ТЕОРІЯ ОПТИМАЛЬНИХ РІШЕНЬ

УДК 519.85

Г.А. ДОНЕЦ, Л.Н. КОЛЕЧКИНА

АЛГОРИТМ ПОИСКА ЗНАЧЕНИЙ ЛИНЕЙНОЙ ФУНКЦИИ НА ЛЕКСИКОГРАФИЧЕСКИ УПОРЯДОЧЕННЫХ ПЕРЕСТАНОВКАХ

Рассматриваются алгоритмы поиска значений линейной функции на лексикографически упорядоченных перестановках, излагается вопрос применения теории графов для построения алгоритмов нахождения перестановки по определенному номеру и наоборот.

Введение. Задачи комбинаторной оптимизации возникают при исследовании многих теоретических и прикладных проблем [1–2]. Для решения таких задач привлекаются довольно много методов, в том числе и методы теории графов [3–4]. С помощью теории графов хорошо описываются многие типы комбинаторных задач. При этом графические представления являются не просто иллюстрациями, но и позволяют получать новые подходы и методы к решению таких задач. Данная работа продолжает исследования [5].

Постановка задачи. В общем виде экстремальные комбинаторные задачи можно сформулировать так: имеется n-множество элементов $A = \{a_1, a_2, ..., a_n\}$, на котором задается конечное множество конфигураций $P(A)$. Под конфигурациями $a_1, a_2, ..., a_r$ можно понимать перестановки, сочетания, различные последовательности и т. п. На множестве $P(A)$ задается функция $F(x)$. Требуется отыскать экстремум $F(x)$ и элементы множества A, которые этот экстремум доставляют. Сама формулировка экстремальных комбинаторных задач диктует выбор операций, применяемых для их решения.

Общая схема связи экстремальных комбинаторных задач с методами линейного программирования выглядит таким образом: элементы $a \in P(A)$ представляются в виде точек евклидова пространства, чтобы целевая функция $F(x)$ стала линейной формой.

Рассматривается задача нахождения экстремума этой функции на выпуклой оболочке заданных точек (т. е. на выпуклом многограннике). В самом деле, экстремум линейной формы на многограннике достигается в одной из вершин, которые входят во множество рассматриваемых элементов. Рассмотрим на примере постановку комбинаторных экстремальных задач оптимизации вида: $Z(\Phi, P(A)) : \max\{\Phi(a) \mid a \in P(A)\}$, которая состоит в максимизации функции $\Phi(a)$ на множестве перестановок $P(A)$, где $A = \{1,2,\ldots,n\}$, а $\Phi(a) = \sum_{j=1}^{n} c_j x_j$. Решение задачи представляет собой перестановку $\pi \in S_n$, а $x_j = \pi(j)$. Каждая перестановка может быть интерпретирована как точка в n-мерном евклидовом пространстве. Тогда, при отображении множества $P(A)$ в евклидово пространство R^n, можно сформулировать задачу линейного программирования $Z(F, X)$ максимизации критерия $F(x)$ на множестве X, причем каждой точке $a \in P(A)$ будет соответствовать точка $x \in X$, такая, что $F(x) = \Phi(a) : Z(F, X) : \max\{F(x) \mid x \in X\}$, где $F(x) = \sum_{j=1}^{n} c_j x_j$, X – непустое множество в R^n, которое определяется таким образом $X = \mathrm{vert}\,\Pi(A)$, $\Pi = \mathrm{conv}\,P(A)$, где Π – многогранник перестановок.

Следует отметить, что иногда целесообразно решить задачу вида:

$$x^* = \arg\max_{x \in \Pi(A)} F(x), \tag{1}$$

для значений функции $y^* = F(x^*)$. Так же имеет смысл рассматривать задачу, где значение целевой функции находится в интервале $F(\underline{x}) \le F(x) \le F(\overline{x})$.

Тогда задача (1) примет вид: определить

$$\underline{x} = \arg\max_{x \in \Pi(A)} F(x) \text{ при } \underline{y} = F(\underline{x}), \quad \overline{x} = \arg\max_{x \in \Pi(A)} F(x) \text{ при } \overline{y} = F(\overline{x}) \tag{2}$$

при условии $|\underline{x} - \overline{x}| \to \min$.

Продолжая исследования и развивая результаты работы [5], в статье предлагаются алгоритмы решения задач, основанные на лексикографическом упорядочивании перестановок и определении значений целевой линейной функции $F(x)$ на графе, который представляет бинарные отношения перестановок в порядке убывания целевой функции. Далее под задачей понимаем задачу (2).

Построение графа перестановок для линейной функции. В принципе задачи (1) и (2) можно решить, если множеству перестановок $P(A)$ поставить в соответствие полный ориентированный граф, в котором перестановки представлены как вершины, а две вершины $p_1, p_2 \in P(A)$ соединены дугой, идущей от p_1

к p_2, если $F(p_1) \geq F(p_2)$. Очевидно, что максимальное значение функция принимает в вершине, в которую не входит ни одна дуга, а минимальное – в вершине, из которой не выходит ни одна дуга. Гамильтонов путь в этом графе будет соответствовать последовательности перестановок, на которых значения функции составляют невозрастающую последовательность. Если перенумеровать эту последовательность, то задачу (2) можно решать путем дихотомии последовательности до тех пор, пока искомый аргумент для заданного значения функции не попадет в интервал минимальной длины. Однако простота алгоритма при его реализации наталкивается на непреодолимые трудности вычислительного характера, если учесть то обстоятельство, что число вершин графа, который необходимо построить, равно $n!$. Для определения направления дуг графа необходимо проделать $n!$ вычислений значений функции F и C_m^2 сравнений этих значений, где $m = n!$. Даже для сравнительно небольших значений n этот алгоритм практически нереализуем. Для преодоления этих трудностей построим специальный подграф искомого полного графа. При этом направление части дуг между некоторыми вершинами можно определить, не вычисляя в них значения функции. Заменим $P(A)$ на $uP(A)$, где u – перестановка, переводящая коэффициенты $F(x)$ в неубывающую последовательность. Поэтому в дальнейшем, не нарушая общности, будем считать, что $c_1 \leq c_2 \leq ... \leq c_n$.

Лемма 1. Если из перестановки $p_1 = (i_1, i_2, ..., i_n) \in P(A)$ получена перестановка $p_2 \in P(A)$ транспозицией двух чисел $i_k < i_l$, где $k < l$, то $F(p_1) \geq F(p_2)$.

Доказательство. $F(p_1) - F(p_2) = c_k i_k + c_l i_l - c_k i_l - c_l i_k = (i_l - i_k)(c_l - c_k) \geq 0$, так как обе величины в скобках неотрицательны.

Будем строить искомый граф G_n по индукции. Для $n = 2$ граф содержит две вершины (1, 2) и (2, 1). По лемме 1 от первой вершины ко второй выходит дуга. Если построен G_n, то G_{n+1} образуется как $n+1$ копий графа G_n по правилу: в первой копии графа G_n каждой перестановке (вершине) справа приписывается число $n+1$. Вторая копия получается из первой транспозицией во всех перестановках чисел $n+1$ и n. И так дальше – из j-й копии транспозицией чисел $(n+2-j)$ и $(n+1-j)$, где $1 \leq j \leq n$, получаем $(j+1)$-ю копию. При этом от всех вершин j-й копии идут дуги в соответствующие вершины $(j+1)$-й копии. Пример построения графов G_3 и G_4 приведен на рис. 1 и 2. На этих графах все дуги построены по лемме 1. Так же можно в G_3 провести дополнительную дугу от вершины (213) к вершине (231), но это не существенно. В G_4 эта дуга воспроизводится в каждой копии G_3. В G_4 эти копии обозначены буквами A, B, C и D.

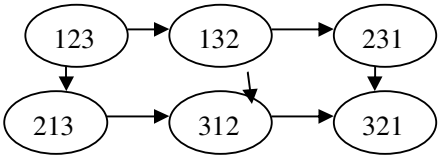

РИС. 1. Представление перестановочного графа G_3

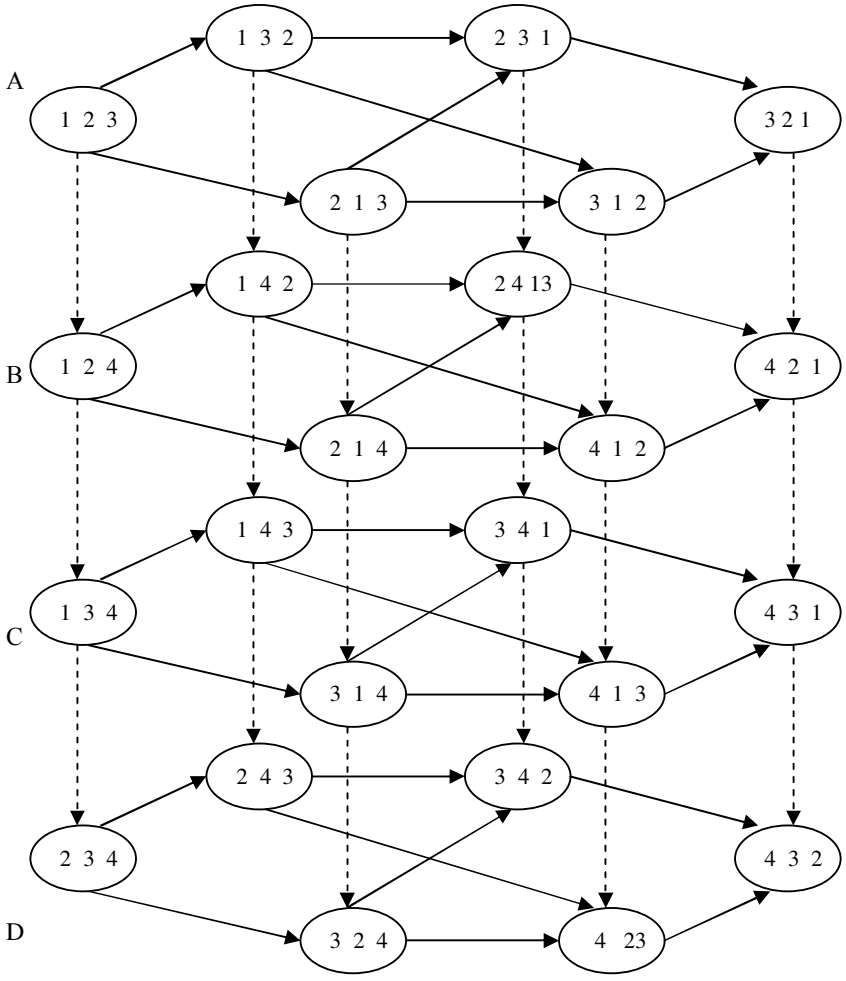

РИС. 2. Представление перестановочного графа G_4

Очень важной задачей, которая возникает при построении зависимости между элементами перестановки по значениям линейной целевой функции, является задача организации упорядочения и выбора элемента перестановки.

Алгоритм отображения по номеру перестановки N **элемента перестановки** $P(A)$. В данном алгоритме элементы перестановок нумеруются в лексикографическом порядке, который строится по правилу: из двух перестановок $a=(a_1,a_2,...,a_k,...,a_n)$, $b=(b_1,b_2,...,b_k,...,b_n)$, $b>a$, если найдется такое k, при котором $b_k>a_k$, а $b_i=a_i$, $i>k$, $(1<k\le n,\ i\le n)$. Алгоритм дает возможность найти по заданному номеру $1\le N\le n!$ соответствующую перестановку. Рассмотрим работу алгоритма на примере для n = 4. Построим последовательность перестановок по вышеизложенному правилу.

1. 1234	7. 1243	13. 1342	19. 2341
2. 2134	8. 2143	14. 3142	20. 3241
3. 1324	9. 1423	15. 1432	21. 2431
4. 3124	10. 4123	16. 3412	22. 4231
5. 2314	11. 2413	17. 3412	23. 3421
6. 3214	12. 4213	18. 4312	24. 4321.

Задача. Задано: n – количество элементов в множестве, тогда $n!$ – количество перестановок, N – номер элемента перестановки. Найти элемент перестановки $x=(x_1,x_2,...,x_n)$.

Начальные данные: $k=n$, $N_0=N$, массив r = (1,2,…, n). Алгоритм работает по шагам: $\Delta=(k-1)!$; $j=\left|\frac{N_0-1}{\Delta}\right|$; $x_k=r_{k-j}$. В массиве r вычеркивается элемент r_{k-j}; $N_0=N_0-j\cdot\Delta$; $k=k-1$; если $k=1$, то $x_1=r_1$ и конец.

В терминах программной реализации алгоритм для N = 19 будет выглядеть следующим образом:

$i=1; k=4; r=[1234]$

$\Delta=3!=6;$	$i=2,\Delta=2;$	$i=3,\Delta=1;$
$j=\left\|\frac{19-1}{6}\right\|=3;\ x_4=r_{4-3}=1;$	$j=\left\lfloor\frac{1-1}{2}\right\rfloor=0, x_3=r_{3-0}=4;$	$j=\left\lfloor\frac{1-1}{1}\right\rfloor=0, x_2=r_{2-0}=3;$
$r=[234];$	$r=[23];$	$r=[2];$
$N_0=19-3\cdot6=1;$	$N_0=1-2\cdot0=1;$	$N_0=1-1\cdot0=1;$
$k=3;\ i=2;$	$k=2;\ i=3;$	$k=1; i=4; x_1=2.$

В результате получаем перестановку x = (2341), что соответствует данным.

Алгоритм отображения номера перестановки N **по элементу перестановки** $P(A)$. Данный алгоритм решает **обратную задачу**: по заданной перестановке найти ее номер в лексикографически упорядоченной последовательности

перестановок. Рассмотрим реализацию данного алгоритма на примере.

Пусть задана перестановка $x=(x_1,x_2,...,x_n)$. Найти N – номер элемента перестановки. Начальные данные: $k=n$, $N=0$, массив r = (1,2,…, n). Работа алгоритма: по значению x_k находим номер элемента j в массиве r и вычеркиваем его. Вычисляем $N=(k-1)!(k-j)+N$; $k=k-1$; если $k=1$, то $N=N+1$ и конец. В терминах программной реализации алгоритм будет выглядеть следующим образом при начальных параметрах: $n=4$, $x_N=(2,4,3,1)$, $N=0$, $r=(1,2,3,4)$, $k=n=4$, $k=4$, $x_4=1$, $j=1$, $N=3!(4-1)=18$; $r=[2,3,4]$;

$k=3$, $x_3=3$, $j=2$, $N=18+1\cdot 2!=20$; $r=[2,4]$;

$k=2, x_2=4$, $j=2$, $N=20+0=20$; $r=[2]$;

$k=1$, $x_1=2$, $j=4$, $N=20+1=21$. В результате получаем $N=21$.

Заключение. Дальнейшее развитие работы будет направлено на реализацию и адаптацию сформулированных алгоритмов, на разработку новых методов решения комбинаторных задач с учетом других комбинаторных множеств.

Г.П. Донець, Л.М. Колєчкіна

АЛГОРИТМ ПОШУКУ ЗНАЧЕНЬ ЛІНІЙНОЇ ФУНКЦІЇ
НА ЛЕКСИКОГРАФІЧНО ВПОРЯДКОВАНИХ ПЕРЕСТАНОВКАХ

Розглядаються алгоритми пошуку значень лінійної функції на лексикографічно впорядкованих перестановках, висвітлюється питання застосування теорії графів для побудови алгоритмів знаходження перестановки за заданим номером і навпаки.

G.A. Donets, L.N. Kolechkina

ALGORITHM OF SEARCH OF LINEAR FUNCTION VALUES
ON LEXICOGRAPHIC ORDERING TRANSPOSITIONS

Algorithms of search of linear function values on lexicographic ordering transpositions are considered. The investigation serves to clarify the issue of possible application the graph theory to construct algorithms of finding a transposition under the basis of certain number and vice versa.

1. *Сергиенко И.В.*, *Шило В.П.* Задачи дискретной оптимизации: проблемы, методы решения и исследования. – Киев: Наук. думка, 2003. – 260 с.
2. *Стоян Ю.Г.*, *Яковлев С.В.* Математические модели и оптимизационные методы геометрического проектирования. – Киев: Наук. думка, 1986. – 265 с.
3. *Донец Г.А.*, *Шулинок И.Э.* О сложности алгоритмов поиска в глубину на модульных графах // Теорія оптимальних рішень. – К.: Ін-т кібернетики ім. В.М. Глушкова НАН України, 2002. – № 1. – С. 105 – 110.
4. *Донец Г.А.*, *Бинь Чжан.* Задачи о математическом сейфе на графах // Кибернетика и системный анализ. – 2006. – № 5. – С. 14 – 22.
5. *Донец Г.А.*, *Колечкина Л.Н.* Метод упорядочения значений линейной функции на множестве перестановок // Там же. – 2009. – № 2. – С. 50 – 61.