УДК 519.173.5
ББК 22.176

АЛГОРИТМЫ БЫСТРОГО ПОИСКА ДЛЯ ДВУХ ЗАДАЧ О МЕТРИЧЕСКИХ ХАРАКТЕРИСТИКАХ ВЗВЕШЕННЫХ ГРАФОВ

Ураков А.Р.[1], **Тимеряев Т.В.**[2]
(Уфимский государственный авиационный технический университет, Уфа)

Рассматриваются две задачи о поиске метрических характеристик взвешенных ненаправленных графов с неотрицательными весами ребер. Первая задача: дан взвешенный неориентированный граф, требуется найти радиус, диаметр, и хотя бы один центр и две периферийные вершины. Во второй задаче дополнительно задана матрица расстояний графа. Для рассматриваемых задач предлагаются алгоритмы быстрого поиска, которые для поиска указанных хар-ик посматривают лишь часть вершин графа. Приводится сравнение предлагаемых алгоритмов с популярными способами решения поставленных задач на различных исходных данных.

Ключевые слова: метрические характеристики графа, радиус графа, диаметр графа, центр графа, периферийные вершины графа.

Введение

Метрическими (или числовыми) характеристиками называют параметры графа, определяемые через кратчайшие расстояния между вершинами: центр, радиус и диаметр. Диаметр определяется как самое длинное из всех кратчайших расстояний между вершинами графа, центр — как вершина, максимальное расстоя-

[1] *Ураков Айрат Ренатович, кандидат физико-математических наук, доцент, (urakov@ufanet.ru).*
[2] *Тимеряев Тимофей Валерьевич, аспирант (timeryaev@yandex.ru).*

ние от которой до всех остальных вершин графа минимально, а это максимальное расстояние от центра есть радиус графа.

Эти характеристики являются существенными свойствами графов и дают важную информацию о структурах, представляемых ими. Например, в задачах обслуживания диаметр может представлять максимальное время ожидания, центр и радиус — наилучшее местоположение пункта обслуживания и максимальное время необходимое для того чтобы добраться до него. В тех случаях, когда графы представляют вычислительные сети, диаметр может обозначать время прохождения сигнала между двумя узлами с наиболее медленным соединением, а радиус указывать на задержку между сервером и самым медленным клиентом и т.д.

В общем случае, метод отыскания этих характеристик состоит в нахождении кратчайших расстояний между всеми парами вершин исследуемого графа и установлении длин и вершин, подходящих под определение центра, радиуса и диаметра. Для взвешенных графов с n вершинами известные алгоритмы, выполняющие эту задачу, имеют сложность от $\tilde{O}(Cn^{2,376})$ для графов имеющих ограниченные C целые веса ребер [11] до $O(n^3/\log^2 n)$ для графов с произвольными вещественными весами ребер [3]. Очевидно, что для структур, состоящих из сотен тысяч, миллионов вершин, применение такого способа является во многих ситуациях нецелесообразным или даже практически невозможным из-за слишком большого времени счета.

Хотелось бы дополнительно отметить ситуацию, когда кратчайшие расстояния между всеми парами вершин графа уже вычислены, а числовые характеристики неизвестны и требуют нахождения. Это, например, происходит в тех случаях, когда изначально вычислять числовые характеристики не требовалось или необходимо найти характеристики не для всего графа, а для различных его подграфов. Причем, состав подграфов может изменяться во времени, и, соответственно, необходимо периодически пересчитывать значения радиуса, центра и диаметра. Тривиальный способ решения в данном случае — просмотр всех вычисленных кратчайших расстояний, он имеет сложность $O(n^2)$. Для

графов с большим количеством вершин или в случае с многократным пересчетом характеристик решение этой задачи также может иметь большое суммарное время счета.

В данной статье рассматриваются задачи двух типов. Задача 1: нахождение центра, радиуса и диаметра графа, заданного лишь множествами вершин V и ребер E и Задача 2: нахождение центра, радиуса и диаметра графа при наличии дополнительной информации в виде вычисленных кратчайших расстояний между всеми парами вершин графа.

Задачи рассматриваются на связных взвешенных ненаправленных графах с неотрицательными вещественными весами ребер. Несвязные графы не рассматриваются, потому что для них, по определению, радиус и диаметр равны бесконечности и центр не определен. В тех случаях, когда характеристики для несвязных графов определяют как некоторую функцию (например, максимум) от характеристик компонент связности, задачу можно свести к рассматриваемой, отыскав отдельные компоненты быстрыми существующими способами [9]. Рассматриваются графы только с неотрицательными весами ребер, потому что любому графу, не имеющему циклов с отрицательным весом, можно поставить в соответствие граф с неотрицательными весами ребер с сохранение кратчайших путей [8].

В данной статье приводятся алгоритмы быстрого поиска метрических характеристик взвешенных графов для двух указанных выше задач. Основное преимущество предлагаемых алгоритмов состоит в том, что для определения метрических характеристик большую часть вершин графа можно не рассматривать. Предложенные алгоритмы показали значительное уменьшение времени счета в сравнении с популярными способами решения этих задач при тестировании на взвешенных графах различной природы.

1. Постановка задачи

Рассматривается связный ненаправленный взвешенный граф $G = (V, E, w)$, где $V = (v_1, v_2, \ldots, v_n)$ — множество вершин графа, $E = (e_1, e_2, \ldots, e_n)$ — множество ребер графа и $e_i \subseteq V \times V$, а

$w : E \to \mathbb{R}^{+,0}$ — неотрицательная вещественная весовая функция на ребрах. Мощности множества вершин и множества ребер будем считать равными $|V| = n$, $|E| = m$. Введем ряд обозначений и определений.

Вес ребра из вершины v_i в вершину v_j будем обозначать $w(i, j)$. Будем называть расстоянием из вершины v_i в вершину v_j кратчайшее расстояние из v_i в v_j и обозначать его m_{ij}. Матрицей расстояний будем называть матрицу $M = (m_{ij})$, состоящую из расстояний между всеми парами вершин графа. Граф называется связным, если между любыми двумя вершинами графа v_i и v_j существует путь, т.е., если расстояние между ними конечно $m_{ij} < \infty$. Взвешенный граф называется ненаправленным, если веса ребер между вершинами равны в обоих направлениях $w(i, j) = w(j, i), \forall i, j$. Метрические характеристики графа определяются следующим образом.

Определение 1. *Эксцентриситетом $\varepsilon(v_i)$ вершины v_i называют максимальное расстояние от этой вершины до всех остальных $\varepsilon(v_i) = \max_{j=\overline{1,n}} m_{ij}$.*

Определение 2. *Радиус графа r — минимальный из эксцентриситетов вершин графа $r = \min_{i=\overline{1,n}} \varepsilon(v_i)$. Вершина c, на которой достигается этот минимум, называется центром графа $c : \varepsilon(c) = r$.*

Определение 3. *Диаметр графа d — максимальное расстояние между вершинами для всех пар вершин графа $d = \max_{i,j=\overline{1,n}} m_{ij}$. Вершины, расстояние между которыми равно диаметру, называются периферийными вершинами.*

Очевидно, что в общем случае граф может иметь несколько центров и несколько пар периферийных вершин. В частности, в полном графе с одинаковыми весами ребер каждая вершина будет одновременно и центром, и периферийной. Нас интересует нахождение хотя бы одного из центров и одной пары периферийных вершин, так как зачастую знание радиуса и диаметра гораздо важнее, чем информация о том, на каких вершинах они достигаются. В статье рассматриваются две задачи о нахождении метрических характеристик графа.

Задача 1. *Дан связный неориентированный взвешенный граф $G = (V, E, w)$ с неотрицательной вещественной весовой функцией на ребрах $w : E \to \mathbb{R}^{+,0}$. Требуется найти радиус r, диаметр d данного графа и хотя бы один из центров графа c и одну пару периферийных вершин.*

Задача 2. *Даны связный неориентированный взвешенный граф $G = (V, E, w)$ с неотрицательной вещественной весовой функцией на ребрах $w : E \to \mathbb{R}^{+,0}$ и вычисленная матрица расстояний M. Требуется найти радиус r, диаметр d данного графа и хотя бы один из центров графа c и одну пару периферийных вершин.*

Введем еще два понятия. Задачей о кратчайших путях с единственным источником (Single Source Shortest Paths, SSSP) для взвешенного графа называется задача, в которой задан взвешенный граф $G = (V, E, w)$ и необходимо найти кратчайшие пути от заданной вершины v_i до всех остальных $v_j \in V$. Поиск кратчайших расстояний от вершины v_i данного графа до всех остальных будем обозначать SSSP(v_i). Будем называть множеством опорных вершин $P = (p_1, p_2, \ldots, p_p)$ данного графа множество любых p вершин, информация о расстояниях от которых используется при вычислении метрических характеристик.

2. Состояние вопроса

Наиболее распространенным способом нахождения метрических характеристик графа в случае с не вычисленной матрицей расстояний является решение задачи о кратчайших путях между всеми парами вершин. Большинство таких алгоритмов не обладает универсальностью и показывает хорошую скорость решения только для графов с определенным набором свойств. В частности, существуют алгоритмы для разреженных графов [6], для графов с целыми ограниченными весами ребер [11] и т.п. И хотя сложность решения задачи о кратчайших путях периодически понижается с появлением новых алгоритмов, нахождение метрических характеристик таким способом для взвешенных графов с боль-

шим числом вершин все еще остается не практичным.

Существуют алгоритмы, находящие метрические характеристики графов, обладающих особым устройством и описывающих специальные структуры. К подобным можно отнести, например, алгоритм поиска диаметра для графов "сетей малого мира"(Small World Networks) [12] или алгоритм поиска центра и диаметра для графов бензеноидных систем (Benzenoid Systems) [4]. Эти алгоритмы для ускорения решения используют особенности рассматриваемых графов и поэтому область их эффективного применения сильно ограничена.

Одной из разновидностей задач о метрических характеристиках графов является приближенное вычисление радиуса и диаметра графа, при этом поиск расстояний между вершинами графа производится с некоторой погрешностью, что позволяет значительно уменьшить время счета. К алгоритмам, решающим эту задачу, можно отнести алгоритм приближения функции окрестности (Approximate Neigborhood Function Algorithm) [2] и [1].

Кроме поиска точных и приближенных характеристик графа широко используется поиск и так называемых эффективных характеристик графа. Эффективный радиус вершины v определяется как 90-й перцентиль всех расстояний от данной вершины. Эффективный диаметр графа определяется как минимальное расстояние, на котором 90% вершин достижимы друг из друга. Алгоритмы решения этих задач предлагаются, например, в [10].

Для Задачи 2 тривиальным способом решения является просмотр всех расстояний и нахождение тех, которые подходят под определение радиуса и диаметра. Решение данной задачи может быть ускорено, если вычисленные расстояния хранятся с использованием специальных структур данных (например, в отсортированном списке). Если же вычисленные расстояния хранятся в виде обыкновенного массива, само построение таких структур будет иметь сложность, превышающую сложность решения тривиальным способом. Другие опубликованные методы решения Задачи 2 нам неизвестны.

3. Алгоритмы быстрого поиска

Кратко основная идея предлагаемых алгоритмов выглядит следующим образом. Вначале мы определенным образом выбираем некоторое количество опорных вершин и вычисляем от них расстояния до всех остальных вершин. Используя эту информацию и определения метрических характеристик, мы находим границы радиуса и диаметра и их оценки на данной итерации. После этого из рассмотрения выбрасываются вершины, расстояния от которых не входят в найденные границы. Если в рассмотрении остались вершины, мы увеличиваем число опорных вершин и снова определяем границы радиуса и диаметра и их оценки. Процесс продолжается до тех пор, пока все вершины не будут исключены. Текущие оценки радиуса и диаметра при этом являются искомыми нами характеристиками.

3.1. ПОИСК ЦЕНТРА И РАДИУСА

Алгоритмы поиска центра и радиуса для Задач 1 и 2 практически идентичны. Единственное различие заключается в возможности использования уже вычисленных расстояний от рассматриваемой вершины. В Задаче 2 мы просто просматриваем уже вычисленные расстояния, хранящиеся в виде столбца матрицы, а в Задаче 1 эти расстояния вычисляются алгоритмом, решающим SSSP. В основе ускорения поиска лежит определение нижней r_l и верхней r_u границ радиуса графа по множеству опорных вершин P и просмотр претендентов на центр c_i, до тех пор, пока для какого-то из них не будет выполнено равенство $r_l = r_u$.

Радиус графа r обладает тем свойством, что вычислив расстояния от всех вершин графа до произвольный вершины v_i и найдя минимальное из них $m_{ji} = \min_{k=\overline{1,n},k\neq i} m_{ki}$ мы получим нижнюю границу r_l для радиуса $r \geqslant r_l = m_{ji}$. Действительно, так как m_{ji} минимальное расстояние до v_i от других вершин, то эксцентриситет центра c будет не меньше этой нижней границы $r_l = m_{ji} \leqslant m_{ci} \leqslant \varepsilon(c) = r$. Более того, величина r_l является нижней границей радиуса и в случае, если она определяется как минимум из максимумов расстояний от всех вершин графа до подграфа

из любых p вершин. В этом случае $r_l = \min_{i=\overline{1,n}}(\max_{p\in P} m_{ip})$ и $r_l = \max_{p\in P} m_{jp} \leqslant \max_{p\in P} m_{cp} \leqslant \varepsilon(c) = r$.

С другой стороны, радиус графа r ограничен сверху эксцентриситетом любой вершины $r \leqslant r_u = \varepsilon(v_j)$. Таким образом, радиус графа r лежит в пределах $r_l \leqslant r \leqslant r_u$ и,

(1) если $r_l = r_u$, то $r = r_l = r_u$

Таким образом, найти радиус графа и один из его центров можно итеративным увеличением нижней границы радиуса r_l и уменьшением его верхней границы r_u до тех пор, пока не будет выполнено равенство $r_l = r_u$. Очевидно, что скорость схождения r_l и r_u к радиусу зависит от выбора претендентов на центр c_i и опорных вершин.

Для быстрого поиска радиуса необходимо находить вершины претенденты на центр c_i с как можно большим

(2) $$\max_{p\in P} m_{c_i p} = r_l = \min_{i=\overline{1,n}}(\max_{p\in P} m_{ip})$$

и с как можно меньшим эксцентриситетом $\varepsilon(c_i)$, здесь $P-$множество опорных вершин. При этом желательно, чтобы число просмотренных претендентов на центр c_i и вершин в P было как можно меньше, это позволит минимизировать число решений SSSP в Задаче 1 и число просмотренных элементов матрицы расстояний в Задаче 2.

Так как центральная вершина у многих графов действительно находится приблизительно в геометрическом центре, если бы таковой можно было определить, то неплохой стратегией является выбор вершин-претендентов, обладающих подобным местоположением. Одним из способов нахождения таких вершин является выбор вершин c_i, максимальное расстояние от которых до периферийных вершин графа минимально $c_i : \max_{d\in D} m_{c_i d} = \min_{i=\overline{1,n}}(\max_{d\in D} m_{id})$, где $D-$множество периферийный вершин графа. Такой подход требует нахождения диаметра и периферийных вершин графа и поэтому является весьма вычислительно сложным. Однако, можно использовать следующий простой способ, позволяющий получить неплохой приближение к периферийным вершинам.

В графе ищутся вершины p_1 и p_2, являющиеся друг для друга самыми удаленными

$$p_1, p_2 : m_{p_1 p_2} = \max_{i=\overline{1,n}} m_{ip_1} = \max_{i=\overline{1,n}} m_{ip_2} \tag{3}$$

Для этого выбирается любая вершина графа d_1 и вершина d_2, определяемая как самая удаленная от d_1 $d_2 : m_{d_1 d_2} = \max_{i=\overline{1,n}} m_{d_1 i}$. Последующие вершины в данной «последовательности» определяются аналогичным образом

$$d_{j+1} : m_{d_j d_{j+1}} = \max_{i=\overline{1,n}} m_{d_j i} \tag{4}$$

Процесс продолжается до тех пор, пока не будет выполнено равенство $d_{j-1} = d_{j+1}$, в этом случае искомые вершины найдены и $p_1 = d_j$, $p_2 = d_{j+1}$.

Найденные вершины p_1 и p_2 включаются в пустое до этого множество опорных вершин $P = \{p_1, p_2\}$. Претенденты на центр c_i ищутся по формуле (2). Если для первого претендента на центр c_1 выполняется равенство $r_l = r_u$, данная вершина является одним из центров графа и радиус графа $r = r_l = r_u$. Если же $r_u > r_l$, самая удаленная от c_1 вершина $p_3 : m_{c_1 p_3} = \varepsilon(c_1)$ добавляется в множество опорных вершин P и выполняется поиск c_2 по формуле (2). Процесс продолжается до тех пор, пока для какого-то из претендентов не будет выполнено равенство (1).

Данный подход к поиску центра и радиуса обладает несколькими преимуществами. Во-первых, вычисление (или просмотр для Задачи 2) кратчайших расстояний происходит по мере поиска и при достаточно раннем выполнении условия (1) для большой части вершин эту операцию выполнять не придется. Во-вторых, не нужно вычислять максимумы в формуле (2) заново при каждом новом поиске претендента на центр, можно хранить вычисленные максимумы, а при добавлении каждой новой вершины p_i в P лишь вычислять максимумы между хранящимися значениями и расстояниями от p_i. Описание алгоритма для обеих постановок задач изображено на рисунке 1.

Алгоритм поиска центра и радиуса (P1, P2)
Вход: граф $G = (V, E, w)$
матрица кратчайших расстояний M (*только для Задачи 2*).
$r_l = 0$, $r_u = \infty$
1. Ускорение поиска
Для вершин d_i из формулы (4) решаем SSSP(d_i) (*только для Задачи 1*).
Определяем p_1, p_2 по формуле (3) и включаем их в P.
2. Находим претендента на центр
Определяем вершину c_i и r_l по формуле (2).
3. Проверка претендента
Если для c_i кратчайшие расстояния не вычислены, решаем SSSP(c_i) (*только для Задачи 1*).
Находим эксцентриситет $\varepsilon(c_i)$ и, если $\varepsilon(c_i) < r_u$, $r_u = \varepsilon(c_i)$.
Если $r_u \neq r_l$, добавляем вершину $p_j : m_{c_i p_j} = \varepsilon(c_i)$ в P и переходим к шагу 2.
Иначе, c_i — один и центров графа, $r = r_l$ — радиус графа, конец алгоритма.

Рис 1: Алгоритм поиска радиуса и центра. Курсивом в скобках обозначены дополнительные данные и действия для Задач 1 и 2.

Сложность выполнения алгоритма, очевидно, зависит от количества просмотренных претендентов на центр c_i. Для графов с различной структурой это количество может сильно отличаться, поэтому невозможно строго оценить сложность для общего случая, можно лишь приблизительно указать нижнюю и верхнюю границы. Рассмотрим лучший и худший случай для Задачи 1. В лучшем случае, когда первый же претендент на центр c_1 является одним из центров графа, нам необходимо решить задачу SSSP для k вершин d_i из первого шага алгоритма, определить c_1 по формуле (2) и найти эксцентриситет c_1, решив SSSP(c_1). Сложность в данном случае определяется формулой $k \cdot diff(SSSP) + k \cdot O(n)$, где $diff(SSSP)$ — сложность выполнения SSSP. В теоретическом худшем случае, когда радиус определяется при просмотре всех возможных претендентов c_i, мы получим $n \cdot diff(SSSP) + O(n^2)$, здесь $O(n^2)$ — поиск эксцентриситетов всех n вершин и опреде-

ление c_i по формуле (2).

3.2. ПОИСК ДИАМЕТРА

Принципиальное различие в поиске диаметра заключается в том, что, в отличие от радиуса, для диаметра в общем случае нельзя определить удобную в использовании верхнюю границу. Это же влечет и второе отличие от поиска радиуса — наличие различных алгоритмов для Задач 1 и 2. Без ограничения диаметра сверху количество просматриваемых пар вершин v_i, v_j — претендентов на периферийные вершины графа — может быть достаточно велико, а просмотр каждой вершины v_k в Задаче 1 требует предварительного решения SSSP(v_k). Поэтому оправданным в Задаче 1 является выполнение дополнительной операции, позволяющей уменьшить число просматриваемых пар вершин v_i, v_j и обладающей небольшой вычислительной сложностью в сравнении с решением SSSP. В качестве такой операции выступает сортировка пар вершин v_i, v_j по потенциальному расстоянию между ними и рассмотрение их в порядке убывания. В Задаче 2 сортировка нецелесообразна ввиду высокой сложности сортировки массива в сравнении с просмотром нескольких дополнительных столбцов матрицы расстояний.

Очевидно, что диаметр графа d ограничен снизу расстоянием между двумя любыми вершинами графа $d \geqslant d_l = m_{ij}, \forall i, j = \overline{1, n}$. Верхняя же граница графа не может быть определена без просмотра расстояний между всеми парами вершин больших нижней границы $m_{kl} > d_l$. Т.е., по сути, верхняя граница может быть вычислена только после определения диаметра и поэтому не имеет практической пользы. В таких условиях, одним из способов ускорить поиск диаметра является нахождение большой нижней границы d_l с наименьшими вычислительными затратами и ограничение просмотра претендентов на периферийные вершины только теми парами вершин v_k, v_l, расстояние между которыми потенциально может превосходить нижнюю границу $m_{kl} > d_l$.

В качестве первого приближения нижней границы диаметра d_l в алгоритмах используется максимальное расстояние между опорными вершинами. Как отмечалось выше, опорные вершины

являются неплохим приближением к периферийным вершинам графа. Для оценки потенциальных расстояний между претендентами на периферийные вершины v_k, v_l в алгоритмах используется расстояния от этих вершин до центра графа c. Выбор c в этом качестве позволяет отбросить из рассмотрения большое количество вершин v_k, v_l не являющихся периферийными $m_{kl} < d_l$.

Алгоритмы поиска диаметра состоят из двух частей. Вначале ищется один из центров графа алгоритмами, описанными в предыдущем разделе. Во второй части алгоритма производится непосредственный поиск диаметра на основе информации о расстояниях от центра до остальных вершин графа. Т.е. алгоритмы поиска диаметра полностью решают поставленные нами Задачи 1, 2 и могут называться алгоритмами поиска радиуса и диаметра.

3.2.1. АЛГОРИТМ ДЛЯ ЗАДАЧИ 2

После нахождения одного из центров графа c просматриваются расстояний только от тех вершин v_i, для которых выполняется неравенство

$$m_{ic} > d_l/2 \tag{5}$$

где d_l — текущая нижняя граница диаметра, вычисляемая по формуле

$$d_l = \max_{i,j \in P} m_{ij} \tag{6}$$

а P — множество опорных вершин, образованное при вычислении радиуса графа. Действительно, если диаметр графа больше текущей нижней границы и v_i — одна из периферийных вершин графа, для некоторой вершины v_j в силу неравенства треугольника для матрицы расстояний имеем $m_{ic} + m_{cj} \geqslant m_{ij} = d > d_l$. Т.е. $m_{ic} + m_{cj} > d_l = 2 \cdot d_l/2$ и необходимо либо $m_{ic} > d_l/2$, либо $m_{cj} > d_l/2$. Что доказывает необходимость рассматривать только вершины v_i, удовлетворяющие (5), в силу симметричности матрицы M. Описание алгоритма изображено на рисунке 2.

Алгоритм поиска диаметра для Задачи 2 (Д2)
Вход: граф $G = (V, E, w)$, матрица кратчайших расстояний M.
$d_l = 0$
1. Поиск центра графа
Используя алгоритм из раздела 3.1, находим один центров графа c.
Вычисляем нижнюю границу диаметра d_l по формуле (6).
2. Поиск диаметра
Просмотр на диаметр расстояний от вершин v_i, для которых выполняется формула (5). Если $m_{ij} > d_l$, $d_l = m_{ij}$.
Выход: d_l — диаметр графа.

Рис 2: Алгоритм поиска диаметра для Задачи 2.

3.2.2. АЛГОРИТМ ДЛЯ ЗАДАЧИ 1

В Задаче 1 просмотр всех вершин v_i, удовлетворяющих (5), может быть довольно длительным ввиду необходимости решения для каждой такой вершины SSSP(v_i). Поэтому для сужения круга претендентов на периферийные вершины графа можно использовать информацию о расстояниях от двух вершин до центра графа.

По определению кратчайшего расстояния для элементов матрицы расстояний M справедливо неравенство треугольника

(7) $$m_{ij} \leqslant m_{ik} + m_{kj}, \forall i, j, k$$

Значит, вычислив расстояния от некоторой вершины v_v до всех остальных, для определения диаметра нужно просматривать расстояния только между теми парами вершин v_k, v_l, для которых справедливо

(8) $$m_{kv} + m_{vl} > d_l$$

Для тех вершин, для которых это неравенство не выполнено, $m_{kl} \leqslant m_{kv} + m_{vl} \leqslant d_l$, т.е. расстояние между ними будет не больше текущей нижней границы диаметра. В качестве вершины v_v в алгоритме используется центр графа c.

Согласно (7) потенциально наибольшим расстоянием между собой обладают вершины v_k, v_l с максимальной суммой

$m_{kv} + m_{vl}$. Поэтому имеет смысл рассматривать вершины v_k, v_l в порядке убывания величины $m_{kv} + m_{vl}$. Если при таком проходе "сверху вниз"для какой-то из пар вершин v_k, v_l неравенство (8) не выполняется, то просмотр остальных пар, обладающих меньшим потенциальным расстоянием между собой, не нужен.

Сортировка n^2 пар v_k, v_l может быть весьма долгой в сравнении с общим временем нахождения диаметра, поэтому можно использовать следующий способ, позволяющий уменьшить время, несколько ухудшив качество сортировки. Вместо сортировки пар v_k, v_l по сумме $m_{kc} + m_{cl}$ следует сортировать вершины v_i по расстоянию до центра m_{ic}, а проверку по формуле (8) выполнять для пар вершин в порядке

(9) $$v_{m_1}v_{m_2}, v_{m_1}v_{m_3}, \ldots, v_{m_1}v_{m_n}, v_{m_2}v_{m_3}, v_{m_2}v_{m_4} \ldots$$

Здесь m_i — номер вершины с i-ым по величине расстоянием до центра графа. Если для некоторых $v_{m_i}v_{m_j}$ (8) не выполняется, переходим к просмотру $v_{m_{i+1}}v_{m_{i+2}}$. Если (8) не выполняется и при этом $j = i + 1$, конец алгоритма и d_l — диаметр графа. Описание алгоритма изображено на рисунке 3.

Алгоритм поиска диаметра для Задачи 1 (Д1)
Вход: граф $G = (V, E, w)$.
$d_l = 0$
1. Поиск центра графа
Используя алгоритм из раздела 3.1, находим один центров графа c.
Вычисляем нижнюю границу диаметра d_l по формуле (6).
2. Сортировка
Сортировка вершин v_i по расстоянию до центра c в порядке убывания.
3. Проверка претендентов
Выбор пары претендентов из списка (9)
Если для пары v_k, v_l выполняется (8), то если для v_k, v_l кратчайшие расстояния не вычислены, решаем SSSP(v_k), SSSP(v_l).
Если $m_{kl} > d_l$, $d_l = m_{kl}$
Выход: d_l — диаметр графа.

Рис 3: Алгоритм поиска диаметра для Задачи 1.

Так же, как и в случае с поиском радиуса, сложность алгоритма сильно зависит от структуры графа и от того насколько удачно были выбраны опорные вершины. В общем случае сложность можно оценить следующей формулой $diff(Ar)+diff(Sort(n))+k \cdot diff(SSSP) + O(x)$. Где $diff(Ar)$ — сложность алгоритма поиска радиуса и центра, $diff(Sort(n))$ — сложность используемого алгоритма сортировки n элементов, $k \cdot diff(SSSP)$ — сложность решения SSSP для k вершин на шаге 3 и $O(x)$ — проверка x пар вершин v_k, v_l на выполнение неравенства (8).

4. Результаты

Все тесты проводились на компьютере с процессором Intel Core i5 530 с частотой 2,93 ГГц и объемом памяти 3 ГБ в 32-рязрядной версии Windows XP. Программный код был написан на языке C++ с использование среды разработки Borland C++ Builder 6. В качестве первого набора тестовых данных использовались взвешенные графы дорожных сетей США из открытого доступа [13]. Вторым набором тестовых данных были сгенерированные нами полные графы со случайными весами ребер. Для полных графов результат брался как среднее по 10 различным графам с одинаковыми характеристиками. Параметры тестовых графов представлены в таблице 1.

Разработанные алгоритмы для Задачи 1 (обозначения Р1 и Д1) сравнивались с алгоритмом Дейкстры в реализации с двоичной кучей [5] для каждой вершины графа для графов с малой средней степенью вершин и алгоритмом Флойда-Уоршелла [7] для полных графов (РС1, ДС1). В силу того, что решение задачи алгоритмом Дейкстры невозможно за приемлемое время для графов большой размерности, время работы алгоритма для графов, начиная с RN_BAY, было приближено методом наименьших квадратов. В качестве данных для аппроксимации выступало время работы алгоритма на разной размерности подграфах тестовых графов, часть этих подграфов представлена в таблице 1 (RN_1, RN_2 и т.п.). Время работы алгоритма Дейкстры для всех вершин было аппроксимировано функциями

Таблица 1. Параметры тестовых графов.

Название	Вершин	Ребер	Средняя степень
RN_1	1001	2432	2,43
RN_2	2007	5288	2,63
RN_5	5000	$1,34 \cdot 10^4$	2,69
RN_7	7000	$1,8 \cdot 10^4$	2,58
RN_10	10^4	$2,7 \cdot 10^4$	2,69
RN_15	$1,5 \cdot 10^4$	$4,38 \cdot 10^4$	2,92
RN_20	$2 \cdot 10^4$	$5,41 \cdot 10^4$	2,71
RN_NY	$2,64 \cdot 10^5$	$7,34 \cdot 10^5$	2,78
RN_BAY	$3,21 \cdot 10^5$	$8 \cdot 10^5$	2,49
RN_COL	$4,36 \cdot 10^5$	$1,06 \cdot 10^6$	2,43
RN_FLA	$1,07 \cdot 10^6$	$2,71 \cdot 10^6$	2,53
RN_NW	$1,21 \cdot 10^6$	$2,84 \cdot 10^6$	2,35
RN_LKS	$2,76 \cdot 10^6$	$6,89 \cdot 10^6$	2,5
RN_E	$3,6 \cdot 10^6$	$8,78 \cdot 10^6$	2,44
RN_W	$6,26 \cdot 10^6$	$1,52 \cdot 10^7$	2,43
F_1	1000	10^6	1000
F_2	2000	$4 \cdot 10^6$	2000
F_5	5000	$2,5 \cdot 10^7$	5000
F_7	7000	$4,9 \cdot 10^7$	7000
F_10	10^4	10^8	10^4

$7,1 \cdot 10^{-7} n^2 \log(n) - 1,1 \cdot 10^{-7} nm \log(n) + 0,66$ для поиска радиуса и $7,2 \cdot 10^{-7} n^2 \log(n) - 1,1 \cdot 10^{-7} nm \log(n) + 0,98$ для поиска диаметра. В разработанных алгоритмах для Задачи 1 в качестве алгоритма решения SSSP используется алгоритм Дейкстры в реализации с двоичной кучей, в качестве сортирующего алгоритма в Д1 используется быстрая сортировка. В таблицах 2 и 3 приводятся результаты тестов для Задачи 1.

*Таблица 2. Результаты для Задачи 1, начало. P1 SSSP, штук — количество решений задач SSSP при использовании P1, P1 SSSP, доля — доля вершин с решенной SSSP от общего кол-ва вершин графа. Аналогично для Д1. * обозначены приближенные результаты.*

Граф	РС1, сек	Р1, сек	Р1 SSSP, штук	Р1 SSSP, доля	ДС1, сек	Д1, сек	Д1 SSSP, штук	Д1 SSSP , доля
RN_1	1,33	0,016	4	0,004	1,31	0	4	0,004
RN_2	5,45	0,031	7	0,0035	5,45	0,031	10	0,005
RN_5	39	0,031	4	0,0008	39	0,61	72	0,0144
RN_7	75	0,063	6	0,00086	75	5,5	492	0,0702
RN_10	170	0,17	10	0,001	170	0,72	41	0,00,1
RN_15	373	0,17	7	0,00047	374	0,77	30	0,002
RN_20	761	0,23	6	0,0003	762	25	635	0,0318
RN_NY	$1,54 \cdot 10^5$	3,5	6	$2,3 \cdot 10^{-5}$	$1,54 \cdot 10^5$	5,3	9	$3,4 \cdot 10^{-5}$
RN_BAY	$2,49 \cdot 10^{5*}$	3,16	4	$1,3 \cdot 10^{-5}$	$2,49 \cdot 10^{5*}$	3,31	4	$1,25 \cdot 10^{-5}$
RN_COL	$4,76 \cdot 10^{5*}$	7,09	6	$1,4 \cdot 10^{-5}$	$4,77 \cdot 10^{5*}$	233	198	0,00045
RN_FLA	$2,99 \cdot 10^{6*}$	13	4	$3,7 \cdot 10^{-6}$	$2,99 \cdot 10^{6*}$	16	5	$4,7 \cdot 10^{-6}$
RN_NW	$4,03 \cdot 10^{6*}$	14	4	$3,3 \cdot 10^{-6}$	$4,03 \cdot 10^{6*}$	15	4	$3,3 \cdot 10^{-6}$
RN_LKS	$2,14 \cdot 10^{7*}$	37	4	$1,5 \cdot 10^{-6}$	$2,14 \cdot 10^{7*}$	38	4	$1,5 \cdot 10^{-6}$
RN_E	$3,76 \cdot 10^{7*}$	49	4	$1,1 \cdot 10^{-6}$	$3,77 \cdot 10^{7*}$	51	4	$1,1 \cdot 10^{-6}$
RN_W	$1,18 \cdot 10^{8*}$	92	4	$6,4 \cdot 10^{-7}$	$1,19 \cdot 10^{8*}$	97	4	$6,4 \cdot 10^{-7}$

*Таблица 3. Результаты для Задачи 1, продолжение. P1 SSSP, штук — количество решений задач SSSP при использовании P1, P1 SSSP, доля — доля вершин с решенной SSSP от общего кол-ва вершин графа. Аналогично для Д1. * обозначены приближенные результаты.*

Граф	РС1, сек	Р1, сек	Р1 SSSP, штук	Р1 SSSP, доля	ДС1, сек	Д1, сек	Д1 SSSP, штук	Д1 SSSP , доля
F_1	10	0,18	9	0,009	10	0,27	13,9	0,0139
F_2	81	1	10,3	0,0052	81	1,75	17,8	0,0089
F_5	1246	12	9,5	0,0019	1246	21	15,4	0,0031
F_7	3401	33	10	0,0014	3401	45	13,7	0,002
F_10	9853	71	9,5	0,00095	9853	123	16,5	0,0017

Заметно значительное сокращение времени поиска радиуса и диаметра всех тестируемых графов разработанными алгоритмами для Задачи 1. Процент вершин, для которых необходимо решить SSSP, при поиске радиуса не превышает 0,9%, а при поиске диаметра 7%.

Ввиду невозможности хранения информации о расстояниях в оперативной памяти и вытекающей некорректной оценки времени счета для графов большой размерности, тесты для Задачи 2 проводились для графов с числом вершин не более 10^4 . Сравнение разработанных алгоритмов (Р2, Д2) проводилось с проходом по всем элементам матрицы расстояний для поиска радиуса (РС2) и с проходом верхнего треугольника матрицы расстояний для поиска диаметра (ДС2). В связи с небольшим временем решения задачи для исследуемых графов каждый алгоритм запускался последовательно 1000 раз. Результаты тестов для Задачи 2 приведены в таблице 3.

Таблица 4. Результаты для Задачи 2.

Граф	РС2, сек	Р2, сек	Р, ускорение	ДС2, сек	Д2, сек	Д, ускорение
RN_1	4,21	0,031	135	1,7	0,047	36
RN_2	156	0,14	1114	7,3	0,17	42
RN_5	95	0,16	593	45	1,38	32
RN_7	181	0,38	476	87	16	5
RN_10	374	0,89	420	177	2,19	80
F_1	4,07	0,072	56	2,08	0,107	19
F_2	16	0,16	100	8,16	0,26	31
F_5	100	0,43	232	50	0,69	72
F_7	196	0,63	311	98	0,86	113
F_10	399	0,81	492	200	1,27	157

Для Задачи 2 на рассматриваемых графах также наблюдается значительное преимущество разработанных алгоритмов над сравниваемыми — ускорение поиска радиуса от 56 раз и ускорение поиска диаметра от 5 и более раз.

Заключение

Разработанные алгоритмы позволяют решать поставленные задачи поиска метрических характеристик 1 и 2 с использованием информации о расстояниях лишь от небольшой части вершин графа. Результаты тестов показывают, что, по крайней мере, на рассматриваемых графах процентная доля используемой информации в общем случае уменьшается с увеличением размерности графа. Такое поведение разработанных алгоритмов говорит об увеличивающемся выигрыше во времени при использовании алгоритмов на графах большой размерности.

Основные результаты проведенных тестов в численном выражении выглядят следующим образом. Для Задачи 1 при поиске радиуса и центра процент вершин, для которых решалась SSSP, не превышает 0,9%, ускорение времени поиска изменяется в пределах от 55 до $1,28 \cdot 10^6$ раз, для поиска диаметра эти цифры составляют, соответственно, 7% и от 14 до $1,22 \cdot 10^6$ раз. Для Задачи 2 при поиске радиуса и центра ускорение времени изменяется в пределах от 56 до $1,14 \cdot 10^3$ раз, при поиске диаметра достигается ускорение от 5 до 157 раз.

В качестве дальнейших исследований по теме работы возможны тестирование разработанных алгоритмов на других видах графов, оценка области применения алгоритмов. Еще одним направлением для исследований является дальнейшая оптимизация алгоритмов за счет использования более быстрого алгоритма сортировки и алгоритмов решения SSSP более точно соответствующих исследуемым графам. Интересным также является вопрос применимости подхода использованного в алгоритмах для поиска всех центров и периферийных вершин графа.

Литература

1. BERMAN P., KASIVISWANATHAN S.P. *Faster Approximation of Distances in Graphs* // Algorithms and Data Structures. Lecture Notes in Computer Science. - 2007. - Vol. 4619. - P. 541-552.
2. BOLDI P., ROSA M., VIGNA S. *HyperANF: approximating the neighbourhood function of very large graphs on a budget* // Proceedings of the 20th international conference on World wide web. - New York: ACM, 2011. - P. 625-634.
3. CHAN T.M. *More Algorithms for All-Pairs Shortest Paths in Weighted Graphs* // Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. - New York: ACM, 2007. - P. 590-598.
4. CHEPOI V., DRAGAN F. ET AL. *Center and Diameter Problems in Plane Triangulations and Quadrangulations* // Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms. - Philadelphia: Society for Industrial and Applied Mathematics, 2002. - P. 346-355.
5. CORMEN T.H., LEISERSON C.E., RIVEST R.L., STEIN C. *Introduction to Algorithms, Second Edition*. MIT Press, 2001. - 1202 p.
6. DIJKSTRA E.W. *A note on two problems in connection with graphs* // Numerische Mathematik 1. - 1959. - P. 83-89.
7. FLOYD R.W. *Algorithm 97: Shortest Path* // Communications of the ACM. - 1962. - Vol. 5, N 6. - P. 345.
8. JOHNSON D.B. *Efficient algorithms for shortest paths in sparse networks* // Journal of the ACM. - 1977. - Vol. 24, N 1. - P.1-13.
9. KNUTH D.E. *The Art Of Computer Programming Vol 1.*: 3rd ed. Boston: Addison-Wesley, 1997. - 650 p.
10. KUNG U., TSOURAKAKIS C.E., APPEL A.P., FALOUTSOS C., LESKOVEC J. *Radius Plots for Mining Tera-byte Scale Graphs: Algorithms, Patterns, and Observations* // SIAM International Conference on Data

Mining 2010. - 2010. - P.548-558.
11. SHOSHAN A., ZWICK U. *All Pairs Shortest Paths in Undirected Graphs with Integer Weights* // Proceedings of the 40th Annual Symposium on Foundations of Computer Science. - Washington: IEEE Computer Society, 1999. - P. 605-614.
12. TAKES F.W., KOSTERS W.A. *Determining the diameter of small world networks* // Proceedings of the 20th ACM international conference on Information and knowledge management. - New York: ACM, 2011. - P. 1191-1196.
13. *9th DIMACS Implementation Challenge — Shortest Paths* [Электронный ресурс]. – Режим доступа: http://www.dis.uniroma1.it/challenge9/download.shtml (дата обращения: ноябрь 2012).

FAST SEARCH ALGORITHMS FOR TWO METRIC CHARACTERISTICS PROBLEMS ON WEIGHTED GRAPHS

Airat Urakov, Ufa State Aviation Technical University, Ufa, Cand.Sc., assistant professor (urakov@ufanet.ru).
Timofey Timeryaev, Ufa State Aviation Technical University, Ufa, graduate student (timeryaev@yandex.ru).

Abstract: Two problems in the search of metric characteristics on weighted undirected graphs with non-negative edge weights are being considered. The first problem: a weighted undirected graph with non-negative edge weight is given. The radius, diameter and at least one center and one pair of peripheral vertices of the graph are to be found. In the second problem we have additionally calculated the distances matrix. For the problems being considered, we proposed fast search algorithms which use only small fraction of graph's vertices for the search of the metric characteristics. The proposed algorithms have been compared to other popular methods of solving problems considered on various inputs.

Keywords: metric characteristics of a graph, graph radius, graph diameter, graph center, graph peripheral vertices.