УДК 519.178

АЛГОРИТМ ПОИСКА КРАТЧАЙШИХ ПУТЕЙ ДЛЯ РАЗРЕЖЕННЫХ ГРАФОВ БОЛЬШОЙ РАЗМЕРНОСТИ

А. Р. Ураков, Т. В. Тимеряев

Уфимский государственный авиационный технический университет, г. Уфа, Россия

E-mail: urakov@ufanet.ru, timeryaev@yandex.ru

Рассматривается задача поиска кратчайших путей между всеми вершинами взвешенного ненаправленного разреженного графа. Предлагается алгоритм «разборки и сборки графа», использующий решение задачи на графе малой размерности для получения решения для исходного графа. Приводится сравнение предлагаемого алгоритма с одним из быстрых классических алгоритмов на данных из открытого доступа.

Ключевые слова: *задача о кратчайших путях, APSP, разборка графа, сборка графа, сжатие графа, разреженные графы.*

Введение

Задача поиска кратчайших путей между всеми парами вершин графа (APSP — All-Pairs Shortest Path problem) является одной их классических в теории графов. Такой её статус обусловлен тем, что при решении оптимизационных задач на графах из многих областей (задача коммивояжёра, транспортная задача и многие другие) необходимо знать расстояния между всеми вершинами графа. Более того, сама задача представляет большой исследовательский интерес, так как её решение позволяет ответить, например, на такие вопросы:

— Как попасть из любого пункта A в любой пункт $Б$ с наименьшими затратами времени/денежных средств?
— Каков ближайший/самый удалённый пункт для A и каково расстояние до него?

Новый приток интереса к задаче произошёл с появлением создаваемых полуавтоматически графов высокой подробности, описывающих структуры реального мира. Подобные графы имеют размерность 10^6 и более, а их подробность со временем будет только увеличиваться. Поэтому актуальным становится вопрос об ускорении поиска кратчайших путей в графах, обладающих большой размерностью.

Для решения задачи APSP существует множество алгоритмов, 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. Известны алгоритмы, в которых входами являются направленные графы [1], полные графы [2], взвешенные графы [3], невзвешенные графы [4], разреженные графы [5].

В данной работе рассматривается задача APSP для взвешенных ненаправленных разреженных графов большой размерности с неотрицательными весами ребер. Предлагается алгоритм, сводящий решение задачи к замене исходного графа графом меньшей размерности и решению задачи на нём. Приводится сравнение предлагаемого алгоритма с быстрым классическим алгоритмом на графах дорожных сетей.

1. Постановка задачи и определения

1.1. Термины и определения

Рассматривается задача поиска кратчайших путей на связном ненаправленном разреженном взвешенном графе $G = (V, E, w)$ с неотрицательными весами рёбер, где $V = \{v_1, v_2, \ldots, v_n\}$ — множество вершин графа; $E = \{e_1, e_2, \ldots, e_m\} \subseteq V \times V$ — множество рёбер графа; $w : E \to [0, \infty)$ — весовая функция на рёбрах. Ребро между вершинами v_i и v_j будем обозначать $e(v_i, v_j)$. Будем рассматривать простые графы — без петель и кратных рёбер.

Граф называется *взвешенным* (относительно рёбер), если каждому его ребру поставлено в соответствие число — вес ребра. 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 рёбер графа, инцидентных v_i. Граф *разреженный*, если справедливо $m \ll n^2$.

Путь в графе — чередующаяся последовательность вершин и рёбер v_0, e_1, v_1, …, v_{k-1}, e_k, v_k, каждое ребро которой инцидентно двум вершинам — непосредственно предшествующей ему и непосредственно следующей за ним. *Длина пути* — сумма весов входящих в него рёбер. *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 ($m(i, i) = 0$, $i = 1, \ldots, n$). *Матрица расстояний* графа — матрица $M_{n\times n}$ с элементами $m_{ij} = m(i, j)$. Граф *связный*, если между любыми двумя его вершинами существует путь, то есть $m_{ij} < \infty$ для всех i, j.

Между двумя вершинами графа может существовать несколько кратчайших путей. Этот аспект в данной работе не является существенным, поэтому при всех упоминаниях кратчайшего пути будем подразумевать любой кратчайший путь. *Матрица предшествования* — матрица $P_{n\times n}$, 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

$$p_{ij} = \begin{cases} v_k, & \exists v_k (p_{ij}^s = \ldots v_k, e(v_k, v_j), v_j), \\ \infty & \text{иначе.} \end{cases}$$

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

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

Обозначим через S и будем называть *сжимающей граф* $G_0 = (V_0, E_0, w_0)$ последовательность графов $S = G_1, G_2, \ldots, G_r$, где $G_p = (V_p, E_p, w_p)$; $V_p = \left\{v_1^p, v_2^p, \ldots, v_{n(p)}^p\right\}$; $E_p = \left\{e_1^p, e_2^p, \ldots, e_{m(p)}^p\right\}$. Каждый следующий граф G_{p+1} последовательности получается из предыдущего G_p удалением k вершин и инцидентных им рёбер, добавлением новых рёбер и пересчётом весов рёбер между вершинами, смежными с удалёнными. Для таких графов справедливо $|V_p| > |V_{p+1}|$, $p = 0, \ldots, r-1$. Будем обозначать v_i^{p+1} вершину графа G_{p+1}, соответствующую вершине v_i^p графа G_p, и $e^{p+1}\left(v_j^{p+1}, v_k^{p+1}\right)$ — 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 рёбер из графа G_p, будем обозначать $G_{p+1} = R_p(v_1^p, v_2^p, \ldots, v_k^p)$. Для такого графа справедливо: $w_{p+1}(i, j) = w_p(i, j)$ для

всех i, j, таких, что $v_i^{p+1}, v_j^{p+1} \in V_{p+1}$. Расстояние между вершинами v_i^p и v_j^p графа G_p будем обозначать $m^p(i,j)$, а его матрицу расстояний — $M_p = \left(m_{ij}^p\right)$. Наконец, обозначим A_i^p множество всех смежных с v_i^p вершин в графе G_p.

1.2. Постановка задачи

Задача. Дан связный ненаправленный разреженный простой взвешенный граф $G = (V, E, w)$ с неотрицательной вещественной весовой функцией на ребрах $w : E \to [0, \infty)$. Необходимо для всех пар вершин графа найти кратчайший путь между вершинами в паре, т. е. найти матрицу расстояний M и матрицу предшествования P.

2. Алгоритм поиска решения

2.1. Общая идея

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

1) сжатие — замена исходного графа графом меньшей размерности;
2) микрорешение — решение задачи о кратчайших путях на графе меньшей размерности с использованием существующих методов;
3) восстановление — перенос решения с графа меньшей размерности на исходный граф; пересчёт кратчайших расстояний для части вершин исходного графа.

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

Алгоритм, использующий похожие идеи, опубликован, например, в [6].

Для разреженных графов 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. Отличие предлагаемого алгоритма от уже известных алгоритмов со сжатием или подменой рёбер состоит в том, что он достаточно прост, выполняет расчёт кратчайших путей сразу между всеми вершинами, при этом гарантированно находит точное решение (не является эвристическим). В частности, в [6] 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.

2.2. Разборка

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 $S = \{G_1, G_2, \ldots, G_r\}$. Будем рассматривать 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 $v_{i_z}^p$, $z = 1, \ldots, k$. Если какой-то из кратчайших путей графа проходит через вершину v_i^p (не считая путей от и до самой v_i^p), то этот путь обязательно содержит подпуть вида $v_{i_j}^p, e^p(v_{i_j}^p, v_i^p), v_i^p, e^p(v_i^p, v_{i_l}^p), v_{i_l}^p$, $j, l \in \{1, 2, \ldots, k\}$. Таким образом, при уда-

лении вершины v_i^p необходимо гарантировать сохранение кратчайших путей только между вершинами, смежными с v_i^p.

Обозначим $w_p^{\mathrm{mv}(1,2,\ldots,h)}(i_j, i_l) = \min\limits_{g=1,2,\ldots,h}(w_p(i_j, g) + w_p(g, i_l))$ минимальную сумму весов двух рёбер, 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

$$w_{p+1}(i_j, i_l) = \begin{cases} \min\left(w_p^{\mathrm{mv}(i)}(i_j, i_l), w_p(i_j, i_l)\right), & \text{если } w_p^{\mathrm{mv}(i)}(i_j, i_l) < w_p^{\mathrm{mv}(h\neq i)}(i_j, i_l), \\ w_p(i_j, i_l) & \text{иначе.} \end{cases} \quad (1)$$

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

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

В частности, если удаляемая вершина v_i^p смежна только с одной вершиной графа G_p, то сохранять кратчайшие пути надобности нет (так как через v_i^p они не проходят), и 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 рёбер при удалении вершины. Вопрос совместного определения $d_{\max}$, $n_{\min}$ и $I_{\max}$ является весьма объёмным, поэтому его рассмотрение остаётся вне рамок данной работы.

Пусть на предмет удаления рассматривается вершина v_i^p с k инцидентными ребрами. Обозначим через $I(v_i^p)$ 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 $I(v_i^p) = -k$. Пересчёт 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 $I(v_i^p)$ согласно формуле

$$I(v_i^p) = \begin{cases} I(v_i^p) + 1, & \text{если } w_p(i_j, i_l) = \infty \ \&\ w_p^{\mathrm{mv}(i)}(i_j, i_l) < w_p^{\mathrm{mv}(h\neq i)}(i_j, i_l), \\ I(v_i^p) & \text{иначе.} \end{cases} \quad (3)$$

Таким образом получим число, на которое изменится количество рёбер в графе G_{p+1} по сравнению с G_p после удаления вершины v_i^p. Если $I(v_i^p) > 0$, число рёбер увеличится, иначе — уменьшится или останется неизменным. Использование формулы (3) предполагает применение ограничения $I_{\max}$ на рост числа рёбер при удалении вершины: удаляются лишь те вершины v_i^p, для которых выполняется неравенство $I(v_i^p) \leqslant I_{\max}$.

В процессе разборки просмотр вершин — претендентов на удаление — происходит следующим образом. Так как при $d_{\max} \geqslant 2$ вершины степени меньше 3 удаляются в любом случае, то рассмотрение вершин происходит в порядке роста их степеней от 1 до $d_{\max}$. Это позволяет уменьшить степени оставшихся вершин и ускорить работу алгоритма за счёт меньшего числа просмотров вершин со степенями, близкими к $d_{\max}$. После удаления вершины v_i^p степени смежных с ней вершин могут измениться. Поэтому для того чтобы обеспечить удаление всех вершин со степенью $d\left(v_j^p\right) \leqslant d_{\max}$, после

удаления вершины v_i^p рекурсивно рассматриваются бывшие смежные с ней вершины, степень которых уменьшилась в результате удаления. После завершения рекурсивного вызова просмотр вершин продолжается в обычном порядке. Алгоритм 1 разборки графа использует вспомогательный алгоритм 2 проверки и удаления вершины.

Алгоритм 1. Разборка графа

Вход: граф $G_0 = (V_0, E_0, w_0)$, $|V_0| = n$; $d_{\max}$, $n_{\min}$, $I_{\max}$; $P'_{n\times n} = \left(p'_{ij}\right)$, $p'_{ij} = \infty$ для всех $i, j = 1, \ldots, n$.

Шаг 0. Подготовка

$d_c = 1$, $i = 0$, $n_c = n$, $p = 0$.

Шаг 1. Выбор вершины

Если $n_c = n_{\min}$, **то**
  конец алгоритма,
иначе
  Если $\exists v_j^p \in V_p$ $(j > i$ & $d\left(v_j^p\right) = d_c)$, **то**
    $i = j$, переход к шагу 2,
  иначе
    Если $d_c < d_{\max}$, **то**
      $d_c = d_c + 1$, $i = 0$ и переход к шагу 1,
    иначе
      конец алгоритма.

Шаг 2. Проверка и удаление вершины

Проверка и удаление вершины $(v_i^p, n_c, I_{\max}, d_{\max}, n_{\min}, p, P')$, переход к шагу 1.

Выход: граф $G_r = G_p$.

2.3. Микрорешение

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

$$p'_{ij} = \begin{cases} p^r_{ij}, & p'_{ij} = \infty \ \& \ p'_{p^r_{ij}j} = \infty, \\ p'_{p^r_{ij}j}, & p'_{ij} = \infty \ \& \ p'_{p^r_{ij}j} \neq \infty; \end{cases} \tag{4}$$

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

где p^r_{ij} — элементы матрицы предшествования P_r графа G_r. Найденные на этом этапе пути между вершинами графа G_r будут гарантированно кратчайшими, так как удаление вершин, произведённое на этапе разборки, сохраняет расстояния. То есть справедливо $m'^r_{ij} = m^r_{ij} = m^0_{ij}$ для всех i, j, таких, что $v^r_i, v^r_j \in V_r$.

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

2.4. Сборка

До начала работы данного этапа (алгоритма 3) 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

Алгоритм 2. Вспомогательный алгоритм проверки и удаления вершины

Вход: вершина v_i^p, текущее количество вершин n_c, $I_{\max}$, $d_{\max}$, $n_{\min}$, p, $P^{'}$.

Шаг 1. Проверка вершины

Если $d\left(v_i^p\right) < 3$, **то**
  переход к шагу 2,
иначе
  $I\left(v_i^p\right) = -d\left(v_i^p\right)$. Рассматриваются все пары вершин множества A_i^p и изменяется значение $I\left(v_i^p\right)$ по формуле (3).
  Если $I\left(v_i^p\right) \leqslant I_{\max}$, **то**
    переход к шагу 2,
  иначе
    конец алгоритма.

Шаг 2. Удаление вершины

Формируется новый граф $G_{p+1} = R_p\left(v_i^p\right)$.
Пересчитываются веса рёбер между вершинами графа G_{p+1}, соответствующими вершинам множества A_i^p, по формуле (1).
Элементы матрицы $P^{'}$ изменяются по формуле (2), $n_c = n_c - 1$, $t = p$, $p = p + 1$.
Если $n_c = n_{min}$, **то**
  конец алгоритма,
иначе
  Пока $n_c > n_{\min}$
    для вершин $v_{i_l}^p$, таких, что $d\left(v_{i_l}^p\right) < d\left(v_{i_l}^t\right)$, выполняем
    Проверка и удаление вершины $(v_{i_l}^p, n_c, I_{\max}, d_{\min}, n_{\min}, p, P^{'})$.

всеми вершинами. На этапе сборки производится обратный ход от G_r к G_0 через графы $G_{r-1}, G_{r-2}, \ldots, G_1$ с расчётом кратчайших путей на каждом шаге p для вершин v_i^{r-p}, таких, что $v_i^{r-p+1} \notin V_{r-p+1}$ и $v_i^{r-p} \in V_{r-p}$.

Вершина v_i^{r-1}, «добавляемая» на первом шаге к графу G_r при переходе к G_{r-1}, соединена ребрами с вершинами $v_{i_z}^{r-1}$, $z = 1, \ldots, k$, 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}, надо вычислить кратчайшие пути от вершины v_i^{r-1} до всех вершин графа G_{r-1}, положив остальные элементы матрицы $M_{r-1}^{'}$ равными соответствующим элементам $M_r^{'}$: $m_{jl}^{'r-1} = m_{jl}^{'r}$ для всех $v_j^r, v_l^r \in V_r$.

Так как кратчайший путь от любой вершины графа G_{r-1} до v_i^{r-1} проходит через одну из вершин $v_{i_z}^{r-1}$, $z = 1, \ldots, k$, кратчайшие расстояния от v_i^{r-1} до любой вершины v_l^{r-1} графа G_{r-1} можно рассчитать по формуле

$$m^{'r-1}(i,l) = \min_{z \in \{1,\ldots,k\}} \left(w_{r-1}(i, i_z) + m^{'r}(i_z, l) \right).$$

Формулы расчёта матрицы расстояний $M_{r-p}^{'}$ при переходе от графа G_{r-p+1} к графу G_{r-p} с добавлением вершины v_i^{r-p} выглядят следующим образом:

$$m_{jl}^{'r-p} = m_{jl}^{'r-p+1}, \quad v_j^{r-p+1}, v_l^{r-p+1} \in V_{r-p+1}; \tag{6}$$

$$m_{il}^{'r-p} = m_{li}^{'r-p} = \min_{z \in \{1,\ldots,k\}} \left(w_{r-p}(i, i_z) + m^{'r-p+1}(i_z, l) \right), \quad v_l^{r-p+1} \in V_{r-p+1}. \tag{7}$$

Обозначим $x(l)$ номер i_z для l, на котором достигается минимум формулы (7). Элементы матрицы $P^{'}$, для вершин которых либо $w_{r-p}(i,l) > m_{il}^{'r-p} \vee w_{r-p}(l,i) > m_{li}^{'r-p}$, либо $p_{il}^{'} = \infty \ \vee \ p_{li}^{'} = \infty$, изменяются по формулам

$$p_{il}^{'} = \begin{cases} v_i, & p_{x(l)l}^{'} = \infty, \\ p_{x(l)l}^{'}, & p_{x(l)l}^{'} \neq \infty; \end{cases} \tag{8}$$

$$p_{li}^{'} = \begin{cases} v_l, & p_{x(l)i}^{'} = \infty, \\ p_{x(l)i}^{'}, & p_{x(l)i}^{'} \neq \infty. \end{cases} \tag{9}$$

Алгоритм 3. Сборка графа

Вход: $S = \{G_0, G_1, \ldots, G_r\}$, $M_r^{'}$, $P^{'}$

1: $p = 1$.
2: **Пока** $p \leqslant r$
3: расчёт матрицы $M_{r-p}^{'}$ по формулам (6), (7) и матрицы $P^{'}$ по формулам (8) и (9); $p = p + 1$.
4: Выход: матрица $M_0^{'}$, матрица $P^{'}$.

2.5. Корректность алгоритма

Утверждение 1. Предложенные алгоритмы «сборки и разборки графа» являются корректным, т. е. найденные в результате сборки матрицы $M_0^{'}$ и $P^{'}$ являются соответственно матрицами расстояний и предшествования исходного графа G_0.

Доказательство. Сначала докажем корректность относительно матрицы $M_0^{'}$. Производимое на этапе разборки графа удаление вершин сохраняет расстояния между оставшимися вершинами, это следует из формулы (1). На втором этапе алгоритма находятся кратчайшие (относительно графа G_0) расстояния: $m_{jl}^{'r} = m_{jl}^0$ для всех $v_j^r, v_l^r \in V_r$. На этапе сборки вычисляются кратчайшие расстояния между всеми вершинами графа; для доказательства этого достаточно показать, что при переходе от графа G_r к G_{r-1} находятся кратчайшие расстояния в графе G_{r-1} (в силу формулы (6) достаточно доказать это для расстояний от «добавляемой» вершины v_i^{r-1} до всех остальных вершин графа G_{r-1}). Это справедливо в силу сохранения расстояний для графа G_{r-1} и того, что кратчайшие пути от любой вершины проходят через смежные с ней вершины, т. е. в силу формулы (7).

Теперь докажем корректность относительно матрицы $P^{'}$. Покажем, что на этапе микрорешения для всех вершин графа G_r находятся корректные элементы матрицы $P^{'}$. В силу сохранения расстояний в графе G_r кратчайший путь $p^s(v_j, v_l)$ от v_j до v_l содержит подпуть $p_{ij}^r, \ldots, v_l$. Согласно формуле (2), если $p_{p_{ij}^r j}^{'} = \infty$, то $p^s\left(p_{ij}^r, v_j\right)$ не содержит вершин, кроме p_{ij}^r и v_j, т. е. $p_{ij} = p_{ij}^r$. Если же $p_{p_{ij}^r j}^{'} \neq \infty$, то, согласно (2), $p^s\left(p_{ij}^r, v_j\right) = p_{ij}^r, \ldots, p_{p_{ij}^r j}^{'}, e(p_{p_{ij}^r j}^{'}, v_j), v_j$, т. е. $p_{ij} = p_{p_{ij}^r j}^{'}$. Расчёт по формулам (4), (5) производится для тех $p_{jl}^{'}$, $v_j^r, v_l^r \in V_r$, которые либо ещё не известны, либо определены неверно. Для $p_{jl}^{'} \neq \infty$ и $w_r(j,l) = m_{jl}^r$ в силу (2) значения уже определены верно на этапе разборки и не требуют пересчёта, т. е. $p_{jl}^{'} = p_{jl}$ для всех $v_j^r, v_l^r \in V_r$.

По индукции, для доказательства корректности матрицы $P^{'}$ достаточно показать, что при переходе от графа G_r к G_{r-1} корректно определяются элементы матрицы $P^{'}$ для «добавляемой» вершины v_i^{r-1}. Это следует из сохранения расстояний для G_{r-1},

корректности P' для $v_j^r, v_l^r \in V_r$ и того, что кратчайшие пути от любой вершины проходят через смежные с ней вершины, т. е. в силу формул (8), (9). ■

3. Результаты эксперимента

Все тесты проводились на компьютере с процессором Intel Core 2 Duo E8400 с частотой 3 ГГц и объемом памяти 2 Гбайт в 32-разрядной версии Windows XP. Программный код написан на языке C++ с использование среды разработки Borland C++ Builder 6. В качестве тестовых данных использовались взвешенные графы дорожных сетей США из открытого доступа ($G1$–$G10$) [7]. Из графов $G1$–$G10$ были получены связные подграфы размерностью от 10^3 до 10^4 в количестве 10 шт. для каждой размерности. Еще одним набором тестовых данных были графы дорожных сетей городов России (GR, подробные характеристики см. в [8]). Характеристики тестовых графов представлены в табл. 1.

Таблица 1

Характеристики тестовых графов

Группа графов	Среднее кол-во вершин	Среднее кол-во рёбер	Средняя степень вершин	Макс. степень вершин	Графов
$G1$	10^3	$2{,}5 \cdot 10^3$	2,48	6	10
$G2$	$2 \cdot 10^3$	$5{,}21 \cdot 10^3$	2,6	5	
$G3$	$3 \cdot 10^3$	$7{,}88 \cdot 10^3$	2,62	6	
$G4$	$4 \cdot 10^3$	$1{,}07 \cdot 10^4$	2,68	6	
$G5$	$5 \cdot 10^3$	$1{,}33 \cdot 10^4$	2,66	6	
$G6$	$6 \cdot 10^3$	$1{,}58 \cdot 10^4$	2,63	7	
$G7$	$7 \cdot 10^3$	$1{,}85 \cdot 10^4$	2,64	6	
$G8$	$8 \cdot 10^3$	$2{,}08 \cdot 10^4$	2,6	6	
$G9$	$9 \cdot 10^3$	$2{,}36 \cdot 10^4$	2,62	7	
$G10$	10^4	$2{,}72 \cdot 10^4$	2,72	7	
GR	$2{,}1 \cdot 10^3$	$6 \cdot 10^3$	2,86	14	20

Тестирование проводилось при $d_{\max} = I_{\max} = \infty$, $n_{\min} = 1$. То есть при разборке удаляются вершины с любой степенью до тех пор, пока в наименьшем графе G_r не останется лишь одна вершина. Соответственно второй этап алгоритма — микрорешение — не выполняется. Разработанный алгоритм (обозначение БП) сравнивался с одним из быстрейших алгоритмов для разреженных графов — алгоритмом Дейкстры в реализации с двоичной кучей (обозначение ДД, [4]), выполняемым для всех вершин тестовых графов. Результаты тестов приведены в табл. 2.

Использование предлагаемого алгоритма позволяет ускорить вычисление кратчайших путей графа в среднем в 47 раз в сравнении с алгоритмом Дейкстры. На всех тестовых графах алгоритм БП оказался быстрее алгоритма Дейстры, наибольшее по всем группам графов время счёта БП оказалось в 34 раза меньше аналогичного времени ДД. Рост максимальной степени вершины в графах последовательности $S = (G_1, G_2, \ldots, G_r)$ во время работы алгоритма явился допустимым и не превысил 17, что говорит о невысоком росте сложности процедуры удаления вершины на этапе разборки для исследованных графов.

Таблица 2

Результаты тестов

Группа графов	БП, среднее время, с	ДД, среднее время, с	БП, макс. время, с	ДД, макс. время, с	Среднее ускор. БП/ДД, раз	БП, макс. степень удал. верш.
$G1$	0,03	1,5	0,05	1,7	50	11
$G2$	0,13	6,7	0,14	7,1	52	12
$G3$	0,31	16	0,33	17	52	12
$G4$	0,67	30	0,88	32	45	22
$G5$	1,1	48	1,2	52	44	16
$G6$	1,5	72	1,8	82	48	20
$G7$	2,1	97	2,2	104	46	17
$G8$	2,6	131	2,8	145	50	21
$G9$	3,7	177	4,7	189	48	24
$G10$	5,4	218	6,4	230	40	23
GR	0,17	7,5	0,4	18,8	45	17

Заключение

Представленный алгоритм позволяет находить кратчайшие пути между всеми вершинами взвешенных ненаправленных графов. Эффективность алгоритма достигается за счёт разборки и сборки исходного графа с поиском промежуточного решения на графе малой размерности. Результаты тестов, проведённых на разреженных графах, позволяют говорить о значительном (в среднем в 47 раз) ускорении счёта в сравнении с одним из быстрых классических алгоритмов, решающих данную задачу.

Объектами дальнейших исследований являются подбор параметров алгоритма на основе быстрого анализа свойств рассматриваемого графа, модификация порядка разборки и сборки графа и вопрос масштабируемости алгоритма с увеличением размерности исследуемых графов. Представляет интерес возможность адаптации использованного в алгоритме подхода для решения задач поиска кратчайших путей в направленных графах и поиска кратчайших путей с заданной погрешностью.

ЛИТЕРАТУРА

1. *Bellman R.* On a routing problem // Quarter. Appl. Math. 1958. No. 16. P. 87–90.
2. *Floyd R. W.* Algorithm 97: Shortest Path // Comm. ACM. 1962. No 5(6). P. 345.
3. *Dijkstra E. W.* A note on two problems in connexion with graphs // Numer. Math. 1959. No. 1. P. 269–271.
4. *Кормен Т. Х. и др.* Алгоритмы: построение и анализ. М.: Вильямс, 2006. 1296 с.
5. *Johnson D. B.* Efficient algorithms for shortest paths in sparse graph // J. ACM. 1977. No. 24. P. 1–13.
6. *Geisberger R., Sanders P., et al.* Contraction hierarchies: faster and simpler hierarchical routing in road networks // International Workshop on Experimental Algorithms (WEA 2008). Provincetown: Springer, 2008. P. 319–333.
7. `http://www.dis.uniroma1.it/challenge9/download.shtml` — 9th DIMACS Implementation Challenge — Shortest Paths (дата обращения: ноябрь 2012).
8. *Ураков А. Р., Тимеряев Т. В.* Использование особенностей взвешенных графов для более быстрого определения их характеристик // Прикладная дискретная математика. 2012. № 2. С. 95–99.