УДК 519.179
DOI: 10.15827/0236-235X.122.321-329
Дата подачи статьи: 10.10.17
2018. Т. 31. № 2. С. 321–329

АДАПТИВНЫЙ АЛГОРИТМ ПОИСКА ОПТИМАЛЬНОГО МАРШРУТА В НЕСТАЦИОНАРНОЙ СЕТИ

А.А. Солдатенко [1], *аспирант, glinckon@gmail.com*

[1] *Сибирский федеральный университет,*
Свободный просп., 79, г. Красноярск, 660041, Россия

Рассматривается задача Time-Dependent Shortest-Path (TDSP), которая является расширением задачи о кратчайшем пути в графе.

Задача TDSP возникает при проектировании и эксплуатации телекоммуникационных и транспортных сетей, когда требуется учитывать временной фактор и возможность возникновения в отдельные промежутки времени снижения объема трафика и наличия пробок в сети. В этих случаях сеть представляется ориентированным графом $G = (V, E)$, в котором для каждой дуги $(x, y) \in E$ определены две функции: время, необходимое для передвижения по этой дуге, и время прибытия в вершину y при условии, что старт из вершины x осуществлен в момент времени t. Такую сеть называют нестационарной, а наименьшее время передвижения из стартовой вершины в целевую интерпретируют как оптимальный маршрут между этими вершинами.

Известно, что задача TDSP для нестационарной сети общего вида является NP-трудной. В данной статье задача TDSP рассматривается для полиномиально разрешимого случая, когда функции прибытия монотонны. Предлагается решать TDSP с помощью двухфазного алгоритма ALT (A* with Landmarks & Triangle) – одного из современных алгоритмов оптимальной маршрутизации, изначально разработанного для решения задачи о кратчайшем пути в графе. Данный алгоритм на первой фазе расставляет некоторое множество ориентиров в вершинах сети и вычисляет потенциальные функции, а на второй с помощью алгоритма A* и потенциальных функций находит оптимальный маршрут.

Предлагается модификация алгоритма ALT, эффективно решающая задачу TDSP для последовательности запросов на поиск оптимальных маршрутов в нестационарной сети. Модификация заключается в применении адаптивной эвристики для расстановки ориентиров, а также специальных формул вычисления потенциальных функций. Адаптивная эвристика использует историю обработки предыдущих запросов и корректирует текущий набор ориентиров для эффективного исполнения последующих запросов. Приводятся описание и оценка времени работы модифицированного алгоритма ALT.

Представлены результаты вычислительных экспериментов, выполненные с помощью разработанных программных средств и подтверждающие высокое быстродействие модифицированного алгоритма ALT по сравнению с его классическими аналогами.

Ключевые слова: *нестационарные сети, графы большой размерности, оптимальная маршрутизация, алгоритм ALT, расстановка ориентиров.*

Поиск *кратчайшего пути в графе* (shortest-paths, SP) – хорошо известная задача комбинаторной оптимизации, имеющая множество реальных приложений. Задача SP состоит в нахождении кратчайшего пути между вершинами s и d (стартовой и целевой соответственно) в заданном графе $G = (V, E)$ [1, 2]. При проектировании и эксплуатации современных телекоммуникационных и транспортных сетей часто приходится иметь дело с расширением задачи SP, когда требуется учитывать временной фактор и возможность возникновения в отдельные промежутки времени таких предсказуемых ситуаций, как снижение объема трафика и наличие пробок в сети. В таких ситуациях сеть представляется ориентированным графом $G = (V, E)$, в котором каждой дуге $(x, y) \in E$ соответствуют две функции: $w_{xy}(t)$ – время, необходимое для передвижения по дуге (x, y), $F_{xy}(t)$ – время прибытия в вершину y при условии, что старт из вершины x осуществлен в момент времени t. Такую сеть принято называть нестационарной, а наименьшее время передвижения из стартовой вершины в целевую – оптимальным маршрутом или кратчайшим путем между этими вершинами, полагая, что время передвижения по этому пути всегда зависит от момента выхода из стартовой вершины. В литературе задача поиска кратчайшего пути в нестационарной сети носит название Time-Dependent Shortest-Path problem, или коротко TDSP [3, 4]. Известно, что TDSP для нестационарной сети общего вида без каких-либо ограничений на топологию сети и функции прибытия является NP-трудной задачей [3]. Когда функции прибытия монотонны, задача TDSP полиномиально разрешима [4, 5]. В данной работе рассматривается именно этот полиномиально разрешимый случай.

К настоящему времени уже предложено много алгоритмов решения задачи TDSP. Самый известный из них – алгоритм Дейкстры, который при условии неотрицательности значений весов дуг находит точное решение задач SP и TDSP в графе $G = (V, E)$ за время $O(|V|^2)$ [2, 5]. Современные телекоммуникационные и транспортные сети могут быть огромными – содержать миллионы узлов, и требуется найти оптимальный маршрут из стартовой вершины в целевую за доли секунды. Экспериментально установлено, что в подобных условиях алгоритм Дейкстры работает неприемлемо долго.

Чтобы справиться с этой проблемой, были предложены различные методы ускорения алгоритма Дейкстры, позволяющие вычислять оптимальный маршрут за несколько микросекунд даже в огромных сетях [6–10].

Большинство этих методов основано на двухфазном подходе нахождения оптимального маршрута в сети. Первая фаза – предобработка графа, моделирующего исходную сеть. На второй фазе с помощью алгоритма Дейкстры (или какой-либо его версии) осуществляется поиск оптимального маршрута в графе, полученном после предобработки. Предобработка сводится к просмотру исходного графа и анализу его структуры с целью извлечения информации, позволяющей ускорить вторую фазу алгоритма. Двухфазные алгоритмы принято разделять на следующие классы:

– иерархические алгоритмы, основанные на многоуровневом представлении исходного графа с выполнением алгоритма Дейкстры для каждого выделенного уровня [8];

– алгоритмы маркировки, в которых исходный граф разбивается на несколько примерно равных частей и маршрутизация осуществляется по меткам дуг, находящимся на стыке этих частей [9];

– алгоритмы маршрутизации по ориентирам, в которых некоторое множество вершин исходного графа рассматривается в роли ориентиров и используется для целенаправленного поиска оптимального маршрута [7, 10].

Большинство двухфазных алгоритмов первоначально разрабатывались для задачи SP, поэтому лишь немногие из них пригодны для решения задачи TDSP и учета динамических сценариев, возникающих в реальных сетях и изменяющих их структуру.

По мнению многих авторов, алгоритм ALT (A* with Landmarks & Triangle), являющийся современным представителем класса алгоритмов маршрутизации по ориентирам, хорошо подходит для TDSP и оптимальной маршрутизации в динамических графах [10, 11]. В пользу этого алгоритма приводятся следующие аргументы. Алгоритм ALT на этапе предобработки расставляет определенное число ориентиров в вершинах графа, не изменяя состава вершин и дуг графа, весов дуг и функций прибытия. Это позволяет при изменении графа не повторять полностью фазу предобработки, а лишь вносить локальные изменения в данные, используемые для ускорения второй фазы алгоритма. На второй фазе ALT для поиска оптимального маршрута применяется алгоритм A* – версия алгоритма Дейкстры, работающая с потенциальными функциями, вычисленными на основе ориентиров [12]. Использование в A* потенциальных функций позволяет на практике находить оптимальный маршрут значительно быстрее алгоритма Дейкстры. Например, для европейской дорожной сети, состоящей из более 1,5 миллиона вершин, алгоритм ALT в среднем работает в 20 раз быстрее алгоритма Дейкстры [10, 13]. Алгоритм ALT был предложен в 2005 году Гольдбергом и Харрельсоном [7]. Сочетание алгоритма ALT с другими методами ускорения, а также совершенствование процедуры расстановки ориентиров – современные направления развития данного алгоритма [6, 10, 11]. Однако алгоритм ALT можно применить не ко всем графам, поскольку для корректной работы алгоритма A* необходимо, чтобы потенциальные функции, вычисленные на основе заданных весов дуг исходного графа, обладали свойствами допустимости и преемственности [12]. В данном случае под корректностью алгоритма понимается его способность находить точное решение поставленной оптимизационной задачи [2].

В статье предлагается модификация двухфазного алгоритма ALT для нестационарных сетей, которая находит точное решение задачи TDSP. Модификация заключается в следующем: вычисление потенциальных функций по формулам, гарантирующим свойства, необходимые для корректной работы алгоритма ALT применительно к задаче TDSP; многократное решение задачи TDSP для последовательности запросов на поиск кратчайшего пути; использование адаптивной стратегии расстановки ориентиров. Приводятся описание модифицированного алгоритма ALT, оценка времени его работы, программные средства реализации, а также результаты сравнительного анализа данного алгоритма с известными – алгоритмом Дейкстры и классическим алгоритмом ALT.

Постановка задачи TDSP

Понятия и обозначения теории графов общеприняты и взяты из [1]. Введем дополнительные обозначения, необходимые для формулировки задачи TDSP.

Пусть задан ориентированный граф $G = (V, E)$ без кратных дуг и петель (далее просто граф), в котором для всякой дуги $(x, y) \in E$ определена вещественная весовая функция

$$w_{xy}(t) = \frac{l_{xy}}{v_{xy}(t)} \geq 0. \qquad (1)$$

В формуле (1) величину l_{xy} интерпретируем как длину дуги (x, y), $v_{xy}(t)$ – как скорость движения по дуге (x, y), а $w_{xy}(t)$ – время передвижения из вершины x в вершину y при условии, что старт из x осуществлен в момент времени t. Считаем, что $v_{xy}(t) > 0$, $l_{xy} \geq 0$, а расстояния между вершинами графа $G = (V, E)$ являются постоянными величинами и подчиняются неравенству треугольника. Кроме того, полагаем, что величина $t \geq 0$ измеряется в условных единицах времени и принимает значения из некоторого конечного множества T. Таким образом, $w_{xy}(t)$ и $v_{xy}(t)$ – дискретные функции с конечным множеством значений. В данном слу-

чае под расстоянием между вершинами графа понимается кратчайший путь между этими вершинами в обычном для теории графов толковании [1].

Сопоставим дуге $(x, y) \in E$ функцию прибытия $F_{xy}(t) = t + w_{xy}(t)$, где t – время отправления из вершины x; $F_{xy}(t)$ – время прибытия в вершину y при движении по дуге (x, y). Поскольку $t \geq 0$, $w_{xy}(t) \geq 0$, всегда $F_{xy}(t) \geq t \geq 0$. (2)

Неравенство (2) отражает непосредственный ход времени: «отправляясь из вершины x в момент времени t, невозможно прибыть в вершину y раньше времени t». Если для любых моментов времени $0 \leq t_1 \leq t_2$ верно $F_{xy}(t_1) \leq F_{xy}(t_2)$, то говорят, что функция прибытия дуги (x, y) монотонная. Граф $G = (V, E)$, отвечающий всем указанным выше предположениям, включая монотонность функций прибытия всех его дуг, принято называть нестационарной сетью, удовлетворяющей условию First-In First-Out, или коротко FIFO [3, 4].

Последовательность вершин $P = \{x_0, x_1, \ldots, x_k\}$ графа $G = (V, E)$, в которой $s = x_0$, $d = x_k$, $(x_i, x_{i+1}) \in E$, $i = 0, 1, \ldots, k-1$, задает некоторый путь из вершины s в вершину d. Если при этом начало движения осуществлено в момент времени t_s, то данный путь будем обозначать (s, d, t_s) или (P, t_s). Существование в графе (s, d, t_s)-пути указывает, что вершина d достижима из вершины s. В этом случае будем записывать $s \underset{P}{\sim\sim\sim>} d$.

По определению, вес пути

$$w(P, t_s) = t_s + \sum_{i=0}^{k-1} w_{x_i x_{i+1}}(t_i), \quad (3)$$

где $t_0 = t_s, t_{i+1} = F_{x_i x_{i+1}}(t_i)$,

а вес кратчайшего пути

$$dist(s, d, t_s) = \min_P \{w(P, t_s) : s \underset{P}{\sim\sim\sim>} d\}. \quad (4)$$

Заметим, что значения величин $w(P, t_s)$ и $dist(s, d, t_s)$ всегда могут быть вычислены по формулам (3) и (4), если вершина d достижима из вершины s. Величину $w(P, t_s)$ будем трактовать как время прибытия в вершину d при прохождении (s, d, t_s)-пути, а величину $dist(s, d, t_s)$ – как самое раннее время прибытия в d при условии, что отправление из вершины s осуществлено в момент времени t_s. Тройку величин (s, d, t_s), где s и d – стартовая и целевая вершины соответственно, а t_s – стартовое время, будем называть запросом на поиск в нестационарной сети $G = (V, E)$ кратчайшего (s, d, t_s)-пути, или кратко (s, d, t_s)-запросом.

С использованием введенных понятий и обозначений задача TDSP формулируется следующим образом.

Time-Dependent Shortest-Path problem

Заданы нестационарная сеть $G = (V, E)$ с условием FIFO и (s, d, t_s)-запрос.

Требуется найти значение $dist(s, d, t_s)$ и последовательность вершин, образующих кратчайший (s, d, t_s)-путь.

Очевидно, что в указанной постановке задача TDSP всегда имеет решение, если вершина d достижима из вершины s. Точное решение данной задачи может быть найдено за время $O(|V|^2)$ с помощью алгоритма Дейкстры. Корректность алгоритма Дейкстры гарантируется условием FIFO для сети $G = (V, E)$ и неотрицательностью значений весов (1) всех ее дуг [5]. При применении других алгоритмов для нахождения точного решения задачи TDSP могут возникать дополнительные требования к $G = (V, E)$. Например, для алгоритма ALT необходимо, чтобы потенциальные функции удовлетворяли свойствам допустимости и преемственности, обеспечивающим корректность алгоритма A* [12].

Вычисление потенциальных функций для задачи TDSP

В работе [14] было доказано, что неравенство треугольника, основанное на оптимистичных весах дуг, гарантирует справедливость свойств допустимости и преемственности потенциальных функций для нестационарной сети. В модифицированном алгоритме ALT оптимистичные веса дуг используются также в формулах расчета потенциальных функций. Приведем эти формулы и дадим краткие пояснения к ним.

Максимальная скорость, с которой можно двигаться по всякой дуге исходного графа, равна величине $v_{\max} = \max\limits_{(x,y) \in E} \{\max\limits_{t \in T}[v_{xy}(t)]\} > 0$, постоянной для $G = (V, E)$ на рассматриваемом промежутке времени T. Оптимистичный вес дуги $(x, y) \in E$ определяется через $v_{\max}$ следующим образом:

$$w'_{xy} = \frac{l_{xy}}{v_{\max}}. \quad (5)$$

Если $dist'(x, y)$ – кратчайшее расстояние между вершинами x и y, вычисленное по формуле (4) с использованием (5), то неравенство треугольника для тройки вершин $x, y, z \in V$ имеет вид

$$dist'(x, y) + dist'(y, z) \geq dist'(x, z). \quad (6)$$

Пусть $L \subseteq V$ – множество вершин графа $G = (V, E)$, в которых установлены ориентиры. Для каждой вершины $x \in V$ и всякого ориентира $l \in L$ могут быть найдены следующие числовые значения:

$$\pi_{l+}(x) = dist'(l, d) - dist'(l, x), \quad (7)$$

$$\pi_{l-}(x) = dist'(x, l) - dist'(d, l), \quad (8)$$

$$\pi_l(x) = \max\{0, \pi_{l+}(x), \pi_{l-}(x)\}. \quad (9)$$

На их основе определяется функция

$$\pi_L(x) = \max_{l \in L} \pi_l(x), \quad (10)$$

называемая потенциальной функцией вершины $x \in V$ относительно множества ориентиров L. Формулы (7)–(10) задают процесс вычисления потенциальных функций через оптимистичные веса дуг для нестационарной сети.

В данной работе рассматриваются нестационарные сети, для которых неравенство (6) справедливо

для любой тройки вершин $x, y, z \in V$. Это достаточное условие корректности применения алгоритма ALT к задаче TDSP [14]. При его справедливости всегда

$$0 \le \pi_L(x) \le dist(x, d, t_x). \quad (11)$$

В работе [14] было показано, что условие (6) всегда справедливо, если верно предположение о выполнимости неравенства треугольника для расстояний между вершинами графа $G = (V, E)$. Заметим, что такое предположение является естественным для графов, моделирующих широкий класс телекоммуникационных и транспортных сетей. Следовательно, для таких графов не требуется специальная проверка условия (6). Этот факт чрезвычайно важен для сетей большой размерности, поскольку подобная проверка может быть трудоемкой и сопоставимой со временем работы алгоритма Дейкстры.

Адаптивное размещение ориентиров в нестационарной сети

В алгоритме Дейкстры направление поиска кратчайшего пути определяется на основе верхних оценок этого пути, вычисляемых для некоторых вершин графа. Алгоритм A* дополнительно использует нижние оценки, задаваемые потенциальными функциями. Очевидно, что, чем лучше потенциальные функции оценивают снизу кратчайший путь, тем алгоритм A* работает более целенаправленно, а значит, быстрее алгоритма Дейкстры. Это справедливо и для случая, когда потенциальные функции определяются через множество ориентиров. Легко убедиться, что различное положение какого-либо отдельного ориентира относительно стартовой и целевой вершин может приводить к различным значениям потенциальных функций. Таким образом, расстановка ориентиров в вершинах графа – фактор, существенно влияющий на быстродействие алгоритма A* [7, 11].

Задача оптимального выбора ориентиров в графе $G = (V, E)$ заключается в определении числа ориентиров и мест их расстановки в вершинах этого графа. Данная задача носит комбинаторный характер и является труднорешаемой. Доказано, что даже в частном случае, когда число ориентиров фиксированное, эта задача NP-трудная [7]. Чтобы исключить на фазе предобработки исходного графа решение NP-трудной задачи, расстановку ориентиров обычно выполняют с помощью эвристических алгоритмов. В настоящее время уже экспериментально установлено разумное число ориентиров, которое обеспечивает эффективность алгоритма A* на больших графах. Это интервал от 9 до 16 ориентиров [7, 15]. На сегодняшний день именно в такой постановке эта задача решается на первой фазе алгоритма ALT при обработке последовательности запросов на поиск кратчайших путей в графе $G = (V, E)$. Для этого применяются следующие эвристики [6, 11]: H_1 – случайная расстановка ориентиров в вершинах графа перед выполнением первого запроса; H_2 – выбор первого ориентира случайным образом, а каждого следующего ориентира как можно дальше от ранее выбранных; H_3 – случайная расстановка ориентиров перед выполнением первого запроса на выпуклой оболочке графа. Все эти эвристики разрабатывались для транспортных сетей. Им присущи следующие особенности: выполнимость за полиномиальное время; применимость только для сетей с евклидовой метрикой, когда вершины определены в некоторой системе координат; однократность расстановки ориентиров. Эти особенности, как правило, ограничивают область применения эвристик H_1–H_3. Так, применение евклидовой метрики в компьютерных и беспроводных сетях в большинстве случаев невозможно. Если задачу TDSP требуется решать многократно, то есть для последовательности (s, d, t_s)-запросов, то целесообразно учитывать историю обработки запросов и использовать адаптивные стратегии. Адаптивные стратегии позволяют периодически изменять места расстановки ориентиров с целью повышения их эффективности для текущего запроса и последующих запросов.

В модифицированном алгоритме ALT применяется полиномиальная по времени адаптивная стратегия расстановки ориентиров, реализованная в виде эвристики AdaHeuris. В рамках этой эвристики считается, что вершины нестационарной сети занумерованы и не привязаны к какой-либо системе координат. Эвристика AdaHeuris с определенной периодичностью изменяет положение ориентиров в сети, учитывая историю обработки запросов. Ее применение существенно расширяет спектр рассматриваемых нестационарных сетей и позволяет решать задачу TDSP для последовательности запросов, поступающих в режиме онлайн.

Суть эвристики AdaHeuris состоит в следующем. Пусть задача TDSP решается для нестационарной сети $G = (V, E)$ большой размерности применительно к последовательности σ, состоящей из конечного числа (s, d, t_s)-запросов. Перед обработкой первого (s, d, t_s)-запроса осуществляется случайная расстановка заданного числа $K = |L|$ ориентиров в вершинах графа $G = (V, E)$, при этом $9 \le K \le 16$. Затем выбранное множество ориентиров L периодически обновляется через каждые $\Delta > 0$ запросов. Полагается, что $|V| >> K$ и $|\sigma| >> \Delta$. При обработке последовательности запросов выполняется сбор информации о результативности текущих ориентиров. Всякий раз, когда ориентир предлагает наилучшую нижнюю оценку пути в соотношении (11) среди всего набора ориентиров, он получает очко. Процесс обновления ориентиров начинается после завершения Δ-го запроса. Пусть l_{old} – низкорезультативный ориентир, то есть ориентир с наименьшим количеством очков, набранных за период Δ. Данный ориентир перемещается в

вершину *new*, удовлетворяющую двум условиям: вершина *new* в алгоритме A* получила временную метку при обработке предыдущих (s, d, t_s)-запросов, и это ее состояние осталось неизменным на момент обновления ориентиров; вершина *new* наиболее удалена от множества ориентиров $L \setminus \{l_{old}\}$ и выбрана с учетом правила тупого угла. Далее информация об эффективности ориентиров формируется заново.

Применение в эвристике AdaHeuris правила тупого угла обеспечивает наилучшую нижнюю оценку (11), поскольку, согласно неравенству треугольника (6), верны соотношения

$dist(x, d, t_s) \geq dist'(x, d) \geq dist'(l, d) - dist'(l, x) = \pi_l(x)$.

Различное положение ориентира l по отношению к вершинам x и d приводит к различным значениям разности $dist(x, d, t_s) - \pi_l(x) \geq 0$. Эта разность тем меньше, чем больше угол при вершине x. Это хорошо иллюстрируют рисунки 1 и 2.

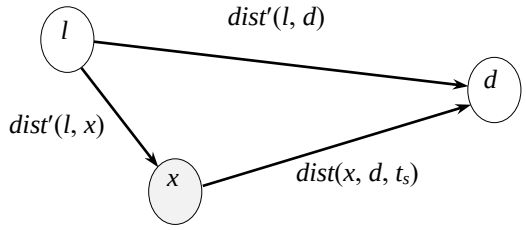

Рис. 1. Положение ориентира l отвечает правилу тупого угла и приводит к значению $dist'(l, d) - dist'(l, x) = \pi_l(x)$

Fig. 1. Landmark l location corresponds to the obtuse angle rule and leads to the value $dist'(l, d) - dist'(l, x) = \pi_l(x)$

На вход эвристики AdaHeuris поступает нестационарная сеть $G = (V, E)$, отвечающая условию FIFO и для которой неравенство (6) справедливо для любой тройки вершин $x, y, z \in V$. Считаются также заданными число ориентиров K, последовательность запросов σ и период Δ обновления множества ориентиров.

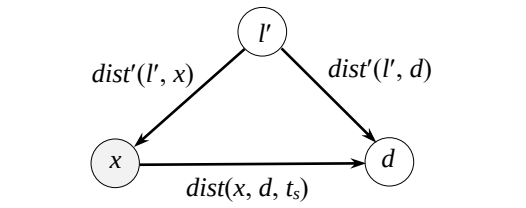

Рис. 2. Положение ориентира l′ не соответствует правилу тупого угла, что дает $dist'(l', d) - dist'(l', x) = \pi_{l'}(x) \leq \pi_l(x)$

Fig. 2. Landmark l′ location does not correspond to the obtuse angle rule, which gives $dist'(l', d) - dist'(l', x) = \pi_{l'}(x) \leq \pi_l(x)$

Эвристика состоит из трех процедур, которые вызываются в соответствующих местах модифицированного алгоритма ALT:

– ИНИЦИАЛИЗАЦИЯ формирует вспомогательные массивы *Landmark*, *Stat*, *Passed*, *Space* и выполняет первоначальную расстановку ориентиров случайным образом;

– ИСТОРИЯ ОБРАБОТКИ ЗАПРОСОВ собирает сведения о результатах выполнения текущего (s, d, t_s)-запроса и заносит их в массивы *Passed* и *Space*;

– ОБНОВЛЕНИЕ ОРИЕНТИРОВ выбирает низкорезультативный ориентир и перемещает его в вершину, выбор которой осуществляется в массиве *Space* с учетом правила тупого угла.

Описание этих процедур на псевдокоде представлено в виде алгоритмов 1–3.

Алгоритм 1. ИНИЦИАЛИЗАЦИЯ

1: $Landmark \leftarrow \varnothing$
2: $Stat \leftarrow \varnothing$
3: $Passed \leftarrow \varnothing$
4: $Space \leftarrow \varnothing$
5: $i \leftarrow \varnothing$
6: **Для** $i < K$
7: $Landmark(i) \leftarrow y$ // $y \in V$ – номер случайно выбранной вершины
8: **Для** каждого $x \in V$
9: **Вычислить** кратчайшие ($Landmark(i)$, x)-пути и (x, $Landmark(i)$)-пути
10: **Конец цикла**
11: $i \leftarrow i + 1$
12: **Конец цикла**

Алгоритм 2. ОБНОВЛЕНИЕ ОРИЕНТИРОВ

1: **Найти** индекс l_{old} такой, что $Stat(l_{old}) = \min(Stat(i) : i = 1, \ldots, K)$
2: **Вычислить** номер вершины *new* такой, что $d_{mark}(new) = \min(d_{mark}(i) : i \in Space)$
3: $Landmark(l_{old}) \leftarrow new$
4: **Вычислить** для вершины *new* и каждой вершины $x \in V$ кратчайшие (new, x)-пути и (x, new)-пути
5: $Stat \leftarrow \varnothing$

Алгоритм 3. ИСТОРИЯ ОБРАБОТКИ ЗАПРОСОВ

1: **Для** каждой вершины $x \in V$
2: **Если** x имеет временную метку и $x \notin Space$, $x \notin Passed$, **то**
3: $Space \leftarrow x$
4: **Конец условия**
5: **Если** x имеет постоянную метку и $x \in Space$, **то**
6: **Удалить** x из *Space*
7: $Passed \leftarrow x$
8: **Конец условия**
9: **Конец цикла**

Формируемые в эвристике AdaHeuris вспомогательные массивы имеют следующее назначение. Массив *Landmark* служит для формирования и хранения множества ориентиров L, причем i-я строка этого массива содержит номер вершины, где установлен i-й ориентир, и длины всех кратчайших путей от этого ориентира до всех других вершин графа $G = (V, E)$, $i = 1, \ldots, K$. В массив *Stat* записывается информация о результативности каждого ориентира из L. Массивы *Passed* и *Space* предназначены для хранения номеров вершин, получивших в алгоритме A* постоянные и временные метки соответственно при обработке предшествующих (s, d, t_s)-запросов. В данных массивах накапливается история обработки запросов – информация о множестве вершин, пройденных алгоритмом A* в процессе обработки выполненной части последовательности σ. Это множество вершин является пространством поиска решения для алгоритма A*, причем вершины из массива *Space* рассматриваются в качестве границы этого пространства, куда перемещаются низкорезультативные ориентиры. В худшем случае мощность пространства поиска решений для графа $G = (V, E)$ составляет $|V|$. Основная цель эвристики AdaHeuris – уменьшение мощности этого пространства.

Кроме представленных выше алгоритмов 1–3, эвристика AdaHeuris содержит также алгоритм 4 – процедуру ВЫЧИСЛЕНИЕ ПОТЕНЦИАЛЬНОЙ ФУНКЦИИ, которая вызывается из алгоритма A*.

Алгоритм 4. ВЫЧИСЛЕНИЕ ПОТЕНЦИАЛЬНОЙ ФУНКЦИИ

1: **Вычислить** $\pi_L(x)$ для текущей вершины $x \in V$
2: **Найти** ориентир l, такой что $\pi_l(x) = \pi_L(x)$
3: $Stat(l) \leftarrow Stat(l) + 1$
4: **Возвратить** $\pi_L(x)$

Описание, анализ и программная реализация модифицированного алгоритма ALT

Порядок выполнения различных составляющих модифицированного алгоритма ALT описан в алгоритме 5.

Алгоритм 5. МОДИФИЦИРОВАННЫЙ АЛГОРИТМ ALT

1: **Выполнить** процедуру ИНИЦИАЛИЗАЦИЯ
2: $i \leftarrow 0$
3: **Для** каждого (s, d, t_s)-запроса из σ
4: **Выполнить** алгоритм A*
5: **Выполнить** процедуру ИСТОРИЯ ОБРАБОТКИ ЗАПРОСОВ
6: $i \leftarrow i + 1$
7: **Если** $i = \Delta$, **то**
8: **Выполнить** процедуру ОБНОВЛЕНИЕ ОРИЕНТИРОВ
9: $i \leftarrow 0$
10: **Конец условия**
11: **Конец цикла**

Оценим время работы алгоритма 5 на основе анализа его неэлементарных шагов 1, 4, 5, 8.

Шаг 1. Время выполнения процедуры ИНИЦИАЛИЗАЦИЯ, предназначенной для первоначальной расстановки ориентиров и формирования массивов *Landmark*, *Stat*, *Passed*, *Space*, составляет $O(|V|^2 \cdot K)$, поскольку наиболее трудоемким действием этой процедуры является создание массива *Landmark*, требующего $O(|V|^2 \cdot K)$ времени.

Шаг 4. Время работы алгоритма A* традиционно составляет $O(|V|^2)$ [7]. Эту оценку не изменяют особенности реализации алгоритма A* в модифицированном ALT. Суть этих особенностей: вычисление потенциальных функций по формулам (7)–(10); сбор информации о результативности ориентиров, применяемой для обновления ориентиров.

Шаг 5. Время, необходимое для выполнения процедуры ИСТОРИЯ ОБРАБОТКИ ЗАПРОСОВ, сопоставимо с мощностью пространства поиска решения и в худшем случае составляет $|V|$.

Шаг 8. В процедуре ОБНОВЛЕНИЕ ОРИЕНТИРОВ осуществляют выбор низкорезультативного ориентира l_{old} и его перемещение в новую вершину *new* из массива *Space*, а также вычисление кратчайших путей от вершины *new* до всех других вершин графа. Для выполнения этих действий требуется $O(|V|^2)$ времени.

Очевидно, что однократная реализация шагов 4–10 модифицированного алгоритма ALT требует $O(|V|^2)$ времени. Поскольку число повторений цикла 3–11 совпадает с числом запросов, входящих в последовательность σ, время выполнения данного цикла в целом составляет $O(|\sigma| \cdot |V|^2)$. Таким образом, модифицированный алгоритм ALT находит точное решение задачи TDSP для последовательности σ в нестационарной сети $G = (V, E)$ с условием FIFO за время $O(|\sigma| \cdot |V|^2)$.

Замечание 1. Если в модифицированном алгоритме ALT число ориентиров $K = |V|$, то время работы процедуры ИНИЦИАЛИЗАЦИЯ составит $O(|V|^3)$. В этом случае ориентиры размещены во всех вершинах графа и время выполнения процедуры ИНИЦИАЛИЗАЦИЯ превосходит время однократной реализации всех других шагов модифицированного алгоритма ALT. Следовательно, для эффективной работы данного алгоритма целесообразно исходить из небольшого по сравнению с $|V|$ фиксированного значения K.

Замечание 2. Полученная выше оценка $O(|\sigma| \cdot |V|^2)$ времени работы модифицированного алгоритма ALT вычислялась применительно к худшему случаю для Δ: при $\Delta = 1$, когда обновление ориентиров происходит после выполнения каждого (s, d, t_s)-запроса. Очевидно, что подобная стратегия обновления ориентиров весьма затратная по времени. Поэтому для эффективной работы модифицированного алгоритма ALT подбор значения Δ

целесообразно осуществлять экспериментальным путем, исходя из типичного для рассматриваемой сети содержания запросов и их возможных последовательностей.

Замечание 3. Известно, что алгоритм Дейкстры находит точное решение задачи TDSP для заданного (s, d, t_s)-запроса за время $O(|V|^2)$ [2, 5]. Для последовательности σ, состоящей из конечного числа (s, d, t_s)-запросов, алгоритму Дейкстры потребуется $O(|\sigma|\cdot|V|^2)$ времени. Это сопоставимо с приведенной выше оценкой времени работы модифицированного алгоритма ALT. Заметим, что в обоих случаях эти оценки получены для худшего случая. Однако на практике время выполнения данных алгоритмов во многом зависит от содержания запросов (положения стартовых и целевых вершин относительно друг друга) и порядка их следования в σ. Время выполнения модифицированного алгоритма ALT также зависит от значений K и Δ. Поэтому экспериментально сравнивать эти алгоритмы между собой целесообразно на различных сетях и последовательностях σ, реально существующих или сгенерированных случайным образом. При сравнении могут быть использованы следующие параметры: $|V_\sigma|$ – мощность пространства поиска решения для последовательности σ в целом; $|V_{avg}|$ – среднее значение мощности пространства поиска, вычисленное для одного запроса из σ. Эти параметры отражают, насколько тот или иной алгоритм в процессе своей работы охватывает вершины исходной сети.

На основе описанных выше алгоритмов 1–5 в среде Visual Studio Professional 2013 C++ была разработана программа TDSPALT, реализующая предложенный модифицированный алгоритм ALT. Программа TDSPALT использует динамические массивы, поэтому размерность исходного графа ограничена только объемом оперативной памяти ЭВМ.

Программа TDSPALT состоит из следующих модулей: main – запуск основных процедур и модулей; loadGraph – загрузка исходных данных; search – обработка текущего (s, d, t_s)-запроса; fillLandmark – установка отдельного ориентира; replaceLandmark – обновление ориентиров; calculatePathsToX – вычисление кратчайших путей от заданного ориентира до каждой вершины исходного графа; calculatePathsfromX – вычисление кратчайших путей от каждой вершины исходного графа до заданного ориентира.

На вход программы TDSPALT подаются два файла. Первый файл содержит в себе описание исходного графа в виде списка смежности вершин и весовых функций дуг. Весовые функции дуг задаются перечислением их значений для дискретных моментов времени. Второй файл включает в себя последовательность (s, d, t_s)-запросов. Дополнительно вводятся значения параметров K и Δ. Выходом программы являются решения задачи TDSP для каждого отдельного (s, d, t_s)-запроса.

Вычислительные эксперименты

Для оценки результативности модифицированного алгоритма ALT были проведены вычислительные эксперименты на случайно сгенерированных графах и дорожных сетях, представленных в БД DIMACS [13]. Последовательности запросов σ выбирались случайным образом, при этом $|\sigma| = 500$. Вычислительные эксперименты выполнялись на компьютере с процессором Intel®Core™ i7-720QM Processor (6M Cache, 1.60 GHz) и ОЗУ размером 4 Гб.

Модифицированный алгоритм ALT сравнивался с алгоритмом Дейкстры и классическим алгоритмом ALT. Напомним, что классический алгоритм ALT предусматривает лишь однократную расстановку ориентиров случайным образом. При сравнении использовался классический алгоритм ALT с эвристикой H_1. Алгоритмы сравнивались по следующим показателям: C_1 – коэффициент сокращения пространства поиска, равный отношению величин $|V_{avg}|$ для сопоставляемых алгоритмов; C_2 – коэффициент ускорения, определяемый как отношение времени работы сравниваемых между собой алгоритмов. Считалось, что соответствующие характеристики ($|V_{avg}|$ и время работы) модифицированного алгоритма ALT находятся в знаменателе этих отношений. Опираясь на результаты предыдущих исследований [15] и замечания 1–3, были выбраны следующие параметры эвристики AdaHeuris: число ориентиров $K = 12$, период обновления ориентиров $\Delta = 30$. Результаты сравнения модифицированного алгоритма ALT с алгоритмом Дейкстры и классическим алгоритмом ALT приведены в таблицах 1 и 2 соответственно. Из 6 графов, представленных в этих таблицах, графы LineGraph и GridGraph сгенерированы случайным образом, а все остальные графы взяты из DIMACS. В таблицах 1 и 2 для каждого графа $G = (V, E)$ указаны число вершин $|V|$ и число дуг $|E|$, графы перечислены в порядке возрастания $|V|$.

Таблица 1

Результаты сравнения модифицированного алгоритма ALT и алгоритма Дейкстры

Table 1

Results of comparison of modified ALT algorithm and Dijksta's algorithm

Граф $G = (V, E)$	**$\|V\|$**	**$\|E\|$**	**C_1**	**C_2**
Rome99	3353	8870	1,12	1,07
LineGraph	10001	20000	1,03	1,20
GrigGraph	62500	498000	1,20	2,19
AK	69082	157662	1,05	1,12
VT	97975	216628	1,08	1,14
CT	153011	375310	1,14	1,30

Таблица 2

Результаты сравнения модифицированного и классического алгоритмов ALT

Table 2

Results of comparison of modified and classic ALT algorithms

Граф $G = (V, E)$	$\|V\|$	$\|E\|$	C_1	C_2
Rome99	3353	8870	1,01	0,97
LineGraph	10001	20000	1,01	1,26
GrigGraph	62500	498000	1,05	1,17
AK	69082	157662	1,01	1,05
VT	97975	216628	1,00	1,12
CT	153011	375310	1,05	1,42

Из таблиц видно, что модифицированный алгоритм ALT дает лишь незначительное сокращение пространства поиска. Однако при выбранных параметрах K и Δ наблюдается уменьшение времени работы данного алгоритма по сравнению с алгоритмом Дейкстры и классическим алгоритмом ALT. Причем уменьшение времени работы тем значительнее, чем больше $|V|$. По результатам экспериментов можно сделать вывод, что предложенная модификация алгоритма ALT по быстродействию не уступает алгоритму Дейкстры и своему классическому аналогу, а при соответствующей настройке параметров K и Δ эвристики AdaHeuris превосходит их. Такая настройка всегда может быть выполнена с учетом особенностей рассматриваемого графа и возможных последовательностей σ.

Заключение

В работе представлена модификация известного алгоритма ALT для задачи оптимальной маршрутизации в нестационарной сети. В предложенном алгоритме применены специальные формулы для вычисления потенциальных функций, которые гарантируют нахождение точного решения задачи TDSP, и адаптивная стратегия расстановки ориентиров. Результаты выполненных вычислительных экспериментов подтвердили эффективность предложенного алгоритма. Перспективны дальнейшие исследования по решению задачи оптимальной маршрутизации для нестационарной сети с учетом вероятностей отказов ее вершин и ребер.

Литература

1. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. М.: Либроком, 2012. 392 с.
2. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. М.: Вильямс, 2013. 1328 с.
3. Sherali H.D., Ozbay K., Subramanian S. The time-dependent shortest pair of disjoint paths problem: Complexity, models, and algorithms. J. Networks, 1998, vol. 31, no. 4, pp. 259–272.
4. Kaufman D.E., Smith R.L. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. J. Intelligent Transportation Systems, 1993, vol. 1, no. 1, pp. 1–11.
5. Гимади Э.Х., Глебов Н.И. Математические модели и методы принятия решений. Н.: Изд-во НГУ, 2008. 162 с.
6. Wagner D. & Willhalm T. Speed-up techniques for shortest-path computations. Proc. STACS, 2007, pp. 23–36.
7. Goldberg A., Kaplan H. & Werneck R. Reach for *A**: Efficient point-to-point shortest path algorithms. Technical Report MSR-TR-2005-132, Microsoft Research, 2005, 41 p.
8. Nejad M.M., Mashayekhy L., Chinnam R.B., Phillips A. Hierarchical time-dependent shortest path algorithms for vehicle routing under ITS. J. IIE Transactions, 2016, vol. 48, no. 2, pp. 158–169.
9. Delling D. Time-dependent SHARC-routing. J. Algorithmica, 2011, vol. 60, no. 1, pp. 60–94.
10. Delling D., Wagner D. Landmark-based routing in dynamic graphs. Proc. Intern. WEA, Springer, 2007, 4525 of LNCS, pp. 52–65.
11. Goldberg A., Kaplan H. & Werneck R. Better landmarks within reach. Proc. Interna. WEA, vol. 4525 LNCS, Springer, 2007, pp. 38–51.
12. Рассел С., Норвинг П. Искусственный интеллект: современный подход. М.: Вильямс, 2006. 1408 с.
13. DIMACS Implementation Challenge. Shortest Paths, 2006. URL: http://www.dis.uniroma1.it/challenge9/ (дата обращения: 28.09.2017).
14. Быкова В.В., Солдатенко А.А. Оптимальная маршрутизация по ориентирам в нестационарных сетях // Прикладная дискретная математика. 2017. № 37. С. 114–123.
15. Быкова В.В., Солдатенко А.А. Адаптивное размещение ориентиров в задаче о кратчайшем пути для графов большой размерности // Программные продукты и системы. 2016. № 1. С. 60–67.

Software & Systems
DOI: 10.15827/0236-235X.122.321-329

Received 10.10.17
2018, vol. 31, no. 2, pp. 321–329

ADAPTIVE ALGORITHM FOR OPTIMAL ROUTE SEARCH IN TIME-DEPENDENT NETWORK

A.A. Soldatenko [1], *Postgraduate Student, glinckon@gmail.com*

[1] *Siberian Federal University, Svobodny Ave. 79, Krasnoyarsk, 660041, Russian Federation*

Abstract. The Time-Dependent Shortest-Path problem (TDSP) is an extension of the shortest path problem in a graph.

TDSP problem arises when designing and operating telecommunications and transport networks. Such networks require considering time and possibility of appearing predictable situations for example traffic jams or traffic reduction. In this case,

network is represented with an oriented graph $G = (V, E)$ where for each arc $(x, y) \in E$, two functions are defined. First function is time required for moving along the arc (x, y). Second function is arrival time in vertex y if the movement started from vertex x in time t. Such graph is called time-dependent network. The minimum time for moving from vertex x to vertex y is an optimal route between these vertices.

It is known that TDSP for a general time-dependent network without any restrictions on network topology or arrival function is NP-hard. When arrival function satisfies FIFO (First-In First-Out) condition, TDSP problem is polynomially solvable. This paper studies TDSP problem for a polynomial case when arrival functions are monotonous. It is proposed to solve TDSP problem using a two-phased algorithm ALT (A* with Landmarks & Triangle). ALT algorithm is one of the modern least cost routing algorithms originally developed for solving the problem of the shortest path in a graph. In the first phase, ALT algorithm places landmarks in network vertices and calculates potential functions. In the second phase it finds the exact value of an optimal route in a graph with A* algorithm.

The paper proposes modification of ALT algorithm which is capable of correct and efficient solving the TDSP problem for a sequence of queries for searching for optimal routes in a time-dependent network. The modification consists in using adaptive heuristic for landmark placement and special formulas for calculating potential functions. This heuristic uses experience of processing all completed queries; adapts current set of landmarks for next queries. There is a description of the modified ALT algorithm and estimation of its working time. The paper also gives a description of the software that implements the proposed algorithm.

The results of computational experiments confirm the effectiveness of the modified ALT algorithm.

Keywords: time-dependent networks, big graph, optimal routing, ALT algorithm, landmarks placement.

References

1. Emelichev V.A., Melnikov O.I., Sarvanov V.I., Tyshkevich R.I. *Lektsii po teorii grafov* [Lectures on Graph Theory]. Moscow, Librokom Publ., 2012, 392 p.

2. Kormen T., Leyzerson Ch., Rivest R., Shtayn K. *Algoritmy. Postroenie i analiz* [Algorithms. Construction and Analysis]. Vilyams Publ., 2013, 1328 p.

3. Sherali H.D., Ozbay K., Subramanian S. The time-dependent shortest pair of disjoint paths problem: Complexity, models, and algorithms. *J. Networks*. 1998, vol. 31, no. 4, pp. 259–272.

4. Kaufman D.E., Smith R.L. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *J. Intelligent Transportation Systems*. 1993, vol. 1, no. 1, pp. 1–11.

5. Gimadi E.Kh., Glebov N.I. *Matematicheskie modeli i metody prinyatiya resheny* [Mathematical Models and Methods of Decision-Making]. Novosibirsk, NGU Publ., 2008, 162 p.

6. Wagner D., Willhalm T. Speed-up techniques for shortest-path computations. *Proc. STACS*. 2007, pp. 23–36.

7. Goldberg A., Kaplan H., Werneck R. *Reach for A*: Efficient point-to-point shortest path algorithms*. Technical Report MSR-TR-2005-132, Microsoft Research, 2005, 41 p.

8. Nejad M.M., Mashayekhy L., Chinnam R.B., Phillips A. Hierarchical time-dependent shortest path algorithms for vehicle routing under ITS. *J. IIE Trans*. 2016, vol. 48, no. 2, pp. 158–169.

9. Delling D. Time-Dependent SHARC-Routing. *J. Algorithmica*. 2011, vol. 60, no. 1, pp. 60–94.

10. Delling D., Wagner D. Landmark-based Routing in Dynamic Graphs. *Proc. Int. Workshop on Experimental Algorithms (WEA)*. Springer Publ., 2007, 4525 of LNCS, pp. 52–65.

11. Goldberg A., Kaplan H., Werneck R. Better landmarks within reach. *Proc. Int. Workshop on Experimental Algorithms (WEA)*. Vol. 4525 of Lecture Notes in Computer Science, Springer Publ., 2007, pp. 38–51.

12. Rassel S., Norving P. *Iskusstvenny intellekt: sovremenny podkhod* [Artificial Intelligence: a Modern Approach]. Vilyams Publ., 2006, 1408 p.

13. *DIMACS Implementation Challenge. Shortest Paths*. 2006. Available at: http://www.dis.uniroma1.it/challenge9/ (accessed September 28, 2017).

14. Bykova V.V., Soldatenko A.A. Optimal routing by landmarks in the timedependent networks. *Prikladnaya diskretnaya matematika* [Applied Discrete Mathematics]. 2017, no. 37, pp. 114–123 (in Russ.).

15. Bykova V.V., Soldatenko A.A. Adaptive landmark selection in shortest-paths problem for a big graph. *Programmnye produkty i sistemy* [Software & Systems]. 2016, no. 1, pp. 60–67 (in Russ.).