ПОСТРОЕНИЕ И ИССЛЕДОВАНИЕ АЛГОРИТМИЧЕСКИХ МОДЕЛЕЙ УПРАВЛЕНИЯ ТРАНСПОРТНЫМИ ПОТОКАМИ

Г. А. Омарова, К. Ю. Чернов

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия

УДК 519.179.2

Работа посвящена анализу и исследованию различных алгоритмов решения задачи о максимальном потоке на графе, представляющем собой реальную транспортную сеть.

Ключевые слова: граф, источник, сток, пропускная способность, максимальный стационарный поток, транспортные потоки.

This work is devoted to the analysis and research of different algorithms of the solution of the task about the maximum flow on the graph, representing a real transport network.

Key words: graph, source, sink, residual capacity, maximum stationary flow, flow networks.

Введение. Современное общество нуждается в постоянном увеличении объема транспортного сообщения, повышении его надежности, безопасности и качества. Это требует увеличения затрат на улучшение инфраструктуры транспортной сети, превращения ее в гибкую, высокоуправляемую логистическую систему. При этом риск инвестиций значительно возрастает, если не учитывать закономерности развития транспортной сети, распределение загрузки ее участков. Игнорирование этих закономерностей приводит к частому образованию транспортных пробок, перегрузке/недогрузке отдельных линий и узлов сети, повышению уровня аварийности, экологическому ущербу.

Для поиска эффективных стратегий управления транспортными потоками в мегаполисе, оптимальных решений по проектированию улично-дорожной сети и организации дорожного движения необходимо учитывать широкий спектр характеристик транспортного потока, закономерности влияния внешних и внутренних факторов на динамические характеристики смешанного транспортного потока.

Теория транспортных потоков развивалась исследователями различных областей знаний — физиками, математиками, специалистами по исследованию операций, транспортниками, экономистами. Накоплен большой опыт исследования процессов движения. Однако общий уровень исследований и их практического использования недостаточен в силу следующих факторов:

— транспортный поток нестабилен и многообразен, получение объективной информации о нем является наиболее сложным и ресурсоемким элементом системы управления;

— критерии качества управления дорожным движением противоречивы: необходимо обеспечивать бесперебойность движения, одновременно снижая ущерб от движения, накладывая ограничения на скорость и направления движения;

— дорожные условия, при всей стабильности, имеют непредсказуемые параметры как в части отклонения погодно-климатических условий, так и, собственно, дороги;

— исполнение решений по управлению дорожным движением всегда неточно при реализации и, учитывая природу процесса дорожного движения, приводит к непредвиденным эффектам.

Таким образом, трудности формализации процесса движения транспортного потока стали серьезной причиной отставания результатов научных исследований от требований практики.

Несмотря на то, что первые фундаментальные работы по математической теории управления транспортными потоками были выпущены десятилетия назад, по мнению ряда известных специалистов, проблема образования предзаторных и заторных ситуаций еще до конца не изучена. Данная область сейчас активно развивается, и появляется множество новых работ. В качестве примеров академических журналов, посвященных динамике автомобильного транспортного движения, можно привести издания „Transportation Research“ и „Operation Research“ [1].

1. Классические алгоритмы транспортных задач. Задача о максимальном стационарном потоке была поставлена в 1955 г. Т. Е. Харрисом, который вместе с генералом (в отставке) Ф. С. Россом предложил упрощенную модель железнодорожного транспортного потока и выдвинул именно эту специальную задачу как центральную задачу, подсказываемую этой моделью. Вскоре после этого был высказан в качестве гипотезы, а затем и установлен главный результат — теорема о максимальном потоке и минимальном разрезе. С тех пор появилось большое число доказательств этой теоремы.

В том же году Л. Форд и Д. Фалкерсон впервые построили алгоритм, специально предназначенный для решения этой задачи. Их алгоритм получил название „алгоритм Форда — Фалкерсона“ [2].

В дальнейшем решение задачи много раз улучшалось.

В 2010 году исследователи Д. Келнер и А. Мондры из МТИ вместе со своими коллегами Д. Спилманом из Йельского университета и Шень-Хуа Тенем из Южно-Калифорнийского университета продемонстрировали очередное улучшение алгоритма впервые за 10 лет [3].

На сегодняшний день существует множество способов непосредственного нахождения максимального стационарного потока. Далее будут рассмотрены некоторые из них.

1.1. *Алгоритмы Форда — Фалкерсона и Эдмондса — Карпа.*

Описание. Общая идея данных алгоритмов заключается в следующем: изначально величине потока присваивается значение 0: $f(v_1, v_2) = 0$ для всех $v_1, v_2 \in V$. Затем величина потока итеративно увеличивается посредством поиска увеличивающего пути (путь от источника s к стоку t, вдоль которого можно послать больший поток). Процесс повторяется, пока можно найти увеличивающий путь.

На вход алгоритмов подается сеть $N = ((V, E), c, s, t)$. На выходе получается максимальный поток f из s в t.

Определения. Назовем остаточной сетью $G_f(V, E_f, c_f, s, t)$ сеть с остаточной пропускной способностью $c_f(u, v) = c(u, v) - f(u, v)$.

Алгоритм.

$f(u, v) \leftarrow 0$ для всех ребер (u, v)

Пока есть путь P из s в t в G_f такой, что для каждого ребра $(u, v) \in P$ значение остаточной пропускной способности $c_f(u, v) > 0$

Найти $c_{min} = \min\{c_f(u, v) \mid (u, v) \in P\}$

Для каждого ребра $(u, v) \in P$

$f(u, v) \leftarrow f(u, v) + c_{min}$
$f(v, u) \leftarrow f(v, u) - c_{min}$

Различие алгоритмов Форда — Фалкерсона и Эдмондса — Карпа в том, что в первом случае путь P — любой увеличивающий путь, тогда как во втором случае P — кратчайший увеличивающий путь.

Сложность. На каждом шаге алгоритмы добавляют поток увеличивающего пути к уже имеющемуся потоку. Если пропускные способности всех ребер — целые числа, то и потоки через все ребра всегда будут целыми. Следовательно, на каждом шаге алгоритмы увеличивают поток по крайней мере на единицу. Значит, они сойдутся не более чем за $O(k)$ шагов, где k — максимальный поток в графе. Можно выполнить каждый шаг за время $O(E)$, где E — число ребер в графе, тогда общее время работы алгоритмов ограничено $O(kE)$.

Если величина пропускной способности хотя бы одного из ребер — иррациональное число, то алгоритм Форда — Фалкерсона может работать бесконечно, даже не обязательно сходясь к правильному решению [4].

В процессе работы алгоритм Эдмондса — Карпа будет находить каждый дополняющий путь за время $O(E)$. Общее число увеличений потока, выполняемое данным алгоритмом, составляет $O(VE)$. Таким образом, сложность алгоритма Эдмондса-Карпа равна $O(VE^2)$.

1.2. *Алгоритм Диница.*

Описание. Алгоритм Диница был предложен в 1970 году израильским (бывшим русским) ученым Ефимом Диницем [5]. Временная сложность алгоритма составляет $O(V^2E)$. Получить такую оценку позволяет введение понятий *вспомогательной сети* и *блокирующего потока*. В сетях с единичными пропускными способностями существует более сильная оценка временной сложности: $O(EV)$.

На вход алгоритма подается сеть $N = ((V, E), c, s, t)$. На выходе получается максимальный поток f из s в t.

Определения. Определим $dist(v)$ как длину кратчайшего пути из s в v в графе G_f. Тогда вспомогательная сеть G_f — сеть $G_L = (V, E_L, c_f|_{E_L}, s, t)$, где

$$E_L = \{(u, v) \in E_f \,|\, dist(v) = dist(u) + 1\}.$$

Блокирующий поток — $s - t$ поток f такой, что граф $G' = (V, E'_L)$ не содержит $s - t$ пути, где

$$E'_L = \{(u, v) \,|\, f(u, v) < c_f|_{E_L}(u, v)\}.$$

Алгоритм.

$f(u, v) \leftarrow 0$ для всех ребер (u, v)

Создать G_L из G_f сети N. Если $dist(t) = \infty$ остановиться и вывести f

Найти блокирующий поток f' в G_L

Дополнить поток f потоком f' и перейти к шагу 2

Сложность. В процессе работы алгоритма количество ребер в блокирующем потоке увеличивается хотя бы на одно, поэтому в алгоритме не более $n - 1$ блокирующих потоков, где n — количество вершин в сети. Вспомогательная сеть G_L может быть построена

обходом в ширину за время $O(E)$, а блокирующий поток на каждом уровне графа может быть найден за время $O(VE)$. Поэтому время работы алгоритма Диница есть $O(V^2E)$.

1.3. *Алгоритм проталкивания предпотока.*

Описание. Данный алгоритм был впервые опубликован в 1986 году А. Голдбергом и Р. Тарьяном [6]. Он не является частным случаем алгоритма Форда — Фалкерсона. Реализованный без специальных усовершенствований, алгоритм выполняется за время $O(V^2E)$.

На вход алгоритма подается сеть $N = ((V, E), c, s, t)$. На выходе получается максимальный поток f из s в t. Алгоритм основан на применении двух операций — *проталкивание* и *подъем*.

Определения. Будем называть *потомком* вершины u любую вершину v, такую, что остаточная сеть содержит ребро (u, v). *Избыточным потоком* будем называть величину $e_u = \sum\limits_{w \in V} f(w, u)$. *Предпотоком* назовем функцию $f : V \times V \to \mathbb{R}$ со следующими свойствами для любых вершин u и v:

1) $f(u, v) \leqslant c(u, v)$ для любых $u, v \in V$ — *ограниченность пропускной способностью*;

2) $f(u, v) = -f(v, u)$ для любых $u, v \in V$ — *антисимметричность*;

3) для любого $u \in V \setminus \{s, t\}$ выполнено неравенство $\sum\limits_{w \in V} f(w, u) \geqslant 0$ — *неотрицательность избыточного потока*.

Вершина называется *переполненной*, если она не является источником или стоком, а избыточный поток в эту вершину строго положителен. Доказано, что предпоток является потоком тогда и только тогда, когда нет переполненных вершин.

Припишем каждой вершине неотрицательное целое число, называемое *высотой*. Высоту вершины u обозначим h_u.

Инициализация.

Изначально предпоток равен пропускной способности для всех ребер, выходящих из источника, и противоположен для обратных пар вершин:

— $f(s, u) = c(s, u)$;

— $f(u, s) = -c(s, u)$.

Для всех остальных пар вершин предпоток равен нулю.

Начальная высота равна $|V|$ для источника и 0 для всех остальных вершин.

1.4. *Проталкивание.* Проталкивание из вершины u к вершине v возможно при выполнении следующих условий:

— вершина u переполнена;

— остаточная сеть содержит ребро (u, v) (v — потомок u);

— v ниже u: $h_u > h_v$.

Проталкивание заключается в том, что поток $f(u, v)$ увеличивается на величину $\delta f(u, v) = \min(e_u, c(u, v) - f(u, v))$. На столько же увеличивается избыточный поток e_v. Обратный поток $f(v, u)$ и избыточный поток e_u уменьшаются на ту же величину.

Подъем. Подъем вершины u возможен при выполнении следующих условий:

— вершина u переполнена;

— ни один потомок u не ниже u.

Подъем заключается в том, что из всех потомков u выбирается вершина v с минимальной высотой, после чего высота вершины u становится равной $h_v + 1$.

Алгоритм.

Инициализировать предпоток, избыточные потоки и высоты.

Пока возможны проталкивание или подъем, выполнить любую возможную операцию.

Сложность. Определения нужного действия и проталкивания выполняются за константное время. Следовательно, все определения нужного действия и проталкивания требуют $O(V^2E)$ операций.

Перенос вершины u в множество потомков с не меньшей высотой требует ее исключения из множества потомков с меньшей высотой. Поскольку множества потомков хранятся как векторы, а исключение элемента вектора требует количества операций, пропорционального его длине, такой перенос может требовать $O(V)$ операций. Значит, выполнение переносов для всех соседей требует $O(d_uV)$ операций, где d_u — степень вершины u. Остальные действия, выполняемые в ходе подъема, требуют меньшего количества операций, значит, подъем требует $O(d_uV)$ операций. Одна вершина может выдержать $2V-1$ подъемов, следовательно, все ее подъемы требуют $O(d_uV^2)$ операций, а все подъемы всех вершин — $O(\sum\limits_u d_uV^2) = O(V^2E)$ операций.

2. Решение задачи на основе дорожной сети Новосибирской области.

2.1. ***Постановка задачи.*** Пусть дана целочисленная транспортная сеть $N = ((V, E), c, s, t)$, где $V = \{v_1, v_2, \ldots, v_n\}$ — множество вершин, $E = \{e_1, e_2, \ldots, e_n\}$ — множество ребер. Для произвольных $v_1, v_2 \in V$ $c(v_1, v_2) \in \mathbb{N} \cup \{0\}$ — пропускная способность ребра (v_1, v_2). Заданы источник $s \in V$ и сток $t \in V$.

Функция $f : V \times V \to \mathbb{Z}$ называется *потоком*, если она удовлетворяет следующим условиям:

1) $f(v_1, v_2) \leq c(v_1, v_2)$ для любых $v_1, v_2 \in V$ — *ограниченность пропускной способностью*;

2) $f(v_1, v_2) = -f(v_2, v_1)$ для любых $v_1, v_2 \in V$ — *антисимметричность*;

3) для любого $v_1 \in V \setminus \{s, t\}$ выполняется равенство $\sum\limits_{v_2 \in V} f(v_1, v_2) = 0$ — *сохранение потока*.

Задача состоит в том, чтобы найти поток f, такой, что $\sum\limits_{v \in V} f(s, v)$ — максимальна, используя различные алгоритмы с целью сравнения производительности. При этом ориентированный граф $G = (V, E)$ представляет собой реальную дорожную сеть Новосибирской области. Ввод исходных данных и вывод результата необходимо сделать в удобной и понятной для конечного пользователя форме (на карте).

2.2. ***Хранение данных.***

В качестве основного средства для организации хранения данных выбрана реляционная СУБД PostgreSQL. В качестве основных достоинств данной СУБД можно отметить:

— возможность реализации алгоритмов непосредственно в БД на языке PL/PgSQL;

— планирование и оптимизация запросов силами БД;

— наличие функций для работы с геоинформационными объектами (с использованием расширения PostGIS);

— наличие готовых решений для импорта данных из OpenStreetMap.

Однако, помимо самой СУБД, также целесообразно использовать имеющиеся расширения: PostGIS и PGRouting. Первое предоставляет полезные функции для работы с геоинформационными объектами, координатами, геометрическими объектами. Второе — функ-

ции для поиска по графу, которые будут необходимы для реализации выбранных алгоритмов.

Представление графа. Учитывая способ организации хранения данных, граф решено представить в виде таблицы „ways“ (пути). В ней будет содержаться информация о ребрах графа. Таблица состоит из следующих столбцов:

— class_id — класс дороги;
— length — длина дороги;
— x1, y1, x2, y2 — координаты начальной и конечной вершин (EPSG:4326);
— reverse_cost — стоимость проезда по ребру в обратном направлении;
— cost — стоимость проезда по ребру в прямом направлении;
— the_geom — данные о геометрии ребра;
— source — внутренний номер начальной вершины;
— target — внутренний номер конечной вершины.

Также для ускорения поиска по таблице вводятся следующие индексы по столбцам source и target.

Необходимости отдельно хранить список вершин нет, поскольку он может быть получен простым SQL-запросом в БД:

```
SELECT DISTINCT vid FROM (
  SELECT source AS vid FROM ways UNION
  SELECT target AS vid FROM ways
) AS q1;
```

Получение и загрузка данных. Первым этапом является получение данных из проекта OpenStreetMap — некоммерческого веб-картографического проекта по созданию силами сообщества участников-пользователей Интернета подробной свободной и бесплатной географической карты мира. Данные извлекаются в формате XML для дальнейшей загрузки в БД. Для ускорения данного этапа было решено воспользоваться готовыми выгрузками данных по областям РФ, которые доступны на сайте http://gis-lab.info/projects/osm_dump/. Файл в формате .osm.bz2 содержит заархивированные данные в формате XML.

Предполагается, что СУБД PostgreSQL, а вместе с ней и расширения PostGIS и PGRouting установлены и настроены.

Вторым этапом является загрузка полученных данных в БД. Производится она с помощью утилиты osm2pgrouting, предварительно сконфигурированной (можно использовать поставляемый с утилитой конфигурационный файл „mapconfig_for_cars.xml“) для сохранения данных о дорогах (остальные данные, соответственно, отбрасываются). Этот этап — основной, именно на нем по имеющимся данным строится интересующий нас граф. По завершении данного этапа будет получена таблица „ways“, с которой мы будем работать в дальнейшем. Помимо этого, на ее основе автоматически создаются служебные таблицы pgrouting, которые необходимы для корректной работы алгоритмов поиска на графе.

Запуск osm2pgrouting можно произвести следующим образом:

```
# osm2pgrouting -file RU-NVS.osm -conf mapconfig_for_cars.xml \
  -dbname routing -user postgres -clean
```

Таблица, используемая при написании данной работы, содержит 103163 строки ($|E| = 103163$). Число вершин — 83284 ($|V| = 83284$).

Построение транспортной сети на основе графа и реализация алгоритмов. После того, как на втором этапе получения данных был построен граф, необходимо задать пропускные способности для ребер графа. В проекте OSM существует несколько типов дорог. Их список можно найти в документации к OSM: http://wiki.openstreetmap.org/wiki/Key:highway.

Каждое ребро имеющегося графа принадлежит одному из этих типов. В таблице ways на это указывает поле class_id. Определять, какую пропускную способность имеет то или иное ребро, будем по значению этого поля.

После того, как была построена транспортная сеть, можно приступить к реализации алгоритмов решения задачи. Все функции написаны непосредственно в БД на языке PL/pgSQL. Это дает определенные преимущества, но также накладывает некоторые ограничения. Из преимуществ можно отметить более тесную интеграцию исполнителя алгоритма и БД, в которой хранится транспортная сеть. Частично пропадает необходимость сильно оптимизировать алгоритм благодаря планировщику PostgreSQL; можно ускорить поиск по таблицам с помощью построения индексов. Из недостатков стоит отметить усложненное хранение временных данных (например, для построения вспомогательных сетей).

В качестве алгоритмов для решения задачи выбраны алгоритм Эдмондса — Карпа и алгоритм Диница. Их выбор обусловлен в частности особенностями выбранного способа хранения данных — использование таких алгоритмов как алгоритм проталкивания предпотока, или алгоритм „поднять в начало“ требует хранения большого количества временных данных (например, при реализации алгоритма проталкивания предпотока требуется хранить для каждой вершины два множества потомков: с неменьшей высотой и с меньшей высотой), в связи с чем временные затраты на организацию хранения подобной информации будут гораздо выше, чем возможный выигрыш в скорости.

Алгоритм Эдмондса — Карпа.

Алгоритм реализован в функции edmonds_carp. В качестве входных параметров передаются номера вершин источника и стока. Выход алгоритма представляет собой таблицу со следующими столбцами:

— gid (integer) — номер ребра;

— flow (integer) — значение потока f на ребре;

— _max_flow (integer) — значение пропускной способности c на ребре;

— _prio (integer) — „приоритет“ ребра (номер итерации).

На этапе инициализации создается временная таблица „temp_ways“, столбцы которой повторяют столбцы исходной таблицы „ways“, за исключением того, что столбец class_id заменяется на значение пропускной способности функцией classid_to_max_flow, а так же добавляется новый столбец flow, в котором будет храниться значение потока на ребре.

Затем в цикле выполняется поиск кратчайшего увеличивающего пути из источника в сток, после чего обновляются значения потока вдоль найденного пути. Алгоритм заканчивает работу, как только не удается найти увеличивающий путь из источника в сток.

Далее представлена вышеописанная функция.

Алгоритм Диница. Алгоритм реализован в функции dinic. В качестве входных параметров передаются номера вершин источника и стока. Выход алгоритма представляет собой таблицу со следующими столбцами:

Рис. 1. Выбор вершин от 10-го Бронного переулка до 5-го Гэсстроевского переулка

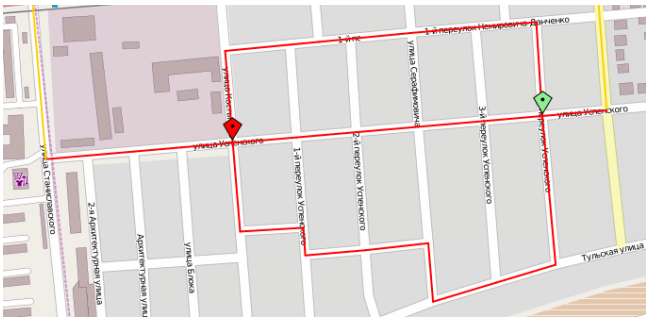

Рис. 2. Обе вершины лежат на одном отрезке ул. Успенского

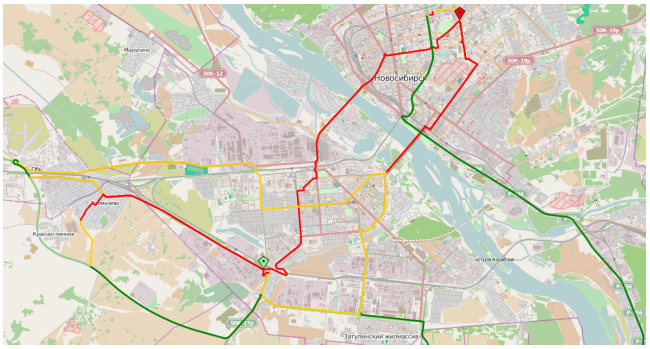

Рис. 3. Выбор вершин от проспекта Дзержинского до Толмачевского шоссе

— gid (integer) — номер ребра;
— flow (integer) — значение потока f на ребре;
— _max_flow (integer) — значение пропускной способности c на ребре;
— _prio (integer) — „приоритет“ ребра (номер итерации).

Инициализация производится аналогично алгоритму Эдмондса — Карпа, однако, помимо таблицы „temp_ways“, создается таблица „temp_paths“, в которой будет храниться блокирующий поток.

Затем в цикле выполняется поиск блокирующего потока и вычисляется его размер, после чего обновляются значения потока в таблице „temp_ways“. Алгоритм заканчивает работу, как только не удается найти блокирующий поток.

Визуализация. Ввод исходных данных и получение результата визуализированы с помощью веб-интерфейса на основе OpenLayers — библиотеки с открытым исходным кодом, написанной на JavaScript, предназначенной для создания карт на основе программного интерфейса (API), подобного GoogleMap API. Она позволяет очень быстро и легко создать веб-интерфейс для отображения картографических материалов, представленных в различных форматах.

Пользователь выбирает две вершины графа на карте — источник и сток. Эти данные передаются в PHP-скрипт (работающий на сервере), который обрабатывает их и вызывает функцию из PostgreSQL, куда передаются обработанные данные. После чего он получает результат работы функции, приводит его к необходимому виду и возвращает на сторону клиента, где эти данные обрабатываются OpenLayers и выводятся пользователю в удобном виде.

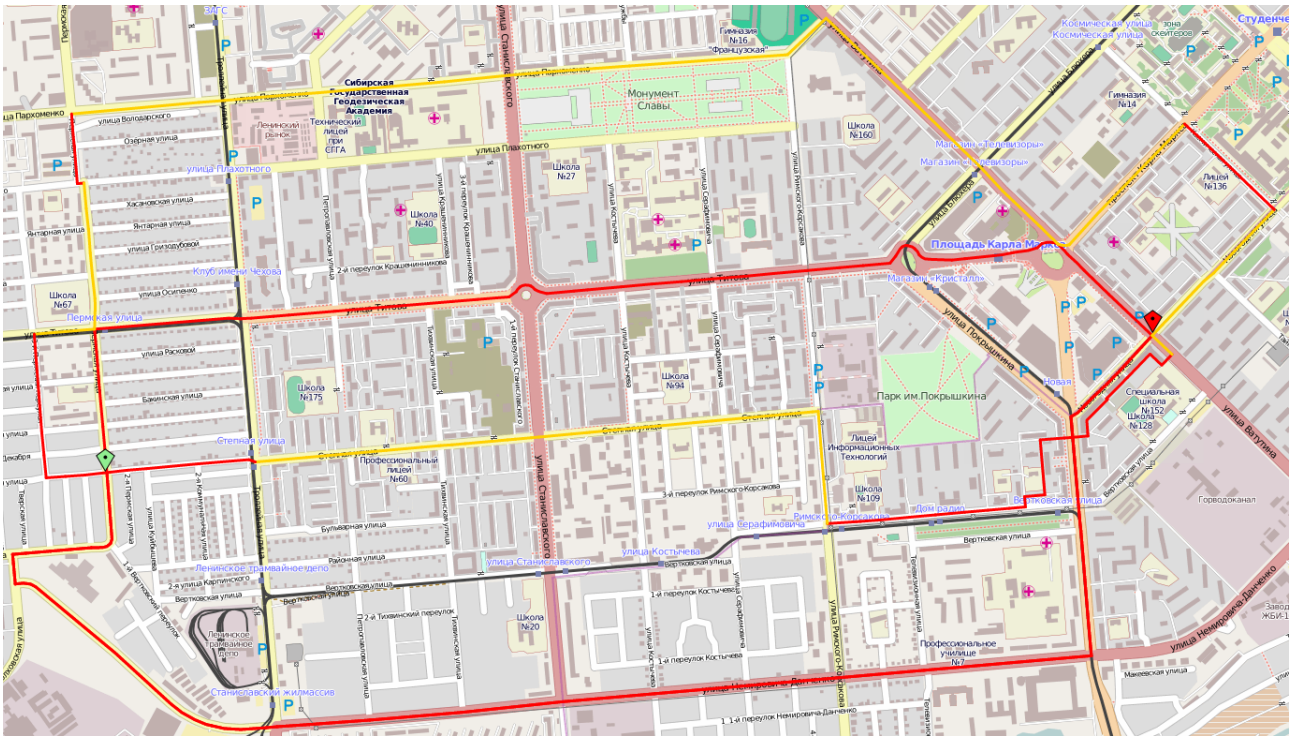

Рис. 4. Выбор вершин от ул. Ватутина до ул. Степная

Таблица

Время работы алгоритмов при различных данных

s	t	N	M	А. Эдмондса-Карпа	А. Диница
3200	6200	1369	350	2,046	2,705
12302	17901	373	100	1,737	1,871
69008	15300	1332	150	1,822	2,299
6341	34195	563	250	1,767	2,042
485	361	366	400	2,128	2,749
714	997	237	300	1,912	2,489
20897	14798	117	200	1,946	2,172
7303	45533	274	700	2,574	3,184
367	13667	233	200	1,955	2,192

Результаты. Используемое оборудование и ПО:

— Intel Core2Duo E8400 3.00GHz / 2Gb DDR2-800 / 250Gb 7200rpm;

— Debian GNU/Linux 3.2.0-4-amd64;

— PostgreSQL 9.1.8 / PostGIS 2.1.0 / PGRouting 2.0.0;

— nginx 1.2.1 / PHP 5.4.4.

Время работы алгоритмов на различных входных данных (см. табл.):

s, t — источник и сток (номера вершин), N — количество задействованных ребер, M — величина максимального потока. Время работы алгоритмов указано в секундах.

Визуализация работы алгоритмов на различных входных данных приведена на рис. 1–4.

Заключение. В ходе работы была создана программная система, состоящая из „клиентской“ и „серверной“ частей, которая решает задачу о максимальном потоке одним из двух алгоритмов по выбору пользователя — алгоритмом Эдмондса — Карпа или алгоритмом Диница.

„Клиентская“ часть представляет собой веб-приложение, отображающее карту OpenStreetMap, на которой происходит ввод исходных данных и вывод результата. Результат отображается в качестве маршрутов от источника к стоку, окрашенных в зависимости от „загруженности“ дорог.

„Серверная“ часть представляет собой БД, в которой хранится граф и реализуются вышеуказанные алгоритмы, а также PHP-скрипт, который используется для обмена данными между „клиентской“ частью и БД.

Возможным продолжением данной работы может являться расширение задачи введением временного фактора — в таком случае поток будет распространяться по дорогам с течением времени, как и происходит в реальной жизни, а не мгновенно.

Список литературы

1. Кравченко П. С., Омарова Г. А. Микроскопические математические модели транспортных потоков. Аналитический обзор // Журнал „Проблемы информатики“. 2014. № 1. С. 71–78.

2. Ford L. R., Fulkerson D. R. Maximal Flow through a Network // Canad. J. Math. 1956. P. 399–404.

3. Christiano P., Kelner J. A., Madry A., Spielman D. A., Shang-Hua Teng. Electrical Flows, Laplacian Systems, and Faster Approximation of Maximum Flow in Undirected Graphs // arXiv:1010.2921 [cs.DS].

4. Zwick U. The smallest networks on which the Ford-Fulkerson maximum flow procedure may fail to terminate // Theoretical Computer Science. 1995. V. 148. P. 165–170.

5. Dinic E. A. Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation // Soviet Math Doklady. 1970. V. 11. P. 1277–1280.

6. Goldberg A. V., Tarjan R. E. A new approach to the maximum flow problem // Journal of the ACM. 1988. V. 35. P. 921–940.

Омарова Гульзира Алимовна — канд. физ.-мат. наук, науч. сотр. Института вычислительной математики и математической геофизики СО РАН;
e-mail: gulzira@rav.sscc.ru

Чернов Константин Юрьевич — студент НГУ;
e-mail: k.j.chernov@gmail.com

Дата поступления — 18.06.2014