РЕГУЛЯРНЫЙ ВСТРЕЧНЫЙ ПОИСК КРАТЧАЙШИХ ПУТЕЙ НА ГРАФАХ

М.П. Ревотюк, Н.В. Хаджинова, А.К. Пушкина
Кафедра информационных технологий автоматизированных систем,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: {rmp, kafitas}@bsuir.by

Предлагается процедура многократного встречного поиска кратчайших путей на графах, когда порядок порождаемых деревьев путей существенно меньше порядка графа. Расширение модели графа снижает сложность поиска путей до линейной зависимости от объема сканируемого пространства.

I. Постановка задачи

Известно, что 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.

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 такой задачи алгоритмом Дийкстры имеет оцеики от $O(m + n\log_2 n)$ до $O(m + nL)$, где $m = |A|$, $n = |N|$, L – максимальная длина дуги графа [1,2].

Встречный поиск принципиально можно реализовать любой процедурой построения дерева на расширенном графе, однако здесь будет рассматриваться лучшая среди известных схема отображения очереди вершин на вектор размером L. Для этого есть достаточно веские причины:

схема отображения очереди вершин наиболее эффективно работает с большими очередями;

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

Достижение потенциально возможной эффективности поиска порождает вопросы выбора способа представления расширенного графа и определения правил остановки поиска.

II. Определение расширенного графа

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

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

$$C(x) = x^+ \cdot (x \in G^+) + x^* \cdot (x \in G^*), \quad (1)$$

где $x \in N^+ \cup N^*$.

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

$$C(x) = 2n - 1 - x, x \in N^+ \cup N^* \quad (2)$$

реализует прямое и обратное отображение номеров вершин $N^+ \leftrightarrow N^*$. Это позволяет использовать (2) 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:

$$N^* = \{x^* = C(x^+), x^+ \in N^+\},$$

$$A^* = \{(C(y^+), C(x^+)), (x^+, y^+) \in A^+\}. \quad (3)$$

Пусть s и t – начальная и конечная вершины искомого кратчайшего пути на исходном графе G^+ . Так как, согласно (4),$N^+ \cap N^* = \oslash$, 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 $s \in N^+$ и $t^* \in C(t)$, $t^* \in N^*$. 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.

III. Правило остановки 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 [2].

Пусть D – массив расстояний от корня дерева, $D = \{D_i, i \in N^+ \cup N^*\}$, а P – массив вершин дерева кратчайших путей, $P = \{P(i), i \in$

$N^+ \cup N^*\}$. 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 – $D_{C(i)} \le D_i$, где i – вершина графа G^+ или графа G^*, получающая постоянную пометку.

Однако проблема состоит в дискретном характере процедуры выбора помечаемых вершин, когда условие $D_{C(i)} \le D_i$ приходится проверять каждый раз после коррекции значений расстояний до временно помеченных вершин.

Обозначим множества листьев встречно растущих деревьев кратчайших путей через $T_x^+ = \{i | x \le D_x < \infty, x \in N^+\}$ и $T_x^* = \{i | x \le D_x < \infty, x \in N^*\}$, 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 $K_x = \{i | i \in \{C(j), j \in T_x^+\} \cap T_x^*\}$. Нетрудно заметить, что такое множество включено в очередь вершин, формируемую алгоритмом Дийкстры. Последнее предлагается использовать для построения корректной и экономной процедуры остановки.

Действительно, перед началом поиска множество $K_0 = \oslash$. Первый элемент в него будет включен лишь после выявления условия $(x \le D_i < \infty) \wedge (x \le D_{C(i)} < \infty)$. Практически проверка такого условия требует лишь включения дополнительной проверки условия $D_{C(i)} \le \infty$ в алгоритм включения вершины в очередь.

На последующих итерациях включения элементов в очередь определим значение $d_{min} = \min\{D_k + D_{C(k)}, k \in K_x\}$. 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 $d_{min} \equiv \min\{D_k + D_{C(k)}, k \in K_x\}$. Так как для любого значения x новые элементы множества K_x будут по определению иметь расстояние до корней, не меньшее D_x , то условием остановки в момент постоянной пометки вершины k будет $d_{min} \equiv D_k + D_{C(k)}, k \in K_x$.

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 $C^{-1}(x)$, обратная $C(x)$. Для случая нумерации вершин расширенного графа по правилу (3) $C^{-1}(x) = C(x)$, где $x \in N^+ \cup N^*$.

IV. Ограничение пространства поиска

Эффективный прием фильтрации просматриваемых дуг графа – каждой дуге поставить в соответствие список вершин, кратчайшие пути к которым 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 – потребность в памяти объемом $O(mn)$. Однако ассоциации кратчайших путей с подмножествами вершин снижает потребность в памяти [2]. 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: $S(I(i,j)) = 0, (i,j) \in A$. Здесь $I(i,j)$ – индекс дуги $i \to j$ 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 вместе с операцией $P(j) \leftarrow i$ сохраняя индекс $R_j \leftarrow I(i,j)$, указывающего позицию дуги $i \to j$ в списке дуг. После этого узлы и листья деревьев кратчайших путей могут эффективно отображаться на дуги графа: $S(k) \leftarrow S(k) \vee B(P(j))$, $(R_j = k) \wedge (P(j) \ne j)$.

Заключение

Вычислительная сложность задачи встречного поиска кратчайших путей для наиболее эффективных адресных схем организации очередей линейно зависит от количества дуг создаваемого дерева. Рассмотренные приемы снижения сложности решения минимизируют количество бесполезных операций для инициализации пространства поиска.

1. Demetrescu, C. Experimental analysis of dynamic all pairs shortest path algorithms/Demetrescu C., Italiano G.F.//ACM Transactions on Algorithms. – 2006. –No. 2(4). –P. 578–601.
2. Ревотюк, М. П. Быстрый поиск кратчайших путей на графах с предопределенными решениями /М. П. Ревотюк, М. К. Кароли, Н. В. Хаджинова //Доклады БГУИР. –2014. – № 4(82). – С. 73–79.