Winter 12-1-2010

An Integrated Bus and Taxi Routes for a Mobile Trip Planning System

Shin-Shiang Lan

Kun-Ting Chen

Chien Chen

Jing-Ying Chen

Rong-Hong Jan

See next page for additional authors

Authors

Shin-Shiang Lan, Kun-Ting Chen, Chien Chen, Jing-Ying Chen, Rong-Hong Jan, and Da-Chun Chang

An Integrated Bus and Taxi Routes for a Mobile Trip Planning System

Shin-Shiang Lan[1], Kun-Ting Chen[1], Chien Chen[1], Jing-Ying Chen[1], Rong-Hong Jan[1], Da-Chun Chang[1,2]
[1]Department of Computer Science, National Chiao Tung University, Hsin-Chu, Taiwan
[2]Emerging Smart Technology Institute, Institute for Information Industry, Taipei, Taiwan
E-mail: {bluelewis.cs96g,quentin2007.cs96g}@g2.nctu.edu.tw,
{chienchen,jyc,rhjan,dcchang}@cs.nctu.edu.tw

Abstract

With the popular usage of Google Maps and smart phones, more and more people are using smart phones to surf and inquire about travel information. As a result, every major city plans to push the existing online public transportation trip planning system beyond traditional computer users to mobile phone users. The trip planning system is based on the starting and ending points that a user inputs, and guides the user to take a bus or metro through an electronic map interface. The system usually provides different kind of alternative travel routes with the estimated time of arrival. However, people who use the public transport system may encounter some uncertainties, such as long waiting times, long routes, long walking distances, etc. In each big city, the taxi is a universal transport vehicle which is available at almost anytime, anywhere. Taxis can save passengers' walking distance and travel time with a deficit of high cost. Therefore, we design a trip planning system to unify the Taipei public transportation system with taxis. The users can inquire of a travel route through the mobile phones. This system uses Google Maps as a base map. The users assign an upper limit of fare which they are willing to pay. The system will balance between travel time and travel cost to obtain a route which may combine usage of the bus and taxi. Because of the high density of bus stations in Taipei city, the route search may consume a lot of system resources. We propose an improvement method to eliminate some intermediate bus stations in route search processing.

Keywords: trip planning, bus information system, Google Maps, Taxi.

1. Introduction

In response to the global effort on the energy conservation and carbon dioxide emission reduction, we must encourage people to use public transportation systems instead of private cars. In order to reach more public transportation passengers, a trip planning system for the public transport system, which provides people with guidance for their trip just like a Global Positioning System (GPS) used in a private sedan, is needed. With the increase of cellphone users, designing a mobile trip planner for the billion cellphone users has a lot of potential impact to popular users of public transit. In a large metropolitan area, a trip planning system is used generally to guide the user to ride on the buses through an electronic map interface. Usually a user makes an inquiry to the trip planner for route information. A trip planner demonstrates a route with bus information and the estimated time of arrival. Some extra travel conditions can be specified in a trip planner to meet the user's demands, such as the shortest travel time, least number of transfers, walking distance limitation, transportation vehicle choice, etc. However, if the users' source and destination locations are far away from any bus station, the current trip planning system, such as Google Maps, will ask you to take a long walk which could take hours. Even if the users' locations are close to a bus station, the bus route could have a long waiting time. Sometime the user may have a very tight schedule. Those reasons may diminish the users' wish to travel by bus. On the contrary, when the users have a time constraint in their travel, the taxis in a city are the transportation vehicles which may supply the users' needs, but have a relatively high cost. If the users do not want to spend much money, but also want to achieve a reasonable travel time, they can plan a trip with a combination of bus and taxi routes. Therefore, the present paper proposes to unite bus and taxi routes in a trip planning system. It lets the users make the best choice to meet their travel time and cost constraints.

Our goal is to design a seamless mobile trip planning system to combine the Taipei bus and taxi route information. Like most modern travel planning systems, we use Web 2.0 to achieve an interactive and integrative application. Our system uses Google Maps as the base map. The system interacts with user by AJAX technological development's Google Map API. Simultaneously, we also design an Android handset client program to allow the users to plan their trip using a mobile device. Taipei's bus density is quite high. The number of bus stations in Taipei is much larger than the number of bus routes. If we take the transportation network path finding method in [1], each bus station must contain the arrival time information which creates a huge memory usage requirement and causes low system efficiency.

Therefore in this paper we propose an enhanced path finding method, which can dynamically produce data, and which may abbreviate some of the station's information after processing, enabling the data processing quantity to be reduced effectively. The remainder of this paper is organized as follows. Section II compares existing trip planning systems in several major cities as well as some related work on path finding algorithms. Section III presents our system architecture. Subsequently, Section IV proposes an effective algorithm to reduce memory requirement of the trip planner. Our path finding algorithm to integrate bus and taxi routes is portrayed in section V. Section VI presents our emulation results, and Section VII concludes the paper.

2. Related Works

We survey several trip planning systems used in various oversea and domestic cities. We compare their functionality through detailed observation and hand on exercise. Overseas cities have New York City [2], Chicago [3], Paris [4], and London [5]; domestic city has Taipei [6]. In addition, we also evaluate the global system Google Maps [7] and the local system UrMap [8] in Taiwan.

Remarkably, New York map demonstrates a 2D and 3D option. London has support for nineteen languages and bike routes. Paris has a fashion web design and allows viewing of incredible details. Google Maps can show the street camera views. We glance over the domestic and foreign systems and summary functionalities of existing systems and our trip planner in Table 1, where we explain each kind of functions as followings:

- Fastest: Find a path with a shortest arrival time
- Walk: The walking distance user can tolerate
- Transfers: Find a path with minimal transfers
- Waiting time: The waiting time user can tolerate
- Cost: The expense user willing to pay
- Map: Have a map to guide the user
- 3D: Have a 3D map interface
- Taxi: Support taxi

As shown in Table 1, the majority of trip planning systems do not have the Cost and Waiting time options. Most of overseas systems implement Walk and the Transfers function. Only two systems have more powerful 3D maps. Our system implements the complete collection of functions, except for the 3D option. Most importantly, we add taxi route into the trip planner, which others cities don't have.

	Fastest	Walk	Cost	Transfers	Waiting	Map	3D	Taxi
New York	O	O	O	O	X	O	O	X
Chicago	O	O	O	O	X	X	X	X
London	O	O	X	O	X	O	X	X
Paris	O	O	X	O	X	O	O	X
Taipei	O	X	X	X	X	O	X	X
UrMap	O	X	X	X	X	O	X	X
Google	O	X	X	X	X	O	X	X
Ours	O	O	O	O	O	O	X	O

Table 1: The functionality of different trip planning systems

In term of path finding algorithms, Florian [9] proposes a shortest time-dependent path algorithm based on Dijkstra's shortest path algorithm. Florian's work considers different routes and transfer station traffic [10][11] at different times in the planning of paths in the public transit network. Moreover, Florian proposes a dynamic module for the public transmit network and models the route with data structures [12],[13]. Huang [1] modifies Dijkstra's shortest path algorithm and presents a path finding algorithm using pattern first search (PFS) approach. Huang sets the time for the source and computes the shortest time-dependent path reachable from source to destination.

Huang's method is built on a graph model, i.e. public transit system graph NET(L, N, S) that consists of bus routes (Line), transfer stations (Node), and stations which cannot transfer (Stop). For each station the bus routes that go through it are recorded, and for each bus route the stations that are on the routes in which the bus passes are also recorded. Each station contains one timestamp for each bus route, as well as the following quantities:

- t: The time at which the user arrives
- r: The bus line which the user takes to this station
- x: The previous transfer station at which the user takes bus line r
- t^x: The departure time at which the user leaves transfer station x

The algorithm needs to define a temporary node set, which records the nodes to update for the next step. It sorts the time value t of nodes in this set, sets the departure time on the source node, and sets infinite time for the others. The algorithm starts at the source node and searches the neighbors reachable from the source node. A new arrival time is calculated and compared with the original time for all the nodes through which the bus line r passes and which are downward nodes from the source. If the new time is shorter than original one, the algorithm updates the station's timestamp (t, r, x, t^x) with the new time value and adds the node to the

set. At each next step, the node with the minimum time in the node set is picked and the process repeats until the destination is found or the set becomes empty.

Note that when the number of Stops is excessively large, the amount of data to process will also increase. We found that when we consider only the bus lines and transfer stations without the stations that cannot be used for transfer, it can reduce unnecessary data processing. We will explain in more details about this reduction in the section 4.

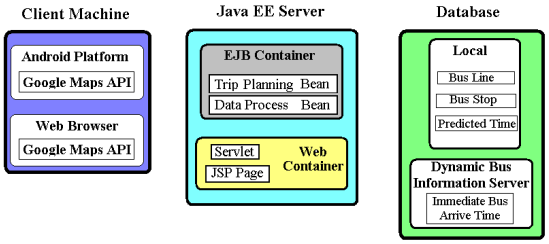

Figure 1: System Architecture

3. System Model

Our trip planning system allows the users to plan for trip with Web browsers as well as with handset running the Android platform. Both the server program and client-side programs are implemented in JAVA technology, which supports both client-side and server-side application development and avoids the need to deal with multiple programming languages during development process. When using the trip planning system, users mark the source and destination points on the map interface, and the system supports various timing constraints that can be specified by the users. People can change their preferences and priorities in different situations. We further include taxi as an additional option to the available transportation types.

The trip planning system contains three main components: the client interface, the travel planning module, and the database. The client interface provides viewing and option setting operations, the trip planning server program implements the trip planning algorithm, and the database includes both static route data and real-time bus and traffic data retrieved from a remote transportation information server. Figure 1 shows the system architecture.

Figure 2.1: Client Interface Model

Figure 2.2: Client Interface Model

The Web client, as shown in Figure 2.1, uses Google Maps as the base interface. The user interaction is further enhanced using the Ajax framework EXT JS [14]. The other handset-based client is based on the Android platform as shown in Figure 2.2. Both clients support timing options such as waiting time and walking distance when planning trips. The system returns a computed path according to the options users select, such as the shortest path with constraints to minimize the waiting time, or the maximum walking distance allowed. We denote different colors for different bus lines and show them on the map interface.

The trip planning module considers both the bus and the taxi to compute the shortest time-dependent path, and returns the resulting path, the transfer station information, and an estimate of total traveling time and arrival time. We will describe the planning algorithm in more details later.

Finally, the database server is based on PostgreSQL and includes PostGIS which provides extensions and many plug-in libraries. We use Taipei public bus stations and lines information in our trip planning algorithm [15] supported by the Taipei city government. When trip planning algorithm is running, we also connect to the remote database server to obtain immediate Taipei public bus information [15].

4. Enhance Method

When the number of public bus stations becomes too large and the bus lines passing them become too dense, the amount of data to process grows tremendously. We propose a method that improves on Huang's algorithm by alleviating some unnecessary computation and reducing the memory usage. The idea is that, generally speaking, the number of stations is more than the number of bus lines, and each station has many bus lines passing through it. Take Taipei public bus transportation as an example, the number of bus lines is 429, while the number of stations is 6782.

Suppose there are two bus stations in public bus line. If the bus lines which pass the downward bus station are the subset of the bus lines which pass the upward station, and moreover, if the bus line passing these two stations goes through the same path between the two stations, it indicates that transfers in downward station is unnecessary, since taking the upward station as transfer station is enough.

For example, Figure 3 shows A, B, C, and D as transfer stations. L1, L2, L3 represent three different bus lines. Take stations A and B as an example, the bus lines passing through A and B go through the same path between A and B, and B is not a destination point. The line set at node B {L1, L2} is a subset of line set in node A, that is, {L1, L2, L3}, which means no transfer in node B is necessary, and we can eliminate the B transfer node.

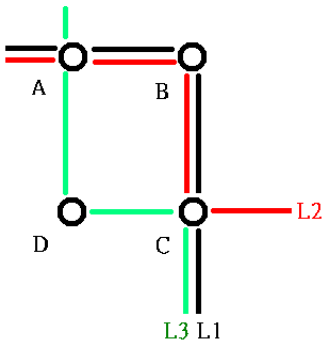

Figure 3: Bus Line Example

4.1 Algorithm

The modified algorithm is as follows. The public transit system graph NET(L, N) consists of the transfer station nodes N, and bus lines L. We use the Huang's method to do path searching and use similar parameters (t, r, x, t^x) as in Huang's. Step (6) is modified with the above mentioned method. During initialization, i.e., in step (1), we start only with the initial source node, unlike the Huang's method which initializes all the transfer nodes in the beginning, and thus the memory usage is less in our algorithm. The algorithm is summarized as follows:

(1) Initialize source node N_S(t_0, null, null, ∞), the set of active lines serving the node, and the set of active nodes serving the line, Set T←{N_S}
(2) If T is an empty set, stop. Destination is unreachable.
(3) Select a node N_j : for all N_j ∈T, such that t_j is minimal
(4) If N_j = N_D, stop, destination arrived.
(5) For each line p' which passes N_j in addition to the line p that has time stamp t_j, let v' be the set of such p' that satisfies the waiting time condition from the user.
(6) If v' is not null, then for each downward node N', if the lines of N' is the subset of the lines of N_j, then it doesn't create time stamp on N';
else, get a new arrival time t^*= travel(p', N_j→N'), If t^*+ t_j < t' ,
then update n' by t'=t^*+ t_j, T←T ∪ { N'}
(7) T ← T − { N_j }
(8) Go to step (2).

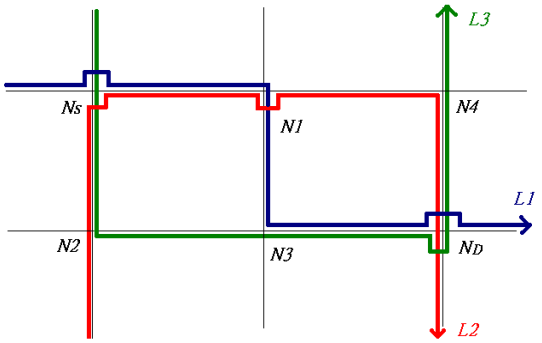

Figure 4: Example

4.2 Example

Figure 4 illustrates the bus transit system graph, including three bus lines (L1, L2, L3) and six bus stations (N_S, N_D, N1, N2, N3, N4). First, the system initializes the bus station set for each bus line and the bus line set for each bus station. We set the departure time at NS as 7:00.

Step 1:
The line set on NS is {L1, L2, L3}:
The station set on L1 is { N_S, N1, N3, N_D}, and we consider only downward stations {N1, N3, N_D}. The line set on N1 is {L1, L2}, which is the subset of line set on NS, so it doesn't update. The line set on N3 is {L1, L3}, and it is the subset of line set on NS, so it doesn't update. N_D is the destination, so it calculates the time from N_S to N_D through line L1, create timestamp on N_D. Figure 5 (a) shows the end of examining L1.
The station set on L2 is {N2, N_S, N1, N4, N_D}, and we consider only downward stations {N1, N4, N_D}. The line set on N1 is {L1, L2}, which is the subset of line set on N_S, so it doesn't update. The line set on N4 is {L2, L3}, which is the subset of line set on N_S, so it doesn't update. N_D is the destination, and since the time from N_S to N_D through route L2 is more than the original time, so it doesn't update. Figure 5 (b) shows the end of examining L2.
The station set on L3 is {N_S, N2, N3, N_D, N4}, and we consider only downward stations {N2, N3, N_D}. The route set on N2 is {L2, L3}, which is the subset of line set on N_S, so it doesn't update. The line set on N3 is {L1, L3}, which is the subset of

line set on N_S, so it doesn't update. N_D is the destination, the time from N_S to N_D through route L3 is more than the original time, so it doesn't update. Figure 5 (c) shows the end of examining L3

Step 2:

Take the node N_D from the set, finish.

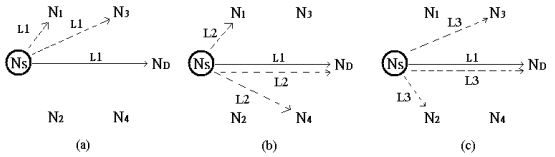

Figure 5: State

5. Path Finding Algorithm Combining Taxi and Bus

In order to have more versatile routes in the planning of trip and to provide more flexible choices for different users' preferences and priority with respect to time, we consider adding taxi support in addition to taking public transportations. We consider the case where a taxi can connect two bus lines which do not intersect, and thus it allows the user to have more trip planning options. Accordingly, in the traffic network illustrated in section 4, in addition to the public transit routes, we add the taxi edges between each pair of transfer stations.

When performing search on the trip planning algorithm, stations except the source and the destination node can have many timestamps. The source and the destination nodes have only one timestamp. For example, suppose the user sets the budget condition. Before the algorithm reaches the destination node, the accumulated cost may exceed the user's budget if taking into account only the shortest time-dependent path, so the transfer station records both the accumulated cost and time conditions. There are two kinds of situations in such transfer station records: either having less time with higher budget or longer time but lower budget.

When performing search on the trip planning algorithm, due to various user conditions and the chosen optimization condition, for example, suppose the user sets the budget upper-bound condition and should search for shortest time-dependent path. Before the planning trip reaches the destination node, the accumulated cost may exceed the user budget condition if taking into account only the shortest time-dependent path, so the transfer station records both the accumulated cost and time conditions.

The algorithm updates the path records in the transfer station depending on the computed time and budget for the current path. We consider three cases. In the first case, it deletes all the worse path records and inserts the new record if the new record has better arrival time and the budget is within upper-bound compared with the path records stored in the station. In the second case, it doesn't update if the arrival time is better than the stored path records but the budget is over upper-bound condition. Finally, if it is not either of the above situations, for example, the arrival time is worse even if the budget is under the bound, it still doesn't update the records.

5.1 Algorithm:

The algorithm considers the routes with respect to bus, walking, and taxi, and computes the arrival time and budget. The detail is as follows.

(1) Under the distance of walk condition, starting from source, search for those stations that are reachable by walking, and those stations that are reachable by taxi under the budget condition.

(2) Find the bus lines set for each station obtained from (1), for each bus line, if the stations it passes are those in (1), update the records on the stations.

(3) For stations from step (2), search for stations that are reachable by walking, check if it can update the time stamp in the records.

(4) From stations by step (2), search for stations that are reachable by taxi, check if it can update the time stamps in the records.

(5) Add the records added from (2), (3), and (4) to record set.

(6) If there is nothing in the record set or the record on the end node, finish.

(7) Take the records from record set in non-decreasing order. Take the node on the record as the start node, if it is the taxi that is the previous path, then neglect (9) (10).

(8) Find the bus lines that pass through the start station by step (7). Check if it can update the records on the stations on the bus route.

(9) For the stations on the records by step (7), search for stations that are reachable by walking, check if it can update records on those stations.

(10) For station on the records by step (7), search for stations that are reachable by taxi, check if it can update records on those stations.

(11) Add records from (8), (9), and (10) to record set, go to (6).

We use an example to illustrate the extended algorithm. In Figure 6, suppose the budget condition is 90, the source node is S, and the destination node is D. N1, N2, N3 are stations.

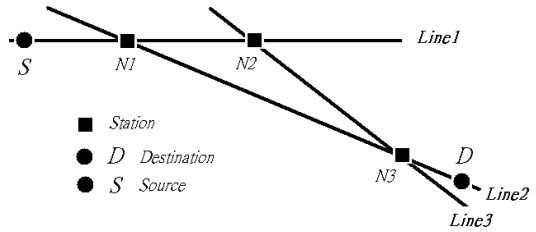

Figure 6: Example

Step 1:

From the source node, S can take bus to N1 and N2, and can take taxi to N1, N2 while still under budget condition. The result is as Table 2.

Record	Previous	Line	Node	Time	Cost
1	0	L1	N1	7:15	15
2	0	Taxi	N1	7:05	70
3	0	L1	N2	7:20	15
4	0	Taxi	N2	7:07	75

Table 2: Step 1

Step 2:

Take record 2 with the smallest time stamp. Node N1 can take bus line L1 to N3 and D. The additional record is as Table 3.

Record	Previous	Line	Node	Time	Cost
5	2	L2	N3	7:21	85
6	2	L2	D	7:25	85

Table 3: Step 2

Step 3:

Take record 4 with the smallest time stamp. Node N_2 can take bus line L2 to N3. The additional record is as Table 4.

Record	Previous	Line	Node	Time	Cost
7	4	L2	N3	7:23	85

Table 4: Step 3

Step 4:

Take record 1 with the smallest time stamp. Node N1 can take bus line L2 to N3 and D, also can take taxi to N2, N3 and D under budget condition. The newly added record is as Table 5.

Record	Previous	Line	Node	Time	Cost
8	1	Taxi	D	7:19	85
9	1	Taxi	N3	7:20	85

Table 5: Step 4

Step 5:

Take record 6 whose node is D, finish.

Record	Previous	Line	Node	Time	Cost
1	0	L1	N1	7:15	15
8	1	Taxi	D	7:19	85

Table 6: Step 5

From Table 6 we know that the shortest time path is from S to N1 by bus line L1 at arrival time 7:15, and then from N1 to D by taxi at arrival time 7:19.

6. Emulation

In order to understand our system's functionalities, we have made a series of emulations. We take a sample area of 3.735 kilometer-long and 2.632 kilometer-wide rectangular regions in the center of Taipei, and randomly create sources and destinations with distances away from each other limited by 2, 3, and 4 kilometers. We also randomly select two end points located on any part of the Taipei metropolitan area with distance away between 5 and 10 kilometers. Each test result is obtained by averaging 100 times of emulation for each kind of distance respectively.

First, we would like to know whether our trip planner can find a quicker path if the user is willing to walk for a little bit longer. The emulation employs the same bus waiting time, and a 4 kilometers/hour walking speed. The trip is for bus only. Figure 7 presents the successful probability of planning a trip versus walk distance. The result shows that when the user is willing to walk a longer distance, the opportunity for finding a path is higher in Taipei city center. However, if the source and destination are not in the city center, such as the results for the 5 and 10 kilometers, the chance of finding bus station is much lower, even the user is willing to walk 500 meters. Therefore, integrating taxi into a trip planning system can be very helpful to encourage people to plan some part of their trip to involve public buses.

Then, our following emulations are for planning a trip with bus and taxi information. Our goal is to show the relation of cost limit, and the successful rate of planning a trip and travel time. Apparently,

the curves in Figure 8 depict that when we combine bus and taxi information with higher upper limits of cost, the chance of finding a path is much higher. The 10 kilometer case eventually will reach a 100% successful rate. Figure 9 further shows when the cost upper limit is higher; the time to arrive at the destination is quicker. Of course, if you don't care how much you spend on the transportation, our trip planner will give you a shortest time taxi route. In 2 kilometers case, the fare surpasses 85 upper limits; you will be given a direct route to the destination by the taxi.

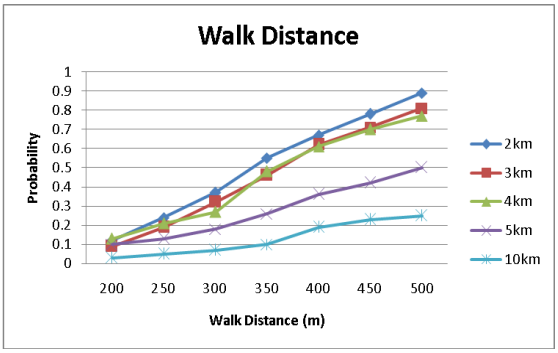

Figure 7 Successful rate verse walking distance

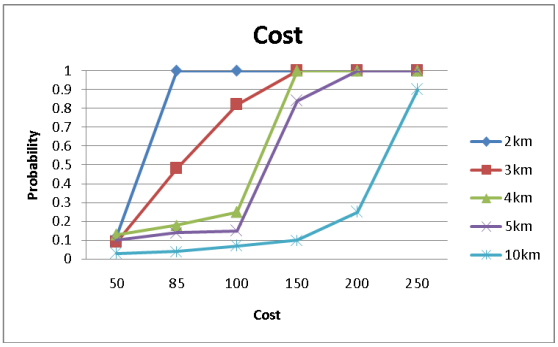

Figure 8 Succesful rate verse cost limit

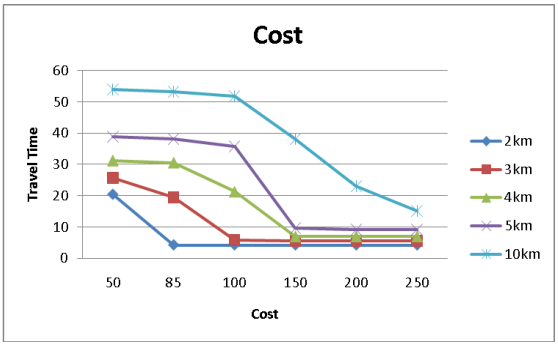

Figure 9 Succesful rate verse cost limit

7. Conclusion

This paper designs and implements a mobile trip planning system for public transportation systems in the Taipei metropolitan area. Since taxis are an important transportation vehicle in almost every major city, we propose to blend the taxi routes into the public transportation trip planning system. It has an advantage of fulfilling user constraints in both travel time and transportation cost. Thus, the people who are located far away from bus stations can still plan part of their trip using buses. Moreover, we propose an enhanced method to reduce the system memory requirement and increase request processing speed. Currently, we are testing our integrated trip planning system with the addition of Taipei Metro Rapid Transit (MRT) system and bicycle routes. People can specify how many calories they would like to burn using a bicycle for their trip. Our system will plan a route to combine MRT, bus, and bicycle accordingly. In the future, we will integrate our system with taxi operators to achieve a seamless transfer. Our final goal is to establish an integrated mobile trip planning system for seamless transportation networks.

Acknowledgements

This study is conducted under the "III Innovative and Prospective Technologies Project" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China.

References

[1] R. Huang, "A Schedule-based Pathfinding Algorithm for Transit Networks Using Pattern First Search," Geoinformatica, vol.1, no. 2, pp.269-285, June 2007.
[2] MTA NYC Transit - Trip Planner http://tripplanner.mta.info/
[3] Plan Your Trip With the RTA! http://tripsweb.rtachicago.com/
[4] RATP Transports en île de France http://www.ratp.info/touristes/
[5] Transport for London: http://www.tfl.gov.uk/
[6] 5284 front page: http://5284.taipei.gov.tw/
[7] Google Maps API http://code.google.com/intl/zh-HK/apis/maps
[8] UrMap: http://www.urmap.com/
[9] M. Florian, "Finding shortest time-dependent paths in schedule-based transit networks: a label setting algorithm," in Niguel H.M. Wilson and Agostino Nuzzolo (Eds.), Schedule-based Dynamic Transit Modeling: Theory and Applications, pp. 43–53, Dordrecht Kluwer, 2004.
[10] R. Huang and Z.-R. Peng, "An object-oriented GIS data model for transit trip planning system," in TRB, National Research Council (Eds.), Transportation Research Record, no. 1804, pp. 205–211, TRB, National Research Council, Washington DC, 2002.
[11] R. Huang and Z. Peng, "A spatiotemporal

data model for dynamic transit networks," International Journal of Geographic Information Science, vol. 22, no. 5, pp. 527-545, 2008.

[12] F. Russo, "Schedule-based dynamic assignment models for public transport networks," in Niguel H.M. Wilson and Agostino Nuzzolo (Eds.), Schedule-based Dynamic Transit Modeling: Theory and Applications, pp. 79–93, Dordrecht Kluwer, 2004.

[13] C.O. Tong and S.C. Wang, "Minimum path algorithms for a schedule-based transit network with a general fare structure," in Niguel H.M. Wilson and Agostino Nuzzolo (Eds.), Schedule-based Dynamic Transit Modeling: Theory and Applications, pp. 241–261, Dordrecht Kluwer, 2004.

[14] Ext JS Cross-Browser Rich Internet Application Framework http://www.sencha.com/products/js/

[15] Taipei City ATIS Web: http://its.taipei.gov.tw/