Определение оптимальных путей транспортной сети на базе графовой декомпозиции

Т.В. Тимеряев
Общенаучный факультет
Уфимский государственный авиационный технический университет
Уфа, Россия
e-mail: timeryaev@yandex.ru

Н.М. Шерыхалина
Общенаучный факультет
Уфимский государственный авиационный технический университет
Уфа, Россия
e-mail: n_sher@mail.ru

Аннотация[1]

Рассматривается задача поиска кратчайших путей между всеми вершинами взвешенного ненаправленного сильно разреженного графа. Для задачи предлагается алгоритм, использующий решение задачи на графе малой размерности для получения решения для исходного графа. Приводится сравнение предлагаемого алгоритма с различными реализациями быстрейшего известного алгоритма на данных из открытого доступа.

1. Введение

Задача поиска кратчайших путей между всеми парами вершин графа (APSP) является одной их классических в теории графов. Такой ее статус обусловлен тем, что при решении оптимизационных задач на графах из многих областей (задача коммивояжера, транспортная задача и многие другие) нам необходимо знать расстояния между всеми вершинами графа.

Новый приток интереса к задаче произошел с появлением создаваемых полуавтоматически графов высокой подробности, описывающих структуры реального мира. Подобные графы обладают размерностью от 106 и более, а их подробность со временем будет только увеличиваться.

Соответственно своему статусу, для задачи APSP существует множество алгоритмов ее решающих, но нет способа, получающего решения одинаково быстро для любого вида входов. В связи с этим алгоритмы решения APSP принято делить по типу графов, на которых задача решается. Известными являются алгоритмы, в которых входы: направленные графы [2], полные графы [4], взвешенные графы [3], невзвешенные графы [1], разреженные графы [6].

В данной статье рассматривается задача APSP для взвешенных ненаправленных сильно разреженных графов с неотрицательными весами ребер. Для рассматриваемой задачи предлагается алгоритм, сводящий решения задачи к замене графа исходного графом меньшей размерности и решению задачи на нем. Приводится сравнение предлагаемого алгоритма с быстрейшим известным алгоритмом на графах дорожных сетей.

2. Термины и определения

Задача поиска кратчайших путей рассматривается на связном ненаправленном разреженном взвешенном графе $G=(V,E,w)$ с неотрицательными весами ребер, где $V=\{v_1,v_2,\ldots,v_n\}$ – множество вершин графа, $E=\{e_1,e_2,\ldots,e_m\}:e_i=e(v_j,v_k)$ – множество ребер графа, а $w:E\rightarrow[0,\infty)$ – весовая функция на ребрах. Будем считать рассматриваемый граф простым, т.е. без петель и кратных ребер. Мощности множества вершин и множества ребер будем считать равными: $|V|=n$, $|E|=m$. Введем ряд определений и обозначений.

Вес ребра между вершинами v_i и v_j обозначается $w(i,j)$, для вершин v_i и v_j не связанных ребром полагается $w(i,j)=\infty$. *Степень* $d(v_i)$ *вершины* v_i – количество ребер графа, инцидентных v_i. Граф *разреженный*, если справедливо $m=n^2$.

Длина пути – сумма весов входящих в него ребер. *Кратчайший путь* $p_{ij}^s = p^s(v_i, v_j)$ между вершинами v_i и v_j – путь между v_i и v_j минимальной длины. *Расстояние* $m(i,j)$ между v_i и v_j – длина кратчайшего пути между v_i и v_j ($m(i,i)=0,\forall i$). *Матрица расстояний* графа – матрица $M=\left(m_{ij}\right)_{i=1,\,j=1}^{n,n}$, на пересечении i-ой строки и j-го столбца которой находится элемент с величиною равной расстоянию из v_i в v_j. Граф *связный*, если между любыми двумя вершинами графа существует путь, то есть $m_{ij}<\infty,\forall i,j$.

Между любой парой вершин графа может существовать несколько кратчайших путей. Этот аспект в данной статье не является существенным, поэтому при всех упоминаниях кратчайшего пути будет подразумеваться любой кратчайший путь.

Труды второй международной конференции "Интеллектуальные технологии обработки информации и управления", 10 - 12 ноября, Уфа, Россия, 2014

Матрица предшествования – матрица $P=\left(p_{ij}\right)_{i=1,\,j=1}^{n,n}$, элемент p_{ij} которой представляет собой метку вершины, которая предшествует вершине v_j в кратчайшем пути от v_i до v_j. То есть элементы матрицы P определяются по формуле

$$p_{ij}=\begin{cases} v_k, \text{ if } \exists v_k : p_{ij}^s = \ldots v_k, e\left(v_k, v_j\right), v_j \\ \infty, \qquad\qquad \text{else} \end{cases} \qquad (1)$$

С использованием матрицы P кратчайший путь p_{ij}^s для вершин $v_i \neq v_j$ в связном графе может быть определен по рекурсивной формуле

$$p_{ij}^s=\begin{cases} p^s\left(v_i, p_{ij}\right), e\left(p_{ij}, v_j\right), v_j, & p_{ij} \neq v_i \\ v_i, e\left(v_i, v_j\right), v_j, & p_{ij}=v_i \end{cases}$$

В статье вводятся следующие дополнительные определения и обозначения. Обозначим через S и будем называть *сжимающей* $G_0=(V_0,E_0,w_0)$ последовательность графов $S=\{G_1,G_2,\ldots,G_r\}$, где $G_p=(V_p,E_p,w_p)$ такой, что $V_p=\left\{v_1^p, v_2^p, \ldots, v_{n(p)}^p\right\}$, $E_p=\left\{e_1^p, e_2^p, \ldots, e_{m(p)}^p\right\}: e_i^p=e^p\left(v_j^p, v_k^p\right)$ и $w_p: E_p \rightarrow [0,\infty)$. Каждый следующий граф G_{p+1} последовательности получается из предыдущего G_p удалением k вершин и инцидентных им ребер, добавлением новых ребер и пересчетом некоторым образом весов ребер между вершинами бывшими смежными с удаленными. Для таких графов справедливо $\left|V_p\right| > \left|V_{p+1}\right|, \forall p=\overline{0,r-1}$. Будем обозначать v_i^{p+1} вершину графа G_{p+1}, соответствующую вершине v_i^p графа G_p. Будем обозначать $e^{p+1}\left(v_j^{p+1}, v_k^{p+1}\right)$ ребро графа G_{p+1}, соответствующее ребру $e^p\left(v_j^p, v_k^p\right)$ графа G_p. Граф, полученный удалением вершин $v_1^p, v_2^p, \ldots, v_k^p$ и инцидентных им ребер из графа G_p, будем обозначать $G_{p+1}=R_p\left(v_1^p, v_2^p, \ldots, v_k^p\right)$. Для такого графа справедливо:

$$w_{p+1}(i,j)=w_p(i,j), \forall i,j : v_i^{p+1}, v_j^{p+1} \in V_{p+1}.$$

Расстояние между вершинами v_i^p и v_j^p графа G_p будем обозначать $m^p(i,j)$, а его матрицу расстояний $M_p=\left(m_{ij}^p\right)$. Будем обозначать $\left\{v_{i_z}^p\right\}$ вершины графа G_p такие, что $\left\{v_{i_z}^p\right\}=\left\{v_{i_l}^p \in V_p : e^0\left(v_i^0, v_{i_l}^0\right) \in E_0\right\}$.

3. Постановка задачи

Дан связный ненаправленный разреженный простой взвешенный граф $G=(V,E,w)$ с неотрицательной вещественной весовой функцией на ребрах $w: E \rightarrow [0,\infty)$. Необходимо найти кратчайшие пути между всеми парами вершин данного графа, т.е. найти матрицу расстояний M и матрицу предшествования P.

Принцип предлагаемого алгоритма состоит в сведении задачи на исходном графе большой размерности к задаче на графе малой размерности. Алгоритм можно разделить на 3 этапа:

- Сжатие. Замена исходного графа большой размерности графом меньшей размерности;
- Микрорешение. Решение задачи о кратчайших путях на графе меньшей размерности, используя существующие методы;
- Восстановление. Перенос решения с графа меньшей размерности на исходный граф. Пересчет кратчайших расстояний для некоторой части вершин исходного графа.

Такой подход требует выполнения следующих условий. а) достоверность – сжатие должно сохранять информацию о кратчайших путях исходного графа; б) скорость – выполнение всех этапов в сумме должно требовать меньше времени, чем решение задачи известными способами на исходном графе.

Для разреженных графов нами предлагается алгоритм «разборки и сборки графа». На этапе разборки из графа последовательно удаляются вершины, далее производится поиск решения на малом графе, после чего исходный граф собирается с получением искомых кратчайших расстояний.

Отличия предлагаемого нами алгоритма от уже известных алгоритмов со сжатием или подменой ребер в том, что он: достаточно прост, выполняет расчет кратчайших путей сразу между всеми вершинами, при этом гарантированно находит точное решение (не является эвристическим). В частности, в [5] решается задача поиска кратчайшего пути от источника до пункта назначения (point-to-point), одной из основных целей которой является уменьшение времени запроса, тогда как время решения задачи APSP для графов большой размерности таким алгоритмом может быть весьма велико

4. Алгоритм

Этап разборки графа является многоступенчатым и состоит в последовательном приближении исходного графа $G_0=(V_0,E_0,w_0)$ графами сжимающей G_0 последовательности $S=\{G_1,G_2,\ldots,G_r\}$. В статье мы рассматриваем частный случай, когда каждый следующий граф G_{p+1} последовательности S получается из предыдущего графа G_p удалением одной вершины.

Пусть удаляемая вершина v_i^p графа G_p имеет степень k, т.е. с ней смежны вершины $\left\{v_{i_z}^p\right\}_{z=1}^{k}$. Если какой-то из кратчайших путей графа проходит через вершину

v_i^p (не считая кратчайших путей от и до самой v_i^p), то этот путь обязательно содержит подпуть вида $v_{i_j}^p, e^p\left(v_{i_j}^p, v_i^p\right), v_i^p, e^p\left(v_i^p, v_{i_l}^p\right), v_{i_l}^p : j,l \in \{1,2,\ldots,n\}$.

Таким образом, при удалении вершины v_i^p необходимо гарантировать сохранение кратчайших путей только между вершинами смежными с v_i^p.

Обозначим
$w_p^{mv(1,2,\ldots,h)}(i_j,i_l) = \min\limits_{g=1,2,\ldots,h}\left(w_p(i_j,g)+w_p(g,i_l)\right)$
минимальную сумму весов двух ребер, соединяющих вершины $v_{i_j}^p$ и $v_{i_l}^p$ и проходящих через одну из вершин $v_1^p, v_2^p, \ldots, v_h^p$ графа G_p. Для сохранения расстояний достаточно для всех пар вершин $\left(v_{i_j}^p, v_{i_l}^p\right)$ смежных с v_i^p в следующем графе G_{p+1} последовательности положить

$$w_{p+1}(i_j,i_l) = \begin{cases} mvi & \text{if}, w_p^{mv(i)}(i_j,i_l) < w_p^{mv(h\neq i)}(i_j,i_l) \\ w_p(i_j,i_l) & ,\text{else} \end{cases} \quad (1)$$

где $mvi = \min\left(w_p^{mv(i)}(i_j,i_l), w_p(i_j,i_l)\right)$. В начале алгоритма вводится матрица $P' = \left(p'_{ij}\right)_{i=1,j=1}^{n,n} : p'_{ij} = \infty, \forall i,j$. Для сохранения информации о путях элементы матрицы P', для соответствующих номеров которых справедливо $w_p^{mv(i)}(i_j,i_l) < \min\left(w_p(i_j,i_l), w_p^{mv(h\neq i)}(i_j,i_l)\right)$, изменяются по формуле

$$p'_{i_j i_l} = \begin{cases} v_i, if\ p'_{ii_l} = \infty \\ p'_{ii_l}, p'_{ii_l} \neq \infty \end{cases} \quad (2)$$

В частности, если удаляемая вершина v_i^p смежна только с одной вершиной графа G_p, то, очевидно, сохранять кратчайшие пути надобности нет (т.к. через v_i^p кратчайшие пути не проходят), и, поэтому, в этом случае вершина v_i^p и инцидентное ей ребро просто удаляются из графа.

В качестве параметров на этапе разборки выступают: $d_{\max}$ –максимальная степень удаляемых вершин, $n_{\min}$ – число вершин в самом малом графе G_r и $I_{\max}$ – ограничение на рост числа ребер при удалении вершины. Вопрос совместного определения $d_{\max}$, $n_{\min}$ и $I_{\max}$ является весьма объемным, поэтому его рассмотрение остается вне рамок данной статьи. В частности, для проведенных тестов, результаты которых приведены в разделе 5: $d_{\max} = I_{\max} = \infty$, $n_{\min} = 1$.

Пусть на предмет удаления рассматривается вершина v_i^p с k инцидентными ребрами. Обозначим через $I\left(v_i^p\right)$ величину изменения количества ребер при удалении вершины v_i^p. Само удаление вершины v_i^p уменьшит число ребер в графе на k, поэтому полагаем $I\left(v_i^p\right) = -k$. Пересчет весов ребер по формуле (1) между каждой парой $\left(v_{i_j}^p, v_{i_l}^p\right)$ смежных с v_i^p вершин изменит величину $I\left(v_i^p\right)$ согласно формуле

$$I\left(v_i^p\right) = \begin{cases} I\left(v_i^p\right)+1 & ,\text{if } w_p(i_j,i_l) = \infty \\ I\left(v_i^p\right)+1 & ,\text{if } w_p^{mv(i)}(i_j,i_l) < w_p^{mv(h\neq i)}(i_j,i_l) \\ I\left(v_i^p\right) & ,\text{otherwise} \end{cases} \quad (3)$$

Таким образом будет получено число, на которое изменится количество ребер в графе G_{p+1} относительно графа G_p после удаления вершины v_i^p. Если $I\left(v_i^p\right) > 0$, число ребер увеличится, иначе – число ребер уменьшится или останется неизменным. Использование формулы (3) предполагает применение ограничения $I_{\max}$ на рост числа ребер при удалении вершины. После задания этого параметра удаляются лишь те вершины v_i^p, для которых выполняется нестрогое равенство $I\left(v_i^p\right) \le I_{\max}$.

В процессе разборки просмотр вершин-претендентов на удаление происходит следующим образом. Так как вершины с $d\left(v_i^p\right) < 3$ удаляются в любом случае (при $d_{\max} \ge 2$), то рассмотрение вершин происходит в порядке роста их степеней от 1 до $d_{\max}$. Это позволяет сразу уменьшить степени оставшихся вершин и ускорить работу алгоритма за счет меньшего числа просмотров вершин со значениями степени близкими к $d_{\max}$. После удаления вершины v_i^p степени смежных с ней вершин $\left\{v_{i_z}^p\right\}_{z=1}^{k}$ в общем случае могут измениться. Поэтому, для того, чтобы обеспечить удаление всех вершин со степенью $d\left(v_j^p\right) \le d_{\max}$, после удаления вершины v_i^p рекурсивно рассматриваются бывшие смежные с ней до удаления вершины, степень которых уменьшилась в результате удаления. После завершения рекурсивного вызова просмотр вершин продолжается в обычном порядке.

На этапе микрорешения решается задача APSP для графа G_r последовательности S, полученного на этапе разборки. Результатом выполнения данного шага алгоритма являются матрица расстояний M_r графа G_r. Вводится матрица $M'_r = M_r$ и производится изменение матрицы P' по формулам

$$p'_{ij} = \begin{cases} p^r_{ij}, p'_{ij} = \infty \wedge p'_{p^r_{ij} j} = \infty \\ p'_{p^r_{ij} j}, p'_{ij} = \infty \wedge p'_{p^r_{ij} j} \neq \infty \end{cases} \quad (4)$$

$$p'_{ij} = \begin{cases} p^r_{ij}, w_r(i,j) > m^r_{ij} \wedge p'_{p^r_{ij} j} = \infty \\ p'_{p^r_{ij} j}, w_r(i,j) > m^r_{ij} \wedge p'_{p^r_{ij} j} \neq \infty \end{cases} \quad (5)$$

где p^r_{ij} – элементы матрицы предшествования P графа G_r. Найденные на данном этапе алгоритма пути между вершинами графа G_r будут гарантированно кратчайшими, так как удаление вершин, произведенное на этапе разборки, сохраняет расстояния. То есть справедливо $m'^r_{ij} = m^r_{ij} = m^0_{ij}, \forall i, j : v^r_i, v^r_j \in V_r$.

Если G_r содержит всего одну вершину $|V_r|=1$, тогда данный этап алгоритм пропускается и сразу выполняется сборка графа.

До начала работы этапа сборки алгоритма определена последовательность графов $S=\{G_0,G_1,\ldots,G_r\}$. Здесь G_0 – исходный граф, G_r – наименьший граф последовательности, у которого на этапе микрорешения найдены кратчайшие пути между всеми парами вершин. На этапе сборки производится обратный ход от G_r к G_0 через графы $G_{r-1},G_{r-2},\ldots,G_1$ с расчетом кратчайших путей на каждом шаге p для вершины $v^{r-p}_i : v^{r-p+1}_i \in V_{r-p+1} \wedge v^{r-p}_i \in V_{r-p}$.

Вершина v^{r-1}_i, «добавляемая» на первом шаге данного этапа алгоритма к графу G_r при переходе к G_{r-1}, соединена ребрами с k вершинами $\left\{v^{r-1}_{i_z}\right\}_{z=1}^{k}$ графа G_{r-1}. У G_r вычислены кратчайшие пути, т.е. известна матрица $M'_r = M_r$. Поэтому, чтобы найти матрицу расстояний M'_{r-1} графа G_{r-1}, нам надо вычислить лишь кратчайшие пути от вершины v^{r-1}_i до всех остальных вершин графа G_{r-1}, положив остальные элементы матрицы M'_{r-1} равными соответствующим элементам M'_r, т.е. $m'^{r-1}_{jl} = m'^r_{jl}, \forall j,l : v^r_j, v^r_l \in V_r$.

Так как кратчайший путь от любой вершины графа G_{r-1} до v^{r-1}_i проходит через $\left\{v^{r-1}_{i_z}\right\}_{z=1}^{k}$, кратчайшие расстояния от v^{r-1}_i до любой вершины v^{r-1}_l графа G_{r-1} можно рассчитать по формуле $m'^{r-1}(i,l) = \min_{z=1,k}\left(w_{r-1}(i,i_z) + m'^r(i_z,l)\right)$.

Формулы расчета матрицы расстояний M'_{r-p} при переходе от графа G_{r-p+1} к графу G_{r-p} с добавлением вершины v^{r-p}_i выглядит следующим образом

$$m'^{r-p}_{jl} = m'^{r-p+1}_{jl} \quad (6)$$

$$m'^{r-p}_{il} = \min_{z=1,k}\left(w_{r-p}(i,i_z) + m'^{r-p+1}(i_z,l)\right) \quad (7)$$

где where $j,l : v^{r-p+1}_j, v^{r-p+1}_l \in V_{r-p+1}$. Обозначим $x(l)$ номер i_z, на котором достигается минимум формулы (7). Элементы матрицы P', для соответствующих вершин которых либо $w_{r-p}(i,l) > m'^{r-p}_{il} \vee w_{r-p}(l,i) > m'^{r-p}_{li}$, либо $p'_{il} = \infty, p'_{li} = \infty$, изменяются по формулам

$$p'_{il} = \begin{cases} v_i, p'_{x(l)l} = \infty \\ p'_{x(l)l}, p'_{x(l)l} \neq \infty \end{cases} \quad (8)$$

$$p'_{li} = \begin{cases} v_l, p'_{x(l)i} = \infty \\ p'_{x(l)i}, p'_{x(l)i} \neq \infty \end{cases} \quad (9)$$

5. Результаты

Все тесты проводились на компьютере с процессором Intel Core 2 Duo E8400 с частотой 3 ГГц и объемом памяти 4 ГБ в 64-рязрядной версии Windows 7. Программный код был написан на языке C++ с использование среды разработки Microsoft Visual Studio 2010. В качестве тестовых данных использовались взвешенные графы дорожных сетей США из открытого доступа [7]. Из этих графов были получены связные подграфы размерностью от 10^3 до 10^4 в количестве 10 штук для каждой размерности ($U1$-$U10$, $U15$, $U20$).. Еще одним набором тестовых данных выступили графы дорожных сетей городов России, составленные экспертами в области пассажирских перевозок (R). Характеристики тестовых графов представлены в таблице 1.

Тестирование проводилось при: $d_{max}=I_{max}=\infty$, $n_{min}=1$. То есть, при разборке удалялись вершины с любой степенью до тех пор, пока в наименьшем графе G_r не оставалась лишь одна вершина. Соответственно, второй этап алгоритма – микрорешение – не выполнялся. Разработанный алгоритм (обозначение РС) сравнивался с быстрейшим известным алгоритмом для разреженных графов – алгоритмом Дейкстры [1] в реализации из библиотеки Boost Graph Library 1.54 [8].

Таблица 1. Тестовые графы

Группа графов	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	$\overline{d(v_i)}$	Макс. $\overline{d(v_i)}$	План./ всего
*U*1	10^3	$2{,}5\cdot10^3$	2,5	6	10/10
U2	$2\cdot10^3$	$5{,}2\cdot10^3$	2,6	5	10/10
U3	$3\cdot10^3$	$7{,}9\cdot10^3$	2,6	6	10/10
U4	$4\cdot10^3$	$1{,}1\cdot10^4$	2,7	6	10/10
U5	$5\cdot10^3$	$1{,}3\cdot10^4$	2,7	6	10/10
U6	$6\cdot10^3$	$1{,}6\cdot10^4$	2,6	7	10/10
U7	$7\cdot10^3$	$1{,}9\cdot10^4$	2,6	6	10/10
U8	$8\cdot10^3$	$2{,}1\cdot10^4$	2,6	6	10/10
U9	$9\cdot10^3$	$2{,}4\cdot10^4$	2,6	7	10/10
*U*10	10^4	$2{,}7\cdot10^4$	2,7	7	10/10
*U*15	$1{,}5\cdot10^4$	$4{,}4\cdot10^4$	2,9	7	10/10
U20	$2\cdot10^4$	$5{,}4\cdot10^4$	2,7	6	10/10
R	$1{,}9\cdot10^3$	$5{,}4\cdot10^3$	2,8	14	6/50

При сравнении в качестве очереди с приоритетом в алгоритме Дейкстры использовались различные структуры данных: 4-арная куча (по умолчанию в Boost, Д4), relaxed куча (Д*R*), Фибоначчиева куча (с лучшей теоретической оценкой, ДФ). Результаты тестов приведены в таблице 2.

Таблица 2. Результаты.

Группа графов	РС среднее время, с	Д4 среднее время, с	ДR среднее время, с	ДФ среднее время, с	Ускор., раз
*U*1	0,011	0,108	0,276	0,295	10,2
U2	0,063	0,458	1,24	1,33	7,2
U3	0,163	1,05	2,88	3,21	6,33
U4	0,358	1,92	5,34	6,23	5,34
U5	0,516	3,06	8,43	10,1	5,93
U6	0,783	4,4	12,5	13,5	5,62
U7	1,1	6,14	17,4	19,6	5,6
U8	1,48	7,98	23,2	25,8	5,4
U9	1,87	10,2	29,6	33,9	5,47
*U*10	2,43	13,1	37,5	43,7	5,38
*U*15	7,17	30,2	90,3	104	4,21
U20	18,7	55,1	160	195	2,95
R	0,081	0,565	1,35	1,72	6,92

Использование разработанного алгоритма позволяет ускорить решение задачи APSP на рассматриваемых графах в среднем в 5,24 раза в сравнении с быстрейшим из сравниваемых. Наибольшее ускорение достигается на графах размерности 10^3 – в среднем в 8 раз, наименьшее на графах размерности $2\cdot10^4$ – в среднем в 2,5 раза.

6. Заключение

Представленный в статье алгоритм позволяет эффективно находить кратчайшие пути между всеми вершинами сильно разреженных взвешенных ненаправленных графов. Результаты тестов, проведенных на графах из открытого доступа, позволяют говорить об ускорении решения задачи APSP в среднем в 5,2 раза в сравнении с быстрейшим известным алгоритмом для разреженных графов.

Список используемых источников

1. Кормен Т., Лейзерстон Ч., Ривест Р., Штайн К. “Алгоритмы: построение и анализ”**.** М.: «Вильямс», 2006, 1296 с.
2. Bellman R. “On a routing problem”. *Quarterly of Applied Mathematics*, 1958, 16, 87-90.
3. Dijkstra E.W. “A note on two problems in connexion with graphs”. *Numerische Mathematik*, 1959, 1, 269-271.
4. Floyd R.W. “Algorithm 97: Shortest Path”. *Communications of the ACM*, 1962, 5(6), 345.
5. Geisberger R. “Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks”. *International Workshop on Experimental Algorithms*, 2008, 319-333.
6. Johnson D.B. “Efficient algorithms for shortest paths in sparse graph”. *Journal of the ACM*, 1977, 24, 1-13.
7. 9th DIMACS Implementation Challenge - Shortest Paths [Электронный ресурс] – Режим доступа: http://www.dis.uniroma1.it/challenge9/download.shtml
8. The Boost Graph Library - 1.54.0 [Электронный ресурс] – Режим доступа: http://www.boost.org/doc/libs/1_54_0/libs/graph/doc/index.html