УДК 519.17

Алгоритмы пересчёта кратчайших путей в графе при изменении весов ребер

Д.Ю. Кривошеин[1], А.М. Марченко[1,2]

[1]Московский государственный университет им. М.В. Ломоносова, dk@cs.msu.su

[2]ООО «Нангейт», alexander.m.marchenko@gmail.com

***Аннотация* — В статье рассматривается задача пересчета кратчайших путей в графе при изменении веса какого-либо ребра. При этом можно выделить два принципиально различных случая: увеличение и уменьшение веса ребра. В первом случае описан алгоритм, сложность которого зависит от структуры данного графа и сложности реализации алгоритма Дейкстры для этого графа, при этом для большинства графов сложность ниже, чем сложность алгоритма Флойда-Уоршелла. В случае уменьшения веса ребра описан алгоритм сложности $O(n^2)$.**

***Ключевые слова* — динамически изменяющийся граф, поиск кратчайших путей, изменение веса ребра, инкрементальный алгоритм.**

I. Введение

Задача поиска кратчайших путей является одной из основных задач теории графов и имеет множество применений в реальной жизни. Большое количество задач требует вычисления расстояний между объектами, к таким задачам относится, например, маршрутизации компьютерных сетей [1] или задача трассировки сверхбольших интегральных схем (СБИС) [2]. Известны несколько классических алгоритмов, решающие задачу поиска кратчайших путей. Алгоритм Флойда-Уоршелла находит расстояния между всеми парами вершин в графе и имеет сложность $O(|V|^3)$ [3]. В графе с неотрицательными весами для поиска расстояний от одной вершины до всех остальных используется алгоритм Дейкстры сложности $O(|E|+|V|^2)$ [3]. Этот алгоритм применяется, например, в протоколе маршрутизации OSPF [4]. В случае разреженного графа, содержащего малое количество ребер, сложность алгоритма Дейкстры можно понизить до $O(|V|\log|V|+|E|)$ [5] с использованием специальных структур данных. Для поиска расстояний между всеми парами вершин в этом случае вместо алгоритма Флойда-Уоршелла можно использовать алгоритм Джонсона сложности $O(|V|^2 \log|V|+|V|\cdot|E|)$ [6],

Указанные алгоритмы целесообразно применять, если граф не изменяется с течением времени. В случае динамически изменяющегося графа эффективность применения данных алгоритмов снижается из-за большой вычислительной сложности. Время работы алгоритма Флойда-Уоршелла для графов с большим числом вершин (несколько тысяч) может измеряться часами, что является неприемлемым результатом для применения на практике. На этапе трассировки СБИС возникает задача построения леса деревьев Штейнера [2], в ходе решения которой необходимо большое число раз пересчитывать кратчайшие пути между вершинами графа трассировки, для чего в настоящее время используется алгоритм Дейкстры [7] или алгоритм Флойда-Уоршелла. Поэтому существует необходимость в алгоритмах, которые бы решали эту задачу более эффективно, используя информацию о структуре графа.

II. Постановка задачи

Пусть $G=(V,E)$ – связный ориентированный граф и каждая дуга $(u,v)\in E$ имеет некоторый вес $w(u,v)\geq 0$. Пусть $|V|=n$ и $|E|=m$. Предполагается, что уже известны кратчайшие пути между всеми парами вершин и рассматривается ситуация, когда вес некоторой дуги (u,v) меняется на величину d. После этого изменения кратчайшие пути в графе могли измениться, и ставится задача пересчитать их, по возможности быстрее, чем за $O(n^3)$ операций, которые бы потребовались для выполнения алгоритма Флойда-Уоршелла. В случае $d>0$ алгоритм будем называть *инкрементальным*, а в случае $d<0$ – *декрементальным*. Будем считать, что $|d|\leq w(u,v)$, в противном случае в графе могли бы возникнуть дуги отрицательного веса.

III. Обзор существующих алгоритмов

В статье [8] авторами предложена инкрементальная модификация алгоритма Флойда-Уоршелла для решения задачи контроля маршрутов самолетов. Недостатком алгоритма является то, что он использует

$O(|V|^3)$ памяти для хранения промежуточных данных. По этой причине в применении к реальной задаче контроля алгоритм показал низкую эффективность.

В статье [9] рассматривается случай удаления ребра из графа и последующий пересчёт всех кратчайших путей. Для решения этой задачи авторами предложен алгоритм сложности $O(\min(n^{3/2}\sqrt{\log n}, \frac{n^3}{m}\log^2 n))$. В [10] предложен алгоритм для поиска кратчайших путей от одной вершины до всех, а также его применения для поиска кратчайших путей между всеми вершинами для случаев удаления и добавления ребер.

Можно заметить, что в текущей постановке задачи удаление и добавление ребер в графе может быть представлено как частный случай изменения веса ребра – соответственно его увеличение на ∞ или уменьшение с ∞ до нужного значения.

IV. Теоретические оценки сложности алгоритмов

Получим теоретически возможные нижние оценки сложности алгоритмов пересчета кратчайших путей, исходя из предположения, что алгоритм Флойда-Уоршелла является оптимальным для задачи вычисления кратчайших путей между всеми парами вершин для графов произвольного вида. Рассмотрим следующий алгоритм:

1) построить граф G^0, который представляет собой граф G, всем ребрам которого присвоено значение 0, при этом матрица расстояний – нулевая матрица размера $n \times n$;

2) на i-ом ($i = 1,\ldots,m$) шаге выбрать некоторое ребро $e_i \in E(G^{i-1})$ и его вес заменить на вес соответствующего ребра из $E(G)$;

3) пересчитать кратчайшие пути с помощью инкрементального алгоритма.

В итоге после m шагов строится граф $G^m = G$ и посчитаны кратчайшие пути между всеми парами вершин. Обозначим сложность инкрементального алгоритма как $L(A_{inc})$, тогда сложность описанного алгоритма составит $m \cdot L(A_{inc})$. Так как сложность алгоритма Флойда-Уоршелла $O(n^3)$, получаем следующую нижнюю оценку для сложности инкрементального алгоритма:

$$m \cdot L(A_{inc}) \geq O(n^3),$$

$$L(A_{inc}) \geq O\left(\frac{n^3}{m}\right).$$

Для декрементального алгоритма можно провести аналогичные рассуждения, но в этом случае начальный граф не содержит ни одного ребра (что соответствует тому, что веса всех ребер равны ∞), и на каждом шаге происходит уменьшение веса ребра. Для плотных графов, с $m = O(n^2)$, для сложности $L(A_{dec})$ декрементального алгоритма можно получить аналогичный результат:

$$L(A_{dec}) \geq O\left(\frac{n^3}{m}\right) = O(n),$$

но при рассмотрении разреженных графов ($m = O(n)$), минимальная сложность декрементального алгоритма составит $O(n^2)$.

V. Инкрементальный алгоритм

Перейдем к описанию инкрементального алгоритма пересчета кратчайших путей в графе. Основные утверждения доказаны и опубликованы в статье [ITA], здесь они приведены без доказательств.

Определение 5.1. Пусть кратчайший путь P между вершинами v' и v'' в графе G проходит через ребро (u,v) и имеет вид $v'P_1(u,v)P_2v''$, где P_1, P_2 – некоторые пути. Тогда вершины v' и v'' называются соответственно *входящей* и *выходящей* вершинами для ребра (u,v).

Пусть *Inc* и *Out* – множества входящих и выходящих вершин, соответственно.

Утверждение 5.1. При пересчёте кратчайших путей в графе достаточно рассмотреть только пути между вершинами из множества $Inc \cup Out$.

Инкрементальный алгоритм A_{inc} вычисления кратчайших путей в графе, основанный на утверждении 5.1, состоит из следующих шагов:

1) из вершины *u* поиском в ширину находятся все вершины, принадлежащие множеству *Inc*; аналогично из вершины *v* находятся все вершины, принадлежащие множеству *Out*;

2) для каждой вершины из множества *Inc* с помощью алгоритма Дейкстры вычисляются значения кратчайших путей до множества *Out*;

3) каждой вершине из множества *Out* приписывается значение длины кратчайшего пути до каждой вершины из множества *Inc*, полученное на шаге 2.

В результате вычисляются новые значения кратчайших путей между всеми вершинами множества $Inc \cup Out$, остальные пути не требуют пересчёта, согласно утверждению 5.1, таким образом, алгоритм корректно пересчитывает все кратчайшие пути в графе.

Утверждение 5.2. Сложность инкрементального алгоритма поиска кратчайших путей в графе не превосходит $O(|Inc|\cdot L_D)$, где L_D – сложность алгоритма Дейкстры.

Замечание 5.1. В случае разреженного графа, алгоритм Дейкстры может быть модифицирован таким образом, что его сложность составит $L_D = O(n\log n + m)$ [5], в этом случае сложность инкрементального алгоритма составит $L(A_{inc}) = O(n^2 \log n)$.

Замечание 5.2. В худшем случае сложность инкрементального алгоритма может составить $O(n^3)$, что возможно, если мощность множества входящих вершин асимптотически равна мощности множества всех вершин (т.е. почти все кратчайшие пути содержат изменившееся ребро), и граф при этом не является разреженным. В этом случае требуется пересчитать все кратчайшие пути, что можно осуществить с аналогичной сложностью с помощью алгоритма Флойда-Уоршелла.

VI. Декрементальный алгоритм

Теперь рассмотрим случай уменьшения веса ребра и соответствующий ему декрементальный алгоритм. Кратчайший путь в графе *G* между вершинами v_1 и v_2 будем обозначать как $P(v_1, v_2)$. Длину пути будем обозначать как $|P(v_1, v_2)|$.

Имеет место следующее очевидное утверждение:

Утверждение 6.1. Рассмотрим вершины $v_1, v_2 \in V$, пусть длина кратчайшего пути $P(v_1, v_2)$ между этими вершинами равна *p*. Пусть $|P(v_1, u)| = p_1,\ P(v, v_2) = p_2$. Тогда, если выполнено условие:

$$p_1 + w(u,v) + p_2 < p \quad , \qquad (6.1)$$

то в *G* существует путь между вершинами v_1 и v_2 с длиной меньше, чем текущая.

Непосредственно из этого утверждения можно сформулировать декрементальный алгоритм, состоящий из следующих шагов:

1) с помощью алгоритма Дейкстры вычислить длины всех кратчайших путей от вершин *u* и *v* до всех вершин графа *G*;

2) для каждой пары вершин v_1 и v_2 проверить условие (6.1) и, если требуется, изменить значение длины кратчайшего пути.

Используя простейшую реализацию алгоритма Дейкстры [3], получаем следующую оценку для сложности алгоритма:

$$L(A_{dec}) \le 2O(n^2 + m) + C_n^2 = O(n^2) .$$

Таким образом, сложность декрементального алгоритма поиска кратчайших путей в графе равна $O(n^2)$. Принимая во внимание рассуждения раздела III о теоретических оценках сложности алгоритмов, можно заметить, что для разреженных графов можно решать задачу вычисления расстояний между всеми парами вершин описанным выше способом, при этом сложность алгоритма будет аналогичной сложности алгоритма Флойда-Уоршелла.

VII. Экспериментальные результаты

Алгоритмы были реализованы на языке C++, было проведено тестирование на случайных графах и сравнение времени выполнения с алгоритмом Флойда-Уоршелла вычисления всех кратчайших путей. На рис. 1 показано среднее время выполнения алгоритмов, для инкрементального алгоритма под худшим случаем понималось максимальное время работы в течение эксперимента.

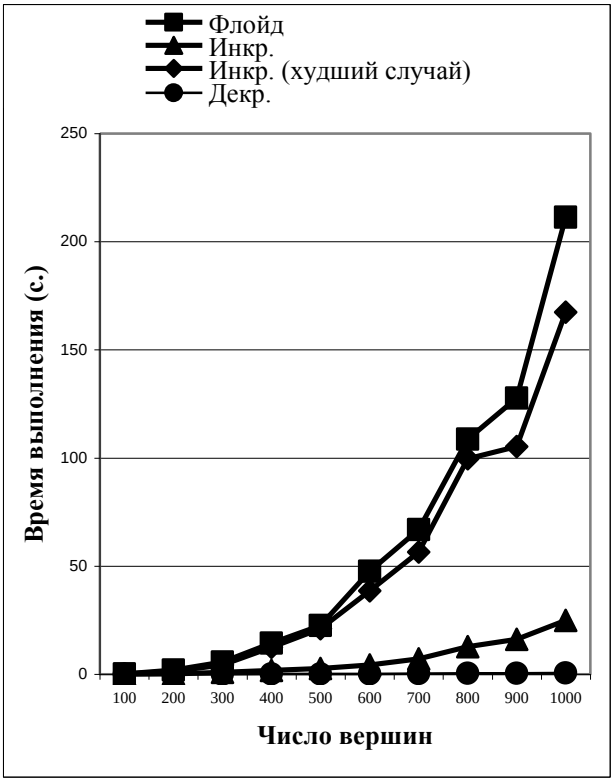

Рис. 1. Время работы алгоритмов на случайных графах

Также были проведены эксперименты, показывающие высокую эффективность инкрементального алгоритма в случае разреженных графов (рис. 2), который даже в худшем случае показал более эффективные результаты, чем алгоритм Флойда-Уоршелла.

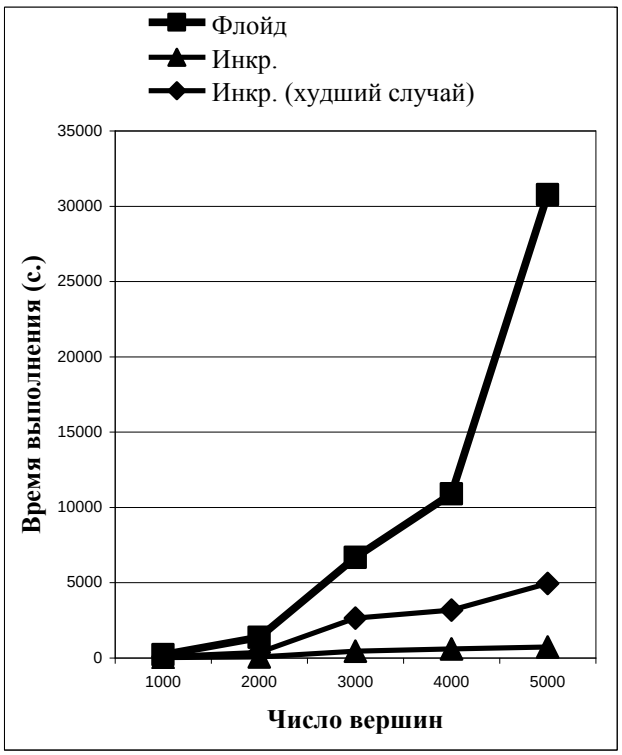

Рис. 2. Время работы инкрементального алгоритма на случайных разреженных графах

VIII. Заключение

В статье описаны алгоритмы пересчета кратчайших путей в графе, если вес какого-либо ребра изменяется. Сначала рассмотрен случай увеличения ребра, показано, что в худшем случае алгоритм имеет сложность $O(n^3)$, но в большинстве случаев сложность меньше и зависит от структуры графа. Также рассмотрен случай уменьшения веса ребра и описан декрементальный алгоритм сложности $O(n^2)$.

Случаи добавления или удаления ребер в графе могут рассматриваться как частные случаи увеличения или уменьшения веса ребер, соответственно, поэтому для этих случаем алгоритм также применим.

Все алгоритмы были реализованы программно, проведены эксперименты на случайных графах, которые подтвердили теоретическую оценку, также проведено сравнение с алгоритмом Флойда-Уоршелла, в котором алгоритмы показал высокую эффективность.

Литература

[1] Zhan F.B., Noon C.E.. Shortest Path Algorithms: An Evaluation Using Real Road Networks // Transportation Science. 1998. Vol. 32. No. 1. P. 65–73.

[2] Alpert C.J., Mehta D.P., Sapatnekar S.S. Handbook of Algorithms for Physical Design Automation. CRC Press, 2008. 1044 p.

[3] Cormen T.H., Leiserson C.L., Rivest R.L., and Stein C. Introduction to Algorithms. Cambridge, Massachusetts. The MIT Press, 2001.

[4] Moy J.. RFC 2328 "OSPF Version 2". The Internet Society. OSPFv2. URL: http://tools.ietf.org/html/rfc2328 (дата обращения 24.04.2012).

[5] Fredman M.L., Tarjan R.E. Fibonacci Heaps And Their Uses In Improved Network Optimization Algorithms // 25th Annual Symposium on Foundations of Computer Science. 1984. P. 338-346.

[6] Johnson D.B. (1977). Efficient algorithms for shortest paths in sparse networks // Journal of the ACM. 1977. Vol. 24. No. 1. P. 1–13.

[7] Cong J., Fang J., Xie M., and Zhang Y.. MARS – A Multilevel Full-Chip Gridless Routing System // Proc. of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Los Angeles, 2005. Vol. 24. No. 3. P. 382-394.

[8] Rivière T., Brisset P. Shortest path in planar graph and air route network // Proc. of 4th EUROCONTROL Innovative Research Workshop & Exhibition. Brétigny-sur-Orge. 2005. P. 306-315.

[9] Baswana S., Hariharan R., Sen S. Improved Decremental Algorithms for Maintaining Transitive Closure and Allpairs Shortest Paths // Journal of Algorithms. 2007. Vol. 62. № 2. P. 74–92.

[10] Ramalingam G., Reps T. An Incremental Algorithm for a Generalization of the Shortest-Path Problem // Journal of Algorithms. 1996. Vol. 21. № 2. P. 267–305.

[11] Кривошеин Д.Ю., Марченко А.М. Инкрементальный алгоритм поиска кратчайших путей в графе // Информационные технологии. 2012. № 7.