ACCOMMODATING USER PREFERENCES IN THE OPTIMIZATION OF PUBLIC TRANSPORT TRAVEL

QIUJIN WU, JOANNA HARTLEY

School of Computing and Technology, The Nottingham Trent University,
Burton Street, Nottingham, NG1 4BU, U.K.
E-mail: qiujin.wu@ntu.ac.uk

Abstract: Traffic congestion is becoming a serious problem in more and more modern cities. Encouraging more private-vehicle drivers to use public transportation is one of the most effective and economical ways to reduce the ever increasing congestion problem on the streets (Hartley and Bargiela, 2001). To make public transport services more attractive and competitive, providing travellers with individual travel advice for journeys becomes crucial. However, with the massive and complex network of a modern city, finding one or several suitable route(s) according to user preferences from one place to another is not a simple task. In this paper, the author presents and compares two solutions to accommodate public transport users' preferences. The first approach is using three different single-purpose shortest path algorithms to accommodate three different user preferences. And the second approach is using the K-shortest paths algorithm to compute a reasonable number of ranked shortest paths, with the ultimate 'most optimal' path being selected by consideration of the preferences. Some experiments have been done based on the public transportation network of Nottingham City.

***Keywords*:** User preferences, K-shortest paths algorithms, Multi-objective algorithms

1. INTRODUCTION

With the increase of cars on the streets, more and more congestion occurs. The traffic congestion causes not only a monetary problem but also a pollution problem at the same time (Peytchev, 2002). Policy debates, promoted by publication of the Transport White Papers at UK and Scottish levels, have identified the need to reduce the number of car journeys and to encourage public transport usage (Hine and Scott, 2000).

As public transport services become more popular, public transportation users need individual route information to help them plan journeys more efficiently. One of the most important pieces of information to be delivered to public transportation users is the quickest bus route(s) between their specified origin and destination according to their preferences.

Route finding is a shortest path problem. Dijkstra's algorithm (Dijkstra, 1959) is often used for solving this problem due to its efficiency and effectiveness. However, Dijkstra's algorithm does not allow for time-dependent links, which is a necessary property of bus routes. Dreyfus (1969) has considered a number of methods incorporating time-dependent links. However, it is still not sufficient because public transport users have various preferences. This paper presents the time-dependent shortest path(s) algorithms which can accommodate public transportation users' preferences such as 'minimum travel time', 'minimum number of bus-changes', and 'minimum walking distance'.

Two different categories of shortest path(s) algorithms are proposed in this paper to solve the problem of accommodating user preferences in the optimisation of public transport travel.

The first solution is found using single-purpose shortest path algorithms. Single-purpose shortest path algorithms are based on standard shortest path algorithms. By adding specific objectives, the developed algorithms can generate routes which satisfy these users' preferences. Although the single-purpose shortest path algorithms work efficiently for accommodating a single user preference, each algorithm computes only one shortest path. The constraints which are added to the algorithms must be defined carefully, to ensure that the correct route is found.

The second solution is found using K-shortest paths (KSP) algorithms. The idea of the KSP algorithm is to compute a reasonable number of ranked shortest paths with the ultimate 'most optimal' path being selected by consideration of the preferences. Using KSP algorithms to accommodate user preferences in the field of public transportation system is new and very challenging. There are a large number of papers concerning different algorithms for solving the KSP (Palmgren and Yuan, 1998) but not many papers dealing with the applications in real world problems so far.

The feasibility of using KSP algorithms to accommodate user preferences for a public transportation network has been studied. The results show that it is feasible, however, at the current stage, there is still a high overlap rate for the generated K shortest routes and the execution time of the KSP algorithm is still long.

The experimental results for both single-purpose algorithms and K-shortest paths algorithm based on the public transportation network of Nottingham City are also given and analyzed.

2. USERS' PREFERENCES AND BI-MODAL TRAVEL

One important feature of this research is that the developed algorithm can accommodate public transportation users' preferences. Public transport users' preferences are various. Actually, it is impractical to take every individual preference into consideration. In this paper, the following preferences are considered:

1. Minimum travel time, which can mean either arriving at the destination at the earliest time or leaving the origin at the latest time.

2. Minimum number of bus-changes. Some travellers do not like changing buses, so they would rather travel on a single bus even if the journey time is longer.

3. Minimum walking distance. A traveller carrying heavy or awkward objects may prefer to walk to the nearest bus stop rather than walk a longer distance to another bus stop on a quicker route.

Another feature is that the route finding is based on a bi-modal travel network which considers not only travelling by bus but also on foot. In reality, all bus stops are linked to each other by foot. To make the generated optimal route more practical in the real world, the developed algorithm is based on a bi-modal travel network. This makes the research more challenging because it causes a fundamental topological change to the network and significantly increases the complexity of the network.

In the network of Figure 1, imagine that node 1 to node 6 are bus-stops, if they are connected only by buses, there are 6 links in the network.

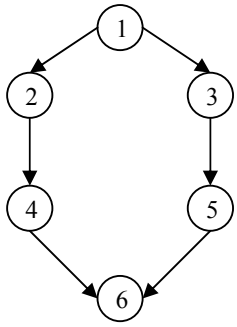

Figure 1 A Simple Transportation Network Connected Only by Buses

If the bus stops are connected by walking as well, the network becomes much more complex. As we can see in Figure 2, there are 21 links in the network now.

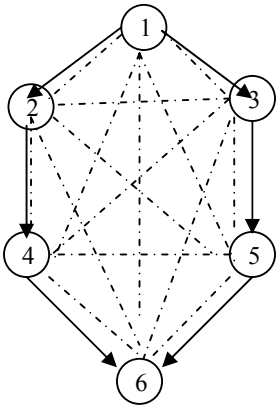

Figure 2 A Simple Transportation Network Connected either by Buses or by walking

3 NETWORK REPRESENTATION

Compared with road network, public transportation network is very time sensitive. Because of the stochastic and time-dependent properties of public transportation network, it is necessary to investigate how it can be best represented.

3.1 Notation and Definitions

The shortest path problem can be modelled as finding the shortest path between two nodes in a weighted and directed network. In some cases, it is of interest to compute not only the shortest path, but an ordered set of alternatives with the aim of finding the shortest one that satisfies user preferences for instance – K-shortest paths (KSP) problems.

Let (N,A) denote a given network, where N = $\{v_1, \ldots, v_n\}$ is a finite set whose elements are called nodes and A =$\{a_1,\ldots, a_m\}$ is a finite set whose elements are called arcs. Each arc a_k is a pair (v_i,v_j) of nodes. In the bus system context, nodes in the graph are bus stops and arcs are links between two bus stops. The input data to the algorithm consists of a description of the bus transportation network (timetables, description of links between bus stops), the bus stop where the journey begins (the source node) and the bus stop at which the journey ends (the destination node). The objective is to find the shortest path(s) between the two specified nodes.

Bus stops are point events in a transit network. A bus stop can be identified by a street name, a street intersection with a corner name or even a street address with a house number. A bus stop is linked to a bus service through the stop sequence. A stop sequence is a many-to-many relation between the bus service and bus stops. To represent the network which contains many bus stops, a very important approach is to use bus-stop codes instead of bus-stop names. The unique bus stop codes are indicated on all bus stops and are widely advertised in the public information literature issued by the Nottingham City Transport (NCT). For example, in the timetable indicated in Figure 3, AR05 and subsequent stops AR08 AR09 AR10 are assigned to indicate the bus stops on the Front Street of Arnold through the bus route 25.

A simple approach to represent the links is to ascribe a cost to every link. The cost can be defined arbitrarily such as time cost or money cost or fuel cost. In this paper, only the time cost is considered.

3.2 Modelling the Bi-modal Travel Network

By adding walking links into the transportation network, a new problem arises. Route finding for travelling on buses is timetable-based which means the length between any pair of nodes is not given directly, the information must be retrieved from the bus timetables. Route finding for travelling on foot requires the distance between any pair of nodes to be calculated according to their location. It is essential to model the two types of arc information consistently.

IDSTOP	LOCATION	ADDRESS	X_CD	Y_CD
Aa	CITY	ANGELRD	457050.58	339922.27
Ab	CITY	ANGELRD	457036.02	339928.82
Ac	CITY	ANGELRD	457005.98	339943.61
Ad	CITY	ANGELRD	456986.82	339951.85

Table 1 Format of Bus Locations Information

3.2.1 Modelling of Bus Timetables

To represent a bus transportation network, bus timetables need to be modelled. Figure 3 is an example of a bus timetable.

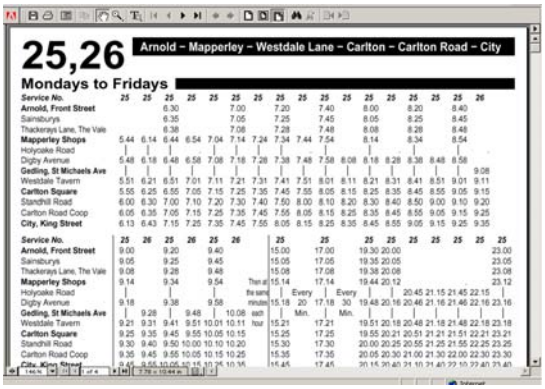

Figure 3 Timetable of Bus 25 and Bus26

(http://www.nctx.co.uk/)

To measure the time it takes to travel between two bus stops, a link is put into the directed graph to represent a connection between two bus stops. A bus departure time and the arrival time to the next bus stop are also needed. This departure time represents one entry in a timetable and the arrival time to the next bus stop can be taken from the same bus timetable at the next node. Each entry in a timetable is associated with the links.

3.2.2 Modelling of Walking Links

The walking links must be modelled consistently to the bus links, so that the information for both travel on buses or on foot can be used by the algorithms. This can be done by translating bus location information (Table 1) into usable time data. The physical distances between two bus stops can be translated into walking time information. For example, the physical distance between node i and j can be calculated by the equation:

distance(i,j)=sqrt((x(i)-x(j))**2+(y(i)-y(j))**2) (1)

Then by assuming average walking speed as 5km/hour, the walking time between i and j is:

walktime(i,j)=distance(i,j)/(5000/60) (2)

Now the two types of information – travelling on buses and travelling on foot -- are consistent.

3.3 Network Representation

With the above definitions and modelling above the public transportation network can be represented as follows. The network for the route finding problem in a bus system is represented as a graph G = (N, A) where N is a finite set of *n* nodes and A is a finite set of *m* arcs. Each arc $(i,j) \in A$ also has a length (or weight) $l_{ij} \geq 0$. In the route finding context, the network is the transportation network. Nodes are bus-stops and arcs represent the time taken to travel between each pair of nodes either by bus or on foot. The task is to find one or a series of ranked shortest paths between two nodes in this transportation network. A graph representation of such a network is shown in Figure 4, where the nodes are shown as numbered circles and the arcs are represented by lines and arrows linking the nodes.

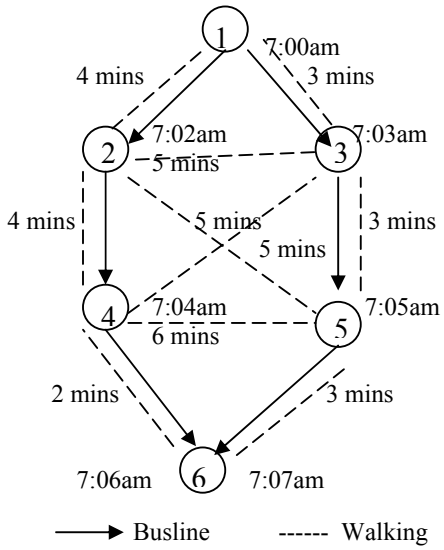

Figure 4 Representation of a Transportation Network

4 SINGLE-PURPOSE SHORTEST PATH ALGORITHMS

The first approach of accommodating user preferences in the optimisation of public transport travel is using single-purpose shortest path algorithms. Single-purpose shortest path algorithms are based on standard shortest path algorithms. The designed algorithms in this paper are based on Dijkstra's shortest path algorithm. Dijkstra's shortest path algorithm is extended to a time-dependent network and schedule-based search. Each algorithm accommodates one user objective in the route finding. Specific conditions are added into the standard shortest path algorithms or various basic shortest path algorithms are combined together, so that the developed algorithms can generate routes which satisfy users' preferences.

4.1 Assumptions and Trip Planning Preferences

To reduce the network redundancy and ensure the efficiency of path finding, some assumptions need to be defined:

(1) There is no congestion in the traffic system. At this stage, traffic uncertainty is not considered.

(2) All passengers are able to get on and get off buses at any stops.

(3) Buses depart from and arrive at every bus-stop on time. The bus arrival and departure time for a non-time point is extrapolated from the schedule of neighbouring time points.

(4) Walking time for transfer at a node is constant.

Although some of the assumptions may not appear to be realistic, they are necessary to reduce the complexity of the algorithm and increase performance. The assumption of on-time bus departure and arrival can be relaxed in the future when the time uncertainty is taken into consideration and real-time traffic data is available.

4.2 Three Single-purpose Shortest Path Algorithms

Demands of public transportation users are many and diverse. For example, some people expect to arrive at their destinations before a specific time; some people may have a planned starting time and want to find the quickest path; some people want a path with minimum bus transfers and some people need a least walking time path. Three criteria are built in the path finding algorithms in this paper: earliest arrival time, minimum walking time and minimum bus tranfers.

4.2.1 The Algorithm for Finding the Route of Earliest Arrival Time

Earliest arrival time or latest departure time is the first user preference to be accommodated. Both of them request a shortest journey time. Two search methods are developed: forward search and backward search. They can be used separately or together.

With the forward search approach, the origin and destination nodes as well as a planned departure time are specified by the passenger. The algorithm starts search from the origin node towards the destination node until all transfer nodes on the network are searched. The earliest possible arrival time at each transfer node and other arrival information are maintained. Walking time is also compared with the bus-travel time in finding the shortest route.

Figure 5 is the pseudo code for the forward search algorithm.

```
Find all branch nodes for each root node,
    Find all accessing traversals from the root
    node for each branch node,
        Do begin
            Record the earliest departure time
            at the root node for the route
            Record arrival time at the branch
            node;
            If the branch node is a new node,
            then
                Record arrival time;
            Else if it is a searched node then
                If new arrival time is earlier
                than exist one then
                    Replace old arrival
                    time by the new one;
                Else
                    Skip the route;
                End if
            End if
        End
        Label the searched branch node;
    End do
    Turn all searched branch nodes to root nodes
    End, close the root node
```

Figure 5 Algorithm for Forward Search

Backward search works quite similarly: the origin and destination nodes and an expected arrival time are specified. The algorithm starts search from destination node towards the origin node until all transfer nodes are searched. The latest possible departure time at each transfer node as well as other departure information are maintained. Walking time is considered as well. When network search is completed successfully, the optimal path can be retrieved from the nodes.

Forward search algorithm and backward search algorithms can be used independently to accommodate user preferences. However, a better performance can be achieved if they are used together.

For example in figure 6, a passenger wants to travel from Arnold to Clifton and arrive at 8:00am, the backward search algorithm finds the route as: taking Bus 58 (route 1) at Arnold at 6:25am and arriving City Centre at 6:55am, transferring to Bus 2 (route 2) at 7:30am and arriving at Clifton at 7:58am. This trip plan is correct because it ensures the passenger arrives at Clifton before 8:00am. However, there is another bus which leaves City Centre at 7:00am and arrives Clifton at 7:30am, which means taking this trip will give the passenger more leisure time in Clifton rather than waiting at City Centre. To solve this problem, a forward search algorithm can be used after a backward search algorithm.

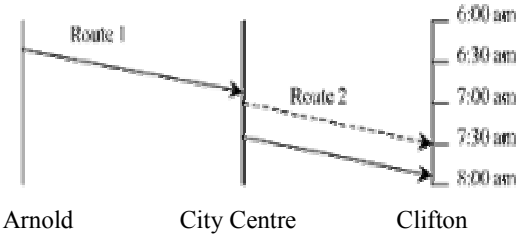

Figure 6 Travel from Arnold to Clifton, expected arrival time 8:00 am

In summary, the forward search algorithm accommodates the user preference of 'earliest arrival time' and the backward search algorithm accommodates the user preference of 'latest departure time'. Using them together can sometimes generate a better route which is closer to human cognition.

4.2.2 The Algorithm for Finding the Route with Minimum Walking Time

The algorithm for finding a shortest path with minimum walking time is an extension of the forward search algorithm. One more parameter is involved to record the total walking time.

Each transfer node takes a walk time property. Walking time is only taken into consideration if a transfer is absolutely essential. This means if it is possible to get to the next bus-stop by staying on a bus, the algorithm will not choose a path by walking even though walking might be quicker than travelling on a bus.

4.2.3 The Algorithm for Finding the Route with Minimum Bus-Changes

The minimum transfer path approach searches from the origin node towards the destination node. Search information maintained at each transfer node

includes the total number of transfers up to this point. The difference between the minimum transfer algorithm and the forward / backward search is that while minimizing the number of transfers, all arrivals with same number of transfers must be maintained at the node. For example, in figure 7, if node C is arrived from node A with 1 transfer and arrived from node B with 1 transfer, information for both arrivals must be maintained in node C.

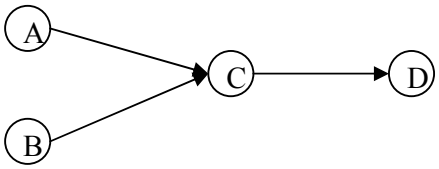

Figure 7 A Bus Transfer Network

Figure 8 is the pseudo code for the minimum transfer search algorithm.

```
Find all branch nodes for each root node,
    Find all accessing traversals for each branch node,
        Do begin
            If the arrival traversal exist in the arrival information of root node, then
                Transfer = root node transfer;
            Else
                Transfer = root node transfer + 1;
            End if
            If branch node is a new node, then
                Record arrival information
            Else
                If transfer is less than existing transfer, then
                        Replace existing arrival information with new one;
                Else if transfer equals existing transfer, then
                        Record new arrival information to existing one;
                Else (transfer is greater than existing transfer)
                        Skip;
                End if
            End if
        End do
        Label a searched branch node;
    End
    Turn all searched branch nodes to root nodes
End, and close root node
```

Figure 8 Algorithm for Minimum Transfer Path Search

The minimum transfer search is non-scheduled due to the difficulties of combining the schedule with other criteria in the search process. Therefore, post-processing is required to provide timetable information. The minimum transfer route may not be the earliest arrival one or the latest departure one, but it is still a valid option and it does accommodate one of the important user preferences. Also it is the one which is similar to human cognition in using the public transportation system.

5 K-SHORTEST PATHS ALGORITHM

The second approach of accommodating public transportation users' preferences is by K-shortest paths algorithm. Compared with single-purpose algorithms, KSP algorithm is based on multi-purpose network search. It computes a number of ranked shortest routes in one time. The background information of finding ranked K-shortest paths is presented first.

Same as the shortest path problem, listing the K shortest paths is also a classic networking programming problem (Martins 1998, Martins 1999). Although it has not been studied as intensively as the shortest path problem, there are numerous papers that can be referred to (http://liinwww.ira.uka.de/bibliography/Theory/k-path.html).

Generally speaking, there are two classes of solution methods to KSP problem. The first class consists of labelling algorithms and the second class of algorithms based on constructing a tree that contains K shortest paths (Palmgren and Yuan, 1998).

Labelling algorithms for KSP problem are very similar to the classical labelling algorithms for shortest path problems. They can be further divided into Label Correcting (Shier, 1974; Shier, 1976; Shier, 1979) and Label Setting (Dreyfus 1969; Martins 1984; 1996, Azevedo et al, 1993; Azevedo et al, 1994) algorithms.

The second class of KSP algorithms is called the deviation algorithm. This class of algorithms attempts to build a tree of K shortest paths. The early references to deviation algorithms are Yen (1971)'s *Finding the k shortest loopless paths in a network* and Martins et al (1997)'s *A new algorithm for ranking loopless paths*.

Again, a large number of papers (http://liinwww.ira.uka.de/bibliography/Theory/k-path.html) concerning different algorithms for solving the KSP problem can be found but not many papers dealing with the applications in real world problem. The reason why it isn't widely used

might be that using KSP algorithms is an expensive way of generating a large number of infeasible paths. For example, according to Desrosiers and Soumis (1991), the 'pickup and delivery problem' is a typical constrained shortest path problem that can be solved by the KSP algorithms, but dynamic programming takes advantages of the additional constraints and its efficiency increases with the number of constraints.

However, there are more and more efficient algorithms existing to solve KSP problems. And by reducing the value of K and taking the additional constraints into consideration, the approach of using KSP algorithms is becoming more and more attractive.

5.1 Finding the Ranked Shortest Paths

The developed KSP algorithm for public transportation network in this paper is based on the principal of label setting KSP algorithms. However, it must be time-dependent and suitable for bus system.

Transportation networks in the real world are very complex. The considered network is a bi-modal travel network. Route finding for travelling on buses is timetable-based which means the length between any pair of nodes are not given directly, the information must be retrieved from the bus timetables. Route finding for travelling on foot requires the distance between any pair of nodes to be calculated according to their location. The two types of arc information are mixed up. The algorithm must be able to distingush between the two different types of information.

Suppose the network in Figure 4 is a transportation network, step-by-step explanations of finding the ranked K-shortest paths from node 1 (origin) to node 6 (destination) are given below: (Wu and Hartley, 2004)

Step 1: Find the closest node to the origin node by bus.

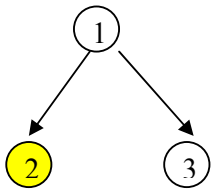

Figure 9: Step 1 of the Algorithm

Suppose node 2 is the closest node to node 1 by bus travelling, record the arrival time timebus(2).

Step 2: Find the closest node to the origin 1 by walking.

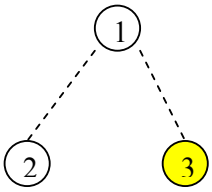

Figure 10: Step 2 of the Algorithm

Suppose node 3 is the closest node to node 1 by walking, record the arrival time timewalk(3)

Step 3: Compare the arrival time of timebus(2) and timewalk(3). If (timebus(2)< timewalk(3)) then label node 2, if (timewalk(3) < timebus(2)) then label node 3.

As explained before, for the KSP problem more than one label is assigned to each node, a flag is used to record how many times a node has been labelled.

Step 4: Upgrade the network according to the labelled node.

Suppose in Step 3: (timebus(2) < timewalk(3)) and node 2 is labelled then the network is upgraded as follows:

In Figure 4, there is a bus link from node 2 to node 4, so the network is upgraded by adding a bus link from node 1 to node 4.

There are also walking links from node 2 to node 3, node 4 and node 5, so the network is upgraded by adding links from node 1 to node 3, node 4 and node 5.

Since there is already a walking link between node 1 and node 3, the walking link 1→2→3 becomes the second walking link from node 1 to 3. It must be recorded separately as a second walking link.

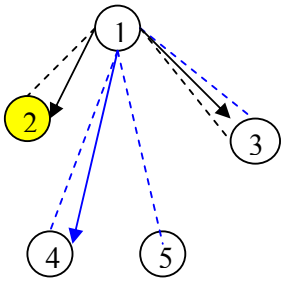

Figure 11: Step 4 of the Algorithm

Or, suppose in Step 3: timewalk(3) < timebus(2) and node 3 is labelled then the network is upgraded using the same principle:

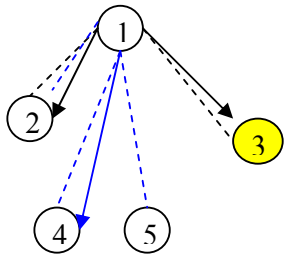

Figure 12: Step 4 of the Algorithm

Step 5: If the labelled node is the destination node, output the route from the origin to the destination.

Step 6: Repeat step 1 to step 5 until the Kth ranked shortest path has been outputted.

Figure 13 is the pseudo code for the KSP algorithm.

count(i) – number of paths that were determined from origin to destination
elm(i) – flag assigned to each node
K – number of routes to be computed

```
For each node i, count(i)=0, elm(i)=0.
For the original node, count(origin)=1, elm(origin)=1
  If there is a bus link or a walking link from origin
  node to node i, then
     elm(i) = 1
  End if

For each node i, while count (destination) < K
  Do begin
     j records the closest node to the origin node
     Time(j) records the arrival time of the closest
     node
     Count(j) = count(j) +1
     If (j=des), then
        Output the route
     End if

     If count (i) ≤ K
        If there is a bus link or walking link from j to
        i, then
           Elm(i) = elm(i) +1
           Upgrade the network by adding nodes and
           links
        End if
     End if
  End do

End, and close root node
```

Figure 13 Algorithm for the KSP algorithm

5.2 Using the KSP Algorithm to Accommodate Users' Preferences

To accommodate users' preferences, the routes must satisfy more conditions. This can be done by either comparing each route with the preferences and choosing the first route which satisfies the preferences or regarding the preferences as constraints and embedding them into the algorithm.

The first solution has been implemented and some efforts have been made to add constraints to the developed KSP algorithm. As mentioned before, public transportation networks are very sensitive to point of time, it is quite difficult to use a weighting system with the schedule-based shortest path algorithms. Furthermore, KSP algorithms are very computational time and memory consuming, the constraints which are added to the KSP algorithm must be defined carefully.

On the current stage of my research, the designed algorithm generates a list of ranked shortest routes and those routes are compared with the users' preferences until the ultimate path is selected. A walking limitation has been added to the developed KSP algorithm as a constraint.

In figure 14, for example, a journey starts from GR01 to GR05 at 7.000am, the developed KSP algorithm generates 4 ranked shortest paths which are described in Table 2.

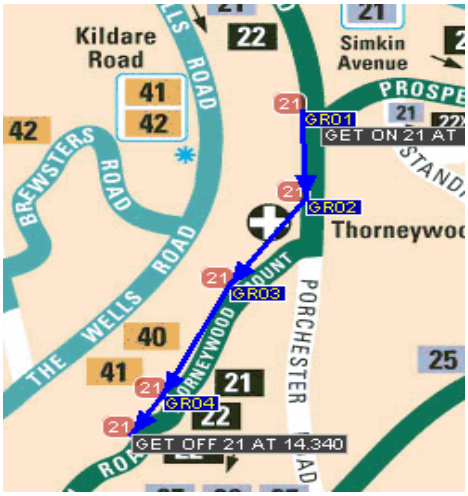

Figure 14 Shorest Route from GR01 to GR05

1st Route			2nd Route		
Bus stop	Time	Services	Bus stop	Time	Services
GR01	7.020	Bus 21y	GR01	7.000	Walking
GR02	7.025	Bus 21y	SA01	7.040	Bus 41y
GR03	7.030	Bus 21y	SA02	7.060	Bus 41y
GR04	7.035	Bus 21y	SA03	7.070	Bus 41y
GR05	7.040	Bus 21y	SA04	7.080	Bus 41y
			SA05	7.090	Walking
			GR15	7.125	Bus 21x
			GR16	7.130	Bus 21x
			GR17	7.135	Bus 21x
			GR05	7.139	walking
Total Walking		0			0.075
Bus Change		0			2
3rd Route			4th Route		
Bus stop	Time	Services	Bus stop	Time	Services
GR01	7.000	Walking	GR01	7.020	Bus 21y
SA12	7.093	Bus 40x	GR02	7.025	Bus 21y
SA13	7.120	Bus 40x	SA03	7.058	Walking
GR05	7.162	Walking	GR15	7.125	Bus 21x
			GR16	7.130	Bus 21x
			GR04	7.235	Walking
			GR05	7.240	Bus 21y
Total Walking		0.132			0.138
Bus Change		0			2

Table 2 Four Ranked Shortest Paths between GR01 and GR05

By comparing the 4 shortest paths, the 1st one is selected as the route of earliest arrival time, minimum walking time and minimum bus changes. The second shortest route with minimum bus changes is the 3rd shortest path. The 2nd and 4th routes have the same number of bus changing, however the 2nd route takes shorter walking time.

To limit the complexity of the network, a walking limitation has been added to the KSP algorithm, because if each pair of bus-stops are linked either by bus-services or by walking or by both of them, the network will be certainly burdened by the walking links. In reality, seldom people would take the advice to walk over 1 hour from one bus node to another if they choose to use public transportation system. Therefore, the the walking time is restricted to 10 mins in the KSP algorithm -- any walking link which is more than 10mins will be removed from the network. This was proved to be very helpful to ensure the efficiency of the algorithm.

However, it remains a problem that how many routes need to be generated to find the 'desired' path for the public transport users. The value of K directly affects the efficiency of the algorithm and determines the capability of the KSP algorithm of finding the desire route.

Furthermore, in Table 2, we can tell that there is still a high overlap ratio between the ranked K shortest paths. Some routes are mostly overlapped which causes a waste of computation resource. A paper by Chen and Feng (1999) propses an approach of using 'Restricted KSP algorithm' to reduce the overlap ratio. By introducing an 'overlap ratio' and a 'route travel difference' into the KSP algorithm, the overlap ratio can be controlled. This solution is under testing now on the public transportation network of Nottingham City.

6 EXPERIMENTAL RESULTS AND ANALYSIS

The experiments are based on the public transportation network of Nottingham City which has 2398 bus-stops in total and 292 bus services running everyday, with the results comparing the performance of the single-purpose (single purpose) shortest path algorithms and the KSP (multi-purpose) algorithm. All the bus stops and links are represented by bus timetables. The departure and arrival time at a particular bus stop can be found in these timetables.

6.1 Perfomance of the Single-purpose Shortest Path Algorithms

Two criteria determine the performance of an algorithm: the ability of finding the shortest bus route and the efficiency of the algorithm. Concerning the efficiency of an algorithm, time-complexity and memory requirement need to be considered. All of the three single-purpose algorithms are based on Dijkstra's algorithm, so that they perform similarly from the time-complexity and memory requirements points of view.

6.1.1 Experimental Results of the Single-purpose Shortest Path Algorithms

A careful implementation of a time consuming shortest path algorithm saves expensive and crucial time.

In figure 15 a passenger wants to travel from SN01 to CA12 with the earliest arrival time. The test result shows that to find the shortest path, a careful implementation of the time-dependent Dijkstra algorithm takes only 0.5 seconds.

Although an execution time of 0.5 seconds is satisfactory, it can be further reduced. Dijkstra algorithm has a bottleneck that is all nodes have to be visited at each iteration in order to select the node with the minimum distance label. A bucket data structure can be used to conquer this problem. Bucket data structure means: an area of storage where items with a common property are stored, so that the labelled nodes in a data structure can be maintained in such way that the nodes are sorted by distance labels. To translate this idea to the transportation network, an up-bound time (this means adding more restrictions into the code) can be set in the iteration. For example, if the leaving time is 7:00am, we may only search the time no later than 9:00am, because in practice, passengers usually won't take bus more than 2 hours to the destination. The test result shows that this implementation improves the shortest path algorithm's efficiency when it works on a very complex transportation network.

Figure 15 Shortest Route from SN01 to CA12

6.1.2 Analysis of the Single-purpose Algorithms

Time complexity

As Dreyfus (1969) pointed out in his article, the algorithm for a network with time dependent costs of links has the same time complexity as the Dijkstra algorithm. It is very well known that the Dijkstra algorithm can work in $O(n^2)$ time, where n is the number of nodes in the network. This time complexity is the worst-case for the Dijkstra algorithm. Since the developed single-purpose algorithms belong to the group of algorithms for a network with time dependent costs of links, they have the same time complexity as the Dijkstra algorithm.
The time efficiency of the algorithm depends on the method used in the implementation. Most principles devised for the Dijkstra algorithm can be very easily used to increase the efficiency of the bus algorithm. The Dijkstra algorithm has been implemented by the author using a priority queue. This implementation runs in $O(kn+lm) = O(n+m)$. The coefficients k and l depend on the implementations of the loops, k expresses the cost of the loop, which processes nodes and l expresses the costs of the loop which processes links.

An article by Cherkassky et al. (1996) examines several implementations of the Dijkstra algorithm. For example: the implementation using Fibonacci heaps, the implementation using R-heaps, the implementation using buckets and others. Fibonacci heaps, R-heaps, and buckets are different data structures, and these three implementations are all aiming to speed up Dijkstra algorithm. And the result from the article by Cherkassky et al. (1996) is that one of the best implementations is with the priority queue. Detailed explanations about these three data structure can be found in any data structure book.

Not only is the priority queue implementation one of the fastest, but it is also simple and clear. The simplicity and efficiency of the priority queue implementation made the author of this paper follow this approach.

Memory requirements

Memory requirements by these single-purpose algorithms are also the same as required by the Dijkstra algorithm. Therefore they need k*n words to represent information about nodes and lm words to represent information about links. The k coefficient depends on the type of data structure used.

In the code the data structure by itself had k=5, because every node stores the following information:

– the node number
– the previous node of the shortest path
– the node number of a link to the previous node
– the time cost of the shortest path to this node
– the status of this node.
The l coefficient equals 2 for the produced code. It is so because every link has two members:

node_no – the number of the finish node
link_no – the number of this link

Therefore the overall memory requirements for the single-purpose algorithms are 5n+2m. This memory is required by plain data to represent the network structure. However, the network is still not completely described. More memory is needed to store timetables. Every timetable implemented in the code needs approximately 30 words (2 words for each entry, and say there are about 15 buses a day of

one bus line). Finally the memory required to store completely the network is 5n+2m+30x, where x is the number of timetables.

The single-purpose algorithms meet the research target very well, as they are fast and have very small memory requirements.

6.2 Performance of the KSP Algorithm

6.2.1 Experimental Results for the KSP Algorithm

The KSP algorithm takes a longer execution time, and the execution time increases significantly with the increase of the length of the journey and the value of K. Taking the travel plan in Figure 14 as an example again, the performance for the KSP algorithm is as follow:

1. To find 2 ranked (K=2) shortest paths from GR01 to GR05, the developed KSP algorithm spends 2.38 seconds. And compared with the results of the three single-purpose algorithms, the 1st shortest path is the path with all the features of earliest arrival time, minimum walking time and minimum bus-changes.

2. To find 4 ranked (K=4) shortest paths from GR01 to GR 05, the KSP algorithm spends 8.6 seconds. Again, compared with the results of the three single-purpose algorithms, the 1st shortest path is the path which satisfies all of the three preferences; and the 3rd path is the one with minimum bus-changes as well, however it arrives at the destination later due to the involvement of walking.

To further analyse the KSP algorithm, more experiments are done by randomly choosing the departure and arrival bus stops. In summary, the developed KSP algorithm performs much better for shorter and medium length of travel. The average run time for a short length of travel (bus stops less than or equal 10) is 2.8 seconds and the average run time for a medium length of travel (bus stops between 10 and 20) is 8.96 seconds. If the K is set to be 4, the successful rate of finding the 3 'desired' shortest paths is more than 95% for a short journey and 91% for a medium journey – depending on the locations of the bus stops. For instance, the algorithm is more likely to fail to find the 'desired' routes if the bus stops are within the city centre where each bus stop is used by many bus services.
Expanding the journey to more than 20 bus stops (long length of travel), say from GR01 to HC, the algorithm takes much longer time (29.8 seconds) to find 4 ranked shortest paths. And there is only about 75% successful rate to accommodate the 3 user preferences by the 4 ranked shortest paths in those long journey. To increase the successful rate, the value of K needs to be set big enough which causes longer execution time consequently. Therefore, heuristic methods are investigated to reduce the infeasible routes produced by the KSP algorithm and different hierarchical methods need to be investigated as well to improve the efficiency of the KSP algorithm.

6.2.2 Analysis of the KSP Algorithm

The time complexity of KSP algorithms is much higher than shortest path algorithms, and it differs with different algorithms. General labelling KSP algorithms take longer execution time but the time complexity can be reduced by advanced data structure. For example, David Eppstein (1997) stated in his paper that the K shortest paths can be found in total time of $O(m+n\log n+Kn)$ for the worst case. Deviation algorithms takes shorter execution time. Martins et al (2000) developed a new implementation of Yen's ranking loopless paths algorithm which takes $O(Kn(m+n\log n))$ computational time.

The KSP algorithm developed by the author is based on the general label setting algorithm, so that the computaional compexity is similar to the generilized Dijkstra's shortest path algorithm.

6.3 Comparison of Results

According to the experimental results, the single-purpose algorithms work much more efficiently than the KSP algorithm. However, the longer execution time of the KSP algorithm is expected as the multi-objective algorithm finds several required routes at the same time. And from an academic point of view, multi-criteria shortest path(s) algorithms for public transportation networks have great research potential and consequently will lead to a great commercial potential as well. Detailed comparison of computational runtime between the KSP algorithm and those single-purpose shortest path algorithms can be found in the Table 3 and Table 4.

	Single Purpose Algorithms' Average Runtime (sec)		
	Earliest arrival	Minimum walking	Minimum bus-change
Short (≤ 10 bus stops)	0.08	0.08	0.09
Medium (≤ 20 bus stops)	0.60	0.61	0.64
Long (≥ 30 bus stops)	0.84	0.86	0.90

Table 3 : Executive Time of Single-purpose Algorithms

	KSP Algorithms' Average Runtime (sec)		
	K=2	K=3	K=4
Short (≤ 10 bus stops)	2.80	4.20	8.60
Medium (≤ 20 bus stops)	8.96	13.78	18.67
Long (≥ 30 bus stops)	21.60	36.57	52.36

Table 4 : Executive Time of KSP Algorithms

7 CONCLUSION

Two different solutions – single-purpose shortest path algorithms and KSP algorithm are developed to accommodate user preference in the optimization of public transportation travel. The feasibility of using KSP algorithms to accommodate users' preferences for a public transportation network is discussed in this paper. Different kinds of KSP algorithms are compared and an algorithm based on the label setting algorithms is developed and implemented on the Nottingham Public Transportation Network.

The experimental results show that the single-purpose shortest path algorithms work efficiently but can only accommodate one single user preference and the K-shortest path algorithm generates multiple routes in one time, however, with a longer computational execution time.

The feasibility study of using KSP algorithms to accommodate user preferences for a public transportation network gives a positive answer as well.

The developed KSP algorithm is ideally expected to find a set of ranked shortest paths which can accommodate users' preferences easily and without many infeasible routes. However, at the current stage, there is still a high overlap ratio for the generated K shortest routes and the execution time of the KSP algorithm is still long.

As expected, all the algorithms developed in this research are ready to be used for other cities. Compared with other cities, such as London, Nottingham is a small city. Only Nottingham City Transport (NCT) bus stops (2398 bus stops) are used by the developed algorithms so far. If all public transport services in Nottingham are used, there might be around 5,000 bus stops. Furthermore, if the developed algorithms are implemented to the London public transportation network which contains more than 45,000 bus stops, the execution time will increase significantly.

8 FUTURE WORK

The objective of this research is to develop an efficient algorithm to accommodate public transportation users' preferences in the real world. Therefore, the next stage of the research will largely focus on improving the efficiency of both single-purpose algorithms and the KSP algorithm.

Both types of algorithm do not take the time uncertainty factor and real-time bus location information into account, so that the time uncertainty factor will be introduced to both the single objective and KSP algorithms. The efficiency of the algorithms will be improved by investigating different hierarchical methods. Neural networks, distributed and parallel computing systems will be investigated and implemented one after another until a satisfying performance is gained.

REFERENCES:

Bellman R. E., 1958, 'On a Routeing Problem', *Quarterly of Applied Mathematics 16*, 87-90

Brander A.W., Sinclair M.C., 1995. 'A Comparative Study of k-Shortest Path Algorithms' *Proc. 11th UK Performance Engineering Worksh. for Computer and Telecommunications Systems*

Chen H.K, Feng G, 1999, Heuristics for the stochastic/dynamic user-optomal route choice problem, European Journal of *Operational Research* 126 (2000) 13-30.

Cherkassky B.V., Goldberg, and T. Radzik A.V., 1996. 'Shortest Paths algorithms: Theory and experimental evaluation'. *Mathematical Programming*, 73: 129-196.

Cooke, K.L., Halsey, E., 1966, The shortest route through a network with time-dependent internodal transit times, *Journal of Mathematical Analysis and Applications.*

Datar M. and Ranade A., 2000, Commuting with delay prone buses. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, 22-29.

Dijkstra, E., 1959. 'A note on two problems in connexion with graphs', *Numerische Mathematik* 1, 269-271.

Dreyfus, S.E., 1969. 'An Appraisal of Some Shortest-path Algorithms'. *Operations Research,* vol.17, no. 395-412.

Dumas, Desrosiers and Soumis, 1991, The pickup and delivery problem with time windows. *European Journal of Operational Research* 54 (1991) 7-22.

Frank H, 1969, Shortest paths in probabilistic graphs, *Operations Res.*, Vol.17, 583-599.

Gallo G., Pallotino S., 1988, Shortest path algorithms, *Annals of operations research*, Vol. 13, 3-79

Gallo G., Pallotino S., 1984, Shortest path methods in transportation models, *Transportation Planning Models*, M.Florian (Editor), Elsevier Science Publishers B.V (North-Holland), pp.227-242.

Jakob R., Marathe M.V., Nagel K., 1998, A computational study of routing algorithms for realistic transportation netwoks, *Proceedings WAE'98*, Saarbrucken, Germany, August 20-2, 167-178.

Hartley, J.K., Bargiela, A., 2001. "Decision Support for Planning Multi-Modal Urban travel", *Proc. of 13th European Simulation Symposium,* Marseille, October 2001, ISBN: 90-77039-02-3, pp 387-391.

Hine J., Scott J., 2000, 'Seamless, accessible travel: user's views of the public transport journey and interchange', *Transport Policy* 7, 217-226.

Hsu-Shih S., 2001, An Interactive Approach for Integrated Multilevel Systems in a Fuzzy Environment, *Mathematical and Computer Modelling* 36 (2002) 569-585.

Kamburowski J., 1985, A note on the stochastic route problem, Operations research, Vol.33, No.3, 696-698.

Martins, E.Q.V., Pascoal, M.M.B., Santos, J.L.E., 1997. A New Algorithm for Ranking Loopless Paths, *Research Report (submitted).*

Martins, E.Q.V., Pascoal, M.M.B., Santos, J.L.E., 1998. The K Shortest Paths Problem, *Research Report.* http://www.mat.uc.pt/~eqvm/cientificos/investigacao/r_papers.html

Martins, E.Q.V., Pascoal, M.M.B., Santos, J.L.E., 1999, Labelling Algorithms for Ranking Shortest Paths, *Research Report.*

Micheal P. Wellman, Matthew F., and Kenneth L., 1995, Path Planning under Time-Dependent Uncertainty, in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (UAI-95).

Miller-Hooks Elise D, Mahmassani Hani S, and Ziliaskopoulos Athanasios, 1994, Path search techniques for transportation networks with time-dependent, stochastic arc costs, *IEEE international conference on systems man and cybernetics: human information and technology*, Vol.2, 1716-1721.

Miller-Hooks E.D., 1997, Optimal routing in time-varying, stochastic networks: algorithms and implementations, PhD thesis, The university of Texas at Austin.

Miller-Hooks E.D. and Mahmassani H.S., 1998, possible time paths in stochastic time-varying networks, *Computers Ops Res*, Vol.25, No.12, 1107-1125.

Miller-Hooks E.D. and Mahmassani H.S, 2000, Least expected time paths in stochastic time-varying transportation networks, *transportation Science*, Vol.34, No.2, 198-215.

Minty, G., 1957, A comment on the shortest route problem. *Operation research*.

Palmgren M. and Yuan D., 'Shortest summary on K shortest path: Algorithms and applications', http://www.esc.auckland.ac.nz/People/Staff/Mason/Courses/LinkopingColGen99/kth.pdf

Peytchev E., Coggan J., 2002. 'See before you go', *Traffic Technology International*

Pallottino S., 1984, 'Implementation and Efficiency of Moore Algorithms for the Shortest Root Problem', *Mathematical Programming 7*, 212-222

Ruihong H. and Zhong-Ren P., 2002. 'An Single-purpose GIS Data Model for Transit Trip Planning Systems', *Journal of the Transportation Research Board: Transportation Research Records*, No.1804, pp. 205-211.

Shier D., 1974. 'Computational experience with an algorithm for finding the k shortest paths in a network', *Journal of Research of the NBS,* 78:139-164

Shier D., 1976. 'Interactive methods for determining the k shortest paths in a network', *Networks,* 6:15 1-159.

Shier D., 1979. 'On algorithms for finding the k shortest paths in a network', *Networks,* 9:195-214.

Yen J.Y. 1971, 'Finding the k shortest loopless paths in a network', *Management Science,* 17:712-716

Orda A. and Rom R., Shortest-Path and Mimimum-Delay Algorithms in Networks with Time-Dependent Edge-Length, *Journal of Association of Computing Machinery*, Vol.37, No.3, 607-625, 1990

Orda A. and Rom R., Shortest path algorithms for time-dependent networks, IEEE INFOCOM '88 – The conference on computer communications proceedings, Seventh annual joint conference of *the IEEE Computer and Communications Societies* – Networks: Evolution or Revolution?, *IEEE*, 282-7, New York, NY, USA, 1998

Sigal C.E., Pritsker A. Alan B and Solberg James J, The stochastic shortest path problem, *Operations research*, Vol.28, No.5, 1122-1129, 1980

Wu Q., Hartley J.K., 2004, Using K-Shortest Paths Algorithms to Accommodate User Preferences in the Optimization of Public Transport Travel, in *Proceeding of UKSIM 2004*, 113-117.

Wu Q., Hartley J.K., 2004, Accommodating User Preferences in the Optimization of Public Transport Travel, in *Proceeding of the 4th European Congress and Exhibition on ITS*.

Zhan F.B. and Noon C.E., Shortest path algorithms: An evaluation using real road networks, *Transportation Science*, Vol.32, No.1, 65-73, 1998

Zhan F.B., Three fastest shortest path algorithms on real road networks: data structures and procedures, *Journal of geographic information and decision analysis*, Vol.1, No.1, 69-82, 1997

Ziliaskopoulos A, 1994, Optimum path algorithms on multidimensional networks: analysis and design, implementation and computational experience, Ph.D dissertation, Department of Civil Engineering The University of Texas at Austin.

Nottingham Trent University, Advanced traffic and travel information system, http://www.doc.ntu.ac.uk/RTTS/Projects/grr32468/public.html

http://www.nctx.co.uk/

http://liinwww.ira.uka.de/bibliography/Theory/k-path.html

BIOGRAPHIES:

Ms. Qiujin Wu is a research student at the School of Computing and Technology, the Nottingham Trent University. She was rewarded a MA degree in Information Technology at the University of Nottingham in 2002 and got her bachelor in engineering in year 1998 from Shanghai University, China. Ms. Qiujin Wu started her PhD study in October 2002 under the supervision of Dr. Joanna Hartley and Professor David Al-Dabass. Her research topic is: "Accommodating User Preferences in the Optimisation of Public Transport Travel".

Dr. Joanna Hartley was awarded a BSc (Hons) degree in Mathematics at the University of Durham in 1991. In 1992, she became a research assistant at The Nottingham Trent University and was awarded a PhD in 1996. The title of her PhD is "Parallel Algorithms for Fuzzy Data Processing with Application to Water Systems". She is now a senior lecture at The Nottingham Trent University and an active member of the simulation and Modelling group. Her current research interests include parallel processing, mathematical modelling and probabilistic state estimation relating to urban traffic networks and water distribution systems.