УДК 517.174

Рибак М. В.

ОДИН ІЗ ПІДХОДІВ ДО РОЗВ'ЯЗАННЯ ЗАДАЧІ ПРО ЗНАХОДЖЕННЯ ОПТИМАЛЬНОЇ ПЕРЕСТАНОВКИ

Серед задач, не розв'язаних за поліноміальний час, існує певний підклас задач, які розв'язуються ітеративно, причому кожна ітерація покращує розв'язок, який уже знайдено після всіх попередніх ітерацій. Таким чином, за умови існування певних обмежень на оціночну функцію розв'язку (наприклад, скінченність множини її значень) можна говорити про певну «поліноміальність» розв'язку – адже час, у гіршому випадку, лінійно залежить від потужності множини значень цієї функції. У праці розглянуто одну з таких задач і наведено відповідний ітеративний алгоритм.

Постановка задачі

Фірма має n програмістів (далі – кодерів) та n робочих місць (далі – ПК). У кожного кодера продуктивність його роботи залежить від ПК, за яким він працює. Матриця $\mathbf{A}_{n,n}$ встановлює цю залежність: продуктивність кодера i за ПК j дорівнює $\mathbf{A}_{i,j}$. Необхідно розподілити ПК між кодерами таким чином, щоб їхня сумарна продуктивність була максимальною. Якщо існує декілька розв'язків, достатньо знайти один з них.

Розв'язання

Перш ніж розглядати розв'язання поставленої задачі, звернемося до допоміжної задачі.

Задача А

Нехай маємо орієнтований зважений граф із v вершинами, у якого вага ребра – ціле число (не обов'язково невід'ємне). Із кожної вершини в кожну іншу вершину (включаючи її саму) прямує рівно одне ребро. Необхідно знайти будь-який простий додатний цикл або впевнитися, що такого не існує (додатним називаємо такий цикл, сума значень ребер якого строго більше нуля). Вага ребер задана в матриці $\mathbf{W}_{v,v}$: $\mathbf{W}_{i,j}$ – вага ребра з вершини i у вершину j. Додаткова умова:

$$\forall i : \mathbf{W}_{i,i} = 0.$$

Розв'язок

Далі вважатимемо, що *шлях* може містити нуль ребер – тобто складатися лише з однієї, початкової та одночасно кінцевої вершини. Для зручності говоритимемо «найбільший шлях», маючи на увазі «найбільший за сумою довжин ребер шлях», а також «довжина шляху», маючи на увазі «суму довжин ребер шляху».

Побудуємо таку матрицю $\boldsymbol{C}_{2v,v}$:$\boldsymbol{C}_{i,j}$ означатиме довжину найбільшого шляху (не обов'язково простого) серед тих, що починаються в будь-якій вершині, складаються рівно з i ребер і завершуються у вершині j. Зауважимо, що в цій матриці нумерацію горизонтальних рядків почнемо з 0: лівий верхній елемент – це $\boldsymbol{C}_{0,1}$, правий нижній – $\boldsymbol{C}_{2v-1,v}$.

Матрицю $\boldsymbol{C}$ будуватимемо по рядках від першого ($\boldsymbol{C}_{0,*}$) до останнього ($\boldsymbol{C}_{2v-1,*}$). У нульовому рядку матимемо нулі, бо довжина єдиного шляху, що завершується у вершині j і містить 0 ребер, дорівнює 0.

Нехай маємо побудований i-й рядок. Побудуємо $(i+1)$-й. Легко бачити, що:

$$\mathbf{C}_{i+1,j} = \max_k(\mathbf{C}_{i,k} + \mathbf{W}_{k,j}).$$

Для відновлення побудованих шляхів будуватимемо разом із матрицею $\mathbf{C}_{2v,v}$ матрицю

$$\mathbf{D}_{2v-1,v}: \forall i = \overline{0,2v-2}, \forall j = \overline{1,v} : \mathbf{D}_{i+1,j} =$$

$$= \arg\max_k(\mathbf{C}_{i,k} + \mathbf{W}_{k,j}).$$

Тоді, очевидно,

$$\boldsymbol{C}_{i+1,j} = \boldsymbol{C}_{i,\boldsymbol{D}_{i+1,j}} + \mathbf{W}_{\boldsymbol{D}_{i+1,j},j}.$$

Тепер подивимося, як за допомогою матриці $\mathbf{C}$ відповісти на питання задачі А. Оскільки ми поряд із $\mathbf{C}$ маємо матрицю $\mathbf{D}$, то для кожної комірки матриці $\boldsymbol{C}$ є можливість легко побудувати конкретний шлях, що їй відповідає,– рухаючись по ньому в зворотному напрямку. Далі вживатимемо «шлях до комірки (i,j)», маючи на увазі «шлях, що відповідає комірці $\mathbf{C}_{i,j}$».

Твердження А. Якщо в заданому графі існує простий додатний цикл, то знайдеться така комірка (i,j) матриці $\mathbf{C}$, шлях до якої містить деякий простий додатний цикл.

Доведення

Нехай в графі існує простий додатний цикл $a_1, a_2, .., a_k, a_1$, що складається з k ребер і k вершин. Оскільки цикл простий, то, очевидно, $k \le v$.

Розглянемо a_1-й стовпець матриці $\mathbf{C}$. Нехай x – найбільше значення в цьому стовпці, і нехай p – перший рядок, в якому воно зустрічається в цьому стовпці:

$$\begin{cases} x = \max\limits_{i=0..2v-1} \mathbf{C}_{i,a_1} \\ p = \min\limits_{i=0..2v-1, \mathbf{C}_{i,a_1}=x} i \end{cases}$$

Нехай $u_0, .., u_p$ – шлях до (p, a_1). Тут, очевидно, $u_p = a_1$. Покажемо, що (p, a_1) – це і є шукана комірка, тобто шлях $u_0, .., u_p$ містить простий додатний цикл.

Зрозуміло, що $p \ge v$. Справді, якщо $p < v$, то довжина шляху $u_0, .., u_p = a_1, a_2, .., a_k, a_1$, що складається з $p+k$ ребер, більша за довжину шляху $u_0, .., u_p$, а $p+k < v+v = 2v$, тому маємо $x = \mathbf{C}_{p,a_1} < \mathbf{C}_{p+k,a_1}$, а це суперечить вибору x. Таким чином, оскільки $p \ge v$, то шлях $u_0, .., u_p$, що складається з $p+1$-ї вершини, містить принаймні 2 однакові вершини і тому містить принаймні один простий цикл. Припустимо, що він не є додатним. Тоді видаливши його, ми отримаємо шлях із меншою кількістю ребер, довжина якого не менша за довжину шляху до (p, a_1), а це суперечить вибору x та p. Твердження А доведено.

Тепер зрозуміло, як відповісти на запитання задачі А. Можна, наприклад, після заповнення i-го рядка матриці $\mathbf{C}$ перевіряти, чи не побудували ми щойно деякий простий додатний цикл, а саме для кожної вершини j будуємо в зворотному порядку шлях до (i,j):

$u_i \leftarrow u_{i-1} \leftarrow \ldots \leftarrow u_1 \leftarrow u_0$.

Під час побудови шляху накопичуємо його поточну довжину в деякій змінній. Якщо зустрічаємо при цьому вершину $u_k = j$, $k < i$, то закінчуємо побудову шляху, бо знайдено цикл, який ми знайшли на i-й ітерації заповнення рядків матриці **C**, причому якщо поточна довжина побудованого відрізка шляху (тобто власне циклу) більше нуля, то це означає, що ми знайшли простий додатний цикл і можемо припинити подальше заповнення матриці **C**.

Повернемося тепер до початкової задачі. Будь-який можливий розподіл ПК між кодерами задаватимемо n-вимірним вектором $\mathbf{V}(s_1, .., s_n)$, де s_i – номер ПК для кодера i. Введемо позначення: $\mathbf{V}^{-1}(i) = j : s_j = i$, тобто це позиція числа i у цьому розв'язку. Введемо також поняття транспозиції. Транспозицією $(a_1, \ldots, a_k)$ у розв'язку $(s_1, .., s_n)$ називатимемо циклічну перестановку вліво значень $s_{a1}, \ldots, s_{ak}$, тобто кодеру a_1 відповідатиме ПК, що відповідав кодеру a_2 і т. д.

Твердження В. Якщо деякий розв'язок $\mathbf{T}(s_1, .., s_n)$, не є оптимальним, то існує транспозиція, яка покращує його, тобто утворює розв'язок $\mathbf{T}'$ з більшою сумарною продуктивністю, ніж у **T**.

Доведення. Справді, нехай маємо поточний розв' язок $\mathbf{T}(s_1, .., s_n)$, і нехай оптимальним є розв'язок $\mathbf{O}$ $(o_1, .., o_n)$ (ми не знаємо його, але можемо розглядати, доводячи твердження). Позначимо P(**A**) сумарну продуктивність розв'язку **A**.

Візьмемо будь-якого кодера i. Побудуємо таку послідовність кодерів:

$q1 = i$
$q2 = \mathbf{T}^{-1}(o_{q1})$
$q3 = \mathbf{T}^{-1}(o_{q2})$
...
$q_k = \mathbf{T}^{-1}(o_{qk-1})$
$q_{k+1} = \mathbf{T}^{-1}(o_{qk}) = i$,

причому k – найменше, для якого $q_{k+1} = i$. Таким чином, маємо транспозицію, породжену кодером i: $(q_1, .., q_k)$. Позначимо $\mathbf{T}'$ результат її виконання на $\mathbf{T}(s_1, .., s_n)$. Очевидно,

$$\mathrm{P}(\mathbf{T}') \geq \mathrm{P}(\mathbf{T}), \qquad (1)$$

бо інакше розв'язок **O** містить неоптимальний фрагмент і може бути покращений.

Розіб'ємо множину кодерів на своєрідні цикли, кожен елемент яких породжує транспозицію, одну й ту саму для елементів одного циклу. Якщо виконати на розв'язку **T** по черзі кожну з цих транспозицій по одному разу, одержимо розв'язок **O**, тому серед усіх цих транспозицій знайдеться хоча б одна, для якої має місце знак «>» у нерівності (1). Це і є шукана транспозиція, яка доводить твердження.

Залишилося вказати спосіб пошуку такої транспозиції для поточного розв'язку $\mathbf{T}(s_1, .., s_n)$. Розглянемо орієнтований зважений граф **F** із n вершинами, в якого з кожної вершини в кожну (включаючи її саму) прямує рівно одне ребро, а вага ребра – ціле число (не обов'язково невід'ємне). Вагу ребра із вершини i у вершину j вважатимемо рівною $\mathbf{W}_{i,j} = \mathbf{A}_{i,s_j} - \mathbf{A}_{i,s_i}$. Встановивши значення ребер таким чином, ми будуємо відповідність між поняттям транспозиції простого циклу в графі. Справді, легко перевірити, що застосування транспозиції $(a_1, a_2, \ldots, a_n)$ на розв'язку **T** змінить його сумарну продуктивність на число, що дорівнює вазі відповідного циклу $(a_1, a_2, \ldots, a_n)$ в графі **F**. Тому, щоб знайти транспозицію, що покращує розв'язок **T**, необхідно знайти простий додаткний цикл у графі **F**, а це є допоміжною задачею А.

Маючи цей доведений метод покращення поточного розв'язку, застосовуватимемо його до початкового розв'язку $(1, 2, \ldots, n)$ доти, доки не одержимо оптимальний.

Доведення коректності алгоритму міститься у доведенні тверджень А та В, оскільки коректність іншої частини алгоритму очевидна і випливає з побудови матриць **C** та **D** та описаних дій.

Проведемо **часову оцінку складності алгоритму**. Оскільки для обчислення $\mathbf{C}_{i,j}$ та $\mathbf{D}_{i,j}$ обраховується максимум з n чисел, шлях до кожної вершини (для перевірки наявності щойно побудованого циклу) будується за час, пропорційний його довжині, яка менша за n, а тому весь алгоритм пошуку додатного циклу виконується за $O(2n^2(n+n)) = O(n^3)$. Отже, складність всього алгоритму складатиме $O(n^3k)$, де k – різниця найбільшого і найменшого можливих значень сумарної продуктивності. Справді, в гіршому випадку ми почнемо з найгіршої (найменшої) продуктивності, і кожна ітерація покращить наш результат не менш ніж на 1, тому всього необхідно не більше k ітерацій.

1. *Таха Хемди А.* Введение в исследование операций. 6-е издание: Пер. с англ.– М.: Издательский дом «Вильямс», 2001.– 912 с.

2. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ.– М.: МЦНМО, 2000.– 960 с.

M. V. Rybak

ONE METHOD FOR SOLUTION OF THE PROBLEM OF OPTIMAL PERMUTATION ESTIMATION

Among the problems which cannot be solved in polynomial time there is a subclass of problems that can be solved by means of certain iterative process where each iteration is used to improve the result achieved by all the previous ones. Hence, provided there are some certain constraints for the criterion function (for instance, finiteness of it's range), there is a polynomial solution in some sense — the solution is linearly dependent on the potency of the range of this function. In this work one can find an example of such iterative algorithm.