Управляемые системы, ИМ СО АН СССР, 1988 г., вып. 28

УДК 519.176+519.854

ЗАДАЧА КОММИВОЯЖЕРА НА ЛЕНТОЧНЫХ ГРАФАХ

В.В.Сервах

Пусть задан n -вершинный ориентированный граф G с множеством дуг U . Занумеруем вершины числами $1, 2, \ldots, n$. Дугу из U будем задавать упорядоченной парой номеров вершин (i, j) . Каждой дуге $(i, j) \in U$ припишем неотрицательный вес $c_{ij} \in Z^+$. Требуется в графе G построить гамильтонов контур минимального веса.

Эта задача, известная как задача коммивояжера, NP -полна в сильном смысле. Ее NP -полнота установлена во многих частных случаях. Например, в [1] показано, что она остается NP -полной, если коммивояжеру на каждом шаге необходимо посетить одну из двух заранее заданных вершин. Учитывая трудность решения задачи коммивояжера, важным направлением в ее исследованиях является поиск полиномиально разрешимых частных случаев. Такие исследования проводились в работах [2 - 4] и др. Этому же посвящена предлагаемая статья.

Зафиксируем натуральное число $t < n$. Граф G называется ленточным (с лентой ширины t), если существует такая нумерация его вершин числами $1, 2, \ldots, n$, что для любой дуги $(i, j) \in U$ имеет место неравенство $|i - j| \leq t$. Соответствующую нумерацию вершин будем считать заданной. В настоящей работе показано, что при любом фиксированном значении t задача поиска гамильтонова контура минимального веса на ленточном графе является полиномиально разрешимой. Для решения этой задачи использовалась схема метода динамического программирования, отличающаяся от предложенной в [5]. Трудоемкость построенного алгоритма растет линейно с ростом n и составляет не более $O(nt!)$ операций.

§ 1. Общая схема алгоритма

Пусть G - ленточный граф. Выделим в нем подграф G_k со следующими свойствами:

1) множеством вершин G_k является $\{1, 2, \ldots, k\}$, где $k \leq n$;

2) подграф G_k не содержит контуров;

3) в каждую вершину входит не более одной дуги;

4) из каждой вершины выходит не более одной дуги.

Такой подграф G_k назовем частичным обходом вершин графа G (или k -обходом). Вес k -обхода определим как суммарный вес входящих в него дуг. Каждая компонента связности G_k является либо простым путем, либо изолированной вершиной. Пусть m - число простых путей некоторого k -обхода. Занумеруем эти пути числами $1, 2, \ldots, m$ и сформируем два упорядоченных множества $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ и $B = \{\beta_1, \beta_2, \ldots, \beta_m\}$, где α_i - начальная, а β_i - конечная вершины i -го пути. Обозначим через $V = \{v_1, v_2, \ldots, v_l\}$ совокупность внутренних вершин путей. Набор $\langle k, m, A, B, V \rangle$ назовем состоянием обхода вершин графа G, или просто состоянием. Заметим, что в информационном наборе $\langle k, m, A, B, V \rangle$ не указывается, каким путям принадлежат вершины из V. Поэтому состояние $\langle k, m, A, B, V \rangle$ может достигаться при различных k -обходах. Тот из них, вес которого минимален, назовем оптимальным. Соответствующий вес обозначим через $S \langle k, m, A, B, V \rangle$.

Суть предлагаемого для решения задачи алгоритма заключается в последовательном достраивании частичных обходов графа G до гамильтонова контура, причем достраивается не каждый k -обход, а только оптимальные. Введем особое состояние $\langle n, 0, \emptyset, \emptyset, \{1, 2, \ldots, n\} \rangle$, называемое конечным и соответствующее гамильтонову обходу вершин графа. Нам необходимо вычислить $S \langle n, 0, \emptyset, \emptyset, \{1, 2, \ldots, n\} \rangle$. Это может быть сделано методом динамического программирования, в котором значение $S \langle n, 0, \emptyset, \emptyset, \{1, 2, \ldots, n\} \rangle$ вычисляется рекуррентно через $S \langle k, m, A, B, V \rangle$. Вывод соответствующих формул приведен в следующем параграфе.

Хорошей иллюстрацией применения метода динамического программирования для поиска минимального по весу гамильтонова контура является работа Беллмана [5]. В ней рассматриваются состояния с $k = n$ и $m = 1$. Множества A и B содержат только по одному элементу. Тем самым состояние $\langle n, 1, \{\alpha\}, \{\beta\}, V \rangle$ может быть задано в более простом виде как $\langle \alpha, \beta, V \rangle$, где α - начальная, β - конечная вершины единственного пути, а V - множество внутренних вершин. Поиск оптимального решения осуществляется индуктивно по мощности множества V. Очевидно, что $S \langle \alpha, \beta, \emptyset \rangle = C_{\alpha\beta}$. Оптимум для состояния $\langle \alpha, \beta, \{v_1, v_2, \ldots, v_l\} \rangle$ вычисляется по рекуррентной формуле

$$S \langle \alpha, \beta, V \rangle = \min_{j = \overline{1,l}} \{ C_{v_j \beta} + S \langle \alpha, v_j, V \setminus \{v_j\} \rangle \}.$$

Длина оптимального гамильтонова контура равна

$$S \langle 0, 0, \{1, 2, \ldots, n\} \rangle =$$

$$= \min_{j=\overline{1,n}} \{c_{nj} + S\langle j, n, \{1,2,\ldots,n\} \setminus \{j,n\}\rangle\}.$$

Сам контур восстанавливается на втором этапе метода динамического программирования. Пусть минимум в последней формуле достигается на некоторой вершине j_0. Тогда дуга (n, j_0) включается в оптимальное решение. Следующая дуга определяется из условия достижения значения $S\langle j_0, n, \{1, 2,\ldots,n\} \setminus \{j_0, n\}\rangle$. Аналогично действуем далее, пока оптимальный контур не будет восстановлен.

§ 2. Основные соотношения для ленточных графов

Пусть G - ленточный граф с лентой ширины t. Поставим ему в соответствие граф G^* с тем же множеством вершин $\{1, 2, \ldots, n\}$ и множеством дуг $U^* = \{(i,j) \mid |i-j| \le t\}$, доопределяя $c_{ij} = \infty$, если $(i,j) \in U^* \setminus U$. Очевидно, что если оптимум на G^* не ограничен, то в G нет допустимого решения. Ниже будем иметь дело только с графом G^*. Рассмотрим простейшие случаи. При $t = 1$ ленточный граф гамильтонова контура не содержит. Если $t = 2$, то достаточно сравнить только два решения:

$$(1, 2, 4, 6, \ldots, 7, 5, 3);$$

$$(1, 3, 5, 7, \ldots, 6, 4, 2).$$

Однако уже при $t \ge 3$ количество гамильтоновых контуров в G^* растет экспоненциально с ростом n.

Назовем (k,t) -обходом k -обход, если вершины $1, 2, \ldots, k-t$ являются внутренними. Сам (k,t) -обход возникает тогда, когда из гамильтонова контура графа G^* удаляют вершины $k+1, k+2, \ldots, n$ и смежные с ними дуги. А так как граф G^* ленточный и, следовательно, между вершинами множеств $\{1, 2, \ldots, k-t\}$ и $\{k+1, k+2, \ldots, n\}$ дуг нет, то $\{1, 2, \ldots, k-t\} \subset V$.

Состояние $\langle k, m, A, B, V\rangle$ назовем (k,t) -состоянием, если $\{1, 2, \ldots, k-t\} \subset V$. Здесь $\{1, 2, \ldots, k-t\} = \emptyset$ при $k \le t$.

Т е о р е м а 1. Для вычисления оптимального решения задачи коммивояжера на ленточном графе G^* достаточно знать оптимумы $S\langle k, m, A, B, V\rangle$ для (k,t) -состояний $(k = 1, 2, \ldots, n-1)$.

Д о к а з а т е л ь с т в о. Необходимо вычислить $S\langle n, 0, \emptyset, \emptyset, \{1, 2, \ldots, n\}\rangle$. Любой гамильтонов контур состоит из двух дуг, смежных с вершиной n, и $(n-1)$-обхода. Существенно, что в ленточном графе

выделенной частью гамильтонова контура будет $(n-1,t)$ -обход. Причем для оптимальности задачи коммивояжера $(n-1,t)$ -обход должен быть оптимальным. Это обычный принцип оптимальности Беллмана. Перебирая возможные варианты расположения дуг, смежных с вершиной n, получаем следующую рекуррентную формулу:

$$S<n,0,\emptyset,\emptyset,\{1,2,\ldots,n\}> =$$

$$= \min_{\substack{i\in\{n-t,\ldots,n-1\}\\ j\in\{n-t,\ldots,n-1\}\\ j\neq i}} \min \{c_{jn}+c_{ni}+S<n-1,1,\{i\},\{j\},$$

$$\{1,2,\ldots,n\}\setminus\{i,j\}>\}.$$

Далее, для $k=n-1,n-2,\ldots,1$ теорема доказывается индуктивно. При вычислении $S<k+1,m,\mathcal{A},\mathcal{B},V>$ возможные $(k+1,t)$ -обходы разбиваем на две части: дуги, смежные с $k+1$ (их может быть 2, 1 или 0), и k -обход, который в ленточном графе является (k,t) -обходом. И снова в соответствии с принципом Беллмана для достижения оптимума $S<k+1, m,\mathcal{A},\mathcal{B},V>$ (k,t) -обход должен быть оптимальным. Теорема доказана.

Приведем некоторые обозначения и предположения. Пусть

$$P_t^k = \begin{cases} \{k-t+1,\ k-t+2,\ \ldots,\ k\} & \text{, если } k>t, \\ \{1,2,\ldots,k\} & \text{, если } k\leq t. \end{cases}$$

В силу теоремы 1 будем рассматривать только (k,t) -состояния $<k,m,\mathcal{A},\mathcal{B},V>$, и, следовательно, все начальные и конечные вершины путей лежат в множестве P_t^k, т.е.

$$\mathcal{A}=\{\alpha_1,\alpha_2,\ldots,\alpha_m\}\subset P_t^k,\quad \mathcal{B}=\{\beta_1,\beta_2,\ldots,\beta_m\}\subset P_t^k.$$

Без ограничения общности пути можно занумеровать таким образом, что в множестве $\mathcal{A}$ выполнено $\alpha_1<\alpha_2<\ldots<\alpha_m$.

Перейдем к описанию алгоритма. Первоначально имеем:

$$S<0,0,\emptyset,\emptyset,\emptyset>=0,$$

$$S<2,0,\emptyset,\emptyset,\emptyset>=0,$$

$$S<2,1,\{1\},\{2\},\emptyset>=c_{12},$$

$$S<2,1,\{2\},\{1\},\emptyset>=c_{21}.$$

Далее, рассматривая состояние $\langle k, m, A, B, V \rangle$ при $3 \le k < n$, считаем, что определены все значения $S \langle k', m', A', B', V' \rangle$ для (k', t) -состояний при $k' < k$. Вычислим теперь $S \langle k, m, A, B, V \rangle$.

Пусть $V = \emptyset$. Это возможно только при $k \le t$. Из-за отсутствия внутренних вершин путь из α_i можно провести только непосредственно в β_i, $i = \overline{1, m}$. Следовательно,

$$S \langle k, m, A, B, \emptyset \rangle = \sum_{i=1}^{m} c_{\alpha_i \beta_i}.$$

При $V \neq \emptyset$ рассмотрим следующие возможные случаи.

Случай 1. Пусть $k \notin A \cup B \cup V$, т.е. вершина k изолирована и не выходит ни в один путь. Тогда

$$S \langle k, m, A, B, V \rangle = S \langle k-1, m, A, B, V \rangle.$$

Случай 2. Пусть $k \in B$, т.е. k является концом некоторого пути. Положим $k = \beta_i$. При $m = 1$ путь из α_1 в k проходит через все вершины множества V. Так как $V \neq \emptyset$, то вершине k должна предшествовать вершина из V. Значит, оптимальный путь из α_1 в k состоит из дуги вида (v, k) и оптимального пути из α_1 в v, проходящего через вершины множества $V \setminus \{v\}$. Для полноты необходимо рассмотреть все возможные значения v. В данном случае это такие $v \in V$, для которых $|v - k| \le t$, т.е. $v \in V \cap P_t^{k-1}$. Получаем

$$S \langle k, 1, \{\alpha_1\}, \{k\}, V \rangle =$$

$$= \min_{v \in V \cap P_t^{k-1}} \{c_{vk} + S \langle k-1, 1, \{\alpha_1\}, \{v\}, V \setminus \{v\} \rangle\}.$$

Пусть теперь $m \ge 2$. Тогда вершине k может предшествовать как вершина $v \in V \cap P_t^{k-1}$, так и непосредственно вершина α_i. В последнем варианте лучшее решение получаем из дуги (α_i, k) и оптимума для состояния $\langle k-1, m-1, A \setminus \{\alpha_i\}, B \setminus \{\beta_i\}, V \rangle$, где

$$A \setminus \{\alpha_i\} = \{\alpha_1, \alpha_2, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_m\},$$

$$B \setminus \{k\} = \{\beta_1, \beta_2, \dots, \beta_{i-1}, \beta_{i+1}, \dots, \beta_m\}.$$

В совокупности имеем

$$S \langle k, m, A, B, V \rangle =$$

$$= min \begin{cases} \min\limits_{v \in V \cap P_t^{k-1}} \{ c_{v,k} + S < k-1, m, A, B \cup \{v\} \setminus \{k\}, V \setminus \{v\} > \}, \\ c_{\alpha_i k} + S < k-1, m, A \setminus \{\alpha_i\}, B \setminus \{k\}, V > . \end{cases}$$

Здесь

$$B \cup \{v\} \setminus \{k\} = \{\beta_1, \beta_2, \ldots, \beta_{i-1}, v, \beta_{i+1}, \ldots, \beta_m\} .$$

Это означает, что $\beta_i = k$ заменено новым $\beta_i = v$, так как концом i-го пути теперь является вершина v.

Случай 3. Пусть $k \in A$, т.е. k является началом некоторого пути. Множество A упорядочено по возрастанию. Поэтому $k = \alpha_m$. Рассуждая как и в предыдущем случае, получаем следующие рекуррентные соотношения:

Если $m = 1$, то

$$S < k, 1, \{k\}, \{\beta_1\}, V > =$$

$$= \min_{v \in V \cap P_t^{k-1}} \{ c_{kv} + S < k-1, 1, \{v\}, \{\beta_1\}, V \setminus \{v\} > \} .$$

Если $m \geq 2$, то

$$S < k, m, A, B, V > =$$

$$= min \begin{cases} \min\limits_{v \in V \cap P_t^{k-1}} \{ c_{kv} + S < k-1, m, A', B', V \setminus \{v\} > \}, \\ c_{k\beta_m} + S < k-1, m-1, A \setminus \{k\}, B \setminus \{\beta_m\}, V > . \end{cases}$$

Здесь

$$A \setminus \{k\} = \{\alpha_1, \alpha_2, \ldots, \alpha_{m-1}\},$$

$$B \setminus \{\beta_m\} = \{\beta_1, \beta_2, \ldots, \beta_{m-1}\} .$$

Множество A' получено из A заменой значения $\alpha_m = k$ новым $\alpha_m = v$. Причем новую последовательность $(\alpha_1, \alpha_2, \ldots, \alpha_{m-1}, v)$ необходимо упорядочить по возрастанию. Если l таково, что $\alpha_l < v < \alpha_{l+1}$, то

$$A' = \{\alpha_1, \alpha_2, \ldots, \alpha_l, v, \alpha_{l+1}, \ldots, \alpha_{m-1}\}$$

и, следовательно,

$$\mathcal{B}' = \{\beta_1, \beta_2, \ldots, \beta_\ell, \beta_m, \beta_{\ell+1}, \ldots, \beta_{m-1}\}.$$

Случай 4. Пусть $k \in V$, т.е. k - внутренняя вершина. Она может входить в любой из m путей. Опишем четыре возможные ситуации, выделяя две дуги, смежные с вершиной k, и то $(k-1, t)$ -состояние, которое возникает после удаления этих дуг:

1) $(\alpha_i, k), (k, \beta_i)$. Полученное состояние $\langle k-1, m-1, A \setminus \{\alpha_i\}, \mathcal{B} \setminus \{\beta_i\}, V \setminus \{k\}\rangle$ обозначим для краткости через $\langle * \rangle$.

2) $(\alpha_i, k), (k, v)$. Состояние $\langle k-1, m, A \cup \{v\} \setminus \{\alpha_i\}, \mathcal{B}, V \setminus \{k, v\}\rangle$ обозначим через $\langle *, * \rangle$. Здесь множество $A \cup \{v\} \setminus \{\alpha_i\}$ необходимо доупорядочить по неубыванию номеров вершин и соответственно изменить $\mathcal{B}$.

3) $(v, k), (k, \beta_i)$. Состояние $\langle k-1, m, A, \mathcal{B} \cup \{v\} \setminus \{\beta_i\}, V \setminus \{v, k\}\rangle$ обозначим через $\langle *, *, * \rangle$.

4) $(v, k), (k, w)$, где $v \in V$ и $w \in V$. Здесь $\langle k-1, m+1, A \cup \{w\}, \mathcal{B} \cup \{v\}, V \setminus \{v, w, k\}\rangle = \langle * * * * \rangle$. (Ситуация, в которой путей при удалении дуг стало больше, в предыдущих случаях не встречалась.)

Окончательно имеем

$$S\langle k, m, A, \mathcal{B}, V\rangle =$$

$$= \min_{i=\overline{1,m}} \min \begin{cases} c_{\alpha_i k} + c_{k\beta_i} + S\langle * \rangle, \\ c_{\alpha_i k} + \min\limits_{v \in V \cap P_t^{k-1} \setminus \{k\}} \{c_{kv} + S\langle *, * \rangle\}, \\ c_{k\beta_i} + \min\limits_{v \in V \cap P_t^{k-1} \setminus \{k\}} \{c_{vk} + S\langle *, *, * \rangle\}, \\ \min\limits_{v \in V \cap P_t^{k-1} \setminus \{k\}} \min\limits_{\substack{w \in V \cap P_t^{k-1} \setminus \{k\} \\ w \neq v}} \{c_{vk} + c_{kw} + S\langle *, *, *, * \rangle\}, \end{cases}$$

Все случаи при $k < n$ описаны. Случай $k = n$ разобран в теореме 1. Для восстановления самого контура всюду в процессе вычислений необходимо фиксировать дугу (или дуги), на которых достигается значение $S\langle k, m, A, \mathcal{B}, V\rangle$. По этим дугам на втором этапе метода динамического программирования восстанавливаем оптимальное решение.

§ 3. Трудоемкость алгоритма

В дискретной оптимизации определяющими характеристиками алгоритма является количество элементарных операций, выполняемых при его реализации, и объем используемой при этом памяти. Для приближенных алгоритмов важно также знать качество полученного решения, т.е. его отклонение от оптимума. В нашем случае имеет место следующая

Т е о р е м а 2. Для получения описанным алгоритмом оптимального решения задачи коммивояжера на ленточном графе G^* требуется не более $O(nt!)$ арифметических операций.

Д о к а з а т е л ь с т в о. В процессе работы алгоритма вычисляются значения $S\langle k, m, \mathcal{A}, \mathcal{B}, V\rangle$ для всех (k,t)-состояний. Посчитаем количество (k,t)-состояний при фиксированном $k \geq t$. Вершины $1,2,\ldots,k-t$ - внутренние. Вершины множества $P_t^k = \{k-t+1, k-t+2, \ldots, k\}$ распределены следующим образом. Некоторое количество из них, пусть ρ, являются внутренними, m - начальными, m - конечными, а остальные - изолированными. Здесь $0 \leq \rho \leq t-2$, $1 \leq m \leq \left[\frac{t-\rho}{2}\right]$. Внутренние вершины выбираем C_t^ρ способами. Остается $t-\rho$ вершин. Множество $\mathcal{A}$ упорядочено, и поэтому его формируем $C_{t-\rho}^m$ способами. При построении $\mathcal{B}$ концом первого пути может стать любая из $t-\rho-m$ оставшихся вершин, концом второй - любая из $t-\rho-m-1$ вершин и т.д. Всего имеем

$$\prod_{i=0}^{m-1}(t-\rho-m-i) = \frac{(t-\rho-m)!}{(t-\rho-2m)!}$$

вариантов.

Получаем, что общее число (k,t)-состояний при фиксированном равно

$$\sum_{\rho=0}^{t-2} C_t^\rho \sum_{m=1}^{\left[\frac{t-\rho}{2}\right]} C_{t-\rho}^m \cdot \frac{(t-\rho-m)!}{(t-\rho-2m)!} =$$

$$= \sum_{\rho=0}^{t-2} \frac{t!}{\rho!(t-\rho)!} \sum_{m=1}^{\left[\frac{t-\rho}{2}\right]} \frac{(t-\rho)!}{m!(t-\rho-m)!} \cdot \frac{(t-\rho-m)!}{(t-\rho-2m)!} =$$

$$= t! \sum_{\rho=0}^{t-2} \frac{1}{\rho!} \sum_{m=1}^{\left[\frac{t-\rho}{2}\right]} \frac{1}{m!(t-\rho-2m)!} \cdot$$

Для вычисления $S<k,m,A,B,V>$ требуется не более $O(m^2(p+1)^2)$ операций. Таким образом, трудоемкость алгоритма с учетом случаев $k<t$ и $k=n$ равна

$$T=O\Big(\sum_{k=1}^{t-1} t!\sum_{p=0}^{k-1}\frac{1}{p!}\sum_{m=1}^{\left[\frac{k-p}{2}\right]}\frac{m^2(p+1)^2}{m!(k-p-2m)!}+$$

$$+\sum_{k=t}^{n-1} t!\sum_{p=0}^{t-2}\frac{1}{p!}\sum_{m=1}^{\left[\frac{t-p}{2}\right]}\frac{m^2(p+1)^2}{m!(t-p-2m)!}+t(t-1)\Big).$$

Оценим сумму

$$G=\sum_{p=0}^{t-2}\frac{1}{p!}\sum_{m=1}^{\left[\frac{t-p}{2}\right]}\frac{m^2(p+1)^2}{m!(t-p-2m)!}=$$

$$=\sum_{p=0}^{t-2}\frac{(p+1)^2}{p!}\sum_{m=1}^{\left[\frac{t-p}{2}\right]}\frac{m^2}{m!(t-p-2m)!}\cdot$$

Так как

$$\sum_{z=0}^{\infty}\frac{z^2}{z!}<2\ell+1,$$

то

$$\sum_{m=1}^{\left[\frac{t-p}{2}\right]}\frac{m^2}{m!(t-p-2m)!}\le\sum_{m=1}^{\left[\frac{t-p}{2}\right]}\frac{m^2}{m!}<2\ell+1.$$

Следовательно

$$G<\sum_{p=0}^{t-2}\frac{(p+1)^2}{p!}\cdot(2\ell+1)=(2\ell+1)\Big(\sum_{p=0}^{t-2}\frac{p^2}{p!}+\sum_{p=0}^{t-2}\frac{2p}{p!}+\sum_{p=0}^{t-2}\frac{1}{p!}\Big)<$$

$$<(2\ell+1)\big[(2\ell+1)+2(\ell+1)+\ell\big]=(2\ell+1)(5\ell+3)=const.$$

Окончательно получаем

$$T=O\Big(\sum_{k=1}^{t-1}t!+\sum_{k=t}^{n-1}t!+t(t-1)\Big)=O(nt!).$$

Теорема доказана.

С л е д с т в и е. При фиксированном t описанный алгоритм имеет линейную трудоемкость в зависимости от числа вершин. Тем самым задача коммивояжера в этом случае принадлежит классу P полиномиально разрешимых задач.

Сделаем несколько замечаний относительно памяти, используемой при работе алгоритма. Для восстановления оптимального контура требуется хранить информацию об элементах, на которых достигается $S\langle k, m, A, B, V\rangle$. При этом объем памяти составит $O(nt!)$. Если же необходимо знать только вес оптимального контура, то достаточно $O(t!)$ ячеек памяти, так как при вычислении $S\langle k, m, A, B, V\rangle$ используются только оптимумы для $(k-1, t)$ -состояний.

§ 4. Приближенный алгоритм решения задачи коммивояжера

В этом параграфе описана реализация приближенного алгоритма решения задачи коммивояжера, известного под названием "подключение к контуру". Использование этого малотрудоемкого алгоритма позволяет получать хорошие приближенные решения на ленточных графах больших размерностей. Трудоемкость получения одного гамильтонова контура с дугами из U^* не превышает $O(nt)$ операций. Причем чем меньше ширина ленты графа G^*, тем быстрее и точнее работает алгоритм.

Первоначально формируется контур $(1,2)$ из двух первых вершин. На каждом шаге алгоритма к построенному контуру будем подключать очередную вершину. После завершения шага k на множестве вершин $\{1, 2, \ldots, k\}$ образуется контур $(\gamma_1, \gamma_2, \ldots, \gamma_k)$ веса S_k. Кроме того, известно множество дуг $U_k = \{(\gamma_i, \gamma_{i+1})\}$, для которых выполнены неравенства $|k - \gamma_i| < t$ и $|k - \gamma_{i+1}| < t$. Опишем шаг $k+1$, на котором к контуру подключается вершина $k+1$.

1. Генерирование решений на множестве $\{1, 2, \ldots, k, k+1\}$. Всего строится $|U_k|$ решений путем замены в контуре дуги $(\gamma_i, \gamma_{i+1}) \in U_k$ на две дуги $(\gamma_i, k+1)$ и $(k+1, \gamma_{i+1})$. Каждый раз вычисляем новый вес:

$$S_k^i = S_k - C_{\gamma_i \gamma_{i+1}} + C_{\gamma_i, k+1} + C_{k+1, \gamma_{i+1}} .$$

Необходимо отметить, что новые дуги всегда допустимы в силу выбора множества U_k.

2. Выбор лучшего решения. Из $|U_k|$ решений выбираем контур $(\gamma_1, \gamma_2, \ldots, \gamma_{i_0}, k+1, \gamma_{i_0+1}, \ldots, \gamma_k)$ с наименьшим весом

$$S_{k+1} = \min_{(\gamma_i, \gamma_{i+1}) \in U_k} S_k^i .$$

Его будем использовать как начальный на следующем шаге.

3. Формирование множества U_{k+1}. Включим в U_{k+1} дуги $(\gamma_{i_0}, k+1)$ и $(k+1, \gamma_{i_0+1})$ и все дуги U_k. После этого исключим дуги, смежные с вершиной $k+1-t$, и дугу $(\gamma_{i_0}, \gamma_{i_0+1})$. На этом шаг k заканчивается.

Алгоритм завершает свою работу через n шагов. Трудоемкость шага k равна $O(|U_k|)$. Так как $|U_k| \leq t$, то трудоемкость всего алгоритма не превосходит $O(nt)$ операций. В заключение необходимо отметить, что описанная схема допускает множество различных вариантов реализации и очень эффективно показала себя на практике.

Поступила в ред.-изд. отдел
23 декабря 1987 г.

Литература

1. Сердюков А.И. О задаче коммивояжера при наличии запретов // Управляемые системы. - Новосибирск. - 1978. - № 17. - С. 80-86.

2. Айзенштат В.С., Максимович Е.П. Некоторые классы задач о бродячем торговце // Кибернетика. - Киев. - 1978. - № 4. - С. 80-83.

3. Кляус П.С. Структура оптимальных решений некоторых классов задач о коммивояжере // Изв. АН БССР. Серия физ.-мат. наук. - Минск. - 1976. - № 6. - С. 95-98.

4. Емеличев В.А., Супруненко Д.А., Танаев В.С. О работе белорусских математиков в области дискретной оптимизации // Изв. АН СССР. - М. - 1982. - № 6. - С. 25-45.

5. Беллман Р. Применение динамического программирования к задаче о коммивояжере // Кибернетический сб. - М.: Мир. - № 9. - С. 219-222.