ЖАДНЫЕ АЛГОРИТМЫ ВСТРЕЧНОГО ПОИСКА КРАТЧАЙШИХ МАРШРУТОВ НА СЕТЯХ

М.П. Ревотюк, Н.В. Хаджинова

Республика Беларусь, Минск

Объект рассмотрения – вычислительные схемы построения серверов поиска кратчайших маршрутов на графах транспортных сетей. Хотя задача поиска кратчайших маршрутов на графах с количеством вершин n характеризуется вычислительной сложностью $n \cdot \log_2 n$ [1], улучшение временных характеристик сервера, обсуживающего поток запросов в реальном времени, когда значение n фиксировано, представляет определенный практический интерес. Предмет обсуждения – структуры данных объектного представления моделей транспортных сетей и алгоритмы быстродействующих процедур оптимизации маршрутов.

Известно, что ключевой прием объектно-ориентированного проектирования – использование и агрегирование наиболее удачных решений. Будем рассматривать в качестве отправных следующие известные идеи построения быстродействующих процедур поиска деревьев оптимальных маршрутов:

применение волновых схем однократного сканирования дуг графа (алгоритма Дийкстры) и организация встречного поиска [1-5];

учет предопределенных решений жадными алгоритмами [2];

отображение очереди вершин дерева на адреса ее элементов [3].

С целью обоснования способа интеграции перечисленных идей шаблоном функции оптимизации маршрутов вначале определим структуры данных представления задачи и результата ее решения.

Известно, что наиболее предпочтительным представлением слабо связного нагруженного графа $G(N,A)$, состоящего из N вершин и A дуг, для волновых алгоритмов построения дерева, реализующих схему Дийкстры, является структура смежности *FSF* (*Forward Star Form*):

$$FSF = \langle n, H, B, W \rangle, \qquad (1)$$

где n – количество вершин графа транспортной сети, $n=|N|$; H – массив указателей списков смежности; B – массив элементов списков смежности (конечных вершин дуг графа); W – массив весов дуг.

Для каждой вершины $x \in N$ множество непосредственно достижимых смежных вершин обозначим x', $x' = \{k \mid w(x,k) \geq 0\}$.

Если вершины графа пронумерованы числами $\overline{0,n-1}$, то элементы (1) фиксируются так:

$$FSF = \begin{cases} H_0 = 0,\ H_{i+1} = H_i + |i'|,\ i = \overline{1,n-1}; \\ B(H_i + j) = i'(j), \\ W(H_i + j) = w(i, i'(j)),\ j = \overline{0,|i'|-1},\ i = \overline{0,n-1}. \end{cases} \quad (2)$$

Очевидно, что $|H|=n+1$, $|B|=|W|=H_n$ (элемент H_n содержит $|A|$).

Для выделения альтернатив развития дерева их вершины x в этом случае используется следующий простой алгоритм:

$$x' = \{(H_x + i) \mid H_x + i < H_{x+1},\ i \geq 0\}.$$

Представление графа в виде (1) достаточно для решения задач поиска маршрутов, но в [2] предложено для каждой вершины графа модели транспортной сети предварительно выделить и пометить входные дуги минимальной длины:

$$T_j = \left\{(i, j) : i = \arg\min_{i,\,j} \{w(i, j) : (i, j) \in A\}\right\},\ j \in N. \quad (3)$$

В случае ветвления процесса через дугу из множества (3) значение расстояния до ее конечной вершины не изменится. Такую вершину можно не включать в очередь, что существенно сокращает время вычислений, особенно при обработке потока запросов на поиск маршрутов.

Для отображения множества (3) достаточно выбрать элементы B или W в (1) со свободным двоичным разрядом и пометить дугу бинарным признаком. Таким образом, учет предопределенных решений практически не требует увеличения объема памяти для хранения описания реальной сети.

Состояние поиска решения представляется тремя элементами:

массивом расстояний $D = \{D_i, i \in N\}$;

массивом номеров предшествующих вершин $P = \{P_i, i \in N\}$;

очередью вершин $Q = \{Q_i, i \in N\}$, элементы которой упорядочены по текущему значению расстояния от корня дерева [6].

Достаточно часто на практике встречаются задачи поиска маршрута между двумя заданными вершинами сети. В этом случае целесообразно организовать процесс поиска путем построения двух встречно растущих деревьев. В результате объем анализируемых данных сокращается в два раза [4-5]. Дерево из конечной вершины должно строиться на графе с обратным направлением дуг, поэтому представление модели сети задается расширенным графом – объединением (1) и инверсии (1).

Встречный поиск принципиально можно реализовать любой процедурой построения дерева от одной вершины до всех остальных на расширенном графе, отражающем фактически существующее пространство поиска. Для этого есть достаточно веские причины:

схема организации списка критических (временно помеченных [1]) вершин имеет узкое место, связанное с холостыми проверками наличия очередей вершин с одинаковыми значениями расстояния;

оптимальный размер деревьев поиска соответствует одинаковому расстоянию от корней дерева [2].

Однако известные процедуры встречного поиска [5,6] не учитывают особенностей остановки процесса при использовании предопределенных решений [4].

Обозначим исходный граф, заданный в виде (1), через $G^{+}(N^{+},A^{+})$, а граф с инвертированием направления дуг – $G^{*}(N^{*},A^{*})$.

Множество вершин и дуг графа $G^{*}(N^{*},A^{*})$ определим так:

$$\begin{cases} N^{*} = \{x^{*} = x^{+} + |N^{+}|,\ x^{+} \in N^{+}\}, \\ A^{*} = \{(x^{*}, y^{*}) = (y^{+} + |N^{+}|, x^{+} + |N^{+}|),\ (x^{+}, y^{+}) \in A^{+}\}. \end{cases} \quad (4)$$

Вершины x^{+} и x^{*} будем далее называть сопряженными. Легко заметить, что условие отбора предопределенных значений (3) для графа $G^{*}(N^{*},A^{*})$ имеет вид

$$T_j^{*} = \left\{ (i,j) : j = \arg\min_{i,j} \{ w(i,j) : (i,j) \in A^{+} \} \right\},\ j \in N^{+}.$$

В общем случае связь сопряженных вершин, не обязательно определяемая (4), пусть задается функцией

$$conj(x) = x^{*} \cdot (x \in G^{+}) + x^{+} \cdot (x \in G^{*}),\ x \in N^{+} \cup N^{*}. \quad (5)$$

В случае же нумерации вершин по правилу (4)

$$conj(x) = (x + |N^{+}|) \cdot (x < |N^{+}|) + (x - |N^{+}|) \cdot (x \geq |N^{+}|)\ x \in N^{+} \cup N^{*}.$$

Пусть заданы s и f – начальная и конечная вершины исходного графа G^{+}.

Так как, по определению (4), $N^{+} \cap N^{*} = \varnothing$, то построение деревьев можно проводить параллельно с синхронным движением от корней дерева на графе $G(N,A)=G^{+}(N^{+},A^{+})+G^{*}(N^{*},A^{*})$. Для этого достаточно начать процесс ветвления из вершин $s \in G^{+}$ и $f^{*} = conj(f)$, $f^{*} \in G^{*}$. В отличие от предложенного в [3] приема поочередного развития деревьев, синхронное движение оказывается оптимальным для “географических” графов.

Организация встречного поиска требует определения правила остановки. Можно показать, что остановка должна соответствовать моменту фиксации постоянной пометки вершины дерева, когда сопряженная вершина уже является постоянно помеченной.

Если для некоторого дерева кратчайших маршрутов максимальное расстояние от помеченных вершин до корня есть *d*, то признаком постоянной пометки вершины *x* является условие $D_x \le d$. В рассматриваемом случае для обоих деревьев значение *d* одинаково.

Отсюда следует, что правило остановки можно определить на значениях текущих расстояний – $D_{conj(i)} \le D_i$, где i – вершина графа $G^+(N^+,A^+)$ или графа $G^*(N^*,A^*)$, получающая постоянную пометку.

Таким образом, использование синхронного движения от корней деревьев не требует хранения пометок, а момент остановки совпадает с моментами выделения критических вершин дерева маршрутов.

После остановки в вершине *x* остается достроить маршрут до конечной вершины в исходном графе.

Так как остановка может быть обнаружена в любом из встречно растущих деревьев, а результат поиска необходимо получить лишь для дерева из исходной вершины, то для перехода в такое дерево используем функцию

$$orig(x) = x \cdot (x \in G^+) + conj(x) \cdot (x \in G^*),\ x \in N^+ \bigcup N^*.$$

В случае нумерации вершин расширенного графа по правилу (4)

$$orig(x) = x \cdot (x < |N^+|) + (x - |N^+|) \cdot (x \ge |N^+|),\ x \in N^+ \bigcup N^*.$$

Пусть описание графа, а также данные представления результатов поиска представлены глобальными объектами программы или элементами данных класса.

Идея реализации алгоритма встречного поиска, учитывающая предопределенные решения (3), может быть представлена следующими процедурами в стиле кодирования [3,6].

```
Procedure SPT(s,f); // Поиск маршрута между вершинами s и f
 begin
  foreach i∈N+ ⋃ N* do Di = ∞, Pi = i;
  Ds = Dconj(f) = 0; // Фиксация корней прямого и обратного дерева
  QINIT(Q,s); // Q - очередь вершин, вначале включает s
  QIN(Q,conj(f)); // Включение в очередь корня обратного дерева
  repeat
   QOUT(Q,i); // Выборка вершины i из очереди Q
  until SPG(i)=false and Q=∅;
 end;
```

```
Function SPE(i):boolean; // Развитие дерева из вершины i
 begin
  SPE = D_conj(i) > D_i ; // Признак остановки процесса поиска
  if SPE then begin
   foreach j ∈ i' do
    if D_j > D_i + w(i,j) then begin
      D_j = D_i + w(i,j), P_j = i ;
     QIN(Q,j); // Включение вершины j в очередь Q
    end;
   Exit;
  end;
  FIN(i);
 end;

Function SPG(i): boolean; // Развитие дерева из вершины i
 begin
  foreach j ∈ t_i do
   if D_j > D_i + w(i,j) then begin
     D_j = D_i + w(i,j), P_j = i ;
    SPG = SPE(j); // Признак остановки процесса поиска
    if not SPG then Exit;
   end;
   foreach j ∈ i'\t_i do
    if D_j > D_i + w(i,j) then begin
      D_j = D_i + w(i,j), P_j = i ;
     QIN(Q,j); // Включение вершины j в очередь Q
    end;
   SPG = true;
  end;

Procedure FIN(i); // Копирование части маршрута из вершины i
 begin
  j = conj(orig(i));
  while P_j ≠ j do
   begin
     D_orig(P_j) = D_orig(j) + D_Pj − D_j , P_orig(P_j) = orig(j);
    j = P_j;
   end;
 end;
```

Здесь процедура FIN реализует дополнение дерева из исходной вершины информацией о найденном участке маршрута от конечной вершины. Нетрудно проверить, что предлагаемый алгоритм остается корректным при распараллеливании процессов

Таким образом, построенные процедуры поиска используют для представления модели сети память в два раза увеличенного объема. Однако структура представления модели сети остается эффективнее

матричных моделей, по крайней мере, с точки зрения удобства коррекции описания сети в реальном времени. Вместе с тем, синхронизация процессов выборки критических вершин позволяет сократить количество холостых проверок состояния очереди для наиболее эффективных адресных схем организации очередей [6].

Проведенные рассуждения базировались на предположении, что состояние процесса поиска отражается на очереди вершин – листьев текущего дерева. В случае наличия ограничений на пути на графе [4] приходится использовать очередь дуг [2]. Однако легко заметить, что и здесь помеченные дуги (3) по тем же соображениям являются пригодными для ускоренного включения в дерево кратчайших путей. Правило остановки остается без изменений.

Эксперименты показывают, что на графе реальной сети автомобильных дорог, где $|N| \cong 10^4$, $|A| \cong 10^5$, среднее время поиска кратчайших маршрутов между случайными парами вершин сокращается более, чем в два раза.

Литература

1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ/Пер. с англ. – М.: МЦМНО, 2002. – 960 с.

2. Ревотюк М.П., Застенчик Н.И., Шешко Е.В. Поглощение предопределенных решений жадными алгоритмами//Известия Белорусской инженерной академии, № 1(17)/2, 2004. – С. 112-114.

3. Ревотюк М.П., Дарадкех Ю.И. Использование предопределенных решений алгоритмами регулярного поиска кратчайших путей// Информационные системы и технологии (IST'2004): Материалы II Межд. конф. (Минск, 8-10 ноября 2004 г.): В 2.ч. Ч.2. – Мн.: Академия управления при Президенте Республики Беларусь, 2004. – С. 127-132.

4. Ревотюк М.П., Дарадкех Ю.И., Кирейчук В.А. Поиск кратчайших путей на графах полиморфных сетей с ограничениями//Известия Белорусской инженерной академии, № 1(17)/1, 2004. – С. 126-129.

5. Стенбринк П.А. Оптимизация транспортных сетей/Пер. с англ. – М.: Транспорт, 1981. – 320 с.

6. Deo N., Chi-yin Pang. Shortest-Path Algorithm: Taxonomy and Annotation//Networks, 1984. – Vol. 14. – P. 275-323.