УДК 681.3.08

Определение кратчайшего маршрута с использованием общественного транспорта

П. Ш. Гейдаров

В работе предлагается алгоритм определения кратчайшего маршрута с использованием общественного транспорта. Приводится базовый алгоритм без географической привязки к местности и с возможностью одной пересадки. Рассматривается описание структуры базы данных, позволяющей в качестве пункта остановки использовать сразу несколько объектов, располагающихся вблизи пунктов остановки. На основе базового алгоритма реализуется алгоритм с двумя или несколькими пересадками, а также описываются возможности расширения функциональности алгоритма применительно к разным видам транспорта и с привязкой к географическим координатам местности. Рассматриваются программные и структурные способы ускорения работы алгоритма.

Ключевые слова: кратчайший маршрут, городской транспорт, общественный транспорт, транспорт в Баку, центр интеллектуального управления транспортом.

1. Введение

Быстрорастущие потребности современных городов повышают необходимость автоматизации сферы транспорта, что становится причиной широкого внедрения информационных технологий при решении дорожно-транспортных задач. Одной из подобных задач является задача определения кратчайшего транспортного пути от одного пункта до другого, и как частный случай этой задачи – задача определения кратчайшего пути с использованием городского транспорта: автобусов, троллейбусов, метро и т.д. Актуальность решения данных задач определяется еще и быстротой изменения облика современных городов: строительством новых зданий, парков, улиц, проспектов; переименованием названий существующих улиц, учреждений; заменой, объединением или перемещением одних организаций в другие; сокращением, переименованием или введением новых транспортных линий. Все это приводит к сложностям ориентирования на местности. К примеру, в г. Баку за последние 20 лет несколько раз менялась вся сеть автобусных маршрутов: траектория их передвижения, номера и количество маршрутов. Менялись также названия улиц, парков, проспектов. Решение данной задачи также полезно для развития сферы туризма, поскольку задача создания условий быстрого и удобного ориентирования в городе без необходимости знания языка и города может в целом способствовать развитию этой области.

2. Основная часть

2.1. Постановка задачи

Нужно отметить, что для задач определения кратчайшего пути существуют различные алгоритмы, которые в целом можно подразделить на два типа: алгоритмы перебора, выдаю-

щие оптимальные результаты [1–4] и эвристические алгоритмы, определяющие приближенно не точные результаты, например, муравьиные или генетические алгоритмы [5–8]. Применение последних обосновано в тех случаях, когда задача является не решаемой иными способами или требует просмотра слишком большого объема информации в ограниченный период времени. В качестве примера алгоритма перебора для задачи определения кратчайшего пути можно было бы привести, например, используемый в системах GPS-навигации алгоритм перебора Дейкстра [1–4], определяющий маршрут на основе перебора вершин графа, предварительно созданного на основе информации о существующих дорогах, маршрутах и т.д. Алгоритм Дейкстра, а также похожие алгоритмы перебора, такие как Беллмана-Форда, Флойда-Уоршелла и др. [1–4] имеют универсальный характер и применимы к разным задачам, но при этом применительно к отдельным задачам, в частности, к рассматриваемой задаче, имеют ряд слабых мест. В частности, потребуется реализация дополнительного алгоритма для создания графа на основе базы существующих транспортных маршрутов. Во время или после реализации этих алгоритмов понадобится также алгоритм выполнения привязки полученного пути маршрута к существующим транспортным маршрутам. При этом количество пересадок для полученного кратчайшего маршрута может получиться чрезмерно большим, что будет неудобно для практического применения этих маршрутов. Кроме того, в ряде случаев универсальные алгоритмы [1–4] будут более медлительными, чем предлагаемый алгоритм. Связано это с тем, что алгоритмы перебора графов последовательно перебирают все смежные вершины графа, двигаясь от одной вершины в другую, таким образом рассматривая большое количество вершин, в том числе и не имеющих отношения к кратчайшему маршруту (рис. 1a). Тогда как в предлагаемом алгоритме благодаря наличию базы маршрутов, а также при наличии малого количества маршрутов и пересадок между начальными и конечными пунктами кратчайшего маршрута будут рассмотрены только эти маршруты (рис. 1b).

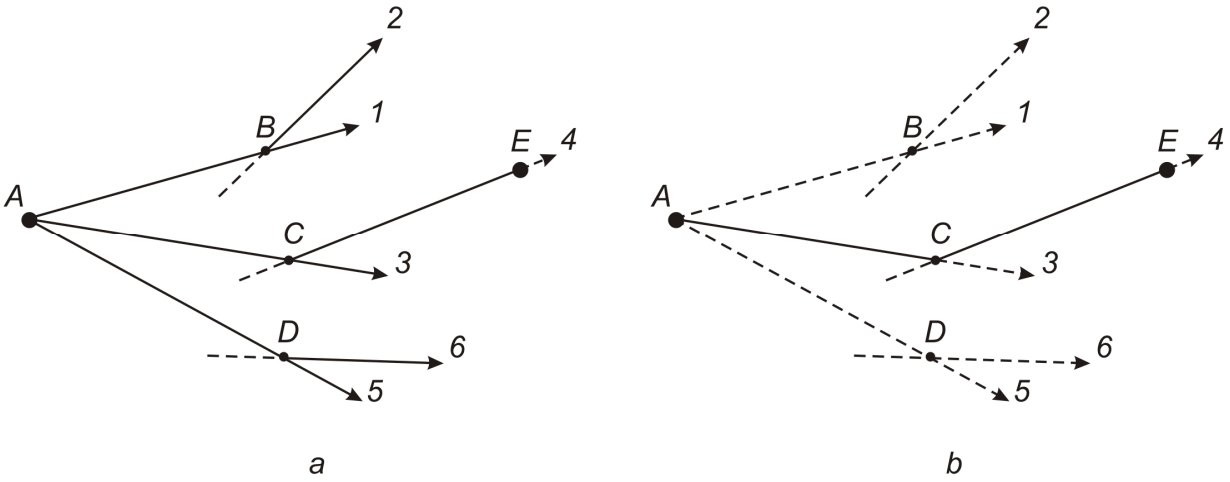


Рис.1. Пример графа, образованного из 6-ти прямолинейных маршрутов с просмотренными (не пунктирными) ребрами графа:
(a) по алгоритму Дейкстра, (b) по рассматриваемому алгоритму

На рис. 1a, b приведен пример графа с количеством рассмотренных ребер (не пунктирные ребра графа) по алгоритму Дейкстра (рис. 1a) и по рассматриваемому алгоритму (рис. 1b) для задачи определения маршрута от точки A до E на графе, построенным из 6-ти прямолинейных маршрутов, пересекающихся в комбинации {1,2} в точке B, {3,4} в точке C и {5,6} в точке D. Можно видеть, что для определения маршрута ACE алгоритму Дейкстра для данного примера понадобится больше шагов (выполняемых операций), чем в случае рассматриваемого алгоритма. Здесь нужно отметить, что при увеличении количества пунктов пересадок в кратчайшем маршруте, а также увеличении количества маршрутов, проходящих через начальный и конечный пункты, количество рассматриваемых вершин графа будет увеличиваться и соответственно скорость работы предлагаемого алгоритма будет уменьшаться.

2.2. Определение структуры базы данных и алгоритма определения кратчайшего маршрута с одной пересадкой без привязки маршрутов к системам отсчета

Рассмотрим для начала структуру и работу алгоритма определения кратчайшего маршрута с одной пересадкой, определяемого по количеству остановок, применительно к автобусным маршрутам города Баку. При этом данный алгоритм реализуется без привязки к географическим координатам пунктов остановок.

Для работы алгоритма предварительно создается структура базы данных, представляющая из себя список всех номеров автобусных маршрутов, после каждого из которых следует последовательный перечень всех наименований пунктов остановок маршрута (пример 1). При этом наименование пункта остановки маршрута может включать в себя наименования сразу нескольких объектов, располагающихся поблизости пункта остановки транспорта (рис. 2). На примере (пример 1) показана структура базы данных, организованная следующим образом. Все пункты маршрутов транспортных средств располагаются построчно друг за другом. Каждый маршрут начинается с номера маршрута транспорта. Каждый пункт остановки состоит из наименований названий различных объектов, располагающихся поблизости данной остановки маршрута. При этом все названия объектов и улиц, касающиеся одной остановки транспорта, располагаются на одной строке и разделяются запятыми, последовательно друг за другом в порядке удаления объектов от пункта остановки. Такая организация структуры базы данных позволяет в дальнейшем гибко управлять и менять базу данных.

Пример 1. Пример фрагмента базы данных

№104

1. м. Дружба народов

2. пл. Украины, м. Ази Асланова

3. радио завод

...

10. Гагаринский мост, больница Нефтяников

11. Консерватория, Центральный банк

12. м. 28 мая, железнодорожный вокзал, Университет нефти и химии

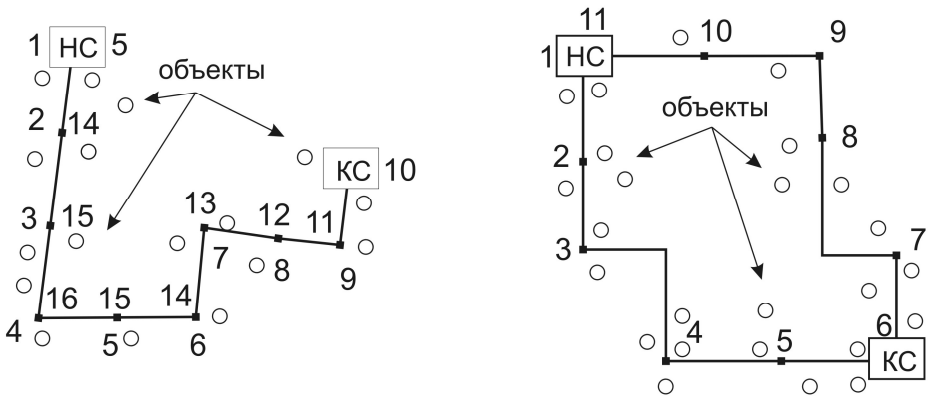
13. дворец Республики, больница №4, ул. Басина

14. пл. Физули, шахматная школа, ул. Гуси Гаджиева, МИД

15. ЦУМ, пл. фонтанов, торговая

Нужно отметить, что передвижение автобусов может происходить как по замкнутым (рис. 2b), так и по незамкнутым маршрутам (рис. 2a). В случае замкнутого маршрута пункты остановок маршрута перечисляются в тексте только один раз сверху вниз по направлению движения транспорта (*пример 1*), в случае незамкнутого маршрута для целостности работы алгоритма пункты остановок маршрута перечисляются дважды: сначала в направлении движения от начальной станции (НС) к конечной станции (КС), затем в обратном направлении – от конечной станции до начальной станции.

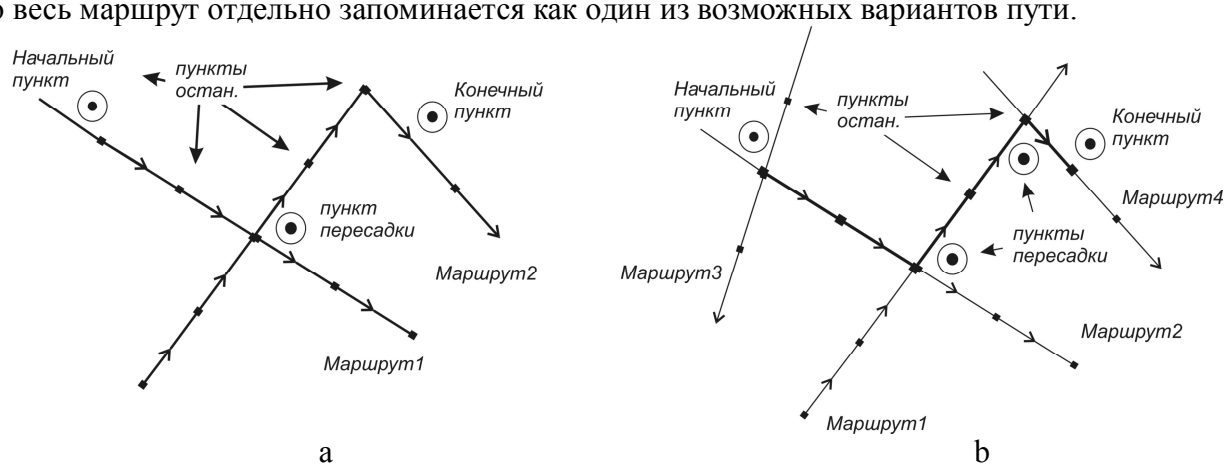
Выполнение алгоритма состоит из 4-х встроенных друг в друга циклов. При этом первый и второй циклы выполняют поиск начальных и конечных пунктов назначения в базе данных, а третий и четвертый циклы определяют пункт пересадки и, соответственно, выполняются только в том случае, когда начальный и конечный пункты назначения в первых двух циклах определяются на разных маршрутах транспорта.



a b

Рис. 2. (a) незамкнутый маршрут, (b) замкнутый маршрут

Алгоритм работает следующим образом. После того как пользователь вводит начальный и конечный пункт назначения (на рис. 4b текстовые поля *1* и *2* с наименованиями: «*откуда», «куда»*), выполняется работа первого цикла, последовательно просматривается база данных начиная с самой верхней строки на предмет определения первого совпадения начального пункта назначения («*откуда*») с записями пунктов остановок маршрутов. При этом просматриваются все записи строки, разделенные запятыми на одной строке. Как только определяется первое совпадение, запоминается номер маршрута, а также номер строки для этой записи. Начиная с этой строки передается выполнение на второй цикл, где последовательно просматриваются строки с целью определения второго пункта назначения («*куда*») начиная с самой верхней строки базы данных. Если второй пункт назначения определяется в пределах этого же маршрута, в этом случае мы получаем маршрут проезда без пересадки. Если же строка второго пункта назначения определяется на маршруте другого номера маршрута автобуса, то выполняется проверка полученных маршрутов на наличие пункта пересадки. С этой целью последовательно выполняется сравнение записей строк найденных маршрутов (рис. 3a) на предмет определения идентичного наименования пункта, который и будет являться пунктом пересадки – остановкой пересечения двух маршрутов. При этом просмотр строк идет в направлении движения первого маршрута (3-ий цикл) начиная с вышеопределенной строки пункта назначения и в обратном направлении для второго маршрута (4-ый цикл) начиная со строки конечного пункта назначения (рис. 4a). В случае если пункт пересадки определяется, то весь маршрут отдельно запоминается как один из возможных вариантов пути.



a b

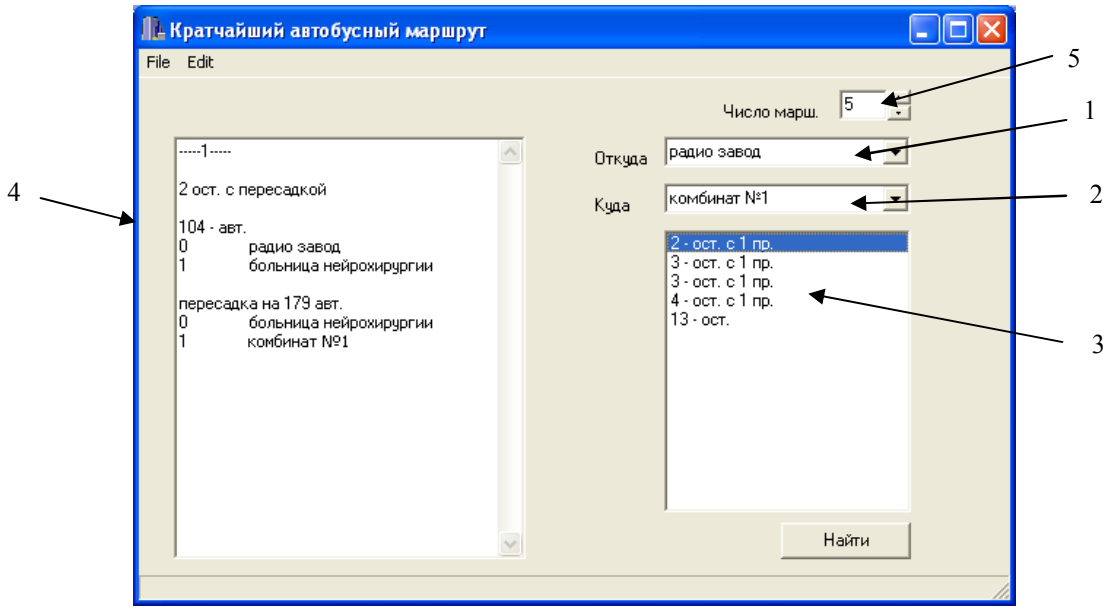
Рис. 3. (a) пересекаемые маршруты, (b) непересекаемые маршруты

При этом при сохранении маршрута помимо начального, конечного и пункта пересадки также дополнительно запоминаются промежуточные пункты проезда объектов остановок, в качестве которых выбираются объекты, указанные первыми в строках как наиболее ближайшие к пунктам остановок.

Если пункт пересечения не определяется, то это будет означать, что маршруты не пересекаются (рис. 3b). В этом случае продолжается процесс выполнения второго цикла на предмет определения следующего совпадения конечного пункта назначения в оставшихся записях базы данных. Процедура 2-го цикла повторяется аналогичным образом до тех пор, пока процесс просмотра не достигнет конца базы данных, после чего алгоритм возвращается к первому циклу и продолжает поиск нового начального пункта со строки последнего определенного начального пункта. Все полученные маршруты, включая номера маршрутов, наименования пунктов остановок, а также места пересадки сохраняются и выводятся на экран (рис. 4b (3, 4)). При этом полученные маршруты являются вариантами маршрутов без пересадки или с одной пересадкой.

	Маршрут 1		Маршрут 2
	остановка 1		остановка 1
	остановка 2		остановка 2
	….		…..
	начальный пункт остановки		…..
	….		…..
↓	пункт пересадки	=	пункт пересадки ↑
	….		….
	…..		конечный пункт остановки
	….		….

a)



b)

Рис. 4. (a) Схема работы 3-го и 4-го циклов,
(b) Программный модуль алгоритма определения кратчайшего маршрута с одной пересадкой без привязки к географическим координатам применительно к автобусным маршрутам г. Баку

2.3. Определение кратчайшего маршрута с двумя или несколькими пересадками

На основе данного алгоритма может быть также получен алгоритм определения кратчайшего маршрута с двумя или несколькими пересадками. Например, для определения кратчайшего маршрута с двумя пересадками выполняется последовательное переназначение каждой *k*-ой остановки *i*-го маршрута новым начальным пунктом ($НП_k^{(i)}$) для нового маршрута {$НП_k^{(i)}$, КП}. При этом конечный пункт (КП) остается неизменным – первоначально заданным. После каждого переназначения начального пункта ($НП_k^{(i)}$) цикл программы перенаправляется в начало первого цикла программы и по аналогичной схеме, приведенной выше, повторяется поиск пункта пересадки. Полученные маршруты сохраняются. Далее по такой же схеме выполняется обратная задача. Начальный пункт остается заданным, а конечный пункт ($КП_n^{(j)}$) меняется путем перебора остановок второго выбранного *j*-го маршрута. Аналогично для каждого полученного маршрута между определенными пунктами {$НП_k^{(i)}$, КП} или {НП, $КП_n^{(j)}$} выполняется описанный выше алгоритм на определение кратчайшего маршрута с одной пересадкой $ПР_{k,n}^{(i,j)}$. В результате получаются итоговые варианты маршрутов с двумя пересадками {НП, $НП_k^{(i)}$, $ПР_{k,0}^{(i,j)}$, КП} и {НП, $ПР_{0,n}^{(i,j)}$, $КП_n^{(j)}$, КП} (рис. 5). Здесь пункты $НП_k^{(i)}$ и $КП_n^{(j)}$ определяют дополнительные пункты пересадки для маршрута с двумя пересадками. На основе полученного списка всех маршрутов вычисляются кратчайшие маршруты. Для приведенного на рис. 5 примера кратчайший маршрут между НП и КП будет: {НП, $ПР_{0,2}^{(i,j)}$, $КП_2^{(j)}$, КП}.

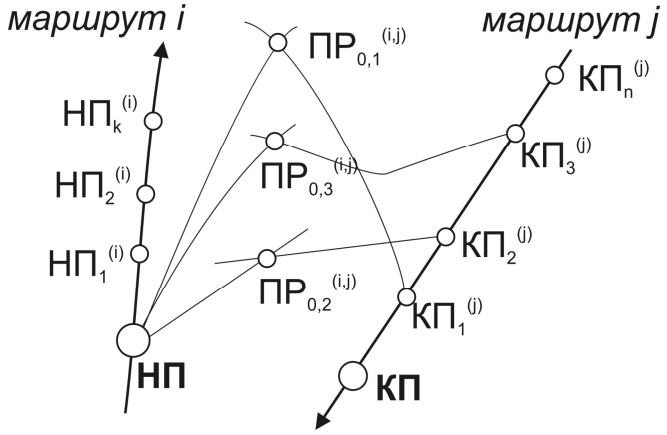


Рис. 5. Пример схемы определения маршрутов с двумя пересадками между пунктами НП и КП, с перебором пунктов КП на транспортном маршруте *j*

При этом если при выполнении основного алгоритма кратчайший маршрут между пунктами {$НП_k^{(i)}$, КП} или {НП, $КП_n^{(j)}$} определяется без пересадки, то в этом случае мы получаем маршрут с одной пересадкой по схеме {НП, $НП_k^{(i)}$, КП} или {НП, К $П_n^{(j)}$, КП}. Если же при назначении пунктов $НП_k^{(i)}$, $КП_n^{(j)}$ выполняется условие $НП_k^{(i)}$=КП или НП= $КП_n^{(j)}$, то получим маршрут без пересадки {НП, КП}. То есть расширенный таким образом алгоритм определения кратчайшего маршрута с двумя пересадками включает в себя определение кратчайшего маршрута и с одной пересадкой, и без пересадки. Аналогичным образом на базе основного алгоритма путем выполнения дополнительных циклов можно получить кратчайшие маршруты с тремя и более пересадками, например, {НП, $НП_k^{(i)}$, $ПР_{k,n}^{(i,j)}$, $КП_n^{(j)}$, КП}. Нужно сказать, что маршруты с большим количеством пересадок создают дополнительные неудобства для самих пассажиров и при этом увеличивают длительность и скорость работы алгоритма. Обычно в условиях большого города бывает вполне достаточно нескольких маршрутов с количеством пересадок от одной до трех.

В процессе поиска маршрута выполняется также перераспределение полученных маршрутов в порядке увеличения их длины. Например, на рис. 4b самый верхний маршрут в списке 3 соответствует самому кратчайшему маршруту. При этом в наиболее простом применении длина маршрута может оцениваться количеством остановок до и после пересадки (рис. 4b) с учетом количества пересадок, например, по выражению:

$$D = \sum_{i=0}^{P} N_i + P, \quad (1)$$

где N_i – количество пунктов остановок для одного *i*-го транспорта маршрута, P – число пересадок. Такой подсчет дает приблизительную оценку и может быть применим, например, в условиях города, где транспортные маршруты, как правило, имеют достаточно много пунктов остановок с короткими и сравнительно равными интервалами пути.

2.4. Расширение возможностей алгоритма и привязка маршрутов к географическим системам отсчета

В приведенном выше алгоритме длина маршрута оценивалась количеством пунктов остановок транспорта. Для более точной оценки пути D в алгоритме необходимо учитывать расстояние между остановками. В этом случае в конце каждой строки базы данных вводится дополнительный параметр, определяющий значение расстояния от данного пункта остановки до следующего. При этом для локализации программой данного пункта в строке необходимо инициализировать значение данного параметра, например, как показано ниже.

10. Гагаринский мост, больница Нефтяников, d377

11. м. Хатаи,_ d232

В процессе работы алгоритм просматривает значение *d* для каждой строки выделенного маршрута и определяет длину пройденного пути по выражению:

$$D = \sum_{i=0}^{P} \sum_{j=0}^{N_i} d_{i,j,j+1}, \quad (2)$$

где $d_{i,j,j-1}$ – расстояние между *j*-м и *(j+1)-м* пунктами остановок для *i*-го маршрута, N_i – количество пройденных пунктов остановок в *i*-ом номере маршрута, P – количество автобусных маршрутов, используемых в данном маршруте пути**.**

В выражении (2) значение $d_{i,j,j+1}$, определяющее расстояние между смежными пунктами *j* и *j+1* в *i*-ом маршруте, определяется предварительно с учетом координат этих пунктов (x_j, y_j), (x_{j+1}, y_{j+1}), а также траектории движения транспорта на этом промежутке. При этом необходимые значения координат могут быть определены путем использования, например, GPS навигаторов или же при помощи геоинформационных системных (ГИС) карт [9]. При этом если траектория движения транспорта между пунктами *j* и *j+1* не прямолинейна, то для определения значения длины $d_{i,j,j+1}$ могут либо использоваться ГИС, для которых встроены функции вычисления непрямолинейных путей, либо данный путь предварительно разбивается на набор K прямолинейных отрезков $d_{i,j,j+1,k}$:

$$d_{i,j,j+1} = \sum_{k=0}^{K} d_{i,j,j+1,k}. \quad (3)$$

При этом значение $d_{i,j,j+1k}$ между двумя точками A и B определяется как

$$d_{i,j,j+1,k} = d_{|A-B|} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}. \quad (4)$$

Если же значения *x* и *y* заданы в географических системах координат (широта и долгота), например, в случае использования геоинформационных систем [9], то расстояние $d_{i,j,j+1,k}$ между двумя заданными точками A и B определяется по выражению:

$$d_{i,j,j+1,k} = d_{|A-B|} = d_\alpha R, \quad (5)$$

где R – средний радиус земного шара ($R = 6372795$ *м*, d_α – расстояние между пунктами A, B, измеряемое в радианах и определяемое, например, по формуле гаверсинусов [10]:

$$d_\alpha = 2\arcsin\left(\sqrt{\sin^2\left(\frac{x_A - x_B}{2}\right) + \cos(x_A)\cos(x_B)\sin^2\left(\frac{y_A - y_B}{2}\right)}\right), \quad (6)$$

где x_A, x_B, y_A, y_B – широты и долготы в точках A и B. Значения $d_{i,j,j+1}$, определяемые по выражениям (3)–(6), могут вычисляться предварительно либо в процессе работы алгоритма.

При этом значения координат *x, y*, определяющие каждый отрезок $d_{i,j,j+1,k}$, могут быть добавлены в базу данных, например, как множества $X\{x_i\}$ и $Y\{y_i\}$, где x_1 и y_1 – значения координат пункта остановки, соответствующей данной строке:

10.парк Самеда Вургуна, железнодорожное управление, d462, X{40.380421, 40.378034, 40.376882},Y{49.850292, 49.852502, 49.853092}

11.Нефтяная Академия, м.28 Мая, d79, X{40.376588, 40.376301},Y{49.852138, 49.85129}

12.отель Empire, посольство Великобритании, d225, X{40.375615,40.37501,40.375402}, Y{49.851644, 49.849745, 49.849434}

Здесь значения координат заданы в географических координатах широт *x* и долгот *y*, полученных с помощью ГИС карт, выраженных в градусах. Соответственно, для применения выражений (5), (6), необходимо значения *x, y* перевести в радианы по выражениям:

$$x_{рад} = \frac{2\pi x_{град}}{180},\ y_{рад} = \frac{2\pi y_{град}}{180}, \tag{7}$$

где $x_{рад}, x_{град}, y_{рад}, y_{град}$ – угловые значения в радианах и градусах, $\pi = 3.141592654$.

Приведенный выше алгоритм также может быть использован применительно к другим видам транспорта (трамвай, метро, троллейбус и т.д.) как отдельно для каждого вида, так и с использованием их комбинаций. При этом в последнем случае необходимо учесть среднюю скорость передвижения каждого транспорта, а также среднее время задержки транспорта в пункте остановок для данного вида транспорта, значения которых могут быть указаны в первой строке рядом с номером маршрута, например, как показано ниже:

№104, v40, tz5

1. м. Дружда народов

2. пл. Украины, м. Ази Асланова, d420

В этом случае вместо значения кратчайшего пути *D* может быть использовано значение потраченного времени пройденного пути *T*, определяемого по выражению:

$$T = \sum_{i=1}^{P}\left(\frac{D_i}{v_i} + tz_i \times N_i\right), \tag{8}$$

где D_i – расстояния пути для *i*-го маршрута, определяемого по выражениям (2)–(3), v_i *и* tz_i – средние значения скорости и времени задержки на остановке для *i*-го маршрута.

При этом в зависимости от вида транспорта вместо номера маршрута могут быть указаны другие наименования, используемые для данного транспорта, например, для метро это может быть название конечной станции линии маршрута метро, а каждый маршрут линии метро будет представляться как отдельный маршрут:

Метро Старая крепость, v50, tz2

...

ст. Дружба народов, торговый цент Лачын, рынок 8 км.

ст. Нефтчиляр

ст. Кара Караева, магазин Орбита

ст. Улдуз, стадион Шяфа

ст. Нариманова, пос. Монтина, детская поликлиника №1

…

Если транспорт работает по расписанию, например, как в случае железнодорожного транспорта, авиатранспорта, междугородних автобусов, то тогда возможно добавить в каждой строке маршрута дополнительную запись времени отправки транспорта с данного пункта маршрута согласно расписанию. В этом случае необходимо дополнительно выполнять сопоставление времени отправки транспорта по расписанию со временем *T*, определенным по выражению (7), для предыдущих рассмотренных регулярно работающих видов транспорта.

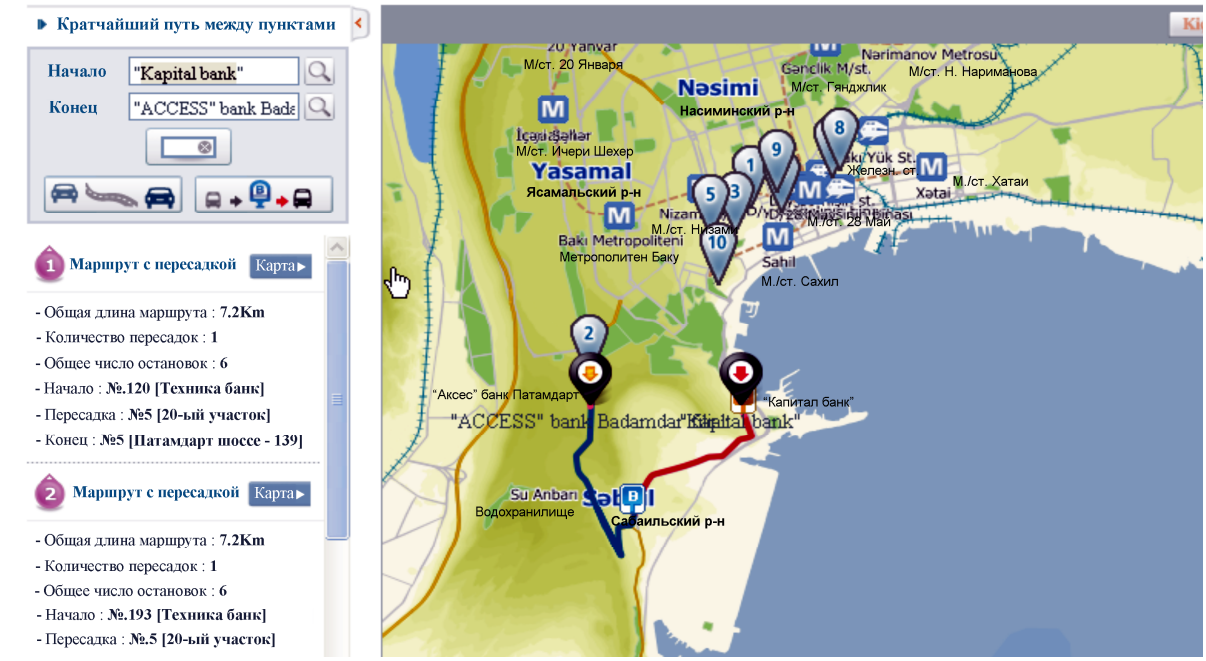


Рис. 6. Применение алгоритма для автобусного транспорта г. Баку

2.5. Методы ускорения работы алгоритма

Для ускорения работы алгоритма возможны как программные способы, так и структурные – путем оптимизации структуры базы данных.

2.5.1. Программные способы ускорения алгоритма

В качестве программных изменений возможно отметить следующие возможности.

Процесс поиска соответствий начального и конечного пункта в базе данных, выполняемый в 1-ом и 2-ом цикле, возможно ускорить до одного цикла. С этой целью в процессе просмотра базы данных уже в 1-ом цикле выполняется поиск сразу всех соответствий как начального, так и конечного пункта. Все найденные соответствия начального и конечного пункта сохраняются в отдельные 2 строки (1-я – для начального пункта, 2-я – для конечного пункта) с указанием номеров строк в базе данных. Например:

НП: Ж/Д Вокзал 15, 23, 55, 72, 117, 149

КП: Кинотеатр Дружба 19, 32, 47, 89, 92, 137

Далее работа алгоритма будет продолжена по аналогичной схеме, но уже с полученным списком номеров строк для найденных начальных и конечных пунктов маршрутов в базе. Ссылка на объект в основной базе будет выполняться по номеру строки. Таким образом, процесс просмотра базы данных уменьшится на порядок одного цикла.

Еще одна возможность ускорения процесса поиска кратчайшего маршрута – это ограничение числа выводимых маршрутов. К примеру, на рис. 4b список выводимых маршрутов в поле *3* ограничивается значением количества маршрутов, введенных в поле *5*. Таким же способом можно получить определение только одного кратчайшего маршрута. Для ускорения процесса возможно также дополнительно ограничить поиск маршрутов некоторыми заданными значениями D_{max} или T_{max}. В этом случае в процессе вычислений значений D или T при превышении этих ограничений рассматриваемый маршрут далее не рассматривается и удаляется из списка выбранных маршрутов.

Нужно отметить, что определение нескольких маршрутов имеет смысл как возможность получения нескольких альтернативных путей. Например, в том случае, когда данная система используется совместно с другими транспортно-дорожными системами, такими как системы

мониторинга дорожно-транспортных происшествий, мониторинга загруженности дорог (пробок) и т.д. В этом случае альтернативой кратчайшего пути может стать другой маршрут – менее короткий, но при этом менее загруженный.

Для ускорения получения результатов работы алгоритма можно также использовать возможности ускоренного ввода наименований объектов пользователем, реализованного с целью быстрого и корректного ввода наименований объектов, которое реализуется путем сравнения вводимого набора букв в списках полей ввода «*откуда*», «*куда*» (рис. 4b (1, 2)) и списках «*начало*», «*конец*» (рис. 6) с содержанием похожих по написанию объектов базы данных. По результатам сравнения выполняется сортировка схожих записей и выводятся наиболее близкие по написанию записи на просмотр пользователю (рис. 7) с возможностью дальнейшего быстрого отбора нужного наименования объекта ввода.

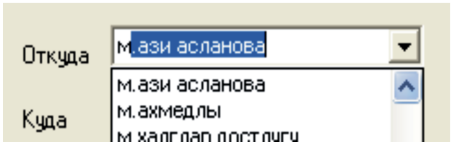


Рис. 7. Пример быстрого ввода наименования объекта пункта отправки в поле списка «откуда»

2.5.2. Оптимизация структуры базы данных

В качестве структурных дополнений возможно привести следующие изменения.

Для случая системы с привязкой к географическим системам координат возможно ускорить процесс выполнения алгоритма путем отделения наименований объектов базы данных от численных значений координатных данных, отображающих пункты остановок и траектории движения транспорта от одной остановки к другой. Последняя база и будет являться основной базой данных, с которой будет работать программа. Наименования объектов базы данных будут храниться в отдельной базе данных, например, в алфавитном порядке. Для привязки каждого объекта к пунктам остановки в базе объектов для каждого объекта будут также указаны географические координаты объекта. В этом случае работа первого и второго цикла алгоритма изменится следующим образом. В начале будет выполнен поиск начального и конечного объекта (НО, КО) в базе объектов с целью определения соответствующих объектам координат $X_{НО}$, $Y_{НО}$ и $X_{КО}$, $Y_{КО}$. Затем по координатам НО и КО будет выполняться поиск начальных и конечных пунктов остановки в основной базе путем выполнения сравнения координат НО и КО в некотором допустимо-заданном диапазоне ближайших к данным объектам координат пунктов остановок. Определение пунктов пересадки в третьем и четвертом цикле выполняется аналогично по описанной в алгоритме схеме, но путем сравнения уже не наименований объектов, а координат пунктов остановок двух маршрутов на предмет определения некоторой близости в допустимом заданном диапазоне расстояний между пунктами пересадки, например, установленным в 50 метров. При таком подходе скорость работы алгоритма увеличится, поскольку исключаются повторы наименования объектов в базе данных.

Еще один способ ускорения работы алгоритма – это преобразование структуры базы данных транспортных маршрутов в структуру графа. Преимущество ускорения работы программы в этом случае будет заключаться в том, что структура графа в отличие от базы данных с обычным перечислением маршрутов является более сжатой, что происходит за счет того, что фрагменты транспортных маршрутов не повторяются. В этом случае каждое ребро графа может соответствовать сразу нескольким маршрутам с разными номерами и типами транспорта. В базе данных в этом случае будут последовательно приводиться не сами маршруты транспорта, разделяемые пустыми строками, а фрагменты маршрутов транспорта, составляющие ребра графа, также разделяемые пустыми строками. Например:

№104:стр45, №95:стр78, №106:стр201

12. м. 28 мая, железнодорожный вокзал, Университет нефти и химии

13. дворец Республики, больница №4, ул. Басина

14. пл. Физули, шахматная школа, ул. Гуси Гаджиева, МИД

При этом для того чтобы сохранить целостность работы алгоритма при таком подходе, необходимо в заголовке каждого фрагмента ребра графа указать все номера маршрутов транспорта, имеющих в своем маршруте данный фрагмент пути. Кроме того, для каждого номера транспорта должен быть также указан номер строки базы данных (стр:) для нового, смежного с данным ребром, ребра, на котором происходит продолжение маршрута для данного номера и типа транспорта. При этом в процедуру определения пункта пересадки, выполняемого в 3-ем и 4-ом цикле, будет добавлен дополнительный цикл, который будет перебирать все номера маршрутов, указанных в оглавлении ребра, и выполнять данную процедуру аналогичным образом для всех указанных в заголовке номеров и типов маршрутов. Переход от одного фрагмента маршрута к другому (одного ребра к другому) будет происходить по номеру строки, указанному для каждого номера маршрута в заголовке ребра графа. При этом если вершина данного ребра графа не имеет смежных ребер или если маршрут транспорта на данном ребре графа завершается, то номер перехода строки в заголовке графа для данного транспорта не указывается. Соответственно, отсутствие номера строки перехода в заголовке будет восприниматься программой как завершение маршрута пути для данного номера транспорта.

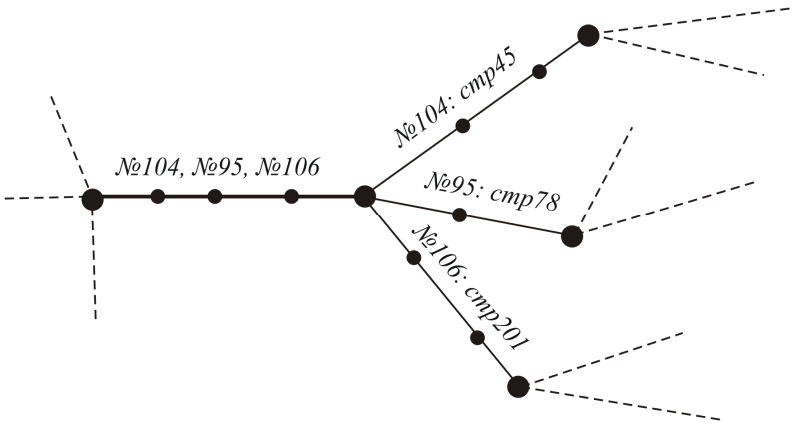


Рис. 8. Пример фрагмента графа

3. Заключение

Результаты работы алгоритма могут быть представлены на вывод как в текстовом виде (рис. 4b (4)), так и в графическом. На рис. 6 результаты работы алгоритма привязаны к информационной электронной карте города Баку, на которой вырисовывается полученный маршрут с указанием начальных и конечных пунктов назначения и сменой маршрутов, отличающихся цветом. При этом при выполнении привязки работы алгоритма к геоинформационным картам необходимо, чтобы объекты маршрутов, а также их данные, используемые в базе данных, были соразмерны наименованиям и данным объектов, используемыми в информационных картах. Описанный алгоритм определения кратчайшего пути применительно к автобусным маршрутам г. Баку (рис. 4b) был представлен в 2008-ом году на рассмотрение в Министерство Транспорта Азербайджанской Республики. В результате данный алгоритм был включен в общий пакет реализации систем интеллектуального управления транспорта г. Баку. В настоящее время реализация алгоритма определения кратчайшего маршрута с использованием одной пересадки представлена на сайте центра интеллектуального управления транспорта Азербайджанской Республики [11] (рис. 6), а также на терминалах, установленных на автобусных остановках на центральных улицах города.

Литература

1. *Кормен Т. Х, Лейзерсон Ч. И., Ривест Р. Л., Штайн К.* Алгоритмы: построение и анализ = Introduction to Algorithms. 2-е изд. М.: Вильямс, 2006.
2. Diestel R. Graph Theory, Electronic Edition. NY: Springer-Verlag, 2005.
3. *Ананий В. Левитин.* Алгоритмы: введение в разработку и анализ. М.: Вильямс, 2006.
4. *Алексеев В. А., Таланов В. А.* Графы. Модели вычислений. Структуры данных. Нижний Новгород: Издательство Нижегородского гос. университета, 2005.
5. *M. Dorigo, V. Maniezzo, A. Colorni.* Ant System: Optimization by a Colony of Cooperating Agents // IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996. 26 (1): P. 29–41.
6. *Кажаров А. А., Курейчик В. М.* Муравьиные алгоритмы для решения транспортных задач. Известия Российской академии наук. Теория и системы управления. 2010. № 1. С. 32–45.
7. *Ватутин Э. И., Титов В. С.* Анализ результатов применения алгоритма муравьиной колонии в задаче поиска пути в графе при наличии ограничений // Известия Южного федерального университета. Технические науки. 2014. № 12. С. 111–120.
8. *Cauvery N. K., Viswanatha K. V.* Routing in Dynamic Network using Ants and Genetic Algorithm // International Journal of Computer Science and Network Security, Vol. 9. No. 3, 2000. P. 36–41.
9. *Longley P. A., Goodchild M. F., Maguire D. J., Rhind D. W.* Geographic Information Systems and Science. John Wiley and Sons, 2nd edn. 2005.
10. *Gade, Kenneth* A non-singular horizontal position representation // The Journal of Navigation, 63, 2010, P. 395–417.
11. Центр интеллектуального управления транспорта Азербайджанской республики. (Nəqliyyat İntellektual İdarəetmə Mərkəzi) `http://www.niim.az/marsrut-secimi`

Статья поступила в редакцию 09.11.2015;
переработанный вариант – 30.11.2015.

Гейдаров Полад Шахмалы оглы
к.т.н., доцент, институт системного управления НАН Азербайджана, e-mail: `plbaku2010@gmail.com`

P. Sh.Geidarov

Determining the shortest route with the use of public transport

This paper describes an algorithm for determining the shortest transport route using public transport. Basic algorithm for determining the shortest route without using geo-referencing and with the possibility of one change is presented. The structure of the database enabling to use several nearby objects as a bus stop is considered. Based on this basic algorithm, algorithm for determining the shortest route with two or more bus changes is implemented. The possibility of expanding the functionality of the algorithm applied to different kinds of transport and geo-referencing is considered. It also considers the program and structural ways of accelerating the algorithm.

Keywords: the shortest route, urban transport, public transport, transport in Baku, Intelligent Transport Management Center.