УДК 656.2-50: 519.8

БЫСТРЫЙ ПОИСК КРАТЧАЙШИХ ПУТЕЙ НА ГРАФАХ С ПРЕДОПРЕДЕЛЕННЫМИ РЕШЕНИЯМИ

М.П. РЕВОТЮК, М.К. КАРОЛИ, Н.В. ХАДЖИНОВА

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 31 марта 2014

Предлагаются приемы ускорения многократного поиска кратчайших путей на графах, когда порядок порождаемых деревьев путей существенно меньше порядка графа. Однократная инициализация переменных состояния и выделение предопределенных решений снижает сложность поиска путей до линейной зависимости от объема сканируемого пространства.

Ключевые слова: транспортные сети, кратчайшие пути, вычислительная сложность.

Постановка задачи

Известно, что при поиске кратчайших путей на нагруженном ориентированном графе $G(N, A)$, где N – множество вершин, A – множество дуг с весовой функцией $W: A \to R^+$, время построения дерева путей одним из лучших для подобной задачи алгоритмом Дийкстры растет в первом приближении по закону $x \cdot \log_2 x$ с увеличением расстояния x от корня дерева. Особенность алгоритма Дийкстры – однократный просмотр дуг формируемого дерева. Это отражается появлением в асимптотиках вычислительной сложности различных реализаций процедур поиска количества дуг. Например, реализация алгоритма Дийкстры с отображением очередей анализируемых вершин на кучи Фибоначчи характеризуется сложностью $O(m + n \cdot \log_2 n)$, где $m = |A|$, $n = |N|$. Отображение очереди вершин на вектор размером L позволяет снизить сложность до величины $O(m + n \cdot L)$, где L – максимальная длина дуги графа [1, 2].

Существует ряд приемов ускорения процесса поиска кратчайших путей, использующих идею сокращения количества анализируемых дуг графа: целенаправленный поиск; встречный поиск; многоуровневый подход; ограничение локальных областей поиска [2, 3]. Реализация перечисленных приемов предполагает предварительное формирование вспомогательного графа, отображающего вершины исходного графа и деревья кратчайших путей. Затем построение дерева путей идет по волновой схеме однократного просмотра дуг, реализуемой алгоритмом Дийкстры. Результат отображается на исходный граф за время $O(m)$.

Наличие в асимптотиках вычислительной сложности параметра n отражает необходимость выполнения перед построением дерева действий, сложность которых $O(n)$. Это снижает эффективность алгоритма Дийкстры в задачах определения множества кратчайших путей. Пример подобной задачи – расчет подматриц кратчайших расстояний: необходимо построить кратчайшие пути от заданного множества исходных вершин $S \in N$ до всех вершин из множества $F \in N$. Практически всегда в таких задачах выполняется условие $|F| < n$.

Расчет подматриц можно проводить прямолинейным использованием агоритма Флойда, но при этом для любых значений $|S|$ и $|F|$ вычислительные затраты имеют оценку $O(n^3)$, а потребность в памяти – $O(n^2)$ [1]. Для разреженных графов реальных транспортных сетей в

таких случаях лучшим оказывается алгоритм Дийкстры, где при потребности в памяти $O(n+m)$ оценка вычислительных затрат – не хуже $|S|O(n^2)$. Далее предлагаются приемы улучшения таких оценок, основанные на исключении излишних операций при построении множества деревьев кратчайших путей для всех элементов $S \in N$.

Ревизия алгоритма построения деревьев кратчайших путей

Известно, что вполне достаточным для реализации алгоритма Дийкстры представлением разреженного нагруженного графа $G(N,A)$ является структура смежности *FSF* (*Forward Star Form*) [1]. В этом случае для каждой вершины x эффективно представлено множество непосредственно достижимых смежных вершин x', $x' = \{k \mid w(x,k) \geq 0\}$, где $w(x,k)$ – вес дуги $x \to k$, $x,k \in N$. Объем памяти для хранения структуры смежности – $O(n+m)$.

Пространство состояний поиска решения алгоритмом Дийкстры включает: D – массив расстояний от корня дерева, $D = \{D(i),\ i \in N\}$; P – массив номеров предшествующих вершин, $P = \{P(i),\ i \in N\}$; Q – очередь вершин, $Q = \{Q_i,\ i \in N\}$, элементы которой упорядочены по текущему значению расстояния от корня дерева [1, 2].

Анализ алгоритма Дийкстры показывает, что количество изменяющихся элементов пространства состояний поиска соответствует порядку итерационно создаваемого дерева кратчайших путей. Однако этот факт при кодировании алгоритма обычно не учитывается, неявно полагая степень исхода вершин графа соизмеримой со значением $n-1$.

Процесс построения дерева кратчайших путей имеет волновой характер до исчерпания возможности его развития из исходной вершины (рис. 1).

```
procedure SPT(r); //Построение дерева кратчайших путей
 begin
  foreach i ∈ N do D(i) ← ∞, P(i) ← i;
  D(r) ← 0; Q ← {r};
  repeat
   i ← Q1, Q ← Q \ {i};
   foreach j ∈ i' begin
    if D(j) > D(i) + w(i, j) then begin
     if D(j) < ∞ then Q ← Q \ {j};
     D(j) ← D(i) + w(i, j), P(j) ← i; Q ← Q ∪ {j};
    end;
   end;
  until Q = ∅;
 end;
```

Рис. 1. Традиционная версия реализации алгоритма Дийкстры

На итерациях построения дерева последовательно выполняются операции:

– выборка вершины из очереди вершин с минимальным расстоянием от корня дерева;

– развитие дерева из выбранной вершины, когда для всех ее выходных дуг выполняется процедура релаксации с включением новых вершин в очередь.

Вершина графа может находиться в следующих последовательно фиксируемых состояниях: не рассматривалась, рассматривается и рассмотрена. Признак состояния явно обычно не используется, а его отображение проводится неявно на элементах $D = \{D(i),\ i \in N\}$.

Все вершины графа перед построением дерева помечаются парами начальных значений $D(i) \leftarrow \infty,\ P(i) \leftarrow i,\ i \in N$. Принадлежность произвольной вершины $x \in N$ дереву путей на любом этапе его построения определяется истинностью выражения $(D(x) = \infty) \vee (P(x) = x)$. На практике именно вершины дерева путей значимы для задач управления транспортом.

Построенное дерево путей – связный граф по определению. Если s – исходная вершина, а t – лист дерева путей, $s,t \in N$, то после завершения поиска кратчайший путь можно восстановить обратным движением из вершины t: $t,\ P(t),\ P(P(T)),\ldots\ P(\ldots P(P(t)))\ldots s$.

Отображение состояния вершин на множествах D и P вынуждает каждый раз перед построением дерева устанавливать начальные значения для всех элементов. Вычислительная сложность такой операции – $O(n)$. Очевидно, что в случае построения кратчайших путей на больших графах количество вершин дерева может оказаться существенно меньше значения n.

Учитывая последовательный характер изменения состояния вершин дерева в соответствии со значением расстояния от его корня, очевидна идея включения такой последовательности в качестве этапа построения множества деревьев. Для ее реализации необходимо обеспечить условие распознавания исходного состояния вершин новых деревьев. Это легко достигается дополнением признака состояния вершин дерева номером такого дерева.

Возможная реализация обсуждаемой идеи (см. рис. 2) базируется на дополнении пространства состояния глобальной переменной e с целью идентификации вершин разных деревьев. Это позволяет выполнить инициализацию массивов P один раз (функция $SPI()$).

```
procedure SPI(); // Инициализация переменных состояния
 begin
  e ← 0;
  foreach i ∈ N  do P(i) ← e;
 end;
procedure SPM(r,t); // Поиск кратчайшего пути r → t
 begin
  D(r) ← 0; Q ← {r}; P(r) ← r + e;
  repeat
   i ← Q1, Q ← Q \ {i};
   if  i = t  then goto finish;
   foreach j ∈ i' begin
   if (P(j) ≤ e) ∨ ((D(j) > D(i) + w(i, j)) ∧ (Q ← Q \ {j})) then begin
     D(j) = D(i) + w(i, j), P(j) = i + e;  Q ← Q ∪ {j};
   end;
  end;
  until Q = ∅;
  finish:  e ← e + n;
 end;
```

Рис. 2. Предлагаемая версия реализации алгоритма Дийкстры

Переменная e содержит начало интервала номеров очередного дерева (функция $SPM(r,t)$). Построение дерева приводит к изменению элементов массива P, соответствующих подвергшимся анализу вершинам графа. Предикат $(P(j) \le e)$ определяет множество не рассмотренных вершин и соответствует состоянию $(D(j) = \infty)$, $j \in N$. Однако использование такого предиката позволяет исключить повторную установку начального состояния множеств D и P. Вычислительная сложность построения дерева путей функцией $SPM(r,t)$ зависит лишь от количества фактически рассматриваемых дуг.

Идея однократной инициализации массивов D и P может быть использована для отказа от очистки очереди Q. Для этого достаточно операцию $Q \leftarrow \{r\}$ в момент фиксации корня дерева в вершине r выполнить с проверкой условия $(r \le e)$. Список элементов очереди не обязательно проецировать на интервал $\overline{1,n}$, если порядок дерева существенно меньше n.

Элементарные предопределенные решения

Можно заметить, что каждая вершина дуг окончательного дерева кратчайших путей как минимум один раз будет представлена в очереди вершин. Однако некоторые из вершин дерева в очереди побывают строго один раз. Очевидно, что операции с очередью (включение и исключение) для таких вершин можно не выполнять. Это ускорит процесс поиска решения. Для реализации подобного предложения достаточно на графе предварительно для каждой вершины выделить и пометить входные дуги минимальной длины [4]:

$$T_j = \left\{ (i, j) : i = \arg\min_{i,\, j} \{ w(i, j) : (i, j) \in A \} \right\},\ j \in N . \qquad (1)$$

В случае ветвления дерева через любую дугу из множества (1) значение расстояния до ее конечной вершины не изменится. Формальным обоснованием этого является то, что любая часть кратчайшего пути является кратчайшим путем. Классические реализации алгоритма Дийкстры (рис. 1) такой факт явно не используют.

Рассмотрим возможности использования (1). Пусть на нагруженном ориентированном графе $G(N, A)$ последней достигнута вершина i дерева кратчайших путей. Если анализируется альтернатива развития дерева по дуге $i \to j$, которая является кратчайшей из входных дуг в вершину j, то такая дуга может быть досрочно включена в дерево без помещения в очередь. Однако продолжение развития дерева из вершины i по дуге $j \to k$, которая является кратчайшей из входных дуг в вершину k, окажется преждевременным, если справедливо $w(i, j) + w(j, k) > \min\{ w(l, k),\ (l, k) \in A \wedge (l \neq j)\}$. Здесь $w(i, j)$ – вес дуги $i \to j, i, j \in N$.

На реальном графе иногда можно выделить более сложный вариант предопределенных решений, когда $w(i, j) + w(j, k) \leq \min\{ w(l, k),\ (l, k) \in A \wedge (l \neq j)\}$. В таком случае количество шагов продолжения развития дерева из вершины *i* досрочно может быть увеличено до двух.

Отсюда по индукции следует, что для любой вершины $i \in N$ можно указать путь $i \to j_1 \to j_2 \to ... \to j_m \to k$, составленный из кратчайших входных дуг промежуточных вершин $j_1,\ j_2, ...\ j_m$, для которых $w(i, j_1) + w(j_1, j_2) + ... + w(j_m, k) \leq \min\{ w(l, k),\ (l, k) \in A \wedge (l \neq j_m)\}$.

Алгоритм выделения множества (1) имеет линейную вычислительную сложность:

$$D(i) \leftarrow \infty,\ i \in N;\quad D(i) \leftarrow \min(D(i), w(i, j)),\ (i, j) \in A;\quad T_j = (D(i) = w(i, j)),\ (i, j) \in A.$$

Предопределенные решения легко фиксируются на структуре смежности графа. Они инвариантны к параметрам решаемых задач многократного поиска путей. Сокращение времени поиска кратчайших путей при использовании предопределенных решений в первом приближении обратно пропорционально средней полустепени захода вершин графа. Для разреженных графов такая величина может оказаться достаточно существенной.

Организация встречного поиска

В случае поиска пути между двумя заданными вершинами графа целесообразно организовать процесс поиска путем построения двух встречно растущих деревьев. В результате объем анализируемых данных сокращается в два раза [1]. Дерево из конечной вершины должно строиться на графе с обратным направлением дуг, поэтому представление модели сети задается расширенным графом – объединением исходного графа и его инверсии.

Встречный поиск принципиально можно реализовать любой процедурой построения дерева на расширенном графе. Для этого есть достаточно веские причины:

– лучшая среди известных схема отображения очереди вершин на вектор размером L наиболее эффективно работает с большими очередями;

– оптимальный размер деревьев поиска соответствует одинаковому расстоянию от корней дерева в точке остановки [2].

С целью определения расширенного графа, пригодного для организации процесса ветвления на общей очереди, обозначим исходный граф через $G^+ = G(N^+, A^+)$, а граф с инвертированием направления дуг – $G^* = G(N^*, A^*)$.

Между вершинами таких графов должно быть взаимно однозначное соответствие. Предлагается для его задания использовать симметричную функцию отображения номеров вершин $N^+ \leftrightarrow N^*$ в виде

$$conj(x) = x^* \cdot (x \in G^+) + x^+ \cdot (x \in G^*),\ x \in N^+ \bigcup N^*. \qquad (2)$$

Пусть номера вершин графа G^+ заданы последовательностью $\overline{0, n-1}$, а номера вершин графа G^* – последовательностью $\overline{2n-1, n}$. Легко проверить, что в таком случае линейная функция

$$conj(x) = 2n-1-x,\ \ x \in N^+ \bigcup N^* \qquad (3)$$

реализует прямое и обратное отображение номеров вершин $N^+ \leftrightarrow N^*$. Это позволяет использовать (3) в качестве адресной функции, так как объединение множеств таких номеров соответствует неразрывной последовательности $\overline{0,2n-1}$.

В результате множество номеров вершин и дуг графа G^* определяется так:

$$N^* = \left\{x^* = conj(x^+),\ x^+ \in N^+\right\},\ \ A^* = \left\{(conj(y^+),\ conj(x^+)),\ (x^+, y^+) \in A^+\right\}. \qquad (4)$$

Пусть s и t – начальная и конечная вершины искомого кратчайшего пути на исходном графе G^+. Так как, согласно (4), $N^+ \bigcap N^* = \varnothing$, то встречный поиск можно проводить синхронным движением волны от корней деревьев на несвязном графе $G^+ \cup G^*$. Для этого достаточно начать процесс ветвления из вершин $s \in G^+$ и $t^* = conj(t)$, $t^* \in G^*$. Последнее соответствует формальному объединению графов фиктивной дугой $s \to t^*$, для которой $w(s, t^*) = \infty$. В отличие от известного приема поочередного развития деревьев [1], синхронное движение оказывается оптимальным.

Улучшение правила остановки встречного поиска

Организация встречного поиска требует определения правила остановки. Известно, что остановка должна соответствовать моменту фиксации постоянной пометки вершины дерева, когда сопряженная вершина уже является постоянно помеченной [5].

Если для некоторого дерева кратчайших путей максимальное расстояние от постоянно 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^+ или графа G^*, получающая постоянную пометку. Однако проблема состоит в дискретном характере процедуры выбора помечаемых вершин, когда условие $D(conj(i)) \le D(i)$ приходится проверять каждый раз после коррекции значений расстояний до временно помеченных вершин.

Обозначим множества листьев встречно растущих деревьев кратчайших путей через $T_x^+ = \{i \mid x \le D(i) < \infty,\ i \in N^+\}$ и $T_x^* = \{i \mid x \le D(i) < \infty,\ i \in N^*\}$, где $x = \min\{D(k),\ k \in N^+ \cup N^*\}$. Очевидно, что на любом этапе процесса развития деревьев остановка может произойти в любой из вершин множества $K(x) = \{i \mid i \in \{conj(i),\ i \in T_x^+\} \cap T_x^*\}$. Нетрудно заметить, что такое множество включено в очередь вершин, формируемую алгоритмом Дийкстры. Условие построения $K(x)$ может включать предикат $(P(x) \ge e)$ вместо $(D(x) < \infty)$, что исключает необходимость повторной инициализации переменных состояния. Последнее предлагается использовать для построения корректной и экономной процедуры остановки.

Действительно, перед началом поиска множество $K(0) = \varnothing$. Первый элемент в него будет включен лишь после выявления условия $(i = conj(i)) \wedge (x \le D(i) < \infty) \wedge (x \le D(conj(i)) < \infty)$. Практически проверка такого условия требует лишь включения дополнительной проверки условия $D(conj(i)) \le \infty$ в алгоритм включения вершины в очередь.

На последующих итерациях включения элементов в очередь определим значение $d_{\min} = \min_k \{D(k) + D(conj(k)),\ k \in K(x)\}$. Значение $D(k)$ только возрастает, а из множества временно помеченных вершин исключаются элементы, для которых $(D(k) < x)$. Это условие соответствует моменту установки постоянной пометки вершины, одна из которых соответствует условию $d_{\min} \equiv \min_k \{D(k) + D(conj(k)),\ k \in K(x)\}$. Так как для любого значения x новые элементы множества K будут по определению иметь расстояние до корней, не меньшее x, то условием остановки в момент постоянной пометки вершины k будет

$d_{\min} \equiv D(k) + D(conj(k))$, $k \in K(x)$.

Таким образом, использование встречного движения от корней деревьев не требует хранения специальных пометок, а момент остановки совпадает с моментами окончательной пометки вершин дерева путей.

После остановки в вершине *x* остается достроить путь 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 требуется функция $orig(x) = x \cdot (x \in G^+) + conj(x) \cdot (x \in G^*)$, $x \in N^+ \bigcup N^*$. Для случая нумерации вершин расширенного графа по правилу (4) $orig(x) = conj(x)$, $x \in N^+ \bigcup N^*$.

Отметим, что правило остановки встречного поиска не использует информацию о предопределенных решениях (3). Однако в любое из встречно растущих деревьев конечная вершина может включаться в момент, когда справедливо $K(x) = \varnothing$, $x > 0$. Очевидно, что в момент включения кратчайший путь уже известен, поэтому процесс поиска следует остановить. В таких случаях работа по анализу другого графа оказывается бесполезной.

Проецирование кратчайших путей на дуги графа

Эффективный прием фильтрации просматриваемых дуг графа – каждой дуге поставить в соответствие список вершин, кратчайшие пути к которым включают такую дугу [2]. Построение подобных списков возможно после предварительного построения всех деревьев кратчайших путей. Очевидный недостаток ассоциации вершин кратчайших путей с дугами графа (с их конечными вершинами) – потребность в памяти объемом $O(mn)$. Однако ассоциации кратчайших путей с подмножествами вершин снижает потребность в памяти [2,3].

Предлагается учесть ассоциации дуг с оптимальными решениями характеристическими множествами признаков вхождения вершин в заранее выделенные любым способом подмножества вершин [4]. Исходный граф будет представлен объединением подграфов, формально включающих вершины одного подмножества. Обозначим $B = \{B(i), i \in N\}$ – множество двоичных векторов классификации вершин графа. Принадлежность вершины i подмножеству k пусть отражается выражением $B(i) = 2^k$.

Алгоритм учета ассоциации дуг следующий. Первоначально каждой дуге графа следует назначить нулевой вектор характеристического множества признаков вхождения ее конечной вершины в кратчайшие пути: $S(I(i, j)) = 0$, $(i, j) \in M$. Здесь $I(i, j) = (k \,|\, (i, j) \in M) \wedge (k \in \overline{1, m})$ – индекс дуги в линейном массиве списка дуг, являющегося частью структуры смежности графа.

Далее для каждой вершины i, $i \in N$, необходимо построить дерево кратчайших путей до всех остальных вершин, но при этом операцию $P(j) \leftarrow i$ следует дополнить сохранением индекса $R_j \leftarrow I(i, j)$, $j \in \overline{1, n}$, указывающего позицию дуги $i \to j$ в списке дуг.

После этого узлы и листья деревьев кратчайших путей могут эффективно отображаться на дуги графа: $S(k) \leftarrow S(k) \vee B(P(j)))$, $(R_j = k) \wedge (P(j) \neq j)$, $j \in N$. В функции $SPM(r,t)$ (рис. 2) условие фильтрации дуг $(P(j) \le e) \vee ((D_j > D_i + w(i, j)) \wedge (Q \leftarrow Q \setminus \{j\}))$ должно иметь вид $((S(I(i, j)) \wedge B(t)) \neq 0) \wedge ((P(j) \le e) \vee ((D_j > D_i + w(i, j)) \wedge (Q \leftarrow Q \setminus \{j\})))$.

В настоящее время процессоры обладают возможностью параллельного выполнения операций над битами в машинных словах размером 16, 32 или 64 разряда. Характеристический вектор реально всегда отображается на массив смежных машинных слов. Операция конъюнкции в выражении $((S(I(i, j)) \wedge B(t)) \neq 0)$ проверяет совпадение единственного разряда на всех итерациях построения дерева путей к целевой вершине t. В таком случае нет необходимости проверки всех разрядов характеристического вектора. Достаточно проверять подмножество $\lfloor \log_2 B(t) / w \rfloor$ вектора $S(I(i, j))$, где w – разрядность процессора.

Экспериментальная оценка времени решения задач

В таблице приведены результаты оценки среднего времени решения задач расчета подматриц кратчайших расстояний предлагаемыми процедурами для графа реальной сети автомобильных дорог с параметрами $n \cong 10^6$, $m \cong 5 \cdot 10^6$. Размеры случайных подматриц – $|S|$=500…900, $|F|$=100. Множества вершин подграфов для проецирования кратчайших путей соответствовали равным по размеру интервалам последовательных номеров вершин исходного графа.

Оценки времени расчета подматриц кратчайших расстояний

Размерность задачи ($\|S\|$)	Среднее время решения, мсек (Celeron 1,7 Ггц, 512 Мбайт)						
	Классическая схема Дийкстры	Предлагаемая схема с различным числом подграфов					
		1	8	16	32	64	128
500	10,823	8,003	1,038	0,719	0,524	0,512	0,503
600	12,234	9,514	1,312	0,828	0,640	0,604	0,602
700	14,534	11,005	1,513	0,947	0,769	0,678	0,688
800	17,002	12,811	1,716	1,102	0,924	0,860	0,772
900	20,712	15,012	2,117	1,417	1,187	1,009	0,901

Результаты эксперимента подтверждают сокращение времени расчета подматриц после учета предопределенных решений и увеличения количества подграфов для проецирования предварительно найденных оптимальных решений. Эффект насыщения объясняется сильной разреженностью графа транспортной сети.

Заключение

Таким образом, построенные процедуры поиска используют для представления модели сети память в два раза большего объема. Предложенный прием нумерации запросов на поиск кратчайших путей позволяет исключить холостые шаги инициализации переменных состояния поиска, В результате вычислительная сложность задачи для наиболее эффективных адресных схем организации очередей линейно зависит от количества дуг дерева кратчайших путей. Пометка входных дуг минимальной длины сокращает количество шагов изменения состояния очереди обратно пропорционально степени захода вершин. Снизить такую степень позволяет проецирование результатов предварительной оптимизации путей на дуги графа.

QUICK SEARCH OF THE SHORTEST PATHS ON THE GRAPH WITH A PREDETERMINED DECISION

M.P. REVOTJUK, M.K. QARALEH, N.V. KHAJYNOVA

Abstract

On the classical problem of searching the shortest paths in graphs considered the possibility of accelerating the search procedure by incorporating a priori information about the search space. Global initialization of state variables predefined search and selection solutions can improve performance of multiple procedures to find paths to a linear dependence on the volume of the scanned area.

Список литературы

1. *N. Deo, Chi-yin Pang* // Networks. 1984. Vol. 14. P. 275–323.
2. *Fredman M.L., Tarjan R.E.* //J. ACM. 1987. Vol. 34(3). P. 596–615.
3. *Demetrescu C., Italiano G.F.* //ACM Transactions on Algorithms. 2006. № 2 (4). P. 578–601.
4. *Ревотюк М.П., Застенчик Н.И., Шешко Е.В.*// Изв. БИА. 2004. № 1 (17)/2. С. 112–114.