HNUE JOURNAL OF SCIENCE DOI: 10.18173/2354-1059.2021-0049
Natural Sciences 2021, Volume 66, Issue 3, pp. 81-90
This paper is available online at http://stdb.hnue.edu.vn

AN IMPROVED DIJKSTRA ALGORITHM TO FIND MINIMUM TIME PATHS FOR BUS USERS IN HANOI

Vu Thai Giang[1], Nguyen Thi Kim Ngan[2], Nguyen Ba Hoan[3] and Bui Minh Duc[4]
[1]*Faculty of Information Technology, Hanoi National University of Education*
[2]*Faculty of Computer Science and Engineering, Thuyloi University*
[3]*Faculty of Information and Language, College of Technology Highlands, Dak Lak province*
[4]*Faculty of Information Technology, Hanoi National University of Education*

Abstract. In Hanoi, many roads are congested during rush hour. When going through congested roads, the movement of vehicles is very slow. As a result, traveling over a short and congested road may take more time than traveling over a longer and uncongested road. Therefore, in this paper, we study the problem of finding optimal bus routes that take less time, considering the traffic jams. We extend Dijkstra's algorithm to compute waiting time at bus stations and traveling time of buses. The experimental results show that our algorithm is suitable.

Keywords: Dijkstra's algorithm, minimum time path, optimal bus route.

1. Introduction

Nowadays, public transport is used more and more, because of its convenience, safety, low cost. Therefore, there have been many studies interested in the problem of finding optimal routes for public transport users. For example, Spiess and Florian [1] studied the problem of finding the minimum time path from the starting station to the destination station. But they consider only one type of transport means and suppose the waiting time depends only on the traffic frequency. Modesti and Sciomachen [2] studied optimization problems in multimedia networks. Wu and Hartley [3] find the optimal route in a multimodal transport network, their goal is the minimum time, the minimum number of bus transfers, and minimum walking distance. Antsfeld and Walsh [4] find optimal multi-objective routes in a multimedia traffic network. Nguyen et al. [5] find shortest routes for Hanoi bus users. Transerco (Hanoi Transport Corporation) has built a website http://timbus.vn, allowing users to look up information about buses and find bus routes. However, this application does not provide a minimum time path. The above-mentioned studies did not consider bus traveling during traffic jams.

Received September 9, 2021. Revised October 21, 2021. Accepted October 29, 2021.
Contact Nguyen Thi Kim Ngan, e-mail address: ngannguyen@tlu.edu.vn

The goal of this study is to research a solution to the minimum time path for bus users in Hanoi. The minimum time path considers both waiting times at bus stations and travels time of buses during rush hour. Dijkstra's algorithm [6] only finds shortest paths on graphs with edge weights, these weights do not change over time. This algorithm does not consider vertex weights and the weight change at different times. Therefore, it can not compute the minimum time path with waiting time at bus stations (vertex weights) and the edge weight change during rush hour. We extend Dijkstra's algorithm to allow finding an optimal bus path considering bus waiting time at bus stations and bus travel time on-road segments.

2. Content

2.1. Survey of the Hanoi bus system

To conduct this study, we surveyed the Hanoi bus system. The work includes surveying bus stations, active bus routes and departure times of buses at stations, congested roads during rush hours. We have obtained some results as follows: Each bus route contains one outbound route and one return route. The outbound/return route is represented by a list of bus stations. Each bus route has some buses. The start time of a bus route is defined as the start time of the earliest bus on this route. The end time of a bus route is defined as the end time of the latest bus on this route. The operating time of buses is usually from 4:30 to 23:15 [7]. In Hanoi, the rush hours are 6:30 am - 9 am and 4 pm - 7 pm, the times during which most people commute. During the times, some roads in Hanoi are congested. Currently, in Hanoi, there are 2210 bus stations, 131 bus routes [7]. Some bus stations have information about the distances of upcoming buses, others have not. When going through congested roads, the movement is very slow. As a result, traveling on a short and congested road may take more time than traveling on a longer and uncongested road.

In the next section, we will describe the problem of finding minimum-time paths for bus users in Hanoi, considering traffic jams.

2.2. Problem description

Some roads in Hanoi are often congested at the time of 6:30 am - 9 am and 4 pm - 7 pm. During these periods, traveling on a congested road is very slow. The problem of finding minimum time paths considers the bus waiting time at stations and the travel time of buses on roads (which may or may not be congested). To perform this problem, we assume that we know the departure time of the buses at every station.

For example, Figure 1 describes a bus diagram. In which, B, C, D, G are bus stations, 1, 2, 3, 4, 5 are names of bus routes. Between two adjacent stations of a bus route is an edge. The weight of this edge is the distance between the two stations. The information on edge BC means we can use bus 1 or bus 2 to go from station B to station C, the distance between B and C is 5000 m. We assume that the departure times of bus routes at every station are described in Table 1. At arc BC, we can use bus 1 or bus 2 to go from B to C. The departure time of bus 1 at B to go to C is 6h00, 6h10, 6h20, 6h30, 6h40, 6h50, 7h00. The departure time of bus 2 at B to go to C is 6h05, 6h15, 6h25, 6h35, 6h45, 6h55, 7h05.

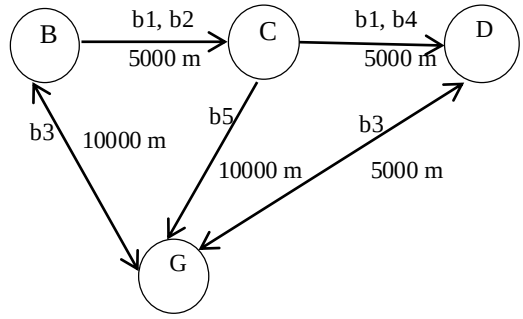

Figure 1. An example of bus movements between stations

Table 1. An example of departure times of buses at stations

Vertex	**B**			**C**			**D**	**G**	
Edge	***BC***		***BG***	***CD***		***CG***	***DG***	***GB***	***GD***
Bus route	*1*	*2*	*3*	*1*	*4*	*5*	*3*	*3*	*3*
Departure time	6h00	6h05	6h00	6h10	6h25	6h00	6h05	6h25	6h20
	6h10	6h15	6h10	6h20	6h35	6h10	6h15	6h35	6h30
	6h20	6h25	6h20	6h30	6h45	6h20	6h25	6h45	6h40
	6h30	6h35	6h30	6h40	6h55	6h30	6h35	6h55	6h50
	6h40	6h45	6h40	6h50	7h05	6h40	6h45	7h05	7h00
	6h50	6h55	6h50	7h00	7h15	6h50	6h55	7h15	7h10
	7h00	7h05	7h00	7h10	7h25	7h00	7h05	7h25	7h20

The problem of finding minimum time paths is described as follows:

Input: Start station, target station, departure time of a customer; information about bus stations in Hanoi, bus routes, departure times of bus routes at stations.

Output: The minimum time paths from the start station to the target station. It means that the customer arrives at his destination the earliest.

The next section presents the method to solve this problem.

2.3. Method

Using the Hanoi bus map, we build a directed static graph. Each bus station corresponds to a vertex of the graph. Between two vertices (B, C) has a directed edge from B to C (B→C or BC) if B and C are two adjacent stations of a bus route (directly from B to C). The attributes of the edge BC are the names of buses going directly from B to C, the distance between B and C. The attributes on each vertex are the departure time of buses going from it to other vertices. An example of this graph is shown in Figure 1, the departure timetable of bus routes at stations is in Table 1.

Assume that the travel speed of the bus when there is no traffic jam is v_1, the travel speed of the bus when there is a traffic jam is v_2 ($v_2<v_1$). In Hanoi, there are many bus

stations on each street, the distance between two adjacent stations (one edge of the bus graph) is small. During rush hour, some streets are completely blocked. Therefore, during rush hour, if an edge belongs to a blocked street, the entire edge is congested, the bus travel speed on this edge is v_2.

The minimum time paths from the start station to the target station consists of both the bus waiting time at stations and the travel time of buses on roads (which may or may not be congested). To find the minimum time paths we have to consider the graph which has both edge weights (travel time between adjacent stations) and vertex weights (bus waiting time at stations). These weights change depending on the departure time of a customer.

Dijkstra's algorithm only finds shortest paths on graphs which have only edge weights, these weights do not change over time. For this reason, Dijkstra's algorithm can not find the minimum time paths for bus users in Hanoi where we have to compute travel time between adjacent stations (edge weights) and bus waiting time at stations (vertex weights) which depend on the departure time of a customer. So, we extend Dijkstra's algorithm to allow finding an optimal bus path.

2.3.1. Problem-solving idea

To find the minimum time path for bus user in Hanoi, we build a graph which has both edge weights and vertex weights. The weight of an edge is the bus travel time. The weight of a vertex is the bus waiting time.

The minimum time path from a source to a destination is a path which arrives earliest at the destination. The idea of Dijkstra's algorithm, the path from a source to a destination is optimal if all sub paths from the source to the vertices of this path are optimal. Based on this idea, we find the minimum time path whose sub paths are minimum time. The time to arrive at a vertex from the source is the sum of waiting time and travel time.

Because between a bus station and a next bus station, there can be many bus routes, we choose the earliest bus route to move (optimizing bus waiting time). The weight of an edge is the ratio of the edge length and the bus speed. If there is no traffic jam, the bus speed is v_1. If there is traffic jam, the bus speed is v_2. We use the edge weights to compute and optimize travel time.

For example, considering the edge BC:

- The earliest departure time at B to go to C: t0;
- The distance between B and C: d;
- The speed of the bus when there is no traffic jam: v1;
- The speed of the bus when there is a traffic jam: v2 (v2<v1).

If t_0 does not belong to the traffic jam time, the weight of BC is $t_0 + d/v1$, otherwise, the weight of BC is $t_0 + d/v2$.

To illustrate the problem-solving idea, we use the bus graph in Figure 1 and the bus departure time-table in Table 1. We assume the bus speed is v1 = 500 m/min, only CD is congested between 6 am and 7 am with v2 = 200m/min. We find the minimum time path for a customer who is B at 6:02 am, and wants to go from B to D.

Let

V: the set of vertices of the graph;

T: the set of vertices not considered;

L: the time when the customer travels from the start vertex to the current vertex; Prev: the previous adjacent vertex;

Bus: bus route chosen to go from the previous adjacent vertex to the current vertex.

Using the bus graph in Figure 1 and the bus departure time table in Table 1, we have the following initialization table:

Table 2. Initializing values of extended Dijkstra algorithm

V	B	C	D	G
T	B	C	D	G
L	0	+∞	+∞	+∞
Prev	∅	∅	∅	∅
Bus	∅	∅	∅	∅

Considering the starting vertex B (Figure 2)

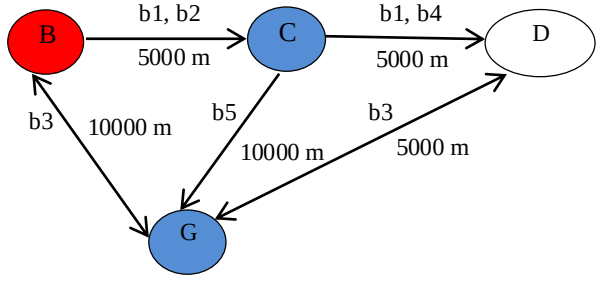

Figure 2. Considering vertex B

At B, the customer can go to C or go to G. Considering the edge B → C, because the customer arrivals B at 6h02 am, bus 2 is the earliest bus from B to C, its departure time is 6h05 am. BC is not congested, the travel time from B to C is BC/v1 = 5000/500 = 10 (minutes). So the customer arrivals C at 6h05 + 10 = 6h15.

Considering the edge B → G, there is only bus 3. Because the customer arrivals B at 6h02am, the earliest departure time is 6h10. BG is not congested, the travel time from B to G is BG/v1=10000/500 = 20 (minutes). Therefore the customer arrives at G at (6h10+20) = 6h30. Updating the values, we have Table 3. Based on this table, we see that the time to vertex C is earlier than vertex G, so in the next step we will choose vertex C.

Table 3. Results after considering vertex B

V	B	C	D	G
T		C	D	G
L	6h02	**6h15**	+∞	6h30
Prev	∅	B	∅	B
Bus	∅	2	∅	3

Considering vertex C (Figure 3)

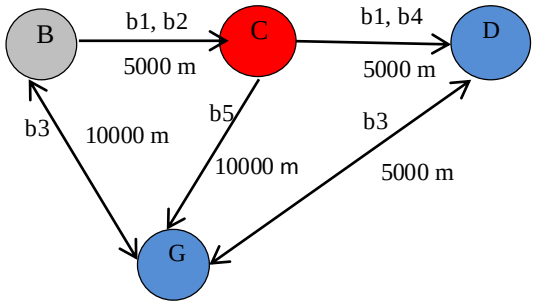

Figure 3. Considering vertex C

At C, the customer can go to D or go to G. Considering the edge C → D, because the customer arrivals C at 6h15 am, the earliest bus from C to D is bus 1, its departure time is 6h20. CD is congested between 6 am and 7 am, the travel time from C to D is CD/v2 = 5000/200 = 25 (minutes). So, the customer arrives at D at (6h20 + 25) = 6h45. In Table 3, at the intersection of row L and column D, the arrival time at D is +∞ (> 6h45), so we change +∞ to 6h45.

Considering arc C → G, there is only bus 5. Because the customer arrivals C at 6h15 am, the earliest departure time is 6h20. CG is not congested between 6 am and 7 am, the travel time from C to G is CG/v1 = 10000/500 = 20 (minutes). So, the customer arrives at G at (6h20+20) = 6h40. In Table 3, at the intersection of row L and column G, the arrival time at G is 6h30, this value is less than 6h40, so we do not change this value.

Updating changed values, we have Table 4. Based on this table, we see that the time to vertex G is earlier than vertex D, so in the next step we will choose vertex G to consider.

Table 4. Results after considering vertex C

V	B	C	D	G
T			D	G
L	6h02	6h15	6h45	**6h30**
Prev	∅	B	C	B
Bus	∅	2	1	3

Considering vertex G (Figure 4)

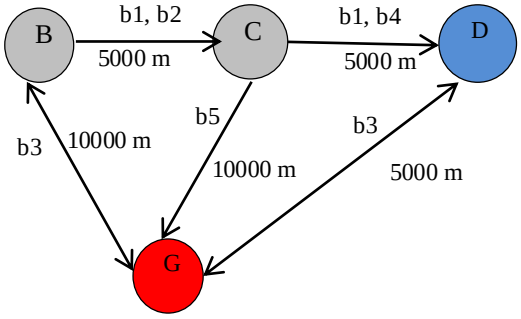

Figure 4. Considering G

At G, the customer can go to B or D. Since vertex B has been considered, in this step, we only consider vertex D. Considering arc G →D, there is only bus 3. Because the customer arrivals G at 6h30 am, the earliest departure time is 6h30. GD is not congested between 6 am and 7 am, the travel time from G to D is GD/v1 = 5000/500 = 10 (minutes). So, the customer arrives at D at (6h30 + 10) = 6h40. In Table 4, at the intersection of row L and column G, the arrival time at D is 6h45, this value is greater than 6h40, so we change 6h45 to 6h40. Then we have Table 5.

Table 5. Results after considering vertex G

V	B	C	D	G
T			D	
L	6h02	6h15	**6h40**	6h30
Prev	∅	B	G	B
Bus	∅	2	3	3

Destination D has been considered, so we stop the algorithm here. Thus, the distance that takes the least time is B → (3) G → (3) D (Figure 5), the customer arrives at D at 6h40.

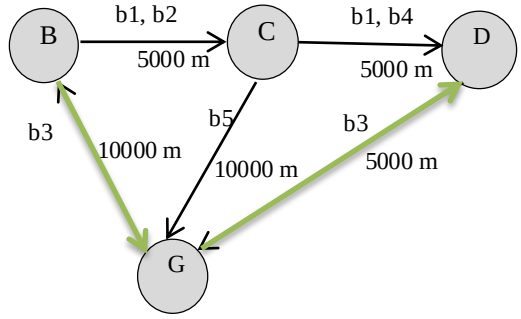

Figure 5. The minimum time path

In the next section, we will present how to choose the earliest bus to travel between two vertices and the extended Dijkstra algorithm.

2.3.2. Selecting the earliest bus

Let:

- d_{ik}: distance between vertex i and vertex k
- t_i: the time when the customer arrives at i
- b_{ik}: the earliest bus to travel from i to k
- t_k: the time when the customer arrives at k
- v_1: bus speed without traffic jam
- v_2: bus speed with traffic jam ($v_2<v_1$)

Suppose, the customer arrives at vertex i at t_i. The earliest_bus function is to select the earliest bus traveling from vertex i to vertex k. The earliest_function(i, k) is as follows:

Input: i, k, t_i

Output: b_{ik}, t_k

Step 1: b_{ik}=earliest bus to travel from i to k;

t=earliest departure time of b_{ik} at i ($t \geq t_i$);

Step 2: if (ik is congested) and (t_i belongs to the rush hour)

$t_k = t + d_{ik}/v_2$

else $t_k = t + d_{ik}/v_1$;

Step 3: Return b_{ik}, t_k;

2.3.3. Extended Dijkstra algorithm

We assume that we know the set of buses, departure time of buses at the stations, set of congested roads during rush hour, time of traffic jam, bus speed without traffic jam (v_1), bus speed with traffic jam (v_2, $v_2 < v_1$).

Let:

G = (V, E) is a single graph

V: set of all bus stations (vertices)

E: set of edges (connecting two adjacent bus stations)

d_{ik}: distance from vertex i to vertex k ($\forall i, k \in V$)

u: start vertex

v: target vertex

t_0: the time when the customer arrives at u

T: set of all the unvisited vertices

L[i]: time to go from the start vertex to vertex i ($\forall i \in V$)

Prev[i]: previous adjacent vertex of vertex i ($\forall i \in V$)

Bus[i]: the bus traveling from Prev[i] to vertex i ($\forall i \in V$)

**** Problem solving ideas***

Initializing, L[u] is 0, L[i] is $+\infty$ ($\forall i \in V$, $i \neq u$), the current vertex is the start vertex (u), T is the set of all the unvisited vertices, T= V.

For the current vertex, consider all adjacent vertices. For each adjacent vertex, use the earliest_bus function to select the earliest bus for going from the current vertex to the adjacent vertex. When all adjacent vertices are considered, select the adjacent vertex whose arrival time is minimum, mark it and remove it from T. If the target vertex (v) has been considered (i.e. not in T), then stop the algorithm. Otherwise, set the marked vertex to be the current vertex and go back to consider its adjacent vertices.

**** Extended Dijkstra algorithm***

Input: u, v, t_0

Output: shortest path (minimum time path) from u to v

Step 1: Set T = V; L[u] = 0; $t_u = t_0$; Prev[u] = $\varnothing$; Bus[u] = $\varnothing$;

for each vertex i in V\{u}:

L[i] = $+\infty$; Prev[i] = $\varnothing$; Bus[i] = $\varnothing$;

Step 2:

while (T is not empty) and ($v \in T$):

i $\leftarrow$ vertex in T with min L[i];

remove i from T;

for each neighbor k of i: // only k that are still in T
earliest_function(i, k);
if (L[k] > t_k) then:
L[k] = t_k; Prev[k] = i; Bus[k]=b_{ik};

Step 3: return L, Prev, Bus

2.4. Experimental study

Currently, in Hanoi, only a few stations have information about bus arrival times, so we cannot build a bus graph with complete vertex weights. We are not yet chosen a bus subgraph where all stations have information about bus arrival times. Our study will be applied when every bus station in Hanoi updates bus arrival times.

Experiments were implemented in the Java programming language. We run the demo application with the bus graph in Figure 1 and the timetable in Table 1. Bus speed without traffic jam is v_1 = 500m/min, bus speed with traffic jam is v_2 = 200m/min, only CD is congested between 6 am and 7 am. We find the minimum time path for a customer who wants to go from B to D. We demo with 2 cases: customer traveling is belongs to the rush hour, customer traveling is not belonged to the rush hour.

**** Case 1: Customer traveling is belongs to the rush hour***

Suppose a customer arrives at B at 6:02 a.m to go to D. In this case, when the customer travel from C to D, the CD road is congested. The result is in Figure 6.

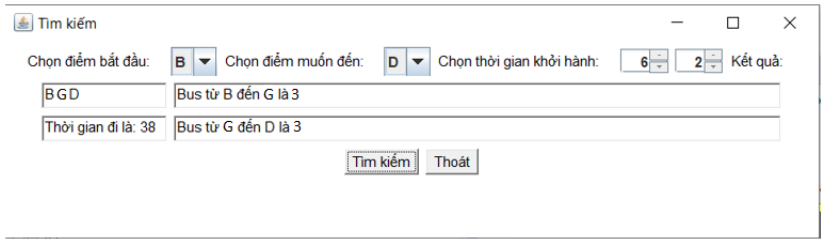

Figure 6. Customer traveling is belongs to the rush hour

**** Case 2: Customer traveling is not belonged to the rush hour***

Suppose the customer arrives at B at 6:50 am to go to D. In this case, when the customer travel from C to D, the CD road is not congested. The result is in Figure 7.

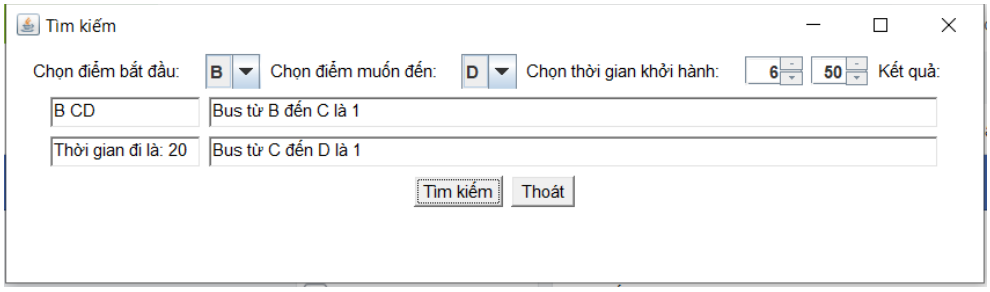

Figure 7. Customer traveling is not belonged to the rush hour

In both cases above, the Dijkstra's algorithm only finds the optimal path that is BCD. This path is not minimum time during the rush hour.

3. Conclusions

We have extended Dijkstra's algorithm to find the minimum time path for Hanoi bus users considering the traffic jam, waiting time and traveling time. However, we have not completed the data of the bus graph. We also want to develop an algorithm to find the optimal paths that satisfy simultaneously multiple constraints, e.g., minimum time, the minimum number of bus transfers, minimum distance, cheapest cost. Besides, we also want to develop a full application for customers who find optimal bus routes.

REFERENCES

[1] Spiess, H., and M. Florian, 1989. Optimal Strategies: A New Assignment Model for Transit Networks. *Transportation Research B*, Vol. 23B, No.2, pp. 83-102.

[2] Modesti, P., and A. Sciomachen, 1998. A Utility Measure for Finding Multiobjective Shortest Paths in Urban Multimodal Transportation Networks. *European Journal of Operational Research*, Vol. 111, pp. 495-508.

[3] Wu, Q., Hartley, J.K., 2004. Accommodating User Preferences in the Optimization of Public Transport Travel (extended version). *International Journal of Simulation Systems, Science & Technology: Applied Modelling & Simulation*, ISSN 1473-8031, pp 12-25.

[4] Antsfeld, L., and Walsh, T. 2012. Finding multi-criteria optimal paths in multi-modal public transportation networks using the transit algorithm. 19th world Congress on Intelligent Transport Systems,Vienna.

[5] Nguyrn Thi Kim Ngan, Tran Thi Thom and Dương Thi Kim Oanh, 2018. Building a solution to find the optimal bus route for Hanoi bus users. *HNUE Journal of Science*, Volume 63, Issue 11A, pp. 3-9 (in Vietnamese).

[6] Dijkstra, E. W, 1959, *A note on two problems in connexion with graphs*. Numerische Mathematik, 1, 269-271. DOI: 10.1007/BF01386390.

[7] Xe buyt noi thanh Ha Noi, https://vi.wikipedia.org/wiki/Xe_bu%C3%BDt_n%E1%BB%99i_th%C3%A0nh_H%C3%A0_N%E1%BB%99i.