АЛГОРИТМЫ ПОЛНОГО ПЕРЕБОРА В СХЕМАХ ВЕТВЛЕНИЯ РЕШЕНИЯ КОМБИНАТОРНЫХ ЗАДАЧ С ПСЕВДОБУЛЕВЫМИ ФУНКЦИЯМИ

Трофимчук А., Васянин В., Ушакова Л.

В статье приводятся полезные сведения для разработчиков алгоритмов и программ об использовании кодов Грея для решения комбинаторных задач с псевдобулевыми функциями. В качестве примера эффективности применения этих кодов рассматривается решение двух комбинаторных задач с булевыми переменными с полным перебором вариантов решения. Представлены результаты экспериментального исследования, которые показывают, что коды Грея можно практически применять в схемах ветвления, например, в методе ветвей и границ, когда количество переменных в узлах ветвления решающего алгоритма не превышает 35.

Ключевые слова. Коды Грея, задачи комбинаторной оптимизации, время решения задачи

ВВЕДЕНИЕ

В статье рассматривается применение двоично-отраженных кодов Грея для решения комбинаторных задач с псевдобулевими функциями (полиномами от булевых переменных). Приводится рекурсивный алгоритм Эрлиха для генерации последовательности строк n - разрядных кодов Грея, в которой каждая следующая строка отличается от предыдущей только одним разрядом (битом). На примерах решения 0-1 задачи о ранце (0-1 Knapsack Problems) [1] и задачи выбора пропускных способностей дуг коммуникационной сети с ограничением на время задержки потоков [2] показано, как эти коды можно использовать для эффективного вычисления значений целевой функции и ограничений.

Цель статьи состоит в том, чтобы показать разработчикам алгоритмов и программ как можно применять коды Грея в различных схемах разветвления решающего алгоритма, например, в методе ветвей и границ, когда количество двоичных (булевых) переменных в узлах дерева ветвления не больше 35.

Методика исследований основана на проведении вычислительного эксперимента решения вышеуказанных задач предложенным алгоритмом перебора вариантов решений с частичным и полным пересчетом значений целевой функции и ограничений.

В 1953 г. физик Фрэнк Грей (Frank Gray) получил патент на изобретение двоично-отраженных n - разрядных кодов, которые и были названы его именем [3]. Изначально эти коды применялись в кодово-импульсной модуляции для управления различными электромеханическими переключателями и методе аналоговой передачи цифровых сигналов. В настоящее время коды Грея используются для выявления и исправления ошибок в системах связи, управления различными цифровыми датчиками, кодирования номеров дорожек в жестких накопителях компьютеров и пр. Кроме того известно о применении кодов Грея для решения комбинаторных задач «Ханойская башня» и «Китайские кольца» [4]. Подробнее о кодах Грея можно узнать в книге Д. Кнута [5].

ЗАДАЧА О РАНЦЕ И ЗАДАЧА ВЫБОРА ПРОПУСКНЫХ СПОСОБНОСТЕЙ ДУГ

Математическая формулировка первой задачи заключается в следующем. Задан набор n предметов, для каждого из которых известна стоимость $c_i \in Z^+$ и вес $a_i \in Z^+$, $i = \overline{1,n}$. Требуется так загрузить ранец предметами, чтобы его суммарная стоимость была максимальной

$$\max \sum_{i=1}^{n} c_i x_i \tag{1}$$

и выполнялось ограничение на суммарный размер ранца $W \in Z^+$

$$\sum_{i=1}^{n} a_i x_i \le W, \tag{2}$$

$$x_i \in \{0,1\}, i = \overline{1,n}. \tag{3}$$

Предполагается, что $a_i \le W$, $i = \overline{1,n}$ и $\sum_{i=1}^{n} a_i > W$.

Вторая задача заключается в выборе пропускных способностей дуг из заданного набора дискретных целочисленных значений при ограничении на максимальное время задержки потоков, которая актуальна при распределении потоков в многопродуктовых коммуникационных сетях. В этой задаче задержки потоков t_{kl} на дугах определяются как $t_{kl} = f_{kl} / (w_{kl} - f_{kl})$, $kl \in E$, а ограничение на время t_{av} задержки потоков в сети имеет вид $t_{av} = 1/U_{\Sigma} \sum_{kl \in E} f_{kl} / (w_{kl} - f_{kl}) \le T_{\max}$. Здесь $f_{kl} \in Z^+$ - фиксированное значение потока по дуге $kl \in E$, E - множество дуг сети, $w_{kl} \in Z^+$ - пропускная способность дуги $kl \in E$,

$T_{\max}$ — максимальное время задержки потоков в сети, $U_{\Sigma} = \sum_{ij \in S} u_{ij}$ - суммарный поток в сети, $u_{ij} \in Z^{+}$ - величина потока из узла i в узел j, S - множество пар индексов корреспондирующихся узлов в сети. При приближении величины потока на дугах к их пропускным способностям задержки увеличиваются и, следовательно, могут возникать перегрузки в сети. Суть задачи заключается в том, что при фиксированных потоках требуется так выбрать пропускные способности дуг из заданного набора целых чисел, чтобы выполнялось ограничение на время задержки потоков, и достигался минимум некоторой целевой функции. Такая задача возникает не только в сетях передачи данных, но и в транспортных сетях при распределении потоков по критерию минимума стоимости сети и заданном ограничении на время задержки потоков [6-8]. Управляя параметром $T_{\max}$ для максимальной задержки, администратор сети передачи данных или диспетчер транспортной сети может обеспечить требуемый ему резерв пропускной способности каналов связи или грузоподъемности транспортных средств при прогнозируемых колебаниях величины потоков на заданных промежутках времени. Уменьшение параметра $T_{\max}$ (увеличение резерва) приводит к удорожанию сети, но уменьшает вероятность перераспределения потоков и технического переоснащения каналов связи или парка транспортных средств при увеличении потоков и угрозе возникновения перегрузок в сети. Увеличение параметра $T_{\max}$ дает возможность уменьшить пропускную способность каналов связи или грузоподъемность транспортных средств и стоимость сети, но увеличивает риск перераспределения потоков и вероятность модернизации сети.

Математическая модель задачи формулируется следующим образом. Пусть задана связная ориентированная сеть $G(N, E)$ с множеством узлов N, $n = \left| N \right|$ и множеством дуг E, $e = \left| E \right|$, где n и e соответственно количество узлов и дуг сети. Будем считать, что сеть такова, что для каждой прямой дуги kl, ($k < l$) существует обратная lk, ($l > k$). Для сети передачи данных дуга представляет коммутированную линию связи, состоящую из одного или пучка элементарных каналов. Для транспортной сети дуга отождествляется с маршрутом транспортного средства, концевые узлы которого совпадают с начальным и конечным узлами дуги. Сеть может содержать петли и параллельные дуги, поскольку допускаются циклические и повторяющиеся маршруты, а также маршруты с одинаковыми концевыми узлами. На сети задана целочисленная матрица потоков

$U = \| u_{ij} \|_{n \times n}$, где u_{ij} - величина потока (количество единиц потока) из узла i в узел j в некоторых транспортных блоках заданного размера. Пусть w_{kl}, $kl \in E$ - искомые пропускные способности дуг сети в транспортных блоках, $w_{kl} \in \{w_1, w_2, ..., w_\alpha\}$, w_i, $i = \overline{1, \alpha}$ - упорядоченные по возрастанию целые положительные числа; d_{kl}, $kl \in E$ - длины дуг; $C_{kl}(w_{kl}, d_{kl}) \in R^+$, $kl \in E$ - дискретные стоимости дуг, такие, что $C_{kl}(w_i, d_{kl}) \le C_{kl}(w_{i+1}, d_{kl})$, $i = \overline{1, \alpha - 1}$; $f_{kl} = \sum_{ij \in S} u_{ij}^{kl}$, $kl \in E$ - фиксированные суммарные потоки в транспортных блоках, протекающие по дугам сети, где u_{ij}^{kl} - поток транспортных блоков из i в j, проходящий по дуге kl.

Требуется найти минимальное значение функции стоимости сети

$$\min_{w_{kl}} \sum_{kl \in E} C_{kl}(w_{kl}, d_{kl}), \; w_{kl} \in \{w_1, w_2, ..., w_\alpha\} \tag{4}$$

при ограничении:

$$\frac{1}{U_\Sigma} \sum_{kl \in E} \frac{f_{kl}}{w_{kl} - f_{kl}} \le T_{\max}, \; w_{kl} > f_{kl}, \; \forall kl \in E. \tag{5}$$

Задачу (4), (5) представим в виде задачи о ранце с булевыми переменными и мультивыбором (0-1 Multiple-choice Knapsack Problem, 0-1 MCKP). Пусть $c_{ij} \in R^+$ - дискретные стоимости дуг i с пропускной способностью $w_{ij} \in \{w_1, w_2, ..., w_\alpha\} \in Z^+$ и длиной d_i, $j = \overline{1, \alpha}$, $i = \overline{1, e}$; $t_{ij} = f_i / (w_{ij} - f_i)$, $w_{ij} > f_i$, $j = \overline{1, \alpha}$, $i = \overline{1, e}$ - задержки потоков на дугах; f_i - поток по дуге i, $i = \overline{1, e}$. Положим $x_{ij} = 1$, если для дуги i выбрана пропускная способность w_{ij}, $j = \overline{1, \alpha}$, $i = \overline{1, e}$ и $x_{ij} = 0$ в противном случае.

Требуется найти

$$\min \sum_{i=1}^{e} \sum_{j=1}^{\alpha} c_{ij} x_{ij} \tag{6}$$

при ограничениях

$$\frac{1}{U_\Sigma} \sum_{i=1}^{e} \sum_{j=1}^{\alpha} t_{ij} x_{ij} \le T_{\max}, \tag{7}$$

$$\sum_{j=1}^{\alpha} x_{ij} = 1, \; i = \overline{1, e}, \tag{8}$$

$$x_{ij} \in \{0,1\}. \quad (9)$$

Здесь искомые пропускные способности w_{ij} соответствуют x_{ij}^* - оптимальному решению задачи (6)-(9).

Нетрудно видеть, что любую индивидуальную задачу, сформулированную в виде (4), (5), можно за время $O(e\alpha)$ преобразовать в соответствующий экземпляр задачи (6)-(9). Для этого необходимо построить две матрицы размером $e \times \alpha$, строки которых соответствуют дугам, столбцы — набору дискретных пропускных способностей, а в качестве элементов матриц принимаются стоимости дуг c_{ij} и задержки на дугах t_{ij}. Справедливо и обратное преобразование. В работе [2] доказано, что оптимизационная задача, сформулированная в виде (4), (5) является NP-трудной, и для ее решения предложено два приближенных алгоритма, основанных на аппроксимации дискретных функций стоимости линейными, и на методе последовательного анализа вариантов. Было показано, что оба алгоритма на заключительной стадии работы сужают область допустимых решений до двух значений $w_{ij} \in \{w_j, w_{j+1}\}$, $j = \overline{k, \alpha - 1}$, $i = \overline{1, e}$, и при полном переборе 2^e вариантов позволяют получить точное решение. Запишем сокращенную задачу в виде

$$\min \sum_{i=1}^{e} \sum_{j=1}^{2} c_{ij} x_{ij} \quad (10)$$

при ограничениях

$$\frac{1}{U_\Sigma} \sum_{i=1}^{e} \sum_{j=1}^{2} t_{ij} x_{ij} \le T_{\max}, \quad (11)$$

$$\sum_{j=1}^{2} x_{ij} = 1, \; i = \overline{1, e}, \quad (12)$$

$$x_{ij} \in \{0,1\}. \quad (13)$$

где c_{ij}, $j = \overline{1,2}$, $i = \overline{1,e}$ - дискретные стоимости дуг; $t_{ij} = f_i / (w_{ij} - f_i)$, $j = \overline{1,2}$, $i = \overline{1,e}$ - задержки потоков на дугах; f_i - поток по дуге i, $i = \overline{1,e}$.

Задачу (10) - (13) можно преобразовать в 0-1 задачу о ранце:

$$\min \sum_{i=1}^{e} (c_{i1} + \Delta c_i x_i) \quad (14)$$

при ограничениях

$$\frac{1}{U_\Sigma}\sum_{i=1}^{e}(t_{i1}-\Delta t_i x_i)\le T_{\max}, \qquad (15)$$

$$x_i\in\{0,1\},\ i=\overline{1,e}, \qquad (16)$$

где $\Delta c_i=c_{i2}-c_{i1}$, $t_{i1}=f_i/(w_{i1}-f_i)$, $\Delta t_i=f_i/(w_{i1}-f_i)-f_i/(w_{i2}-f_i)$, $i=\overline{1,e}$, $\frac{1}{U_\Sigma}\sum_{i=1}^{e}t_{i1}>T_{\max}$, $\frac{1}{U_\Sigma}\sum_{i=1}^{e}t_{i2}\le T_{\max}$.

Все сформулированные выше задачи являются NP-трудными, т.е. в общем случае для их решения не существует точных полиномиальных алгоритмов [9]. В книгах [10, 11] можно найти описание трех точных псевдополиномиальных алгоритмов решения задач (1)-(3) и (14)-(16), основанных на методах ветвей и границ и динамического программирования с использованием Лагранжевой и LP релаксации. Листинги программ этих алгоритмов (expknap, minknap и combo) на C++ приведены на странице D. Pisinger в Интернете (www.diku.dk/~pisinger/). В этих же книгах приведены также полностью полиномиальные приближенные схемы решения задачи 0-1 KP (FPTAS — Fully Polynomial Time Approximation Scheme) — алгоритмы, которые за полиномиальное время от размера входа задач и $1/\varepsilon$, позволяют получить $(1-\varepsilon)$ для задачи на максимум или $(1+\varepsilon)$ для задачи на минимум гарантированное приближенное решение, где ε — сколь угодно малое положительное число.

АЛГОРИТМ ПОЛНОГО ПЕРЕБОРА НА ОСНОВЕ КОДОВ ГРЕЯ

Для полного перебора вариантов решения задач (1)-(3) и (14)-(16) будем использовать алгоритм генерации последовательности двоично-отраженных n-разрядных кодов Грея, предложенный Эрлихом [12]. Алгоритм дает возможность в процессе решения эффективно вычислять значения целевой функции (1) или (14) и ограничения (2) или (15). Двоично-отраженный (зеркальный, рефлексивный) код Грея определяется по следующим рекурсивным правилам:

$B_0=$", $B_{n+1}=0B_n1B_n^r$, $n=0,1,2,3,\ldots$, (разряды добавляются слева) или

$B_0=$", $B_{n+1}=B_n0B_n^r1$, $n=0,1,2,3,\ldots$, (разряды добавляются справа).

где $B_0=$" — пустая строка, B_n — бинарная последовательность Грея, состоящая из n - битовых строк, $0B_n$ и B_n0 — последовательность B_n с префиксом 0 в начале и конце каждой строки, $1B_n^r$ и B_n^r1 — последовательность B_n в обратном порядке с префиксом 1 в начале и

конце каждой строки. Поскольку последняя строка в B_n эквивалентна первой строке в B_n^r, то ясно, что на каждом шаге B_{n+1} изменяется ровно один бит, если B_n обладает тем же свойством. С каждым шагом длина строк увеличивается на 1, а их количество — вдвое. Таким образом n-разрядный код Грея есть упорядоченная циклическая последовательность 2^n n-разрядных строк, в которой последовательные строки различаются только одним битом (разрядом). В алгоритмах полного перебора для вычисления значений различных функций с помощью кодов Грея удобно эти коды представлять упорядоченным списком номеров разрядов, изменяющих свое значение на противоположное при переходе от текущей строки к следующей. Такую последовательность переходов P_n можно определить по следующим рекурсивным правилам: $P_1 = 1$, $P_n = P_{n-1}, n, P_{n-1}$, $n = 2,3,4,\ldots$. Длина последовательности P_n равна $2^n - 1$, а нумерацию разрядов в последовательности можно выполнять справа налево или наоборот. Например, для $n = 4$ и начальной строки 0000 $P_4 = 1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,$ а ее длина равна $2^4 - 1 = 15$. Соответствующая последовательность бинарных строк при нумерации разрядов слева направо имеет вид: 0000, 1000, 1100, 0100, 0110, 1110, 1010, 0010, 0011,1011, 1111, 0111,0101, 1101, 1001, 0001. При нумерации разрядов справа налево получим перевернутую последовательность, соответствующую первому рекурсивному определению B_{n+1}. Интересно заметить, что если построить граф, вершины которого соответствуют бинарным последовательностям длины n, а ребра соединяют две вершины отличающиеся только одним разрядом, то такой граф представляет двоичный n-мерный куб. При этом построенная бинарная последовательность соответствует гамильтонову пути в таком графе.

Для генерации последовательности переходов P_n в бинарной строке $B = \| b_i \|$, $i = \overline{1, n+1}$, определим вектор указателей $P = \| p_i \|$, $i = \overline{1, n+2}$, который имитирует стек для рекурсивного определения P_n. Оптимальное решение x_i^*, $i = \overline{1, n}$ будем сохранять в векторе $B^{opt} = \| b_i^{opt} \|$, $i = \overline{1, n+1}$, где: если $b_i^{opt} = 0$, то $x_i^* = 0$; если $b_i^{opt} = 1$, то $x_i^* = 1$; $i = \overline{1, n}$. Размерности векторов P, B и B^{opt} увеличены соответственно на две и одну единицу из-за специфики работы алгоритма. Знак « $\leftarrow$ » означает операцию присваивания.

Алгоритм OPT1 с частичным пересчетом целевой функции и ограничения для задачи (1)-(3)

1. $CSUM \leftarrow c_1$; $ASUM \leftarrow a_1$; $CSUMOPT \leftarrow 0$; $B \leftarrow 0$; $B^{opt} \leftarrow 0$.

2. Для $\{ i \mid i = \overline{1, n+2} \}$ выполнить $p_i \leftarrow i$.

3. $i \leftarrow 1$.

4. Пока $i < n+1$ выполнять шаги 5-7.

5. Если $b_i = 0$, то $CSUM \leftarrow CSUM - c_i$; $ASUM \leftarrow ASUM - a_i$, иначе $CSUM \leftarrow CSUM + c_i$; $ASUM \leftarrow ASUM + a_i$.

6. Если $ASUM \le W$, то если $CSUM > CSUMOPT$ выполнить: $CSUMOPT \leftarrow CSUM$; $B^{opt} \leftarrow B$; $ASUMOPT \leftarrow ASUM$.

7. $i \leftarrow p_1$; $b_i \leftarrow 1 - b_i$; $p_1 \leftarrow 1$; $p_i \leftarrow p_{i+1}$; $p_{i+1} \leftarrow i+1$.

8. Конец алгоритма. Вывести значения: $\max \sum_{i=1}^{n} c_i x_i^* = CSUMOPT$; $\sum_{i=1}^{n} a_i x_i^* = ASUMOPT$; W; B^{opt}.

В варианте решения задачи с полным пересчетом целевой функции и ограничения (**алгоритм OPT2**) шаг 1 заменится на

1. $CSUMOPT \leftarrow 0$; $B \leftarrow 0$; $B^{opt} \leftarrow 0$,

а шаг 5 заменится на

5. $CSUM \leftarrow 0$; $ASUM \leftarrow 0$. Для $\{ j \mid j = \overline{1, n} \}$ выполнить $CSUM \leftarrow CSUM + c_j * b_j$; $ASUM \leftarrow ASUM + a_j * b_j$.

Для алгоритма с частичным пересчетом целевой функции и ограничения для задачи (14)-(16) псевдокод будет таким:

1. $T_\Sigma \leftarrow 0.0$; $C_\Sigma \leftarrow 0.0$; $D_\Sigma \leftarrow 0.0$.

2. Для $\{ i \mid i = \overline{1, e} \}$ выполнить: $T_\Sigma \leftarrow T_\Sigma + t_{i1}$; $C_\Sigma \leftarrow C_\Sigma + c_{i1}$; $D_\Sigma \leftarrow D_\Sigma + \Delta c_i$.

3. $D_\Sigma \leftarrow C_\Sigma + D_\Sigma$; $T_s \leftarrow T_{\max} \times U_\Sigma$; $B \leftarrow 0$; $B^{opt} \leftarrow 0$; $C_\Sigma \leftarrow C_\Sigma + \Delta c_1$; $T_\Sigma \leftarrow T_\Sigma - \Delta t_1$.

4. Для $\{ i \mid i = \overline{1, e+2} \}$ выполнить $p_i \leftarrow i$.

5. Пока $i < e+1$ выполнить пп. 6-8.

6. Если $b_i = 0$, то $C_\Sigma \leftarrow C_\Sigma - \Delta c_i$; $T_\Sigma \leftarrow T_\Sigma + \Delta t_i$, иначе $C_\Sigma \leftarrow C_\Sigma + \Delta c_i$; $T_\Sigma \leftarrow T_\Sigma - \Delta t_i$.

7. Если $T_\Sigma \le T_s$, то если $C_\Sigma < D_\Sigma$ выполнить: $D_\Sigma \leftarrow C_\Sigma$; $B^{opt} \leftarrow B$; $t_{av} \leftarrow T_\Sigma / U_\Sigma$.

8. $i \leftarrow p_1$; $b_i \leftarrow 1 - b_i$; $p_1 \leftarrow 1$; $p_i \leftarrow p_{i+1}$; $p_{i+1} \leftarrow i+1$.

9. Конец алгоритма. Вывести значения: min $C_\Sigma = D_\Sigma$; t_{av}; B^{opt}.

ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ АЛГОРИТМОВ OPT1 И OPT2

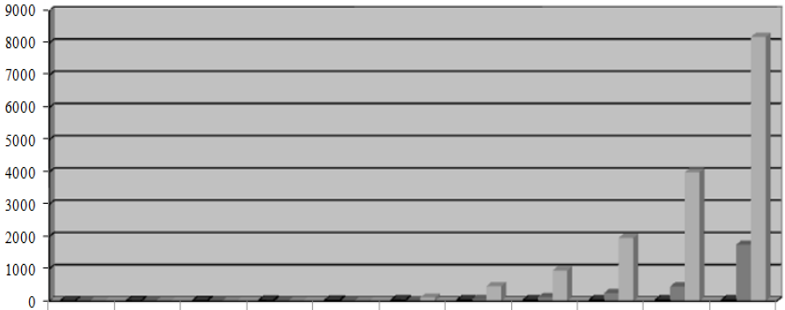
Сравнение быстродействия алгоритмов OPT1 и OPT2 проводилось для задач (1) - (3) и (14) - (16) на примерах, сгенерированных датчиком псевдослучайных чисел. Здесь приводятся результаты решения только для задачи (1)-(3), так как для задачи (14)-(16) были получены аналогичные по характеристикам данные. Для всех размерностей задачи (1) - (3), которая изменялась от $n = 5$ до $n = 36$, стоимости предметов c_i и их веса a_i генерировались в диапазоне от 5 до 10 и от 1 до 20 соответственно. При проведении эксперимента проверялась также точность решения задачи (1) - (3) «жадным» эвристическим алгоритмом OPT3 с временной сложностью $O(n \log n)$. Алгоритм основан на предварительном упорядочении предметов в заданном наборе по невозрастанию их удельных стоимостей c_i / a_i, $i = \overline{1,n}$ с последующим выбором предметов в ранец до тех пор, пока выполняется ограничение на размер ранца W. Хорошо известно, см. например, [11,13], что решения OPT3 = $\max \sum_{i=1}^{n} c_i x_i^*$, полученные «жадным» алгоритмом, могут отличаться от оптимальных не более чем в два раза, при выборе окончательной стоимости ранца из условия $\max \sum_{i=1}^{n} c_i x_i^* = \{\max \sum_{i=1}^{n} c_i x_i^*, \max c_i\}$, где $\max c_i$, $i = \overline{1,n}$ — максимальная стоимость предмета в заданном наборе.

На рис. 1 показано время решения задачи (1) - (3) (с точностью до двух знаков после запятой) на ПК с тактовой частотой 2.66 GHz для алгоритмов OPT1 и OPT2 с частичным и полным пересчетом целевой функции и ограничения. Как видно из рис. 1, алгоритм OPT1 может применяться для практических расчетов в схемах ветвления, когда количество переменных в узлах дерева ветвления не превышает 35 (для $n = 35$ время счета около 8 минут). Алгоритм OPT1 для вариантов 5-10 быстрее алгоритма OPT2 в среднем в 7 раз.

На рис. 2 приведены результаты решения задачи (1) - (3) точным алгоритмом OPT1 и «жадным» алгоритмом OPT3 (результаты решений алгоритмов OPT1 и OPT2 естественно совпадают).

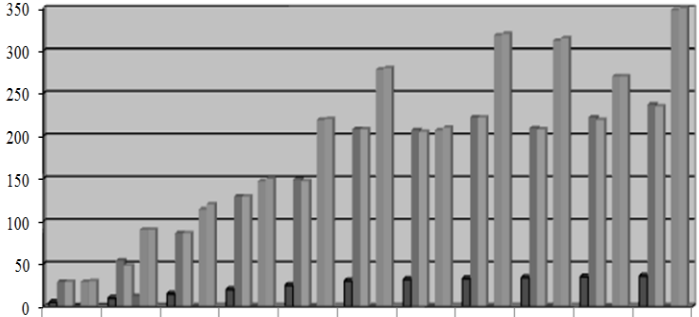
Значения $\sum_{i=1}^{n} a_i x_i^* = ASUMOPT$ показаны для алгоритма OPT1. Значения целевой функции, полученные «жадным» алгоритмом, отличаются от оптимальных в пределах DEL% от 0 до 12,5 %. Алгоритм OPT3 имеет полиномиальную оценку временной сложности и его можно применять на практике для решения задачи (1) - (3) большой размерности, когда лицу принимающему решение необходимо

достаточно быстро получить приближенное значение целевой функции при ограниченных вычислительных ресурсах.



	1	2	3	4	5	6	7	8	9	10	11
n	5	10	15	20	25	30	32	33	34	35	36
T1	0	0	0	0,02	0,66	14,9	60,89	120,32	248,06	455,18	1751
T2	0	0	0	0,09	4,37	107,84	462,45	950,97	1964,46	3994,53	8191,27

Рис.1 – Время решения задачи в сек. С частичным (T1, алгоритм OPT1) и полным (T2, алгоритм OPT2) пересчетом целевой функции и ограничений



	1	2	3	4	5	6	7	8	9	10	11
n	5	10	15	20	25	30	32	33	34	35	36
OPT1	29	54	86	129	149	208	207	222	209	222	237
OPT3	29	48	86	129	147	208	205	222	208	219	235
DEL%	0	12,5	0	0	1,36	0	0,98	0	0,48	1,37	0,85
ASUMOPT	29	90	114	147	219	278	207	318	312	270	348
W	30	90	120	150	220	280	210	320	315	270	350

Рис.2 – Результаты оптимального (OPT1) и приближенного (OPT3) решения задачи

Так, например, задача (1)-(3) была решена для $n = 10000$ с теми же границами изменения значений c_i и a_i за 0,11 сек. При этом OPT3 = 70598, ASUMOPT = 103985 при W = 104000.

Все программы написаны на языке Фортран в среде Microsoft Developer Visual Studio и могут быть адаптированы для работы в системе параллельного программирования Intel® Parallel Studio XE 2018, в которую вошли последние версии компиляторов C/C++ и Фортран (https://software.intel.com/ru-ru/try-buy-tools).

ВЫВОДЫ

В статье показано, как можно применять на практике двоично-отраженные коды Грея для решения комбинаторных задач с псевдобулевыми функциями, когда количество переменных в узлах дерева ветвления решающего алгоритма не превышает 35. На примере решения двух комбинаторных задач с булевыми переменными продемонстрирована вычислительная эффективность предложенного алгоритма полного перебора решений с ограниченным пересчетом значений целевой функции и ограничений, и он может применяться на практике в различных схемах разветвления решающего алгоритма.

СПИСОК ЛИТЕРАТУРЫ

[1] Васянин В.А., Ушакова Л.П. Коды Грея в задачах комбинаторной оптимизации // Математичне моделювання в економіці, 2019. – № 1-2. – С. 63 – 69.

[2] Trofymchuk O.M., Vasyanin V.A. Choosing the Capacity of Arcs with Constraint on Flow Delay Time // Cybernetics and Systems Analysis, 2019. – Vol. – 55. – Issue 4. – P. 561 –569. DOI: https://doi.org/10.1007/s10559-019-00165-0

[3] Gray F. Pulse code communication // U.S. Patent 2632058, March 17, 1953.

[4] Гарднер М. Математические головоломки и развлечения: 2-е изд., испр. и дополн. / Пер. с англ. – М.: «Мир», 1999. – 447 с.

[5] Knuth D.E. The Art of Computer Programming. Volume 4A / Combinatorial Algorithms, Part 1. - Addison Wesley Longman, Inc., 2011. – 933 p.

[6] Trofymchuk O.M., Vasyanin V.A. Simulation of Packing, Distribution and Routing of Small-Size Discrete Flows in a Multicommodity Network // Journal of Automation and Information Sciences, 2015. – Vol. – 47. – Issue 7. – P. 15 – 30.

[7] Vasyanin V.A. Problem of Distribution and Routing of Transport Blocks with Mixed Attachments and Its Decomposition // Journal of Automation and Information Sciences, 2015. – Vol. – 47. – Issue 2. – P. 56 – 69.

[8] Васянин В.А., Трофимчук А.Н., Ушакова Л.П. Экономико-математические модели задачи распределения потоков в многопродуктовой коммуникационной сети // Математичне моделювання в економіці, 2016. – № 2. – С. 5 – 21.
[9] Garey M.R., Johnson D.S. Computers and Intractability: A Guide to the Theory of NP-Completeness. - W. H. Freeman & Co. New York, NY, USA, 1979. – 338 p.
[10] Martelo S., Toth P. Knapsack problems: algorithms and computer implementations. - Great Britain: Wiley, 1990. – 296 p.
[11] Kellerer H., Pferschy U., Pisinger D. Knapsack Problems. - Springer-Verlag Berlin Heidelberg, 2004. – 548 p.
[12] Bitner J.R., Ehrlich G., Reingold E.M. Efficient Generation of the Binary Reflected Gray Code and its Applications // Comm. ACM, 1976. – 19. –P. 517 – 521.
[13] Кузюрин Н.Н., Фомин С.А. Эффективные алгоритмы и сложность вычислений. - М: МФТИ, 2008. – 326 с.

FULL SEARCH ALGORITHMS IN THE SCHEMES OF BRANCHING OF COMBINATORIAL PROBLEMS WITH PSEUDOBOOLEAN FUNCTIONS

Trofymchuk O., Vasyanin V., Ushakova L.

The article provides useful information for developers of algorithms and programs on the use of Gray codes for solving combinatorial problems with pseudoBoolean functions. As an example of the effectiveness of the use of these codes, the solution on two combinatorial problems with Boolean variables with a full search of the solutions is considered. The results of an experimental study are presented, which show that Gray codes can be practically applied in branching schemes, for example, in the branch and bound method, when the number of variables in the branch nodes of the decision algorithm does not exceed 35.

Keywords. Gray codes, combinatorial optimization problems, problem solving time