УДК 519.168

https://orcid.org/0000-0003-4046-5243
https://orcid.org/0000-0002-9020-1329

В.А. ВАСЯНИН, Л.П. УШАКОВА

КОДЫ ГРЕЯ В ЗАДАЧАХ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ

Аннотация. *В статье приводятся полезные сведения для разработчиков алгоритмов и программ об использовании кодов Грея для решения комбинаторных задач с псевдобулевыми функциями (полиномами от булевых переменных). В качестве примера эффективности применения этих кодов рассматривается решение 0-1 задачи о ранце 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.*

DOI: 10.35350/2409-8876-2019-14-1-63-69

Введение

В 1953 г. физик Фрэнк Грей (Frank Gray) изобрел коды Грея и получил на них патент [1]. 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» [2]. Подробнее о кодах Грея можно узнать в книге Д. Кнута [3].

В данной статье показано, как можно применять коды Грея для эффективного вычисления значения целевой функции и проверки ограничений в различных схемах ветвления, например, в методах ветвей и границ, динамического программирования и др., когда количество двоичных (булевых) переменных в узлах ветвления невелико (менее 35). В этом случае в узлах ветвления для каждого допустимого варианта решения можно использовать частичный пересчет значений целевой функции и ограничений.

Рассматривается 0-1 задача о ранце (0-1 Knapsack Problems, 0-1 KP), на примере которой показано, насколько уменьшается время решения задачи за счет частичного пересчета значений целевой функции и ограничений по сравнению с их полным пересчетом.

1. 0-1 задача о ранце

Математическая формулировка 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 $W \in Z^+$

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

$$x_{ij} \in \{0,1\} \,. \tag{3}$$

Предполагается, что $a_i \le W$, $i = \overline{1,n}$ и $\sum_{i=1}^{n} a_i > W$. Как известно, сформулированная задача является NP-трудной, т.е. в общем случае для ее решения не существует точного полиномиального алгоритма [4]. В книгах [5, 6] можно найти описание трех точных псевдополиномиальных алгоритмов решения задачи (1)-(3), 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 $(1-\varepsilon)$ гарантированное приближенное решение, где ε — сколь угодно малое положительное число.

2. Алгоритм полного перебора на основе кодов Грея

Для полного перебора вариантов решения задачи (1)-(3) будем использовать алгоритм генерации последовательности двоично-отраженных n-разрядных кодов Грея, предложенный Эрлихом [7]. Алгоритм дает возможность в процессе решения эффективно вычислять значения целевой функции (1) и ограничения (2). Двоично-отраженный (зеркальный, рефлексивный) код Грея определяется по следующим рекурсивным правилам:

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

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

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

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

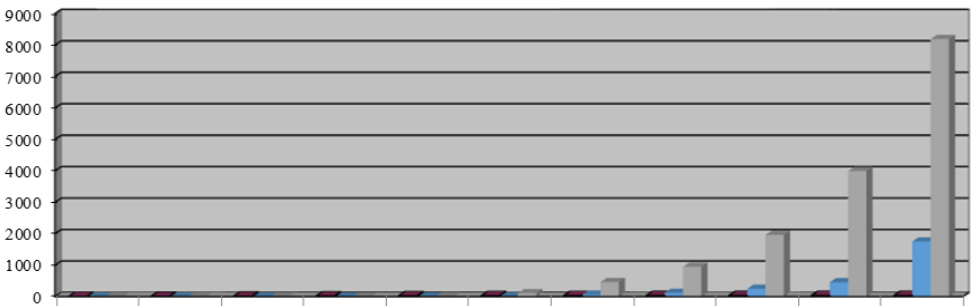
а шаг 5 заменится на

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

3. Экспериментальное сравнение алгоритмов OPT1 И OPT2

Сравнение быстродействия алгоритмов OPT1 и OPT2 проводилось на примерах, сгенерированных датчиком псевдослучайных чисел. Для всех размерностей задачи, 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, [6, 8], 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.

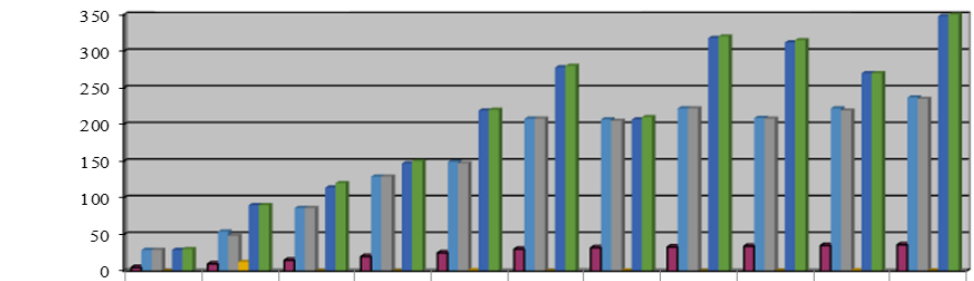
На рис. 1 показано время решения задачи (1)-(3) (с точностью до двух знаков после запятой) на ПК с тактовой частотой 2.66 GHz для алгоритмов OPT1 и OPT2 с частичным и полным пересчетом целевой функции и ограничения. Как видно из рис. 1, алгоритм OPT1 может применяться для практических расчетов в схемах 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.



	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

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

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



	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

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

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. 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).

Заключение

1. В статье показано, как можно применять на практике двоично-отраженные коды Грея для решения комбинаторных задач с псевдобулевыми функциями (полиномами от булевых переменных), когда количество переменных в узлах дерева ветвления решающего алгоритма не превышает 35.

2. На примере решения 0-1 задачи о ранце наглядно продемонстрирована вычислительная эффективность предложенного алгоритма полного перебора решений с ограниченным пересчетом значения целевой функции и ограничения задачи.

3. Приведены результаты вычислительного эксперимента, которые показали возможность использования «жадного» алгоритма решения 0-1 задачи о ранце на практике для инженерных расчетов в задачах большой размерности ($n = 10000$ и более).

СПИСОК ЛИТЕРАТУРЫ

1. Gray F. Pulse code communication // U.S. Patent 2632058, March 17, 1953.
2. Гарднер М. Математические головоломки и развлечения: 2-е изд., испр. и дополн. / Пер. с англ. — М.: «Мир», 1999, 447 с.
3. Knuth D.E. The Art of Computer Programming. Volume 4A / Combinatorial Algorithms, Part 1. — Addison Wesley Longman, Inc., 2011. — 933 p.
4. 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.
5. Martelo S., Toth P. Knapsack problems: algorithms and computer implementations. — Great Britain: Wiley, 1990. — 296 p.
6. Kellerer H., Pferschy U., Pisinger D. Knapsack Problems. — Springer-Verlag Berlin Heidelberg, 2004. — 548 p.
7. 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.
8. Кузюрин Н.Н., Фомин С.А. Эффективные алгоритмы и сложность вычислений. — М: МФТИ, 2008. — 326 с.

REFERENCES

1. Gray F. Pulse code communication // U.S. Patent 2632058, March 17, 1953.
2. Gardner M. Mathematical puzzles and diversions. — Penguin Press Science S., New edition, 1991. — 160 p.
3. Knuth D.E. The Art of Computer Programming. Volume 4A / Combinatorial Algorithms, Part 1. — Addison Wesley Longman, Inc., 2011. — 933 p.
4. 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.
5. Martelo S., Toth P. Knapsack problems: algorithms and computer implementations. — Great Britain: Wiley, 1990. — 296 p.
6. Kellerer H., Pferschy U., Pisinger D. Knapsack Problems. — Springer-Verlag Berlin Heidelberg, 2004. — 548 p.
7. 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.
8. Kuzyrin N.N., Fomin S.A. Effektivnie algoritmi i slognost vithisleni. — M: MFTI, 2008. — 326 s. (In Russian).

Стаття надійшла до редакції 20.12.2018.