен В. О.* Выбор оптимальной технологии ранжирования // Устойчивое развитие горных территорий. 2014. Томь, № 3 (21). С. 3–7.

ABOUT ONE APPROACH TO FORECASTING OF THE STUDENT PERFORMANCE

Budaeva A. A.

Proposed procedure for predicting personal performance of students. The procedure of forecasting is based on the search for analogical students on the basis of taxonomy methods and standards. The personal forecast of the effectiveness of further education is based on precedents. Formal statements of this problem, conditions for experimental verification of the proposed approach and experimental accuracy of the forecast confirming the high efficiency of the proposed approach are presented.

Keywords: *personal performance, forecasting, analytical models, analogs, distance, optimal taxonomy, standard.*

Сведения об авторах

Будаева А. А.,
к.т.н., доцент кафедры "Автоматизированная обработка информации", программист НИИ теоретической и прикладной информатики. Владикавказ, Россия, Северо-Кавказский горно-металлургический институт (государственный технологический университет).
Тел.: +7672407518. E-mail: budalina@yandex.ru

A. A. Budaeva,
Ph.D., assistant professor of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia,
Tel.: +7672407518, e-mail: budalina@yandex.ru

ТЕХНОЛОГИИ ОПТИМИЗАЦИИ ПОЛЬЗОВАТЕЛЬСКИХ КОДОВ

УДК 681.343.001

Томаев М. Х., Алборов М. А.

ОЦЕНКА ПОТОКОБЕЗОПАСНОСТИ ИСХОДНОГО ПРОГРАММНОГО КОДА ПРИЛОЖЕНИЙ, ФУНКЦИОНИРУЮЩИХ В МНОГОЗАДАЧНОЙ СРЕДЕ

Предлагается подход к оценке степени потокобезопасности пользовательских исходных программных кодов вычислительных систем, проектируемых для работы на аппаратных платформах, поддерживающих технологии многозадачности. Работа содержит формальную постановку задачи оценки потокобезопасности пользовательского кода, приводятся примеры прикладного приложения в практике проектирования программных систем. Разработанный математический аппарат позволяет систематизировать подходы к верификации качества многопоточных приложений.

Ключевые слова: *программа, поток, потокобезопасность, модель, качество, синхронизация, доступ, переменная, запись, чтение.*

1. Введение

Особенностью современных аппаратных платформ вычислительных систем, является поддержка многоядерной и многопроцессорной архитектур [1] уже в системах начального уровня, доступных широкому кругу пользователей. Разработка новых программных систем, как правило, всегда включает этап поиска возможностей адаптации тех или иных прикладных алгоритмов к работе в многозадачной среде. Новой качественной характеристикой отдельных участков кода является «потокобезопасность» – это термин, связанный с проблемой обеспечения безопасного доступа к данным из параллельно функционирующих участков кода [2]. Для безопасной синхронизации участков кода, использующих доступ к совместно используемым данным, современные ОС предлагают широкий круг инструментальных средств [3]: «Критические секции» (CriticalSec-

tions) для синхронизации потоков в рамках одного приложения; «мютексы» и «семафоры», поддерживающие синхронизацию как потоков, так и различных приложений. К сожалению, использование данных механизмов сопряжено со значительными затратами процессорного времени. Избыточное применение перечисленных средств синхронизации приводит к заметному снижению производительности программного кода, в связи с чем при проектировании вычислительных систем, для которых время поиска решения является важнейшим фактором качества, количество участков синхронизации стремятся минимизировать. В связи с этим в системах больших масштабов сложно избежать ошибок синхронизации, т. е. коллизий доступа к совместным данным, когда к одному и тому же объекту одновременно получают доступ по записи несколько параллельных потоков (либо процессов). Данная работа предлагает способ формализации подобных ошибок на основе дискретных подходов.

2. Формулировка задачи оценки потокобезопасности

Введем термин «блокированной переменной» для совместно используемых элементов данных, использующих механизмы синхронизации – т. е. организована безопасная работа с этими переменными в условиях многопоточности (многозадачности). Блокированной переменной мы считаем ту, которая заключена в блок ожидания мьютекса, семафора, события или в критическую секцию. Также введем следующие обозначения:

N – количество незаблокированных переменных.

M – количество одновременно выполняющихся потоков, в которых происходит чтение и запись этих переменных.

R, W – матрицы размера N x M для которых справедливо:

Элемент Ww_{ij} равен единице, если в -ю переменную происходит запись в j-м потоке. и нулю – в противном случае.

Элемент Rr_{ij} равен единице, если в -м потоке i-я переменная только считывается (то есть в том же потоке нет операций записи), и нулю – в противном случае. Таким образом, если в -м потоке i-я переменная и считывается, и записывается, то $r_{ij} = 0$.

Ошибкой одновременного доступа являются два случая:

1) когда в одну переменную запись выполняется в двух или более потоках – независимо от числа потоков чтения для этой переменной;

2) когда в переменную выполняется запись только в одном пото-

ке, но при этом количество потоков чтения этой переменной больше нуля.

Введем вспомогательные переменные:

p_i – для обозначения количества потоков записи *i*-й переменной:

$$p_i = \sum_{j=1}^{M} w_{ij} \tag{1}$$

b_i – для обозначения количества потоков чтения *j*-й переменной:

$$b_i = \sum_{j=1}^{M} r_{ij} \,. \tag{2}$$

Тогда необходимость использования функций синхронизации для -й переменной можно сформулировать в виде следующей функции:

$$E(i) = signum(signum(p_i)\,(signum(b_i) + signum(p_i - 1))). \tag{3}$$

Так как количество потоков чтения и записи для переменной – значение неотрицательное, т. е.

$$\forall i : b_i \geq 0, \quad p_i \geq 0, \tag{4}$$

то $E(i)$ будет всегда будет возвращать только 2 значения:

0 – когда ошибок синхронизации при работе *c*-й переменной не было обнаружено. Этому случаю соответствуют следующие варианты:

когда $b_i = 0, p_i = 0$;

когда $b_i > 0, p_i = 0$ (если нет потоков записи, то ошибки нет, независимо от количества потоков чтения);

когда $b_i = 0, p_i = 1$ (когда нет потоков чтения, то ошибки нет, если есть только один поток записи);

1 – в случае, когда имеет место ошибка синхронизации.

Используя обозначения, задачу определения переменных программы, для которых имеет место ошибка синхронизации, можно сформулировать следующим образом:

$$\begin{cases} F = \prod_{i=1}^{N} (E(i) + 1 - z_i) = 1; \\ E(i) = signum(signum(p_i)(signum(b_i) + signum(p_i - 1))); \\ p_i = \sum_{j=1}^{M} w_{ij}; \\ b_i = \sum_{j=1}^{M} r_{ij}; \\ \forall (i, j): w_{ij} = 0,1; r_{ij} = 0,1; \\ \forall i: z_i = 0,1. \end{cases} \quad (5)$$

где z_i – булева переменная, равная единице, если при работе с -ой переменной допущена ошибка синхронизации, и нулю в противном случае.

Неизвестными в данной модели являются булевы переменные z_i. В случае, если для i-ой переменной имеет место ошибка синхронизации, (то есть $E(i) = 1$), то для компенсации этой ошибки (т. е. для того чтобы произведение F было равно 1) z_i должно быть равно 1. Если же для i-й переменной ошибок синхронизации не обнаружено (то есть $E(i) = 0$), то z_i должно быть равно 1.

3. Прикладное применение в технологии проектирования

Оценка многопоточных приложений начинается с выделения в пользовательском коде подмножества всех функций, используемых в качестве главной функции-потока. Для международно-стандартизированных языков программирования данный этап представляет собой поиск наименований стандартных средств запуска потока, которые в качестве одного из параметров принимают имя пользовательской функции.

К примеру в язык «C++», начиная с версии «11», включен стандартный класс std::thread, позволяющий работать с асинхронными потоками на основе пользовательских функций (Листинг1).

```
#include <thread>
int x=2, k=5, q= 4;
```

```
void F1()
{int y = x + 1 + q;}
void F2()
{int z = x * 10 * q;}
void F3()
{ for (inti=0; i<1000; i++) x *= 3+q;}
void F4()
{for (inti=0; i<1000; i++){ x /= 3;k *= 2; }
}

void main ()
{           std::thread thread1( F1 );std::thread thread2( F2 );
              std::thread thread3( F3 );std::thread thread4( F4 );
              // ожиданиезавершенияпотоков
              thread1.join();thread2.join(); thread3.join(); thread4.join();
}
```

В приведенном выше листинге функции *F* 1, *F* 2, *F* 3 и *F*4 выполняются в параллельных потоках. Переменные *x, k, q* используются для совместного доступа. Переменные *y* и *z* являются локальными – не используются для совместного доступа, следовательно не рассматриваются. В результате анализа приведенного выше кода получаем матрицы *R* и *W*, описывающие доступ по чтению и записи – они соответственно будут иметь вид:

{R}:

	«F1»	«F2»	«F3»	«F4»
«x»	1	1	0	0
«k»	0	0	0	1
«q»	1	1	1	0

{W}:

	«F1»	«F2»	«F3»	«F4»
«x»	0	0	1	1
«k»	0	0	0	1
«q»	0	0	0	0

Используя выражение (3) получаем для переменной «*x*» (ей соответствует индекс «1») значение функции *E*:

$$E(1)=signum(signum(2)(signum(2)+signum(2-1)))=1$$

т. е. для данной переменной необходимо использовать механизмы синхронизации, для переменной «*k*» (её индекс – 2)значение:

$$E(2)=signum(signum(1)(signum(0)+signum(1-1)))=0,$$

т. е. переменная «*k*» не требует дополнительного кода синхронизации, так как запись в неё осуществляется только из одного потокаи нет читающих потоков. для переменной «*q*» (её индекс – 2) получим значение:

$$E(3) = signum(signum(0)(signum(3) + signum(0-1))) = 0,$$

т. е. переменная «q»также не требует синхронизации.

4. Заключение

Оценка потокобезопасности кода на основе предложенных в главе 2 формулировок может использоваться как самостоятельный этап в составе технологии разработки многопоточных приложений и поможет избежать с одной стороны ошибок совместного доступа, а с другой стороны избыточного применения инструментов синхронизации, снижающего производительность программного продукта. Не менее важным представляется использование предложенных моделей в качестве дополнительного ограничения в составе моделей и методов оптимизации программного кода [4, 5, 6], чувствительных к условиям многопоточности, позволяющего избежать критических ошибок в результате оптимизационных преобразований, меняющих способы размещения переменных в памяти приложения.

Литература

1. *Таненбаум Э., Бос Х.* Современные операционные системы, СПб.: Питер, 2017.

2. *Побегайло А. П.* Системное программирование в Windows. Спб.: БХВ-Петербург, 2017.

3. *Уильямс Энтони.* Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. М.: ДМК-Пресс, 2012.

4. *Гроппен В. О., Томаев М. Х.* Модели, алгоритмы и средства программной поддержки проектирования оптимальных программных продуктов //Автоматика и телемеханика. 2000. № 11. С. 175–184.

5. *Томаев М. Х., Панарин В. Е.* Выбор оптимального класса памяти для данных, используемых в программах на языке «C++». Материалы XIV международной научно-технической конференции / "IT-технологии: развитие и приложения". Владикавказ: СКГМИ (ГТУ), 2013. С. 215.

6. *Томаев М. Х.* Использование оптимизационных моделей «экстремального программирования» в проектировании ПО. Выбор оптимальной стратегии макрозамен. IT-технологии: теория и практика. Материалы семинара. Владикавказ: СКГМИ (ГТУ), 2017. С. 39–55.

EVALUATING THE THREAD SAFETY OF APPLICATIONS SOURCE PROGRAM CODE, RUNNING IN A MULTI-TASK ENVIRONMENT

Tomaev M. K., Alborov M. A.

An approach is proposed for assessing the degree of thread safety of user-defined source software codes of computing systems designed to work on hardware platforms supporting multitasking technologies. The work contains a formal statement of the task of assessing the thread safety of user code, examples of an application proposed in the practice of designing software systems are given. The developed mathematical apparatus allows to systematize approaches to verification of the quality of multi-threaded applications.

Keywords: *program, thread, thread safety, model, quality, synchronization, access, variable, write, read.*

Сведения об авторах

Томаев М. Х.,
к. т. н, доцент кафедры "Автоматизированная обработка информации; программист НИИ теоретической и прикладной информатики. Северо-Кавказский горно-металлургический институт (государственный технологический институт), Владикавказ, Россия,
E-mail: tmxwork@mail.ru

M. K. Tomaev,
Ph.D., assistant professor of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia, e-mail: tmxwork@mail.ru

Алборов М. А.,
бакалавр информатики и вычислительной техники, магистрант кафедры "Автоматизированная обработка информации", программист НИИ теоретической и прикладной информатики. Северо-Кавказский горно-металлургический институт (государственный технологический институт). Владикавказ, Россия.

M. A. Alborov,
Bachelor of Computer Science, Graduate student of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia

УДК 681.343.001

Томаев М. Х., Погребицкий В. В.

ИСПОЛЬЗОВАНИЕ ДИСКРЕТНЫХ ПОДХОДОВ ДЛЯ МИНИМИЗАЦИИ ЧИСЛА ВЫЗОВОВ ФУНКЦИЙ В ПРОГРАММНОМ КОДЕ

Предлагаются дискретные подходы к решению задачи минимизации числа вызовов функций в условиях наличия верхней границы на размер исполняемого кода модифицируемой программы. Рассматриваются два способа формулировки критериев оптимальности: Один из них, основанный на методе эталонов [1], заключается в оценке качества программного продукта по степени удаленности множества его характеристик-координат в многомерном пространстве критериев от идеальной точки, каждая координата которой представляет собой «идеальное» значение одной из характеристик качества разрабатываемого ПО. Другой способ заключается в оценке качества макрозамен без учета структуры программы – критерий в этом случае формулируется в виде суммарного выигрыша от макрозамен, полученного по числу вызовов функций в исходном коде. В работе приводится сравнительный анализ подходов на конкретных примерах. В заключительной части работы формулируются выводы и оценка перспектив продолженных подходов в рамках более общего направления оптимизации программного обеспечения.

Ключевые слова: *программа, оптимизация, модель, качество, эталон, метод, inline, force_inline, макрозамена, память.*

1. Введение

Один из популярных методов экстремального программирования [4] является вставка кода подпрограмм в местах их вызова. В языках низкого уровня, таких как «C» (Си) и «C++», поддерживаются встроенные средства для автоматического преобразования функции в макрос посредством специального ключевого слова «inline». В сравнении с «обычной» функцией использование макроса позволяет получить выигрыш в производительности, который складывается из времени на передачу управления в функцию и возврат из неё. Макрозамена функций является достаточно популярным методом оптимизации [2], однако его использование требует учета важного ограничения: бескон-

трольное применение макрофункций, может привести к чрезмерному росту размера исполняемого кода программы. Как правило, максимально возможный размер исполняемого кода программы выбирается разработчиками исходя из оценки объема оперативной памяти, доступной клиентским вычислительным устройствам. В следующей главе это значение используется в качестве верхней границы оптимизационной модели.

2. Формальная постановка оптимизационных задач

Один из подходов к формулировке задачи поиска оптимальной стратегии макрозамен – это формулировка критерия качества в виде суммарного времени, затрачиваемого программой на вызов функций, минимальное значение которого будет соответствовать оптимуму. Соответствующая задача может быть представлена в виде следующей модели (1):

$$\begin{cases} F_1 = t_{call} \sum_{i=1}^{N} ((1 - z_i) q_i) \to \min; \\ \forall i : \sum_{i=1}^{N} (z_i q_i v_i) \le V; \\ z_k = 1{,}0. \end{cases} \qquad (1)$$

t^{call} – время на вызов каждой функции (на передачу ей управления и возврат из нее);

q_i – количество вызовов i-й функции в исходном коде программы;

z_i – булева переменная, равная 1, если для i-й функции выполняется inline-подстановка и 0 – в противном случае;

v_i – размер i-й функции;

V – верхняя граница выделенного на оптимизацию дополнительного объема памяти.

Модель (1) обеспечивает процесс оптимизации программы, не противоречащий возможностям пользовательской рабочей станции.

Однако её недостатком является игнорирование особенностей функционирования программного алгоритма. В большинстве программных алгоритмов можно выделить конечные участки, переводящие программу из начального состояния в конечное (соответствующее завершению), и участки зацикливания, образованные условными операторами в сочетании с оператором «goto», либо операторами цикла с вычисляемым количеством итераций. В общем случае участки зацикливания имеют более высокий приоритет по сравнению с конечными, так время работы в контурах при отсутствии информации о вероятностях переходов в условных операторах, стремится к бесконечности. В этой связи производительность в каждом из контуров, образованных операторами программы, можно принять как отдельный критерий качества [3]. Объединить их в один «супер»-критерий можно с помощью метода эталонов, суть которого в интерпретации объекта оптимизации как точки в многомерном пространстве, каждое измерение которого – это значение одного из критериев качества. Оптимальное значение переменных соответствует минимальному расстоянию оптимизируемого объекта до эталонного объекта, обладающего «идеальными» характеристиками качества [5]. Применительно к рассматриваемой задаче количество измерений-критериев будет соответствовать числу контуров в программе, значением качества в соответствии с каждым критерием является время, затрачиваемое на вызов функций в каждом контуре, а в качестве эталона примем вариант программы с идеальными (пусть и недостижимыми) характеристиками, равными нулю. Описанный подход используется в формулировке задачи (2):

$$\begin{cases} F_2 = \sqrt{\sum_{k=1}^{M} C_k^2} \to \min; \\ \forall k : C_k = \sum_{i=1}^{N} (n_{ik} z_i); \\ \forall i : \sum_{i=1}^{N} (z_i q_i v_i) \le V; \\ z_k = 1,0, \end{cases} \qquad (2)$$

где по сравнению с моделью (1) введены следующие новые обозначения:

C_k – время, затрачиваемое программой на вызов функций при однократном зацикливании в k-м контуре.

n_{ik} – количество вызовов k-й функции в i-м контуре.

В следующей главе приводятся примеры решения обеих задач.

3. Практическое использование технологии оптимальных «inline»-подстановок

3.1. Решение задачи на основе модели (1)

Следующий листинг содержит код программы, включающий вызовы 5 функций. Дополнительный размер, на который может быть увеличен размер исполняемого кода, не должен превысить 90 байт.

```
#include <iostream>
using namespace std;
int f1(int x, int y);
int f2(int z, int y);
int f3(int x, int y);
int f4(int x, int y);
int f5(int x, int y);

void main()
{
 int x,y;
 cout << "intput x:";
 cin >> x;
 cout << "intput y:";
 cin >> y;
 int z = f1(x,y);
 z = f1(z+x,y) +f2(x,y);
 z += f3(x,y) + f2(z,y);
 z -= f2(z,y) * f2(z+1,y) + f2(z-1,y)*f3(x-1,y-1);
 z += z + f3(z,y)*f4(x,y);
 z += f4(x,y)*f5(z-2,y);
 z += f5(z-3,y);
 z += f5(z-4,y);
```

```
 z += f5(z-5,y);
 cout << z;
}

int f1(int x, int y)
{
  if (y==0) return 0;
  return (x+y) % y;
}

int f2(int z, int y)
{
  return z*((z+y) % y);
}

int f3(int x, int y)
{
  if (x<=0) return 0;
  if (y==0) return 0;
  return x % y;
}

int f4(int x, int y)
{
  if (y==0) return 0;
  if ((x+y)==0) return 0;
  return (x % y)/(x+y);
}

int f5(int x, int y)
{
  return x * x - y;
}
```

Листинг 1. Исходный код пользовательской программы

Оптимальная стратегия макрозамен, минимизирующая суммарное количество вызовов приведена в следующей табл. 1.

Таблица 1

Оптимальная стратегия макрозамен

	z1	**z2**	**z3**	**z4**	**z5**
Оптимальная стратегия макрозамен z[i]	1	1	0	1	0
Используется ли вызов функции	0	0	1	0	1
Число вызовов функции (входные константы)	2	5	3	1	4
Целевая функция	**7**				
Размер каждой функции (входные константы)	20	16	24	32	12
Объем макросов, сгенерированных на основе функции	20	64	0	0	0
Суммарный размер макросов	84				
Ограничение на объем (входная константа) V =	90				

3.2. Решение задачи на основе метода эталонов

В качестве исходного кода для второго примера выбран алгоритм, имеющий сложную структуру, включающую четыре контура: в первом контуре вызываются функции «f1» и «f2»; во втором – «f2», «f3», «f5»; в третьем и четвертом контуре вызываются все, кроме «f1»

```
#include <iostream>
using namespace std;
int f1(int x, int y);
int f2(int z, int y);
int f3(int x, int y);
int f4(int x, int y);
int f5(int x, int y);

void main()
{
 int x,y;
 cout << "intput x:";
 cin >> x;
 cout << "intput y:";
 cin >> y;
 int z = 0;
 m1:
 z +=f1(x,y);
```

```
z = f1(z+x,y) /f2(x,y);
if (z<x*y) goto m1;
z -= f2(z-1,y);
z += f3(x,y) / f2(z-1,y) / f2(z-2,y);
m2:
z -= f2(z-1,y)*f3(x-1,y-1);
z += f3(z,y)*f5(x,y);
if (z>5*x) goto m2;
z += f4(x,y)*f5(z-2,y);
if (z>5*x) goto m2;
z += f5(z-3,y)*f5(z-4,y);
cout << z;
goto m1;
}
```

Листинг 2. Исходный код пользовательской программы

Решение задачи методом эталонов приведено в табл. 2.

Таблица 2

Оптимальная стратегия макрозамен, найденная методом эталонов

	z1	**z2**	**z3**	**z4**	**z5**		
Оптимальная стратегия макрозамен z[i]	**0**	**0**	**1**	**1**	**1**		
Число вызовов функции (входные константы q[i])	2	5	3	1	4		
Число вызовов функции в каждом из контуров (входные константы n[I,k])						C_k	C_k ^2
В контуре 1	2	1	0	0	0	**3**	**9**
В контуре 2	0	1	2	0	1	**1**	**1**
В контуре 3	0	1	2	1	2	**1**	**1**
В контуре 4	0	1	2	1	4	**1**	**1**
Целевая функция	**3,464102**						
Размер каждой функции (входная константа)	20	16	24	32	12		
суммарный объем макросов, сгенерированных на основе каждой функции	**0**	**0**	**48**	**0**	**36**		
Суммарный размер макросов	**84**						
Ограничение на объем (входная константа V)	90						

Для сравнения в табл. 3 приведена оптимальная стратегия для того же исходного кода решением задачи (1).

Таблица 3

Оптимальная стратегия макрозамен (решением задачи (1))

	z1	**z2**		**z3**	**z4**	**z5**
Стратегия макрозамен	1	1		0	1	0
Число вызовов функции (входные константы q[i])	2	5		3	1	4
Целевая функция	7					
Размер каждой функции	20	16		24	32	12
объем макросов, сгенерированных на основе функции	20	64		0	0	0
Суммарный размер макросов	84					
Ограничение на объем (входная константа V)	90					

Как видно из таблиц оптимальная стратегия, полученная методом макрозамен, заключается в макрозамене, в первую очередь, тех функций, которые состоят в наибольшем количестве контуров.

4. Заключение – выводы и перспективы

Основным преимуществом метода эталонов является возможность формулировки взвешенного подхода к оптимизации пользовательского кода, учитывающей качество наиболее важных участков программы. Использование аналогичного подхода в сочетании с другими алгоритмами оптимизации позволит создать новый класс автоматизированных инструментальных средств, улучшающих качество проектируемых приложений на основе объективных критериев качества.

Литература

1. *Гроппен В. О.* Принципы принятия решений с помощью эталонов // Автоматика и телемеханика. 2006. № 4. С. 167–184.

2. *Гроппен В. О.* Принципы оптимизации программного обеспечения ЭВМ. Ростов-н/Д: Изд. Ростовского университета, 1993.

3. *Гроппен В. О., Томаев М. Х.* Модели, алгоритмы и средства программной поддержки проектирования оптимальных программных продуктов //Автоматика и телемеханика. 2000. № 11. С. 175–184.

4. *Томаев М. Х.* Использование оптимизационных моделей «экстремального программирования» в проектировании ПО. Выбор оптимальной стратегии макрозамен. IT-технологии: теория и практика / Материалы семинара. Владикавказ, 2017. С. 39–55.

5. *Будаева. А. А.* Использование метода эталонов для повышения качества группировок в задачах таксономии / Материалы семинара "IT-технологии развитие и приложения". Владикавказ: СКГМИ (ГТУ), 2016.

USING OF DISCRETE APPROACHES TO MINIMIZE THE NUMBER OF FUNCTION CALLS IN THE PROGRAM CODE

Tomaev M. K., Pogrebizkiy V. V.

The article offers discrete approaches to solving the problem of minimizing the number of function calls in the presence of an upper bound on the size of the executable code of the program being modified. Two ways of formulating the criteria for optimality are considered: One of them, based on the method of standards [1], consists in evaluating the quality of a software product by the degree of remoteness of a set of its characteristic coordinates in a multidimensional space of criteria from an ideal point, each coordinate of which is the "ideal" value of one of the quality characteristics of the software being developed. Another way is to assess the quality of macro changes without taking into account the structure of the program - the criterion in this case is formulated in the form of a sum of winnings from macro changes obtained by the number of function calls in the source code.

The paper provides a comparative analysis of approaches based on specific examples. In the final part of the paper conclusions are drawn and an assessment of the perspectives of continued approaches within the framework of a more general direction of software optimization.

Keywords: *program, optimization, model, quality, standard, method, inline, force_inline, macro-replacing, memory*

Сведения об авторах

Томаев М. Х.,
к. т. н, доцент кафедры "Автоматизированная обработка информации; программист НИИ теоретической и прикладной информатики. Северо-Кавказский горно-металлургический институт (государственный технологический институт), Владикавказ, Россия,
E-mail: tmxwork@mail.ru
***M. K. Tomaev*,**
Ph.D., assistant professor of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia, e-mail: tmxwork@mail.ru

Погребицкий В. В.,
бакалавр информатики и вычислительной техники, магистрант кафедры "Автоматизированная обработка информации; программист НИИ теоретической и прикладной информатики. Северо-Кавказский горно-металлургический институт (государственный технологический институт). Владикавказ, Россия.
V. V. Pogrebizkiv,
Bachelor of Computer Science, Graduate student of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia

УДК 004.021

Томаев М. Х., Кусов О. В.

ВЫБОР ОПТИМАЛЬНОЙ МОДУЛЬНОЙ СТРУКТУРЫ ПОЛЬЗОВАТЕЛЬСКИХ ПРОГРАММНЫХ СИСТЕМ

В статье предложены дискретные подходы к формулировке задачи выбора оптимальной модульной структуры программных систем. Предложены два подхода. Основная идея первого – упорядочение участков программы по важности и последовательная их декомпозиция; В основе второго подхода – метода эталонов, предлагается формулировка задачи для циклящихся алгоритмов. В программе приведено описание оптимизационного модуля, разработанного на основе первого подхода. Предлагаются пути и способы развития технологии.

Ключевые слова: *автоматизация, оптимизация, программа, подсистема, модуль, критерий, модель, метод, качество, производительность.*

1. Введение

Задача поиска оптимальной модульной структуры программной системы является одной из моделей, улучшающих качество работы пользовательского приложения с внешней медленной памятью [1]. Затраты программы на обслуживание медленной памяти зависит от числа операций доступа к внешним носителям [2], в том числе, от количества подгружаемых модулей [3]. При анализе эффективности модульного состава больших систем [4] важно также учитывать особенности алгоритма. В настоящей работе предлагается подход, основанный на безусловном приоритете циклящихся участков пользовательских кодов, по сравнению с конечными [5] (приводящими к завершению программы). В следующей главе приведены формулировки задач поиска оптимальной модульной структуры.

2. Формальные постановки задач

2.1. Формальные постановки оптимизационных задач

В сложных программных алгоритмах можно выделить два критерия качества:

1) верхняя граница времени однократного зацикливания;

2) верхняя граница времени поиска решения (времени завершения).

Первая из этих двух характеристик применяется к циклящимся участкам, а вторая – к конечным участкам, переводящим программу из начального состояния в конечное, соответствующее завершению. Оптимальность модульной структуры согласно первому критерию подразумевает разбиение программных единиц, составляющих максимальный контур, на минимальное число подсистем, что минимизирует время работы с внешними носителями. Аналогично оптимальный состав подпрограмм на максимальном конечном участке минимизирует время работы с медленной памятью в соответствии со вторым критерием. При формулировке задач следует учитывать тот факт, что время работы в контурах может стремиться к бесконечности – из этого следует безусловный приоритет первого критерия качества над вторым [6]. Данное положение позволяет выделить два этапа поиска решения оптимального состава подсистем:

На первом минимизируется время работы с внешней памятью в максимальном контуре:

$$\begin{cases} F_1 = \sum\limits_{i \in H(a_{\max})} b_i t_i \to \min; \\ \forall j, \sum\limits_{i \in H(a_{\max})} (b_i r_{ij}) = 1; \\ \max\limits_i (v_i) \le V; \\ b_i = 1{,}0; i = 1,2,.., N, \end{cases} \quad (1)$$

где b_i – булева переменная, равна единице, если i-й вариант подсистемы включен в решение, и нулю – в противном случае. Данная переменная является неизвестной в модели (1);

r_{ij} – это часть входных данных, вспомогательная булева переменная, значение которой равно 1, если i-й вариант подсистемы включает j-й оператор, и нулю – в противном случае;

t_i – время загрузки i-й варианта подсистемы ($t_i > 0$);.

v_i – размер i-го варианта подсистемы;

V – верхняя граница доступной оперативной памяти на клиентском устройстве;

$a_{\max}$ – это контур, соответствующий верхней границе однократного зацикливания;

$H(a_{\max})$ – подмножество индексов таких вариантов подпрограмм, которые включают операторы контура $a_{\max}$.

Неизвестными в задаче (8) являются переменные b_i, значения которых представляют собой оптимальный состав подсистем, минимизирующий время доступа к медленной памяти в максимальном контуре.

На втором этапе выполняется поиск оптимальной модульной структуры участка кода, соответствующего верхней границе времени завершения.

$$\begin{cases} F_1 = \sum\limits_{i \in \{L(0, q_{\max})\}} b_i^{linear} t_i \to \min; \\ \forall i \in H(a_{\max}) : \grave{=} b_i; \\ \forall j, \sum\limits_{i \in H(a_{\max})} (b_i^{linear} r_{ij}) = 1; \\ \max\limits_i (v_i) \le V; \\ b_i^{linear} = 1{,}0; i = 1,2,.., N; \end{cases} \quad (2)$$

где $L(0,q_{\max})$ – это путь из начального состояния программы в конечное, соответствующий верхней границе времени работы программы.

Оптимальная стратегия b_i^{linear} модульной декомпозиции, согласно модели (2), не должна противоречить стратегии b_i, полученной при решении задачи (1). В общем случае решение задач (1), (3) требует использования различного рода комбинаторных [7] алгоритмов, имеющих переборный характер, и соответственно высокую вычислительную сложность.

3. Прикладные результаты

Прикладным результатом исследований стал программный продукт «Оптимальная декомпозиция программных алгоритмов», в котором были реализованы подходы, описанные в предыдущей главе.

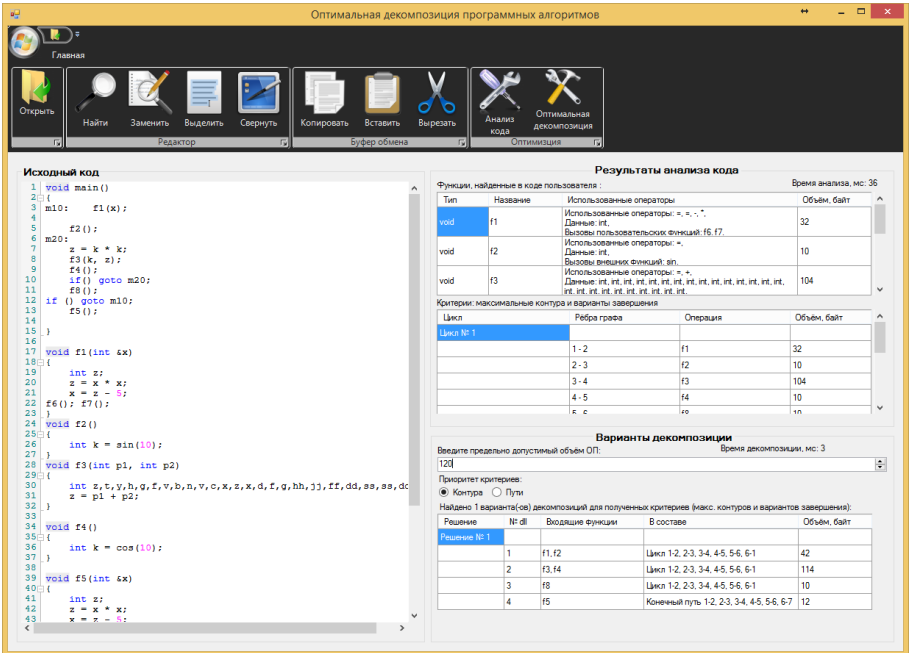

Рис.1. Окно модуля оптимальной декомпозиции

Процесс оптимизации исходного кода включает в себя следующие шаги:

1) загрузка исходного кода;

2) анализ исходного кода – является первым и обязательным этапом оптимизации. Для его запуска необходимо воспользоваться кноп-

кой «Анализ» на панели (в группе «Оптимизации» в правой области). Результаты анализа отражаются в таблицах «список функций» и «Критерии»;

3) вычисление оптимальной стратегии декомпозиции – выполняется с помощью кнопки «Оптимальная декомпозиция». Состав каждого модуля указывается в третьей таблице «Варианты декомпозиции». Полученные результаты являются руководством для разработчика по оптимальному размещению функций в подсистемах (например, в DLL-библиотеках в ОС Windows).

Практическая реализация полученных с помощью разработанного программного продукта рекомендаций зависит от версии ОС и реализованных в ней технологий поддержки модульности. В частности, в ОС Windows декомпозиция исходного варианта кода на DLL-библиотеки будет заключаться в замене вызовов функций, помещенных в библиотеку, обращением к библиотеке с помощью пары API-функций «LoadLibrary» / «FreeLibrary», а также вызовом «GetProcAddress» для получения адреса экспортируемых функций.

4. Результаты

В результате проведенных исследований разработаны модели поиска оптимального функционального состава подсистем, минимизирующего время работы с медленной внешней памятью и соответственно улучшающего производительность пользовательского кода.

Выбранная модель программной реализации – надстройка «Add-In» к среде «Microsoft Visual Studio» – позволила обойти ряд технических вопросов (например, проблема валидации пользовательского кода), не находящихся в центре данного исследования, но являющихся технологически необходимыми. В дальнейшем планируется к реализации ряд инструментов на базе методов так называемого «экстремального программирования» (т. е. таких, которые позволяют улучшить выбранный пользователем критерий качества программы за счет использования дополнительного объема того или иного типа вычислительных ресурсов).

Литература

1. *Рихтер Д., Назар К.* Windows via C/C++. Программирование на языке Visual C++. СПб.: Питер, 2008.

2. *Несвижский В*. Программирование аппаратных средств в Windows. СПб.: BHV-СПб, 2008.

3. *Джонсон Х.* Системное программирование в среде Windows. М.: Вильямс, 2005.

4. *Гроппен В. О.* Принципы оптимизации программного обеспечения ЭВМ. Ростов-н/Д: Изд. Ростовского университета, 1993.

5. *Гроппен В. О., Томаев М. Х.* Модели, алгоритмы и средства программной поддержки проектирования оптимальных программных продуктов //Автоматика и телемеханика. 2000. № 11. С. 175–184.

6. *Гроппен В. О. Мазин В. В*. Эффективная реализация одной стратегии взаимодействия внешней и оперативной памяти ЭВМ // Тезисы докладов XI всесоюзного совещания по проблемам управления. Ташкент, 1989. С. 202.

7. *Томаев М. Х.* Технологии глобальной оптимизации пользовательских программных кодов. Автоматизация и управление в технических системах. 2015. № 3. С. 16-30. DOI: 10.12731/2306-1561-2015-3-2. URL: auts.esrae.ru/15-277.

OPTIMAL MODULAR STRUCTURE SELECTION OF THE USER SOFTWARE SYSTEMS

Tomaev M. K., Kusov O. V.

The article suggests discrete approaches to the formulation of the problem of choosing the optimal modular structure of software systems. Two approaches are proposed: The main idea of the first is the ordering of the program sections by importance and their sequential decomposition; The second approach is based on the method of standards, and the formulation of the problem for cycling algorithms is proposed. The program describes the optimization module developed based on the first approach. The ways and methods of technology development are proposed.

Keywords: *automation, optimization, program, subsystem, module, criterion, model, method, quality, productivity/*

Сведения об авторах

Томаев М. Х.,
к. т. н, доцент кафедры "Автоматизированная обработка информации; программист НИИ теоретической и прикладной информатики. Северо-Кавказский горно-металлургический институт (государственный технологический институт), Владикавказ, Россия,
E-mail: tmxwork@mail.ru

M. K. Tomaev,
Ph.D., assistant professor of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia, e-mail: tmxwork@mail.ru

Кусов О. В.,
бакалавр информатики и вычислительной техники, магистрант кафедры "Автоматизированная обработка информации"; программист НИИ теоретической и прикладной информатики. Северо-Кавказский горно-металлургический институт (государственный технологический университет). Владикавказ, Россия,
O. V. Kusov,
Bachelor of Computer Science, Graduate student of dept. «Automated information processing» North-Caucasian Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia

УДК 004.93

Проскурин А. Е., Уруцкоева М. А.

ОЦЕНКА ДЕФОРМАЦИИ ПОВЕРХНОСТИ РЕЛЬЕФА НА 3D-ИЗОБРАЖЕНИЯХ С ПОМОЩЬЮ МЕТОДА ЭТАЛОНОВ

В статье рассматривается метод измерения деформаций поверхностей твердых тел, представленных их трёхмерными моделями. В качестве инструмента для анализа изменений на поверхности объектов применён метод эталонов. Такой подход значительно упрощает процесс сравнительного анализа, делает его доступнее, нагляднее, быстрее и может быть адаптирован под множество разных задач.

Ключевые слова: *деформация, поверхность, рельеф, метод эталонов.*

1. Введение

С развитием технологий различные отрасли неоднократно нуждались в способах представления множества физических объектов в виде 3D-моделей. Тенденция сохраняется по сей день и получила широкое распространение, что не странно, ведь мы живём в мире, пространство которого описывается тремя измерениями. Трёхмерные модели и их анализ ускорили и качественно улучшили развитие в разных областях науки и промышленности. Сегодня мы получили возможность создавать сколь угодно сложные трёхмерные модели самых разнообразных объектов, и сейчас быстрыми темпами разрабатываются и совершенствуются методы и способы анализа таких моделей в виртуальной среде[1–3].В настоящей статье будет рассмотрена такая технология, как анализ деформации поверхности рельефа объекта с помощью метода эталонов [4; 5].

2. Содержательная постановка задачи

Трехмерное моделирование в наше время заняло свою, довольно большую нишу в сфере цифровых технологий и сегодня уже помогает решать многие задачи в таких областях как наука, образование, строительство и архитектура, медицина, военное дело, производство, ма-

шиностроение и множество других отраслей. Визуализация объектов сейчас уже на высшем уровне, более того, распространение уже получили так называемые 3D-принтеры, и смоделированный объект можно уже распечатать и получить уменьшенную копию, скажем, автомобиля. Визуализируются также и физические, химические процессы в твердых телах, жидкостях, что, несомненно, находит себя в науке и образовании. Нашей целью станет рассмотрение возможности измерения некоторых характеристик исследуемого объекта по его трёхмерной модели, в частности деформации поверхности объекта и ее оценка в соответствии с некими эталонными образцами.

Измерение величины деформации поверхности объекта можно применять в таких отраслях, как геоинформатика – в качестве одной из составляющих в комплексных системах для анализа изменения поверхности рельефа горных территорий; машиностроение и автомобильная индустрия – для отслеживания деформаций металлических каркасов, деталей машин и кузовов автомобилей при внешних механических воздействиях; архитектура и строительство – для оценки изменения рельефа зданий и сооружений при внешних природных воздействиях и в ходе реконструкций. Технологию можно будет применять в самых различных задачах, где требуется с течением времени отслеживать изменения на поверхностях материалов, что тоже уже давно актуально в сфере металлургии, а также фиксировать и измерять деформации рельефа территорий.

Технология предполагает возможность оцифровки данных, описывающих поверхность объекта, что облегчает анализ на ЭВМ и позволяет объективно оценить степень отклонения деформированных поверхностей от нормальных. Это удобно в случаях, когда оценить деформацию объекта наглядно достаточно сложно, и необходимо фиксировать некие эталонные состояния для проведения дальнейшего сравнения состояний трёхмерного объекта до и после его изменения.

Такой подход значительно упрощает процесс сравнительного анализа, делает его доступнее, нагляднее, быстрее и может быть адаптирован под множество разных задач.

3. Описание выбранной технологии

Для упрощения пользования данная технология включает в себя в качестве входных данных несколько фотоснимков объекта с разных ракурсов с пересечением не менее 80 %. По исходным фотоснимкам собирается 3D-модель исследуемого объекта. Исходный объект имеет

некоторые, обозначенные опорные точки. Если поверхность однозначна (рис. 1), т. е. не имеет самопересечений, то опорные точки задаются границами поверхности, если же это полноценная объёмная фигура (рис. 2), то опорные точки на её поверхности определяются алгоритмами распознавания.

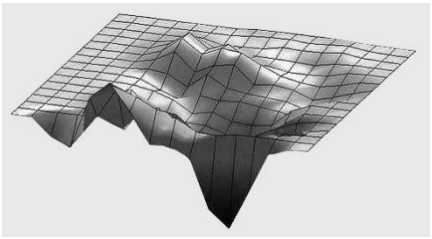

Рис. 1

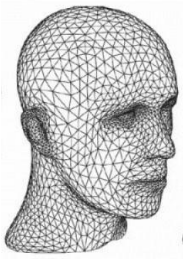

Рис. 2

Выбирается множество точек N на поверхности объекта, заданное количеством граней полигональной сетки объёмной фигуры. У всех деформированных образцов количество граней полигональной сетки должно быть одинаковым и равняться количеству граней в сетке эталонной модели. Таким образом, каждый образец, включая эталонный, описывается облаком из N точек, имеющих свои координаты в трёхмерном пространстве. Поверхность каждой деформированной модели поточечно сравнивается с лучшим эталоном. Из всех деформированных образцов выбирается худший, величина деформации которого является наибольшей.

Сама же величина деформации представляет собой сумму расстояний между точками деформированной поверхности и соответствующими точками поверхности эталонного образца. Расстояние между такими точками вычисляется по формуле:

$$|AB| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}\,, \qquad (1)$$

где A – точка поверхности деформированного образца, а x_A, y_A, z_A – её координаты в пространстве, B – точка поверхности образца-оригинала с координатами x_B, y_B, z_B, которая соответствует точке A в исследуемой модели. Тогда разность между образцами определяется формулой:

$$\sum_{i=1}^{n} |A_i B_i|, \qquad (2)$$

где n – это общее число точек, описывающих каждую трёхмерную модель поверхности.

4. Формальная постановка задачи

Рассматриваемая задача является безусловной, поэтому мы не относим её к задачам минимизации или максимизации, нужно вычислить только определенное значение отклонения. Формальная постановка имеет вид:

$$\begin{cases} D_{max} = \sum_{i=1}^{n} dw_i(a,b), \\ D_c = \frac{1}{D}\sum_{i=1}^{n} d_i(a,c), \\ d_i = \sqrt{(c_{ix} - a_{ix})^2 + \left(c_{iy} - a_{iy}\right)^2 + (c_{iz} - a_{iz})^2}, \\ d_{bi} = \sqrt{(b_{ix} - a_{ix})^2 + \left(b_{iy} - a_{iy}\right)^2 + (b_{iz} - a_{iz})^2}, \end{cases} \quad (3)$$

где $a_{x,y,z}$ – координаты точек эталона, представляющего наилучший образец,

$b_{x,y,z}$ – координаты точек эталона, представляющего наихудший образец,

$c_{x,y,z}$ – координаты точек поверхности исследуемого объекта,

D_{max} – максимальная величина деформации, определяемая у наихудшего эталона относительно наилучшего, берется как верхняя граница при нормировании,

D_c – величина деформации выбранного образца относительно лучшего эталона,

n – общее количество точек в каждом из образцов.

В случае, когда величина деформации D_c равна 0, значит выбранный объект полностью соответствует объекту-эталону A. Если же выбранный объект соответствует эталону B, то величина деформации приравнивается к 1, а значение D_c представляется верхней границей D_{max}. Все исследуемые образцы, значения критерия которых попадают в диапазон $[0, D_{max}]$ нормируются и представляются на шкале деформации от 0 до 1, где единице соответствует значение D_{max}.

5. Пример

Пусть некоторая фигура A в пространстве задается набором точек с координатами x_A, y_A, z_A и является наилучшим эталоном. Также есть фигура B, которая описывается координатами x_B, y_B, z_B, получена путём деформации фигуры A и представляет наихудший эталон. Исследуемая фигура Cтоже задана координатами в пространстве и получена в результате деформации фигуры A. Все фигуры представляют собой

неправильные шестигранники. Точки фигур приведены в соответствие с помощью некоторых маркеров и соединяющих рёбер. Требуется вычислить величину деформации фигуры *C* относительно фигуры *A*. Зададим таблично значения координат точек для всех фигур (Табл. 1–3):

Таблица 1

A		
x	**y**	**z**
1	1	2
4	1	2
4	4	2
1	4	2
1	1	8
4	1	8
4	4	8
1	4	8

Таблица 2

B		
x	**y**	**z**
2	1	2
4	1	2
4	4	5
1	4	2
1	2	8
2	1	8
4	4	8
1	4	6

Таблица 3

C		
x	**y**	**z**
2	1	2
4	1	2
4	4	2
1	4	2
1	2	8
4	1	8
4	4	8
1	4	8

Подсчитав сумму разности координат всех точек между фигурами-эталонами *A* и *B*, получим значение D_{max}, равное 9. Для образца-эталона *A* и исследуемого объекта *C* эта сумма D_c оказалась равна 2. Осуществим нормирование и построим шкалу деформации. Минимальное возможное значение деформации – 0 (деформации нет), максимальное – 9 (деформация объекта соответствует деформации наихудшего образца). После нормирования, мы получили следующую шкалу деформации (рис. 3):

Рис. 3.

Таким образом, величина деформации D_c поверхности исследуемого объекта относительно объекта-оригинала равна 0,22.

6. Выводы

В проделанной работе был рассмотрен и применён метод эталонов в качестве инструмента для оценки деформации поверхностей трёхмерных объектов. Результаты показали, что метод вполне применим к решению подобных задач и может быть внедрён в комплексные системы по анализу рельефов различных объектов и территорий.

7. Литература

1. *Энджел Э.* Интерактивная компьютерная графика. Вводный курс на базе OpenGL. 2-е изд. М.: Вильямс, 2001. 592 с.

2. *Снук Г.* 3D-ландшафты в реальном времени на C++ и DirectX 9. 2-е изд. М.: Кудиц-пресс, 2007. 368 с. ISBN 5-9579-0090-7.

3. *Иванов В. П., Батраков А. С.* Трёхмерная компьютерная графика / Под ред. Г. М. Полищука. М.: Радио и связь, 1995. 224 с. ISBN 5-256-01204-5.

4. *Будаева А. А.* Использование метода эталонов для решения задач дискретной многокритериальной оптимизации // Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. 2015. Т. 15. № 1. С. 22–27.

5. *Гроппен В. О.* Принципы решения многокритериальных задач с помощью эталонов. Труды XII Всероссийской научно-методической конференции Телематика, Санкт Петербург, 6–9 июня 2005. Том 1. Стр. 125–128.

ESTIMATION OF THE DEFORMATION OF THE RELIEF SURFACE ON 3D IMAGES WITH THE METHOD OF STANDARDS

Proskurin A. E.,Uruzkoeva M. A.

The articleexplains the deformations measuring method of the solids surfaces, represented by their three-dimensional. The method of standards is applied as a tool for analyzing changes on the surface of objects. This approach greatly simplifies the process of comparative analysis, makes it more accessible, more visible, faster and can be adapted to a variety of different tasks.

Keywords: *deformation, surface, relief, method of standards.*

Сведения об авторах

Проскурин А. Е.,
к.т.н, доцент кафедры "Автоматизированная обработка информации", программист НИИ теоретической и прикладной информатики, Северо-Кавказский горно-металлургический институт (государственный технологический университет). Владикавказ, Россия.
E-mail: alexander@skgmi-gtu.ru

A. E. Proskurin,
Ph.D., assistant professor of dept. «Automated information processing» *North-Caucasian* Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia, e-mail: alexander@skgmi-gtu.ru

Уруцкоева М. А.,
бакалавр информатики и вычислительной техники, магистрант кафедры "Автоматизированная обработка информации", программист НИИ теоретической и прикладной информатики. Северо-Кавказский горно-металлургический институт (государственный технологический университет). Владикавказ, Россия.
E-mail: madina-urutskoeva@mail.ru

M. A. Urutskoeva,
Bachelor of Computer Science, Graduate student of dept. «Automated information processing» North-Caucasian Institute of Mining and Metallurgy (State Technology University). Vladikavkaz, Russia,
e-mail: madina-urutskoeva@mail.ru

СОДЕРЖАНИЕ

НОВЫЕ ТЕХНОЛОГИИ ПРИНЯТИЯ РЕШЕНИЙ

ТЕХНОЛОГИИ ОПТИМИЗАЦИИ ПОЛЬЗОВАТЕЛЬСКИХ КОДОВ

ТЕХНОЛОГИИ ОБРАБОТКИ ИЗОБРАЖЕНИЙ