ISSN 0002-306X. Изв. НАН РА и ГИУА. Сер. ТН. 2003. Т. LVI, № 3.

УДК 519.857.6.004.4:002

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ИНФОРМАТИКА

Д.Н. ВАРТАНЯН

ПРИМЕНЕНИЕ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ В ЗАДАЧЕ О ДОКУМЕНТООБОРОТЕ

Для оптимального формирования казначейского документооборота к нему применена задача коммивояжера. Дан алгоритм ее решения методом динамического программирования. Показано, что при этом достигается существенное сокращение количества вычислений за счет заметного увеличения объема памяти.

Ключевые слова: казначейский документооборот, задача коммивояжера, динамическое программирование, количество вычислений, объем памяти.

Центральную идею глобальных методов решения задачи коммивояжера составляет замена полного перебора всех планов их частичным перебором, что осуществляется путем отбрасывания некоторых подмножеств вариантов, заведомо не дающих оптимума. Далее перебор ведется лишь среди оставшихся вариантов, являющихся в определенном смысле «перспективными» [1].

Напомним постановку задачи о коммивояжере. Требуется найти кратчайший замкнутый маршрут (цикл), проходящий через каждый из заданных пунктов 0,1,2,…,n ровно по одному разу. Применительно к документообороту задача заключается в необходимости формирования каждого выходного документа ровно один раз с возвращением в конце к начальному для итоговой сверки [2].

Говоря формально, задана матрица (Cij), i,j=0,1,…,r,…,n (i(j), и требуется найти перестановку

$$\pi = (i_1, i_2, \ldots, i_n) \qquad (1)$$

чисел 1,2,…,n, минимизирующую суммарное время формирования всех документов

$$C_{0i_1} + C_{i_1 i_2} + C_{i_2 i_3} + \ldots + C_{i_{n-1} i_n} + C_{i_n 0}.$$

Ясно, что в силу замкнутости искомого цикла безразлично, какой документ считается начальным, однако в практике казначейского документооборота этим документом всегда бывает реестр о поступлении доходов на счет 090, который и будет считаться нулевым.

Пусть $(i_1, i_2, \ldots, i_{k-1})$ – некоторые различные документы, отличные от начального ($i_s \neq i_r$ при $s \neq r$; $i_s \neq 0$ для всех $s = 1,2,\ldots,k-1$). Пусть i_k – документ, отличный от $i_1, i_2, \ldots, i_{k-1}$. Документ i_k может совпадать с начальным ($i_k = 0$) только при k=n.

Обозначим через

$$f_{k-1}(0; i_1, i_2, \ldots, i_{k-1}; i_k) \qquad (2)$$

минимальную длительность при формировании всего списка документов, начиная с документа 0 и заканчивая документом k, проходящего в произвольном порядке

через документы $i_1,i_2,\ldots,i_{k-1}$. Заметим, что функция (2) симметрична по своим аргументам, то есть не изменяется при любой их перестановке.

Перестановку $(i'_1,i'_2,\ldots,i'_{k-1})$ документов (i_1, i_2, …, i_k), на которой реализуется фигурирующая в определении (2) минимальная длительность, обозначим через

$$\pi_{k-1}(0;i_1,i_2,\ldots,i_{k-1};i_k). \quad (3)$$

Теперь перейдем к изложению алгоритма.

Алгоритм

Шаг 0. Вычисляем функцию

$$f_0(0;i) = C_{0i}, \quad i = 1,2,\ldots, n. \quad (4)$$

Количество значений P_0 функции $f_0(0;i)$, подлежащих вычислению, равно

$$P_0 = n. \quad (5)$$

При этом для вычисления одного значения функции требуется одно действие (выбор из массива). Количество значений Q_0 функции $f_0(0;i)$, подлежащих запоминанию, равно

$$Q_0 = n. \quad (6)$$

Шаг 1. Вычисляем функцию

$$f_1(0;i;j)=f_0(0;i)+C_{ij},\ i,j=1,2,\ldots,n;\ i\neq j. \quad (7)$$

Количество значений P_1 функции $f_1(0;i;j)$, подлежащих вычислению, равно

$$P_1=n(n-1)=n\,C_{n-1}^{1}. \quad (8)$$

Для вычисления одного значения функции требуются три действия: два выбора из массива и одно сложение. Необходимый объем памяти (до вычеркивания из памяти вектора $f_0(0;i)$)

$$Q_1 = n+n(n-1). \quad (9)$$

Шаг k ($2 \le k \le n-1$). Вычисляем функцию $f_k(0;i_1,i_2,\ldots,i_k;i_{k+1})$ для всех значений i_1, i_2, …, i_{k+1} ($i_r\neq i_s$, при $r\neq s$, $i_r=1,2,\ldots,n$; $r=1,2,\ldots,k+1$):

$$f_k(0;i_1,i_2,\ldots,i_k;i_{k+1})=\min\{[\,f_{k-1}(0;i_2,\ldots,i_k;i_1)+C_{i_1i_{k+1}}],\ [f_{k-1}(0;i_1,i_3,\ldots,i_k;i_2)+C_{i_2i_{k+1}}],\ldots,$$
$$\ldots,\ [f_{k-1}(0;i_1,i_2,\ldots,i_{k-1},i_k;i_k)+C_{i_ki_{k+1}}]\}. \quad (10)$$

Для каждого значения функции $f_k(0;i_1,i_2,\ldots,i_k;i_{k+1})$ запоминаем оптимальную перестановку $\pi_k(0;i_1,i_2,\ldots,i_k;i_{k+1})$. По окончании вычислений вычеркиваем из памяти вектор значений $f_{k-1}(0;i_1,i_2,\ldots,i_{k-1};i_k)$ (а при $k\ge 3$ – и путей $\pi_{k-1}(0;i_1,i_2,\ldots,i_{k-1};i_k)$).

Количество значений P_k функции f_k, подлежащих вычислению, равно

$$P_k=n\,C_{n-1}^{k}. \quad (11)$$

Для вычисления одного значения функции требуется $4k-1$ действий ($2k$ выбора из массива, k сложений, $k-1$ сравнений). Необходимый объем памяти Q_k (до вычеркивания из памяти векторов f_{k-1} и π_{k-1}) составляет

$$Q_k=n\,C_{n-1}^{k-1}+n\,C_{n-1}^{k}. \quad (12)$$

Шаг n. Вычисляем фукнцию

$$f_n(0;1,2,\ldots,n;0)=\min\{[\,f_{n-1}(0;2,\ldots,n;1)+C_{10}],\ [f_{n-1}(0;1,3,\ldots,n;2)+C_{20}],\ldots,$$

$$\ldots, [f_{n-1}(0;1,2,\ldots,n-1;n)+C_{n0}]\}. \qquad (13)$$

Перестановка $\pi_n(0;1;2;\ldots;n;0)$, на которой достигается минимум, и будет оптимальным планом исходной задачи. При этом количество значений P_n функции f_n, подлежащих вычислению, будет равно

$$P_n=1. \qquad (14)$$

Для вычисления одного значения функции требуется 4n-1 действий (2n выбора из массива, n сложений, n-1 сравнений). Необходимый объем памяти

$$Q_n=n\,C_{n-1}^{n-1}+1=n+1. \qquad (15)$$

Теперь получим выражение для суммарного объема необходимых вычислений:

$$P=P_0^{'}+P_1^{'}+\sum_{k=1}^{n-1}P_k^{'}+P_n^{'},$$

где $P_r^{'}=P_rd_r$, причем P_r – количество значений функции $f_r(0;i_1,i_2,\ldots,i_r;i_{r+1})$, подлежащих вычислению, а d_r – количество операций, приходящихся на одно значение. Находим

$$P_0^{'}=P_0d_0=n\cdot 1=n,$$
$$P_1^{'}=P_1d_1=n\cdot C_{n-1}^1\cdot 3,$$
$$P_n^{'}=P_nd_n=1\cdot(4n-1).$$

При всех k=2,3,…,n-1 имеем

$$P_k^{'}=P_kd_k=nC_{n-1}^k(4k-1).$$

Таким образом,

$$P=n+3nC_{n-1}^1+\sum_{k=2}^{n-1}(4k-1)nC_{n-1}^k+1\cdot(4n-1)=$$
$$=(5n-1)+\sum_{k=1}^{n-1}(4k-1)nC_{n-1}^k+(4\cdot 0-1)nC_{n-1}^0-(4\cdot 0-1)nC_{n-1}^0=$$
$$=(5n-1)+n+n\sum_{k=0}^{n-1}(4k-1)C_{n-1}^k=(6n-1)+n\sum_{k=0}^{n-1}(4k-1)C_{n-1}^k=$$
$$=(6n-1)-n\sum_{k=0}^{n-1}C_{n-1}^k+4n\sum_{k=0}^{n-1}kC_{n-1}^k=(6n-1)-n2^{n-1}+4n\sum_{k=0}^{n-1}kC_{n-1}^k.$$

Преобразуем теперь последнюю сумму в выражении для P. Обозначим

$$S=\sum_{k=0}^{n-1}kC_{n-1}^k.$$

Тогда

$$S=\sum_{k=0}^{n-1}kC_{n-1}^k=\sum_{k=0}^{n-1}kC_{n-1}^{n-1-k}=\sum_{v=0}^{n-1}(n-1-v)C_{n-1}^v=\sum_{k=0}^{n-1}(n-1-k)C_{n-1}^k,$$

откуда

$$S + S = \sum_{k=0}^{n-1} kC_{n-1}^{k} + \sum_{k=0}^{n-1} (n-1-k)C_{n-1}^{k},$$

или

$$2S = \sum_{k=0}^{n-1} (n-1)C_{n-1}^{k} = (n-1)2^{n-1}$$

и, следовательно,

$$S = (n-1)2^{n-2}.$$

Окончательно имеем

$$P = (6n-1) - n2^{n-1} + 4n(n-1)2^{n-2}. \quad (16)$$

Отметим, что при полном переборе имеем (согласно формуле Стирлинга для больших n) [3]

$$n! = n^{n} e^{-n} \sqrt{2\pi n},$$

что существенно больше P.

Получим теперь выражение для максимального потребного объема памяти Q. Имеем

$$Q = \max_{0 \le k \le n} Q_k = \max\left\{Q_0, Q_1, Q_n, \max_{2 \le k \le n-1} Q_k\right\} = \max\{n, n + n(n-1), n+1,$$

$$\max_{2 \le k \le n-1} \left(nC_{n=1}^{k-1} + nC_{n-1}^{k}\right)\} = n \max_{2 \le k \le n-1} \left(nC_{n=1}^{k-1} + nC_{n-1}^{k}\right).$$

Однако

$$C_{n-1}^{k-1} + C_{n-1}^{k} = \frac{(n-1)!}{(k-1)!(n-k)!} + \frac{(n-1)!}{k!(n-k-1)!} = \frac{(n-1)!(k+(n-k)!)}{k!(n-k)!} =$$

$$= \frac{n!}{k!(n-k)!} = C_{n}^{k},$$

так что

$$Q = n \max_{2 \le k \le n-1} C_{n}^{k} = nC_{n}^{\left[\frac{n}{2}\right]},$$

где $\left[\frac{n}{2}\right]$ - целая часть от $\frac{n}{2}$.

Используя формулу Стирлинга в отдельности в случае четных и нечетных n, приходим к выражению

$$Q \sim 2^{n} \sqrt{\frac{2n}{\pi}}. \quad (17)$$

Итак, удалось (по сравнению с полным перебором, для которого P'=n! и Q'=2) достигнуть существенного сокращения количества вычислений за счет заметного увеличения объема памяти.

СПИСОК ЛИТЕРАТУРЫ

1. **Гроппен В.О.** Принципы оптимизации программного обеспечения. - Ростов-на-Дону: Изд-во Рост. у-та, 1993. – 242 с.
2. **Корбут А.А., Финкельштейн Ю.Ю.** Дискретное программирование. - М.: Просвещение, 1987. – 530 с.
3. **Евстигнеев В.А.** Применение теории графов в программировании. - М.: Наука, 1985. – 168 с.

Материал поступил в редакцию 15.01.2002.

Դ. Ն. ՎԱՐԴԱՆՅԱՆ

ԴԻՆԱՄԻԿ ԾՐԱԳՐԱՎՈՐՄԱՆ ԿԻՐԱՌՈՒՄԸ ՓԱՍՏԱԹՂԹԵՐԻ ՇՐՋԱՆԱՌՈՒԹՅԱՆ ԽՆԴՐՈՒՄ

Գանձապետական փաստաթղթերի օպտիմալ ձևավորման խնդրի լուծման համար կիրառվում է շրջիկ վաճառականի (կոմիվոյաժոր) խնդիրը: Տրված է դրա լուծման ալգորիթմը դինամիկ ծրագրավորման մեթոդով: Ցույց է տրված, որ հիշողության ծավալի նկատելի ավելացման գնով ապահովվում է հաշվարկների ծավալի էական կրճատում:

D.N. VARTANYAN

APPLICATION OF DYNAMIC PROGRAMMING IN DOCUMENT TURNOVER PROBLEM

To form optimally treasury document turnover, a run book problem is applied. The algorithm for its decision by method of dynamic programming is given.It is shown that essential reduction of calculation quantity is reached at the expense of appreciable storage increase.