УДК 004.421

П. Ш. Гейдаров, канд. техн. наук, доцент,
Институт Системного Управления НАН Азербайджана, г. Баку

Алгоритм кратчайшего маршрута на основе выделенного набора маршрутов

Рассмотрен алгоритм определения кратчайшего маршрута на основе существующего набора маршрутов. Алгоритм рассматривался применительно к общественному транспорту путем последовательного усложнения и расширения возможностей алгоритма. Приведен базовый алгоритм без географической привязки к местности и с возможностью одной пересадки. Предложено описание структуры базы данных, позволяющей в качестве наименований пунктов остановок гибко использовать названия различного числа объектов, имеющих соответствие с пунктами остановок. На основе базового алгоритма рассмотрен алгоритм с двумя или несколькими пересадками, а также приведено описание возможностей расширения функциональности алгоритма применительно к разным видам транспорта и с привязкой к географическим координатам местности. Также дано описание программных и структурных способов ускорения работы алгоритма.

Ключевые слова: *кратчайший маршрут, городской транспорт, общественный транспорт, транспорт в Баку, Центр интеллектуального управления транспортом*

Введение

Одной из актуальных задач является задача определения кратчайшего 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. В отличие от классической постановки задачи кратчайшего маршрута, особенность решения данной задачи заключается в наличии изначально известного набора маршрутов, на основе которых и требуется определение кратчайшего маршрута. Решение данной задачи является актуальным на фоне современных темпов и масштабов развития городов, определяемых такими событиями, как строительство новых зданий, высоток, парков, улиц, проспектов; переименование названий улиц, учреждений; замена, объединение или перемещение одних организаций в другие; а также сокращение, переименование или введение новых транспортных линий. Все это зачастую приводит к сложностям ориентирования на местности. Например, 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 сферы.

1. Постановка задачи

Нужно отметить, что для задач определения кратчайшего пути существуют различные алгоритмы, которые в целом можно подразделить на два типа: алгоритмы перебора, выдающие оптимальные результаты [1—4], и эвристические алгоритмы, определяющие приближенно неточные результаты (например, муравьиные, генетические алгоритмы... [5—12]). Применение последних обосновано в тех случаях, когда задача является нерешаемой иными способами или требует просмотра слишком большого объема информации в ограниченный период времени. В качестве же примера алгоритма перебора для задачи определения кратчайшего пути, можно было бы привести используемый в системах GPS-навигации алгоритм перебора Дейкстра [1, 4], определяющий маршрут на основе перебора вершин графа, предварительно созданных 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] имеют универсальный характер и применимы к разным задачам, но при этом применительно к отдельным задачам, и в частности в 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 числа пересадок для полученного кратчайшего маршрута может оказаться чрезмерно много, что будет неудобно и неестественно для практического применения этих маршрутов. Кроме того, в ряде случаев универсальные алгоритмы перебора [1, 4] будут более медлительными, чем предлагаемый алгоритм. Связано это с тем, что алгоритм перебора графов [4] последовательно перебирает все смежные вершины графа, двигаясь от одной вершины к другой, таким образом до определения кратчайшего маршрута

будет охвачено значительное число вершин графа (рис. 1, *а*). Тогда как в предлагаемом алгоритме благодаря наличию базы маршрутов, а также начального и конечного пунктов {НП, КП} число рассмотренных вершин может быть значительно меньше, поскольку поиск ведется на основе тех маршрутов, в которых имеются эти пункты. Последнее особо заметно при наличии малого числа маршрутов и пересадок между начальными и конечными пунктами кратчайшего маршрута (рис. 1, *а*, *б*).

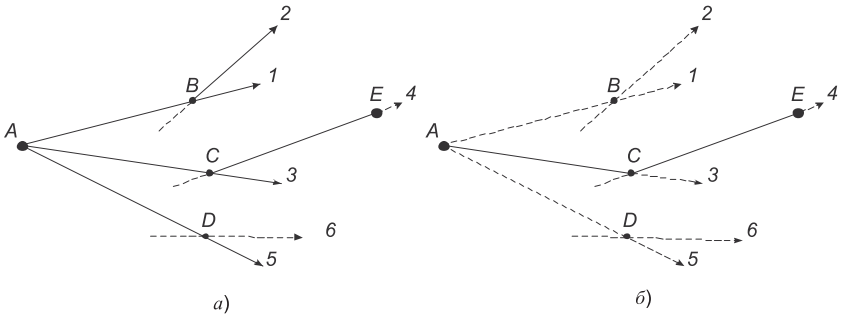

Рис. 1. Пример графа, образованного из шести прямолинейных маршрутов с просмотренными (сплошные линии) ребрами графа:
а — по алгоритму Дейкстра; *б* — по рассматриваемому алгоритму

На рис. 1 приведен пример графа с числом рассмотренных ребер (сплошные линии графа) по алгоритму Дейкстра (рис. 1, *а*), и по рассматриваемому алгоритму (рис. 1, *б*), для задачи определения маршрута от точки *A* до *E* на графе, построенном из шести 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 пять шагов для достижения точки *E*, тогда как в рассматриваемом алгоритме — два шага. Здесь нужно отметить, что при увеличении числа пунктов пересадок в кратчайшем маршруте, а также увеличение числа маршрутов, проходящих через начальный и конечный пункты, число рассматриваемых вершин графа также будет увеличиваться и соответственно скорость работы предлагаемого алгоритма будет уменьшаться.

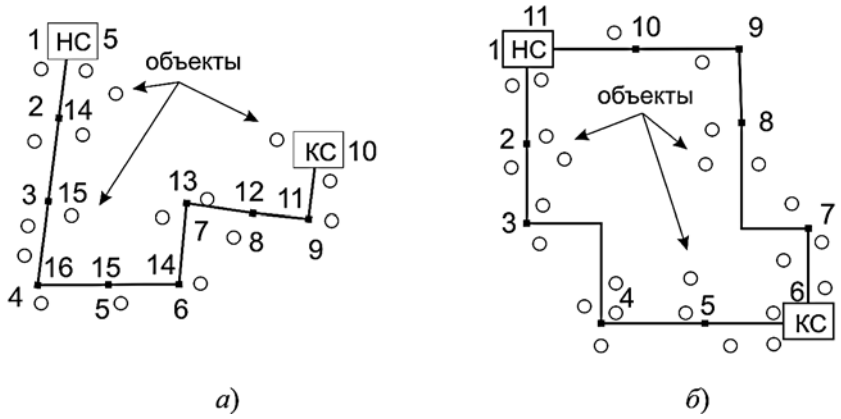

Рис. 2. Незамкнутый маршрут (*а*), замкнутый маршрут (*б*): НС, КС — начальная и конечная станция (остановка) маршрута

2. Определение структуры базы данных и алгоритм определения кратчайшего маршрута с одной пересадкой, без привязки маршрутов к системам отсчета

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

Для работы алгоритма предварительно создается структура базы данных, представляющая собой 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). Структура базы данных организована следующим образом. Все пункты маршрутов транспортных средств расположены построчно друг за другом. Каждый маршрут начинается с номера маршрута транспорта. Название каждого пункта остановки состоит из наименований различных объектов, располагающихся поблизости данной остановки маршрута. При этом все названия объектов и улиц, касательные одной остановки транспорта, расположены на одной строке и разделены 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, добавляя новые или удаляя старые объекты.

Пример 1. Пример фрагмента базы данных
№ 104
1. М. Дружба народов
2. Пл. Украины, м. Ази Асланова
3. Радиозавод
...
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

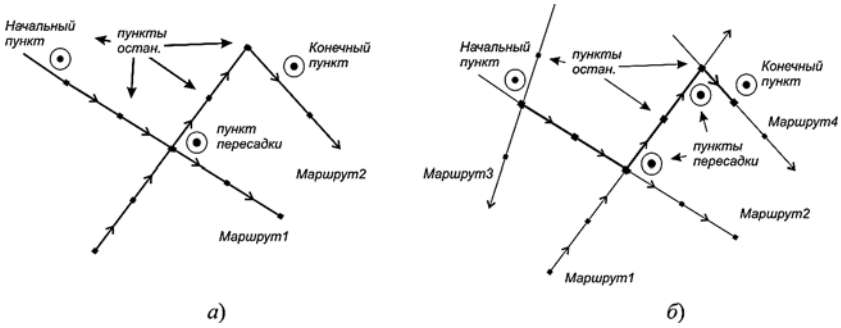

Рис. 3. Пересекаемые (*а*) и непересекаемые (*б*) маршруты

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. 2, *б*), так и по незамкнутым маршрутам (рис. 2, *а*). В случае замкнутого маршрута пункты остановок маршрута перечислены в тексте только один раз сверху вниз по направлению движения транспорта (пример 1), в случае незамкнутого маршрута для целостности работы алгоритма пункты остановок маршрута перечислены дважды, сначала в направлении движения от начальной станции (НС) к конечной станции (КС), затем в обратном направлении от КС до НС.

Выполнение алгоритма состоит из четырех циклов (*алгоритм 1*). При этом 1-й и 2-й циклы выполняют поиск начальных и конечных пунктов назначения в базе данных, а 3-й и 4-й циклы определяют пункт пересадки, и, соответственно, выполняются только в том случае, когда начальный и конечные пункты назначения в первых двух циклах определяются на разных маршрутах транспорта.

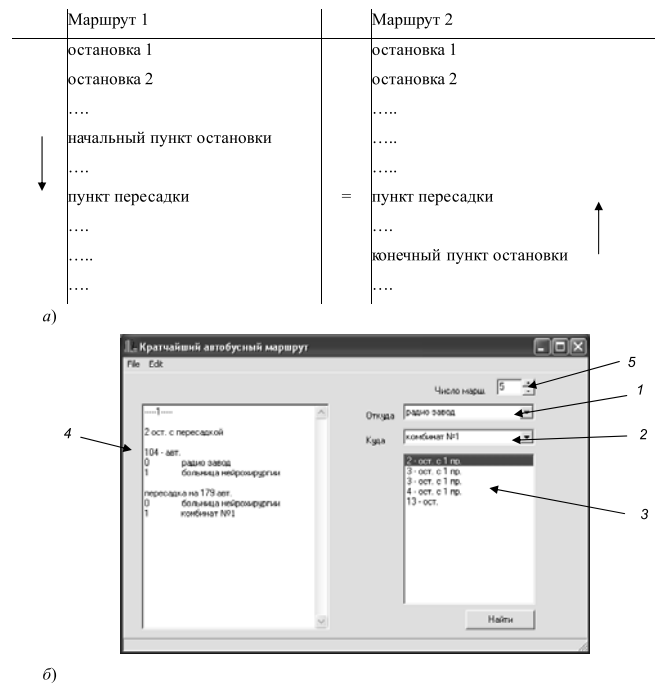

Рис. 4. Схема работы 3-го и 4-го циклов (*а*) и программный модуль алгоритма определения кратчайшего маршрута с одной пересадкой без привязки к географическим координатам, в применении к автобусным маршрутам г. Баку (*б*)

Алгоритм работает следующим образом. После того как пользователь вводит начальный и конечный пункты назначения на рис. 4, *б* текстовые поля *1* и *2* с наименованиями: "*откуда*", "*куда*" (выполняется работа 1-го 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 2-й цикл, где последовательно просматриваются строки в целях определения второго пункта назначения ("*куда*"), начиная с самой верхней строки базы данных. Если второй пункт назначения определяется в пределах этого же маршрута, 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. 3, *а*) 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-й цикл, алгоритм 1*), начиная с определенной выше строки пункта назначения, и в обратном направлении для второго маршрута (*4-й цикл, алгоритм 1*), начиная со строки конечного пункта назначения (рис. 4, *а*). В случае если пункт пересадки определяется, то весь маршрут отдельно запоминается как один из возможных вариантов пути.

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 ближайшие к пунктам остановок.

Если пункт пересечения не определяется, то это означает, что маршруты не пересекаются (рис. 3, *б*). В этом случае продолжается процесс выполнения 2-го 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 1-му циклу и продолжает поиск нового начального пункта со строки последнего определенного начального пункта. Все полученные маршруты, включая номера маршрутов, наименования пунктов остановок, а также место пересадки маршрутов сохраняются и выводятся на экран (см., например, рис. 4, *б* (*3*, *4*)). При этом полученные маршруты являются вариантами маршрутов без пересадки или с одной пересадкой.

Алгоритм 1. Определения кратчайшего маршрута с одной пересадкой

Блок А — определение строк i, j для НП и КП.

Цикл 1, определение строки НП.

1. *Начальные данные: k = 0, НП, КП, база всех маршрутов (N — число строк в базе).*
2. *k = k + 1. Переход на k + 1-ую строку в базе маршрутов, i = k.*
3. *Если k = N: переход в 14.*
4. *Если НП = {Объекту строки i}, то k = 0, переход в 5 (цикл 2), если нет, переход в 2.*

Цикл 2, определение строки КП

5. *k = k + 1. Переход на k + 1-ую строку в базе маршрутов, j = k.*
6. *Если k = N: переход в 2.*
7. *Если КП = {Объекту строки j} переход в 8 (цикл 3), если нет, переход в 5.*

Блок Б — определение пункта пересадки.

Циклы 3, 4.

8. *n = j, k = i.*
9. *k = k + 1.*
10. *Если строка пуста, k = "", переход в 12 (цикл 4).*
11. *Если {Объект строки k} = {Объект строки n}, то k — строка пересадки (ПР = k) сохраняем, вычисляем длину маршрута, и выполняем переход в 5, если нет, то переход на 9.*
12. *n = n − 1, k = i.*
13. *Если в строке n есть "номер маршрута или знак №", то k = j переход в 5 (цикл 2).*
14. *Переход в 9.*

Блок С — обработка и вывод результатов алгоритма.

15. *Сортируем полученные маршруты по возрастанию длины маршрута и выводим пользователю.*
16. *Завершение работы алгоритма.*

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 (*алгоритм 2*) выполняется последовательное переназначение каждой *k*-й 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 цикл программы перенаправляется в начало 1-го цикла программы и по аналогичной схеме повторяется поиск пункта пересадки. Полученные маршруты сохраняются. Далее по такой же схеме выполняется обратная задача. Начальный пункт (НП) остается заданным, а конечный пункт ($КП_n^{(j)}$) меняется путем перебора остановок второго выбранного *j*-го маршрута. 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. Например, на рис. 5 кратчайший маршрут будет: $\{НП, ПР_{0,2}^{(i,j)}, КП_2^{(j)}, КП\}$.

Если при выполнении основного алгоритма кратчайший маршрут между пунктами $\{НП_k^{(i)}, КП\}$ или $\{НП, КП_n^{(j)}\}$ определяется без пересадки, то мы получаем маршрут с одной пересадкой по схеме — $\{НП, НП_k^{(i)}, КП\}$ или $\{НП, КП_n^{(j)}, КП\}$. Если же при назначении пунктов $НП_k^{(i)}$, $КП_n^{(j)}$ выполняется ус-

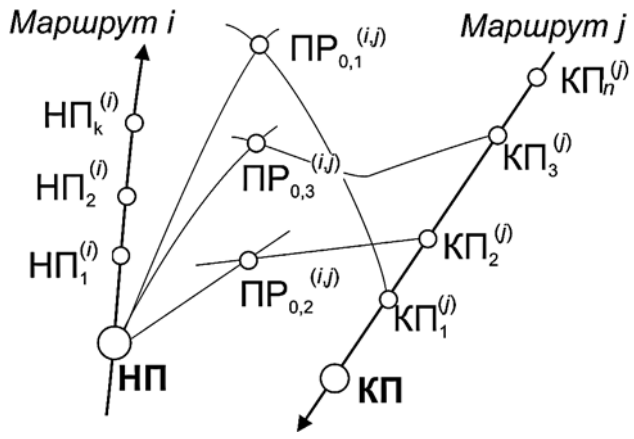

Рис. 5. Пример схемы определения маршрутов с двумя пересадками между пунктами НП и КП, с перебором пунктов КП на транспортном маршруте *j*

ловие $НП_k^{(i)}$ = 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 и определение кратчайшего маршрута с одной пересадкой и без пересадки.

Алгоритм 2. Определение кратчайшего маршрута с двумя пересадками.

1. *Блок А (алгоритм 1): определяем номера строк i, j для НП и КП.*
   Цикл 1.
2. *k = i − 1.*
3. *k = k + 1, — выбираем новый начальный пункт, $НП_k$ = Объект строки k.*
4. *Если строка k пуста: k = "", то переходим в 8, цикл 2.*
5. *Переход в 13, (блок Б) с переменными $НП_k$ и КП.*
6. *Сохраняем общий маршрут {НП, $НП_k$, ПР, КП}, и вычисляем длину маршрута.*
7. *Переход в 3.*

   Цикл 2.
8. *k = j + 1.*
9. *k = k — 1, — выбираем новый конечный пункт, $КП_k$ = Объект строки k.*
10. *Если в строке k есть "номер маршрута или знак № ", то переход в 1, Блок А, если нет, то выполняем переход в 13, Блок Б с переменными НП и $КП_k$.*
11. *Сохраняем общий маршрут {НП, ПР, $КП_k$, КП}, вычисляем длину маршрута.*
12. *Переход в 9.*

13. *Блок Б (алгоритм 1): определяем пункт пересадки (ПР), возвращаемся в пункт вызова: 5, 9.*

14. *Блок С (алгоритм 1): сортируем по длине маршрута и выводим пользователю полученные маршруты.*
15. *Завершение алгоритма.*

Аналогичным образом на базе основного *алгоритма 1* путем выполнения дополнительных циклов можно получить кратчайшие маршруты с тремя и более числом пересадок, например {НП, $НП_k^{(i)}$, $ПР_{k,n}^{(i,j)}$, $КП_n^{(j)}$, КП}. Нужно сказать, что маршруты с большим числом пересадок создают дополнительные неудобства для самих пассажиров и при этом увеличивают длительность и скорость работы алгоритма. Обычно в условиях большого города достаточны маршруты с числом пересадок 1—3.

В процессе работы алгоритма или после выполнения поиска списка маршрутов, выполняется обработка (*алгоритм 1, Блок С*) полученных маршрутов. Определяется длина маршрутов, выполняется перераспределение (сортировка) полученных маршрутов в порядке увеличения их длины . Например, на рис. 4, *б* самый верхний маршрут в списке *3* соответствует самому кратчайшему маршруту. При этом в наиболее простом применении длина маршрута может оцениваться числом остановок до и после пересадки (рис. 4, *б*), с учетом также числа пересадок, например, по выражению:

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

где N_i — число пунктов остановок для *i*-го 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.

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

В приведенном выше алгоритме длина маршрута оценивалась числом пунктов остановок транспорта. Для более точной оценки пути D в алгоритме необходимо учитывать расстояние между остановками. В этом случае в каждой строке, в конце 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, буквой *d*.

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

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

где $d_{i,j,j-1}$ — расстояние между *j*-й и $j + 1$-й пунктами остановок для *i*-го маршрута; N_i — число пройденных пунктов остановок в *i*-м маршруте; P — число 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*-м маршруте, определяется предварительно с учетом координат этих пунктов (x_j, y_j), (x_{j+1}, y_{j+1}), а также траектории движения транспорта на данном промежутке. При этом необходимые значения координат могут быть определены путем использования, например, GPS-навигаторов, или же с помощью геоинформационных системных (ГИС) карт [13]. Если траектория движения транспорта между пунктами j и $j + 1$ не прямолинейна, то для определения значения длины $d_{i,j,j+1}$ могут быть либо использованы 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 $d_{i,\,j,\,j+1,\,k}$ между двумя точками A и B определится 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 [13], то расстояние $d_{i,\,j,\,j+1,\,k}$ между двумя заданными точками A и B определится по выражению:

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

где R — средний радиус земного шара (R = 6 372 795 м, d_α — расстояние между пунктами A, B, измеряемое в радианах, и определяемое, например, по формуле гаверсинусов [14]:

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

где x_A, x_B, y_A, y_B — значения широты и долготы в точках A и B. Значения $d_{i,\,j,\,j+1}$, определяемые по выражениям (3—6), можно вычислять предварительно, либо в процессе работы алгоритма, последнее может быть необходимо в случае необходимости вырисовывания траектории пути. 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 широты x и долготы y, полученных путем ГИС-карт и выраженные в градусах. Соответственно для применения выражений (5, 6) необходимо значения x, y перевести в радианы, по выражениям

$$x_{\text{рад}} = \frac{2\pi x_{\text{град}}}{180};\ y_{\text{рад}} = \frac{2\pi y_{\text{град}}}{180}, \quad (7)$$

где $x_{\text{рад}}$, $x_{\text{град}}$, $y_{\text{рад}}$, $y_{\text{град}}$ — угловые значения в радианах и градусах, π = 3,141592654.

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

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

где D_i — расстояние пути для i-го маршрута, определяемое 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 транспорта 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 центр 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 возможно добавить в каждой строке маршрута транспорта дополнительную запись времени отправки с данного пункта маршрута согласно расписанию. В этом случае необходимо дополнительно выполнять проверку сопоставления времени отправки транспорта по расписанию, со временем T, определенным по выражению (8), для предыдущих рассмотренных регулярно работающих видов транспорта.

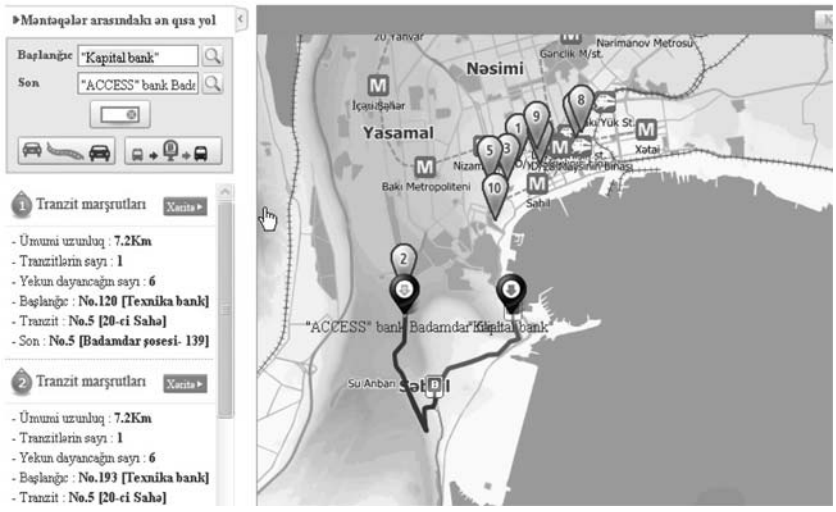

Рис. 6. Применение алгоритма для автобусного транспорта г. Баку

5. Методы ускорения работы алгоритма

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

Программные способы ускорения алгоритма. В качестве программных изменений возможно отметить следующие возможности.

Процесс поиска соответствий начального и конечного пунктов в базе данных, выполняемых в 1-м и 2-м цикле, можно ускорить до выполнения одного цикла. С этой целью в процессе просмотра базы данных уже в 1-м цикле выполняется поиск сразу всех соответствий как начального, так и конечного пункта. Все найденные соответствия начального и конечного пунктов сохраняются в отдельные две строки (первая для начального пункта, вторая 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. Например, на рис. 4, *б* список выводимых маршрутов в поле *3* ограничивается числом маршрутов, введенных в поле *5*. Таким же способом можно получить определение только одного кратчайшего маршрута. Для ускорения процесса можно 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 будет далее не рассматриваться и удаляться из списка выбранных маршрутов.

Нужно отметить, что определение нескольких маршрутов имеет смысл как возможность получения нескольких альтернативных путей. Например, в 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 также в целях быстрого и корректного ввода наименований объектов путем сравнения вводимого набора букв в списках полей ввода "*откуда*", "*куда*" (см. рис. 4, *б* (*1*, *2*)) и списках "*başlanğıc*", "*son*" (рис. 6) с содержанием похожих по написанию объектов базы данных. По результатам сравнения выполняется также сортировка схожих записей и выводятся наиболее близкие по написанию записи на просмотр пользователю (рис. 7) с возможностью дальнейшего быстрого отбора нужного наименования объекта ввода.

Оптимизация структуры базы данных. В качестве структурных дополнений возможно привести следующие изменения.

Для случая системы с привязкой к географическим системам координат можно 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

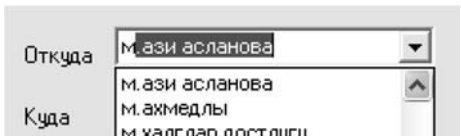

Рис. 7. Пример быстрого ввода наименования объекта пункта отправки в поле списка "Откуда"

работа 1-го и 2-го циклов алгоритма изменится следующим образом. В начале будет выполнен поиск начального и конечного объектов (НО, КО) в базе объектов в целях 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 3-м и 4-м циклах 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 расстояния между пунктами пересадки, например, установленным в 50 м. 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. Ускорение работы алгоритма в этом случае будет происходить за счет того, что структура графа, в отличие от базы данных с обычным перечислением маршрутов, является более сжатой, это происходит за счет того что фрагменты транспортных маршрутов не повторяются. В этом случае каждому ребру графа может соответствовать сразу несколько маршрутов с разными номерами и типами транспорта. В базе данных в этом случае будут последовательно приводиться не сами маршруты транспорта, а фрагменты маршрутов транспорта (рис. 8), 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, 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 процедуре определения пункта пересадки, выполняемого в 3-м и 4-м циклах, добавлен дополнительный цикл, который будет перебирать все номера маршрутов, указанных в оглавлении ребра, и соответственно выполнять данную процедуру аналогичным же 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 также для каждого номера маршрута в заголовке ребра графа. 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 приниматься алгоритмом, как завершение маршрута пути для данного номера транспорта.

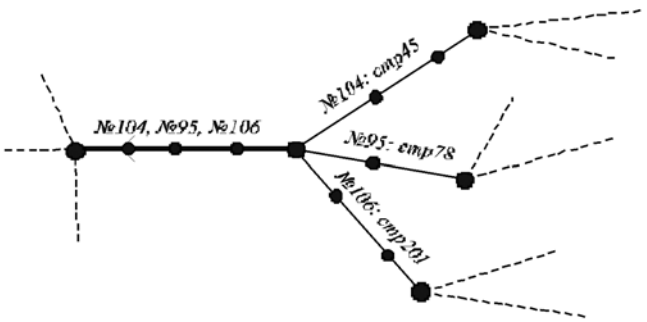

Рис. 8. Пример фрагмента графа

Заключение

Результаты работы алгоритма могут быть представлены на вывод как в текстовом виде (см. рис. 4, *б* (*4*)), так и в графическом виде. На рис. 6 результаты работы алгоритма привязаны к информационной электронной карте г. Баку, на которой вырисовывается полученный маршрут с указанием начальных и конечных пунктов назначения и сменой маршрутов, отличающихся цветами. При выполнении привязки работы алгоритма к геоинформационным картам необходимо, чтобы объекты маршрутов, а также их данные, используемые в базе данных, были соразмерны с наименованиями и данными объектов в информационных картах. Описанный алгоритм определения кратчайшего пути применительно к автобусным маршрутам г. Баку (см. рис. 4, *б*) был представлен в 2008 г. 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 [15] (см. рис. 6), а также на терминалах, установленных на автобусных остановках на центральных улицах города.

Предложенный алгоритм может быть использован не только применительно к задаче определения маршрутов общественного транспорта, но и для лю-

бых других задач, где имеются предварительно выделенные наборы маршрутов. Кроме того, алгоритм может быть использован и в задачах определения кратчайшего пути, где нет выделенных наборов маршрутов. В этом случае будет необходимо предварительно выбрать или создать набор маршрутов (например, случайным образом), к которым затем будет применен алгоритм. При таком подходе близость полученных маршрутов к кратчайшему маршруту, при, будет определяться и зависеть от способа и подходов в выборе набора маршрутов.

Список литературы

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. 1296 с.
2. **Кузнецов О. П., Адельсон-Вельский Г. М.** Дискретная математика для инженера. М.: Энергоатомиздат, 1988. 480 с.
3. **Оре О.** Теория графов. М.: Наука, 1980. 336 с.
4. **Dijkstra E. W.** A note on two problems in connexion with graphs // Numerische Mathematik. 1959. Vol. 1. P. 269—271
5. **Moyson F., Manderick B.** The collective behaviour of Ants: an Example of Self-Organization in Massive Parallelism // Actes de AAAI Spring Symposium on Parallel Models of Intelligence, Stanford, Californie, 1988.
6. **Dorigo M., Maniezzo V., Colorni A.** Ant System: Optimization by a Colony of Cooperating Agents // IEEE Transactions on Systems, Man, and Cybernetics-Part B. 1996. 26 (1): P. 29—41.
7. **Cauvery N. K., Viswanatha K. V.** Routing in Dynamic Network using Ants and Genetic Algorithm // International Journal of Computer Science and Network Security. 2000. Vol. 9, no. 3. P. 36—41.
8. **Казаков П. В.** Использование дифференциальной эволюции при определении множества Парето генетическими алгоритмами многокритериальной оптимизации // Информационные технологии. 2015. Т. 21, № 2. С. 109—116.
9. **Аллилуева Н. В.** Применение генетических алгоритмов решения задачи маршрутизации беспилотных летательных аппаратов // Вопросы радиоэлектроники. 2016. № 1. С. 47—53.
10. **Лунин Д. В., Скворцов С. В.** Разработка параллельного генетического алгоритма для решения задачи коммивояжера на платформе cuda // Системы управления и информационные технологии. 2015. Т. 60, № 2. С. 50—55.
11. **Новиков А. К.** Применение муравьиного алгоритма в задачах маршрутизации транспорта // Молодежный научно-технический вестник. 2015. № 11. С. 32.
12. **Кирилина А. С., Тарасова Л. Г., Казакова А. Е., Видная К. А.** Применение роевых алгоритмов в решении транспортно-экспедиционных задач // Актуальные направления научных исследований XXI века: теория и практика. 2015. Т. 3. № 7—4 (18—4). С. 296—300.
13. **Longley P. A., Goodchild M. F., Maguire D. J., Rhind D. W.** Geographic Information Systems and Science. 2nd edn. John Wiley and Sons, 2005. 454 p.
14. **Gade K.** A non-singular horizontal position representation // The Journal of Navigation. 2010. Vol. 63. P. 395—417.
15. **Центр** Интеллектуального Управления Транспорта Азербайджанской Республики. (Nəqliyyat İntellektual İdarəetmə Mərkəzi) http://www.niim.az/marsrut-secimi

P. Sh. Geidarov, PhD, Associate Professor,
Institute of Azerbaijan National Academy of Sciences of the System Management

Algorithm for the Shortest Route Based on the Selected Set of Routes

This paper describes an algorithm for determining the shortest transport route based on the selected set of routes. The algorithm is considere in relation to public transport by successive complications and extension algorithm features. Initially described an algorithm for determining the shortest route without the use of geographical coordinates of the area and with one change of transport. On the basis of the underlying algorithm examines the algorithm with two or more transfers. Also describe the possibilities of expanding the functionality of the algorithm, applied to different transport, with the possibilities of geo-referenced location. Also give a description of software and structural means to speed up the algorithm.

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

References

1. **Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. T.** *Algoritmy: postroenie i analiz = Introduction to Algorithms*, 2nd edn. Moscow: Vilyams, 2006, 1296 p.
2. **Kuznecov O. P., Adelson-Velskiy G. M.** *Diskretnaya matematika dlya injenera*, Moscow: Energoatomizdat, 1988, 480 p.
3. **Ore O.** *Teoriya grafov*, Moscow: Nauka, 1980, 336 p.
4. **Dijkstra E. W.** A note on two problems in connexion with graphs, *Numerische Mathematik*, 1959, vol. 1, pp. 269—271.
5. **Moyson F., Manderick B.** The collective behaviour of Ants: an Example of Self-Organization in Massive Parallelism, *Actes de AAAI Spring Symposium on Parallel Models of Intelligence, Stanford, Californie*, 1988.
6. **Dorigo M., Maniezzo V., Colorni A.** Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 1996, vol. 26 (1), pp. 29—41.
7. **Cauvery N. K., Viswanatha K. V.** Routing in Dynamic Network using Ants and Genetic Algorithm, *International Journal of Computer Science and Network Security*, 2000, vol. 9, no. 3, pp. 36—41.
8. **Kazakov P. V.** Ispolzovanie differencialnoy evolyucii pri opredelenii mnojestva Pareto geneticheskimi algoritmami mnogokriterialnoy optimizacii, *Informacionnye tekhnologii*, 2015, vol. 21, no. 2, pp. 109—116.
9. **Allilueva N. V.** Primenenie geneticheskih algoritmov resheniya zadachi marshrutizacii bespilotnyh letatelnyh apparatov, *Voprosy radioelektroniki,* 2016, no. 1, pp. 47—53.
10. **Lunin D. V., Skvorcov S. V.** Razrabotka parallelnogo geneticheskogo algoritma dlya resheniya zadachi kommivoyajera na platforme cuda, *Sistemy upravleniya i informacionnye tekhnologii*, 2015, vol. 60, no. 2, pp. 50—55.
11. **Novikov A. K.** Primenenie muravinogo algoritma v zadachah marshrutizacii transporta, *Molodejnyy nauchno-tekhnicheskiy vestnik*, 2015, no. 11, pp. 32.
12. **Kirilina A. S., Tarasova L. G., Kazakova A. E., Vidnaya K. A.** Primenenie roevyh algoritmov v reshenii transportno-ekspedicionnyh zadach, *Aktualnye napravleniya nauchnih issledovaniy XXI veka: teoriya i praktika*, 2015, vol. 3, no. 7—4 (18—4), pp. 296—300.
13. **Longley P. A., Goodchild M. F., Maguire D. J., Rhind D. W.** *Geographic Information, Systems and Science*. John Wiley and Sons, 2nd edn. 2005, 454 p.
14. **Gade K.** A non-singular horizontal position representation, *The Journal of Navigation*, vol. 63, 2010, pp. 395—417.
15. **Centr** *Intellektualnogo Upravleniya* Transporta Azerbaydjanskoy Respubliki. (Nəqliyyat İntellektual İdarəetmə Mərkəzi), URL: http:// www.niim.az/marsrut-secimi