УДК 004.023:629.3.072.1-022.233

# УСКОРЕНИЕ ВСТРЕЧНОГО ПОИСКА КРАТЧАЙШИХ ПУТЕЙ НА БОЛЬШИХ ДИНАМИЧЕСКИХ ГРАФАХ МЕТОДОМ БУТСТРЭППИНГА

***М.П. Ревотюк***
*Доцент кафедры ИТАС БГУИР, кандидат технических наук, доцент*

***Н.В. Хаджинова***
*Старший преподаватель кафедры ИТАС БГУИР*

*Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь*
*E-mail: rmp@bsuir.by*

***М.П. Ревотюк***
*Окончил Минский радиотехнический институт. Доцент кафедры информационных технологий автоматизированных систем БГУИР. Направления научных исследований: моделирование и оптимизация управления дискретными процессами в реальном времени, защита информации, системное и объектно-ориентированное программирование и проектирование.*

***Н.В. Хаджинова***
*Окончила Белорусский государственный университет информатики и радиоэлектроники. Заместитель декана по учебно-методической работе факультета инфокоммуникаций, старший преподаватель кафедры информационных технологий автоматизированных систем БГУИР. Направления научных исследований: моделирование и оптимизация управления взаимодействующими процессами, объектно-ориентированное программирование и проектирование.*

**Аннотация.** Предлагаются приемы ускорения встречного поиска кратчайших путей на больших динамических графах, когда порядок порождаемых деревьев путей существенно меньше порядка графа. Однократная инициализация области переменных состояния и расширение деревьев кратчайших путей методом бутстрэппинга снижает сложность поиска путей до линейной зависимости от объема сканируемого пространства.

**Ключевые слова:** транспортные сети, кратчайшие пути, вычислительная сложность.

*Постановка задачи.* Известно, что при поиске кратчайших путей на нагруженном ориентированном графе $G(N, A)$, где $N$ – множество вершин, $A$ – множество дуг с весовой функцией $W: A \to R^+$, время построения дерева путей одним из лучших для подобной задачи алгоритмом Дейкстры растет в первом приближении по закону $x \cdot \log_2 x$ с увеличением расстояния $x$ от корня дерева [1]. Реализация процедур поиска с отображением очередей анализируемых вершин на кучи Фибоначчи характеризуется сложностью $O(m + n \cdot \log_2 n)$, где $m = |A|$, $n = |N|$. Отображение очереди вершин на вектор размером $L_{\max}$ позволяет снизить сложность до величины $O(m + n \cdot L_{\max})$, где $L_{\max}$ – максимальная длина дуги графа [1,2].

В случае поиска пути между двумя заданными вершинами графа целесообразно организовать процесс поиска путем построения двух встречно растущих деревьев. В результате объем анализируемых данных сокращается в два раза [3]. Дерево из конечной вершины должно строиться на графе с обратным направлением дуг, поэтому модель сети задается расширенным графом – объединением исходного графа и его инверсии.

Реализация алгоритма Дейкстры на графах с изменяемой структурой порождает проблему выбора способа представления графа и пространства состояний процесса поиска решения. Нагруженный граф $G(N, A)$, представляющий транспортную сеть, обычно имеет незначительную степень связности вершин, поэтому практически задается списком дуг и полностью определяется тернарным отношением $G(x, y, w)$, где $x$, $y$ – концевые вершины дуги с весом $w$. В таких случаях описание графа задано легко модифицируемым списком дуг с расширяемыми множествами вершин и дуг.

Информация о возмущениях транспортной сети может быть задана списком дуг с изменившимися весами, представленным отношением $R(x, y, w)$, где $x$, $y$ – концевые вершины дуги с весом $w$ (значение $w = \infty$ означает запрос удаления дуги $x \to y$). При этом не обязательно выполнение условия $x, y \in N$, что означает отсутствие запрета на коррекцию любой части графа транспортной сети.

Очевидно, что учет возмущений транспортной сети в терминах операций над отношениями $G(x, y, w)$ и $R(x, y, w)$ – рутинная операция в системах баз данных.

Реляционная модель графа и пространства поиска решения – основа эффективной по памяти реализации схемы Дейкстры для поиска дерева кратчайших путей методом бутстрэппинга [4]. Идея такого метода основана на аналогии складывания ветвей дерева от корня до последнего постоянно включаемого в дерево узла, что ограничивает потребность в памяти параметрами результата поиска, а не модели исходного графа.

Предмет обсуждения – реализация метода бутстрэппинга для встречного поиска дерева кратчайших путей на графе транспортной сети. В отличие от известных постановок подобных задач, здесь будет учитываться технологичность реализации такого метода на реляционных базах данных, что обеспечит открытость модели транспортных операций для расширения с целью учета различных ограничений.

*Модель графа и пространства поиска решения*. Рассмотрим вначале форму задания графа, а затем модель переменных состояния поиска кратчайших путей. Здесь будут использованы понятия отношения и представлений (view) таблиц в терминах реляционных баз данных.

Пространство состояний поиска решения в традиционных реализациях алгоритма Дейкстры соответствует графу всей транспортной сети и включает $D$ – массив расстояний от корня дерева, $P$ – массив номеров предшествующих вершин, $Q$ – очередь вершин, элементы которой упорядочены по текущему значению расстояния от корня дерева [1,2]. При этом $|D| = |N|$, $|P| = |N|$, $|Q| \leq |N|$. Реально возникающие транспортные задачи обычно не нуждаются в наличии описания всего графа сети.

Метод бутстрэппинга всегда работает с подграфом исходного графа. В случае решения задач поиска кратчайших путей на множестве вершин некоторой компактной части графа возможна несложная модификация алгоритма Дейкстры, когда поиск проводится на расширяемом подграфе с последовательно включаемыми вершинами создаваемого дерева кратчайших путей от заданной корневой вершины.

Для реализации алгоритма Дейкстры для каждой вершины $x$ требуется представление множества непосредственно достижимых смежных вершин $x', x' = \{k \mid w(x,k) > 0\}$, где $w(x,k)$ – вес дуги $x \to k$, $x, k \in N$. На основе таблицы отношения $G(x, y, w)$ легко построить его виртуальное представление $G_x(x, y, w)$ в форме FSF (Forward Star Form). Индексации кортежей такой таблицы по ключу $x$ формально определяет множество связей $\{(x, y, w),\ y \in x'\}$, $x \in N$. После этого проверка условия $x \in N$ состоит в оценке успеха поиска кортежа с ключом $x$ в таблице $G_x$.

Обозначим для каждой вершины $y$ множество входных смежных вершин $'y$, $'y = \{k \mid w(k,y) > 0\}$, где $k, y \in N$. Виртуальное представление $G_y(x,y,w)$ в форме BSF (Backward Star Form) инверсии графа образуется после индексации таблицы отношения *Graph* по ключу $y$. Такое представление определяет множество кортежей $\{(x,y,w),\ x \in 'y\}$, $y \in N$. Успех поиска кортежа с ключом $y$ в таблице $G_y$ означает истинность условия $y \in N$.

Таким образом, имеем $N = N_x \cap N_y$, что означает отсутствие необходимости явного отдельного определения множества $N$. Операция расширения графа сети $G(N,A)$ реализуется на логическом уровне модели данных без использования понятия адресной функции. Это влечет отсутствие проблем комплексирования существующих моделей транспортной сети из разных источников и разной степени детализации. Кроме этого, доступно на высоком описание ограничений на условия перемещения на сети в терминах функций как отношений.

С целью определения расширенного графа, пригодного для организации процесса ветвления на общей очереди, обозначим исходный граф через $G^+ = G(N^+, A^+)$, а граф с инвертированием направления дуг – $G^* = G(N^*, A^*)$. Между вершинами таких графов должно быть взаимно однозначное соответствие. Для его задания необходимо использовать симметричную функцию отображения номеров вершин $x^+ \leftrightarrow x^*$, $x^+ \in N^+$ и $x^* \in N^*$. Учитывая, что динамический граф транспортной сети задан списком дуг $G(x,y,w)$, а отображение номеров вершин $x^+ \leftrightarrow x^*$ потребуется лишь для реализации схемы Дейкстры, предлагается в операциях с узлами дерева кратчайших путей номер вершины исходного графа дополнить признаком принадлежности к графам $G^+$ или $G^*$.

Пусть $s$ и $t$ – начальная и конечная вершины искомого кратчайшего пути на исходном графе $G^+$. Так $N^+ \cap N^* = \varnothing$, то встречный поиск можно проводить синхронным движением волны от корней деревьев на несвязном графе $G^+ \cup G^*$. Для этого достаточно начать процесс ветвления из вершин $s^+ \in N^+$ и $t^* \in N^*$. Последнее соответствует формальному объединению графов фиктивной дугой $s^+ \to t^*$, для которой $w(s^+, t^*) = \infty$. Синхронное движение оказывается оптимальным. Таким образом, расширенный граф $G^+ \cup G^*$ оказывается проекцией двух виртуальных графов на список дуг $G(x,y,w)$.

Результат поиска кратчайшего пути от вершины $s$ к вершине $t$ при встречном поиске будет представлен деревьями, где узлы идентифицируются конкатенацией номеров вершин исходного графа и признаками принадлежности к виртуальным графам $G^+$ или $G^*$: $x^s = x + 's',\ x \in N^s$; $x^t = x + 't',\ x \in N^t$. Здесь множества вершин $N^s \subseteq N^+$ и $N^t \subseteq N^*$ встречных деревьев формируются рекурсивным переносом на очередном шаге ветвления буквенными символами признаков принадлежности определенному дереву (направления поиска) от корней из начального состояния: $s^s = s + 's';\ t^t = t + 't'$.

Независимо от направления поиска, дерево кратчайших путей или любое его поддерево – связный граф по определению. Если $s$ – исходная вершина, а $x$ – произвольный узел или лист дерева путей, $s, x \in N$, то текущий или кратчайший путь $s \to x$ можно восстановить обратным движением из листа $x$:

$$p(x,s) = \{x, P(x), P(P(x)), \ldots s\}. \qquad (1)$$

Элементы (1) упорядочены по невозрастанию значений расстояния от корня дерева путей. Альтернативы формирования дерева кратчайших путей отражаются листьями, путь от

корня до которых не обязательно кратчайший, но восстанавливается по правилу построения $p(x,s)$. Далее они будут ключами доступа к переменным состояния поиска.

Пусть $L_k$ – множество листьев текущего дерева на этапе $k$, $L_k^*$ – подмножество листьев без постоянной пометки. Очевидно, что в любой момент построения дерева кратчайших путей его узлы можно отобразить на элементы множества

$$L_k = \Upsilon\{p(s,x), x \in L_k\}. \quad (2)$$

Расширение дерева кратчайших путей происходит только из некоторого листа $x_k$ без пометки, $x_k = \arg\min_x\{d(x), x \in L_k^*\}$. Лист $x_k$ превращается в узел ветвления после операции $L_{k+1}^* = L_k^* \setminus \{x_k\}$ и до окончания поиска становятся пассивным (получает постоянную метку). Ветвление из узла $x_k$ может привести к включению новых или коррекции в сторону уменьшения расстояния от корня $s$ дерева существующих в $L_{k+1}^*$ листьев из множества $\{x_k'\}$. Так как каждому элементу $x$, $x \in L_k$, соответствует $d(x)$ – длина пути до корня $s$, то элементы множества $L_k^* = \{x \mid d(x) \geq d(x_k), x \in L_k\}$ представляют приоритетную очередь [1-3].

Очевидно, элементы $x \in L_k \setminus L_k^*$ остаются упорядоченными по невозрастанию значений $d(x)$. Таким образом, необходимо отображение ассоциаций $x \to p(x)$, $x \to d(x)$ и $d(x) \to x$. Для этого представим переменные состояния процесса построения дерева кортежами отношения $T(p,q,d,o,\mathrm{z})$, где $p$ – номер предшествующего узла дерева не обязательно кратчайшего пути в узел $q$; $d$ – расстояние от корня до узла $q$, $o$ – признак направления поиска $(o \in \{'s','t'\})$; $z$ – признак постоянной пометки узла дерева.

Состояние процесса построения деревьев встречного поиска на любом его этапе $k$ должно быть представлено четверками $(x, p(x), d(x), o(x)), x \in L_k^*$. Начальное состояние процесса построения деревьев соответствует четверкам $(s,s,0,'s')$ и $(t,t,0,'t')$. Условие завершения процесса – $(L_k^* = \varnothing)$ или фиксация постоянной пометки вершины некоторого дерева, когда сопряженная вершина уже является постоянно помеченной [3].

После остановки в вершине *x* остается достроить путь до конечной вершины в исходном графе. Так как остановка может быть обнаружена в любом из встречно растущих деревьев, результат поиска необходимо получить лишь для дерева из исходной вершины.

*Представления модели поиска решения*. Список дуг графа транспортной сети может быть задан вариантами представлений отношения *Graph*, отличающимися законами упорядочения кортежей:

$G_0$ – размещение строк таблицы $G(x,y,w)$ в памяти на физическом уровне;

$G_s$ – справочник выходных дуг вершин графа (ключ $x$);

$G_t$ – справочник входных дуг вершин графа (ключ $y$).

Дерево кратчайших путей, согласно определению (2), в табличной форме задается вариантами представлений отношения *Trace*:

$T_0$ – размещение строк таблицы $T(p,q,d,o,z)$ в памяти на физическом уровне;

$T_v$ – справочник узлов деревьев кратчайших путей (ключ $q+o$);

$T_d$ – элементы приоритетной очереди (ключ $d$).

Виртуальное представление любого отношения может определяться разными способами, конкретизируемыми в среде программирования баз данных. Например, поддержка понятия структурных индексов в системах семейства FoxPro [5] обеспечивает возможность отображения нескольких законов упорядочения одной таблицы отношения:

```
proc test && Пример процедуры встречного поиска на графах
 para filein && Тестовый файл в формате DIMACS
 close data
 set safe off
 create table Graph (a C(1), x C(8), y C(8), w N(8)) && Список дуг графа сети
 append from (m->filein) delimited with blank for a='a'
 index on x tag s && Представление исходного графа
 index on y tag t && Представление инвертированного графа
 create table Trace (p C(8), q C(8), d N(12), o C(1),z C(1)) && Трасса поиска
 index on q+o tag v && Представление узлов дерева кратчайших путей
 index on d tag d && Представление приоритетной очереди
 priv infinity
 infinity=val(replicate('9',fsize("d","Trace")))
 * . . .
 ? sptp(vertx,verty) && Поиск кратчайшего пути от vertx до verty
 *
 close data
retu
```

Здесь приведен фрагмент программы на языке программирования dBase для тестирования рассматриваемых далее процедур встречного поиска на примерах моделей транспортной сети, заданных в формате DIMACS[6].

*Реализация встречного поиска.* Формирование на прямом и инвертированном графах транспортной сети деревьев кратчайших путей с корнями в вершинах $s$ и $t$ проводится по схеме Дейкстры в представлении $T_d$:

```
proc sptp && Реализация встречного поиска
 para s,t
 insert into Trace (p,q,d,o) values (s,s,0,'s') && Исходная вершина в очереди
 insert into Trace (p,q,d,o) values (t,t,0,'t') && Конечная вершина в очереди
 priv f && Длина искомого кратчайшего пути
 m.f=infinity
 do prct && Построение деревьев кратчайших путей
retu m.f
```

Сканирование приоритетной очереди и пометка выбираемых вершин преследует цель построения дерева кратчайших путей, определяемого представлением $T_v$:

```
 proc prct && Построение деревьев кратчайших путей
  priv q,d,o,r
  select Trace
  set order to d && Контекст приоритетной очереди
  scan all && Сканирование очереди вершин
   repl z with '=' && Вершина постоянно фиксирована в дереве |
   scatter memvar fields q,d,o
   m.r=recno()
   set order to v t && Контекст списка узлов дерева
   if !iif(m.o='s',exps(),expt())&& Контроль первой встречи деревьев
    do tail && Завершение поиска
    exit && Кратчайший путь найден
   endi
   select Trace
   set order to d && Контекст приоритетной очереди
   go record m.r
  ends
 retu
```

Контроль первой встречи деревьев выполняется функциями, различаемыми принадлежностью выбираемой вершины прямому или инвертированному графу.

Ветвление в узле исходного графа обеспечивает перебор выходных дуг узла:

```
proc exps && Ветвление в узле исходного графа
 if seek(m.q+'t').and.(z='=') && Сопряженная вершина постоянно помечена
 retu .f. && Кратчайший путь найден
 endi
 select Graph
 set order to s && Контекст исходного графа
 seek m.q && Позиция узла ветвления
 scan rest while x=m.q && Сканирование выходных дуг
  do plan with Graph.y && Попытка расширения пути
 ends
retu
```

Ветвление в узле инвертированного графа обеспечивает перебор входных дуг узла:

```
proc expt && Ветвление в узле инвертированного графа
 if seek(m.q+'s').and.(z='=') && Сопряженная вершина постоянно помечена
  retu .f. && Кратчайший путь найден
 endi
 select Graph
 set order to t && Контекст инвертированного графа
 seek m.q && Позиция узла ветвления
 scan rest while y=m.q && Сканирование входных дуг
  do plan with Graph.x && Попытка расширения пути
 ends
retu
```

Использование раздельных реализаций процедур ветвления позволяет сократить количество переключений представлений графа до одного. Для каждой дуги исходного или инвертированного графа выполняются попытки расширения пути:

```
proc plan && Попытка расширения пути
 para v && Вершина продолжения пути
 priv e
 m.e=m.d+w && Оценка расстояния до корня графа
 select Trace
 if seek(m.v+m.o) && Вершина уже в очереди
  if d>m.e
   repl p with m.q, d with m.e && Обновление пути в вершину
  endi
 else && Включение вершины в очередь
  insert into Trace (p,q,d,o) values (m.q,m.v,m.e,m.o)
 endi
  select Graph
retu
```

Отказ от ветвления в некотором узле исходного или инвертированного графа приводит к переходу к завершению поиска. Необходимость перехода выявляется приведенными выше функциями ветвления проверкой факта постоянной пометки сопряженной вершины. Однако сумма расстояний от корней встречных деревьев в этот момент гарантированно является лишь верхней границей длины кратчайшего пути. Отклонение от оптимального значения может достигать значения $L_{\max}/2$.

Уточнение длины кратчайшего пути проводится посредством сканирования остатка приоритетной очереди, где остаются узлы деревьев кратчайших путей без постоянной пометки. В результате определяется узел встречи деревьев поиска, от которого сумма расстояний от корней встречных деревьев минимальна:

```
proc tail && Завершение поиска
 priv u,o
 select Trace
 m.f=m.d+d && Верхняя граница длины кратчайшего пути
 m.u=q && Узел сопряжения деревьев поиска
 set order to d
 goto record m.r
 scan rest && Сканирование остатка приоритетной очереди
  m.r=recno()
  scatter memvar fields d,q,o
  set order to v
  if seek(m.q+iif(m.o='s','t','s')).and.(m.f>m.d+d)
   m.f=m.d+d && Уточнение длины кратчайшего пути
   m.u=q && Узел сопряжения деревьев поиска
  endi
  set order to d
  goto record m.r
 ends
 do back && Инвертирование вершин пути к конечной вершине
retu
```

Можно заметить, что до перехода к завершению поиска бесполезная оценка длины кратчайшего пути не проводится.

Использование процедуры инвертирования вершин пути от вершины встречи деревьев к конечной вершине позволяет представить результат поиска в стандартной форме представления пути алгоритмом Дейкстры:

```
proc back && Инвертирование вершин пути к конечной вершине
 priv p,q,d,e,w
 set order to v
 seek m.u+'s' && Узел сопряжения дерева от исходной вершины
 m.e=d
 seek m.u+'t' && Узел сопряжения дерева от конечной вершины
 scatter memvar fields p,q,d
 do while (m.q!=t).and.seek(m.p+'t') && Сканирование пути к конечной вершине
  m.w=m.d-d
  scatter memvar fields p,d
  repl d with m.e+m.w, p with m.q
  m.e=d
  m.q=q
 endd
retu
```

Рассмотренный вариант ветвлений в узлах исходного и инвертированного графа допускает получение списка и параметров смежных дуг узла по мере потребности. Таким образом, для решения локальных транспортных задач нет необходимости наличия громоздкого описания полной модели динамически меняющейся транспортной сети.

*Реоптимизация результатов встречного поиска.* Изменения дуг графа транспортной сети, определяемые отношением $R(x,y,w)$, должны быть учтены в новой версии графа $G(x,y,w)$. Очевидно, что кратчайшие пути от корней $s$ и $t$ вне изменений сохраняются.

Отсюда следует целесообразность определения на основе отношения *Trace* значений

$$x_{\min} = \arg\min_x\{d(x) \mid (x \in N_s)\wedge(x,?,?) \in R(x,?,?)\},\ d_{\min} = d(x_{\min});$$
$$x_{\max} = \arg\max_x\{d(x) \mid (x \in N_s)\wedge(x,?,?) \in R(x,?,?)\},\ d_{\max} = d(x_{\max}).$$

Далее необходимо уточнить деревья встречного поиска для узлов, которые удалены от корня $s$ на расстояниях из интервала $(d_{\min}, d_{\max})$. Не составляет труда восстановить состояние

поиска по известным значениям $x_{\min}$ и $x_{\max}$, а затем продолжить сканирование приоритетной очереди до остановки встречного поиска.

Такая процедура характеризуется надежностью учета любых условий ветвления и ограничений на структуру кратчайших путей, а ее вычислительная сложность в среднем в два раза лучше повторного выполнения встречного поиска. Однако она может быть заменена одной из известных процедур реоптимизации текущего дерева кратчайших путей (в случае выполнения необходимых условий их корректного применения) [7].

*Заключение.* Таким образом, процесс построения кратчайших путей на графах давно изученным алгоритмом Дейкстры представлен в форме, согласованной с возможностями современных технологий быстрого проектирования приложений. Предложенная структуризация процедуры встречного поиска непосредственно проецируется на понятия объектно-ориентированного моделирования и программирования, идиомы систем управления баз данных для создания открытых для расширения и детализации систем. Динамическое выделение области поиска решения ограничивает потребность в памяти и вычислительную сложность поиска деревьев кратчайших путей на больших графах размерностью локальных транспортных задач.

***Список литературы***

[1]. Deo, N. Shortest-Path Algorithm: Taxonomy and Annotation/Deo N., Chi-yin Pang//Networks, 1984. – Vol. 14. – P. 275–323.

[2]. Fredman, M.L. Fibonacci heaps and their uses in improved network optimization algorithms/ M.L. Fredman, R.E. Tarjan// Journal of the Association for Computing Machinery, vol. 34, 1987. – P. 596–615.

[3]. Nicholson, T.A. J. Finding the Shortest Route between Two Points in a Network / T.A.J. Nicholson // The Computer Journal, 9(3), 1966. – P. 275–280.

[4]. Ревотюк, М. П. Многократный поиск кратчайших путей на больших графах методом бутстрэппинга / М. П. Ревотюк, Н. В. Хаджинова // BIG DATA and Advanced Analytics = BIG DATA и анализ высокого уровня : сборник материалов V Международной научно-практической конференции, Минск, 13–14 марта 2019 г. В 2 ч. Ч. 1 / Белорусский государственный университет информатики и радиоэлектроники; редкол. : В. А. Богуш [и др.]. – Минск, 2019. – С. 301 – 310.

[5]. Клепинин, В. Visual FoxPro в подлиннике/ В. Клепинин, Т. Агафонова //СПб.: БХВ-Петербург, 2007. – 1216 с.

[6]. 9th DIMACS Implementation Challenge – Shortest Paths [Electronic resource]. – Access mode: http://users.diag.uniroma1.it/challenge9/download.shtml – Date of access: 11.01.2020.

[7]. Shortest paths on dynamic graphs: a survey/ Ferone D. [et al]//Pesquisa Operacional. – 2017. Vol. 37, iss. 3. – P. 487-508.

# POINT-TO-POINT SEARCHING OF THE SHORTEST PATHS ON THE BIG DYNAMIC GRAPHS BY THE BOOTSTRAPPING METHOD

***M.P. REVOTJUK, PhD***
*Department of Information Technologies of Automated Systems of BSUIR, Associate Professor*

***N.V. KHAJYNOVA***
*Department of Information Technologies of Automated Systems of BSUIR, Senior Lecturer*

*Belarusian State University of Informatics and Radioelectronics, Republic of Belarus*
*E-mail: rmp@bsuir.by*

**Abstract.** On the classical problem of the point-to-point searching off the shortest paths in massive dynamic graphs considered the possibility of accelerating the search procedure by incorporating a priori information about the search space. Global initialization of state variables predefined search and selection solutions can improve performance of multiple procedures to find paths to a linear dependence on the volume of the scanned area.

**Keywords:** transport networks, shortest paths in graphs, computational complexity.