Route-Information Management and Provision for Public Transportation Systems

Chao-Lin Liu[†] Tun-Wen Pai[‡] Shang-Ming Huang[†] Chun-Tien Chang[‡]

[†]Department of Computer Science
National Chengchi University
Wen-Shan, Taipei 11605, Taiwan
chaolin@nccu.edu.tw

[‡]Department of Computer Science
National Taiwan Ocean University
Chung-Cheng, Keelung 20224, Taiwan
twp@cs.ntou.edu.tw

Abstract

We propose a framework for management of route information and provision of travel information in the context of public transportation systems. The information-management component combines information collected from diverse sources, and processes the information for the interests of service providers and travelers. Travelers may place queries via different types of devices and interfaces, and receives travel information in HTML, WML, graphic, and audio formats. The information-provision component computes the desired information and converts the information into appropriate output formats. In particular, we present two algorithms that take advantage of special characteristics of public transportation systems to compute travel plans efficiently. We discuss possible ways for integrating our algorithms with existing path-planning algorithms for prioritizing alternative travels plans based on time-independent and time-dependent costs. Results and feedbacks collected from an open field test, that is available at <http://iris.cs.ntou.edu.tw>*, indicate that our algorithms can compute satisfactory travel plans within a couple of seconds for a public transportation system with approximately 280 routes serving 2500 stops.*

INTRODUCTION

Public transportation systems have become an indispensable part of a modern metropolis. By providing a complex service network that covers the metropolis, public transportation systems have made commuting so convenient that driving has become a less desirable alternative. This in turn contributes to the reduction of gasoline consumption and air pollution—two of the major goals of intelligent transportation systems.

Most people take advantage of public transportation systems only for recurrent trips, however. For non-recurrent trips, it is quite hard even for local people to find out how to travel between unfamiliar locations via the public transportation systems. Ironically, this phenomenon is not a result of the unavailability of services, but typically results from the complexity of the service networks of the public transportation systems. This paper presents a framework for facilitating information management and provision for public transportation systems, hoping to further encourage people to use public transportation systems.

Our system serves both service providers and commuters. The framework includes two major components, as shown in the following figure. The *route-information manager* allows service providers to monitor and manage the route information when necessary. This manager

will also relay information about real-time vehicle locations to message boards located at bus stations. (For brevity, we will not distinguish trains, ground buses, and massive rapid transportation vehicles now.) Our system allows various ways for specifying the queries for travel plans, and returns the travel plans that are computed by the *travel-plan finder*.

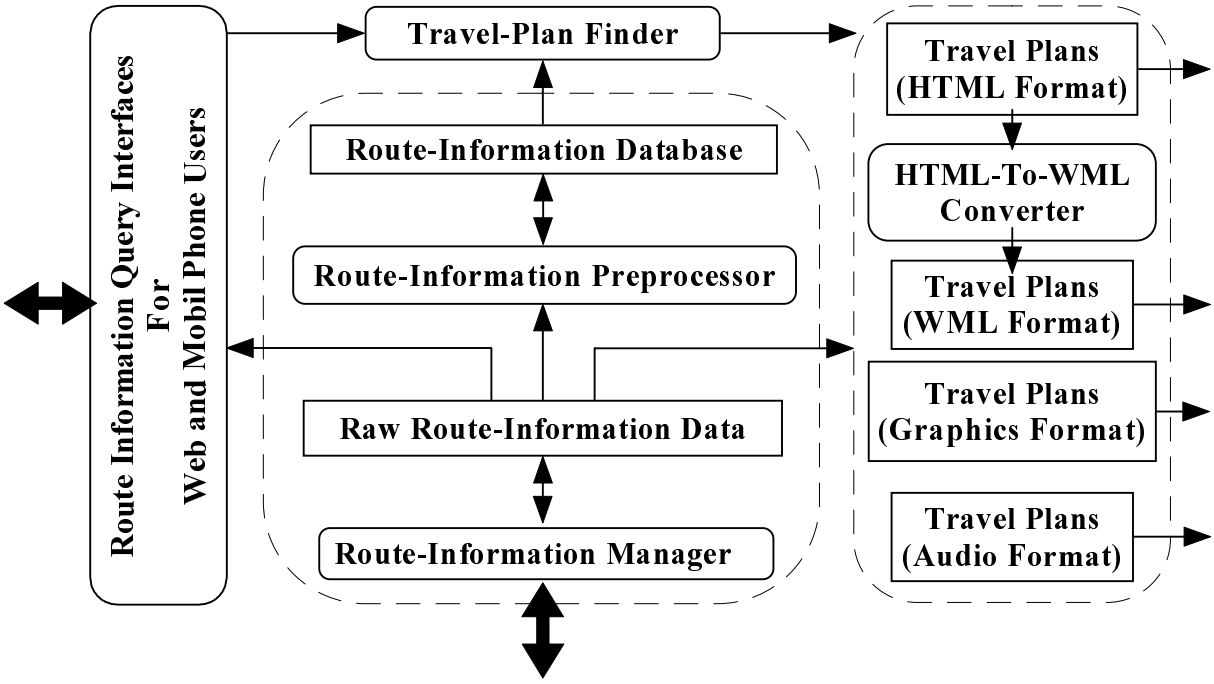

Figure 1. System Block Diagram

A collection of path-planning algorithms form the core of the travel-plan finder. We present algorithms that compute travel plans between two locations, and discuss algorithms that prioritize alternative paths based on time-dependent costs. Our algorithms for computing travel plans differ from traditional shortest-path planning algorithms, both label-setting and label-correcting algorithms [1, 13], in that we explicitly model the constraint imposed by the fact that public buses serve on pre-selected paths. Explicitly modeling and taking advantage of such *route constraint* allow ourselves a chance to find travel plan more efficiently than otherwise.

It is typical that there are multiple alternative ways to travel from one location to another. Hence, there is a strong demand for ordering the alternative plans based on users' preferences. Users of public transportation systems may have a wide variety of preferences, however [2]. Techniques employed by traditional shortest-path algorithms suffice to prioritize travel plans only with time-independent costs [5]. We discuss algorithms useful for prioritizing travel plans with time-dependent costs based on the notion of *stochastic consistency* [14].

We introduce the route-information manager and the preprocessing of route information in the next section. We then spend two sections on deliberation of the design of route planning and prioritizing algorithms. We conclude this paper after discussing the user interface and extended applications of our system.

INFORMATION MANAGEMENT

The management of route information consists of two components. The *route-information manager* allows service providers to monitor and manage operation-related information, and the *route-information preprocessor* converts raw route-information data into a database of information tailored for searching travel plans.

ROUTE-INFORMATION MANAGER

The route-information manager takes as input information from several sources, and integrates and processes the raw data into several useful formats. The information may come from the request to update route information when buses are to be rerouted. The information may also be real-time operation data, such as current locations of vehicles in service [8], the actual number of passengers on a bus, the actual travel time between bus stops, and the number of people waiting at the stops.

A very important function of the route-information manager is to allow service providers to update the route information. Once in while, the bus services may be interrupted for some unexpected reasons, and service providers may need to change the routes of some bus services temporarily. No matter what reasons that cause these changes of bus services, a route-information provision system must attempt to instantly reflect such changes in its recommended travel plans. Therefore, the route-information manager ought to offer a secure and distributed interface on the Internet so that service providers may conveniently update route information.

The route-information manager also disseminates appropriate information to users of different needs. Service providers can lay out their business strategies based on the utilization of their services [4]. For instance, one may adjust the frequencies of bus services if the demand for that bus route has changed significantly either temporarily or over an extended period. Information about real-time bus locations is very important for the implementation of bus priority systems [8, 12]. Passing current locations of buses to people who are waiting at the bus stops helps to alleviate potential anxiety, thereby improving the service quality.

ROUTE-INFORMATION PREPROCESSING

To facilitate efficient search of travel plans of interest, the route-information preprocessor specifically compile the raw route-information data into a database for the path-planning task. We illustrate the definitions of the data formats and basic terms used in our algorithms in a simplified context shown in the following figure. Each black circle represents a group of stops that serve the marked locations, and directed lines represent service routes. For clarity, we do not distinguish stop names and location names until later in this paper.

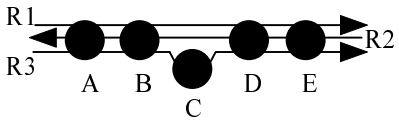

Figure 2. Route information assignment

We assign a number to each different service route. Service routes, e.g., $R1$ and $R3$, that do not serve the same set of stops will be assigned different numbers. Service routes, e.g., $R1$ and $R2$, that serve the same set of stops but not in the same order will get different numbers.

We also assign an ordinal number to each stop on a route. Smaller numbers will be assigned to stops that are served earlier on a bus route, and we use $K(r, s)$ to denote the number assigned to the stop s on the route r. When a route r does not serve a stop s, we set $K(r, s) = 0$. Assume that the route number of $R1$, $R2$, and $R3$ are 1, 2, and 3, respectively. If $K(1, \mathsf{A}) = l$, $K(2, \mathsf{A}) = m$, and $K(3, \mathsf{A}) = n$, then we will also have $K(1, \mathsf{B}) = l + 1$, $K(2, \mathsf{B}) = m - 1$, and $K(3, \mathsf{B}) = n + 1$. Our algorithms use these numbers to determine whether one may travel from one stop to another on a particular route.

Our algorithms employ two functions that relate bus stops and bus routes. The algorithms need to know the set of service routes, *SR*(s), that serve a given bus stop s. For instance,

SR(B) $= \{1, 2, 3\}$ and *SR*(C) $= \{3\}$ in Figure 2. The algorithms also need to know the bus stops that are served by two routes, say $r1$ and $r2$, and we denote these common stops by *CS*$(r1, r2)$. For instance, *CS*$(1, 2) = \{$A, B, D, E$\}$. We can compute *SR*(s) and *CS*$(r1, r2)$ from $K(r, s)$:

$$\begin{aligned} \textit{SR}(s) &= \{r | K(r, s) > 0\} \qquad \text{and} \\ \textit{CS}(r1, r2) &= \{s | K(r1, s) > 0 \text{ and } K(r2, s) > 0\}. \end{aligned}$$

CONNECTIVITY MATRICES

In a related work, we propose *connectivity matrices* as the basis for a path-planning algorithm designed for the context of public transportation systems [10]. Connectivity matrices, produced by the route-information preprocessor, play a key role in our algorithms for finding bus connections between locations. Let i and j be the numbers assigned to two service routes. We set a cell, say $T_{i,j}$, of a connectivity matrix T to the number of stops in *CS*(i, j). Namely, $T_{i,j}$ is the number of ways that we can transfer from route i to j. We set $T_{i,i}$ to 0, although route i and itself obviously have common stops.

Consider the network shown in the following figure. We assume that we may transfer from one route to another at intersections. Therefore, $T_{1,3} = 1$ and $T_{2,4} = 1$ indicate that we can transfer from R1 to R3 and from R2 to R4, respectively. The information encoded in the connectivity matrices can be misleading, however. For instance, $T_{1,4} = 1$ suggests that we may directly transfer from route $R1$ to $R4$ no matter where we catch a bus on route $R1$. In fact, we *cannot* transfer directly from $R1$ to $R4$ once we pass C.

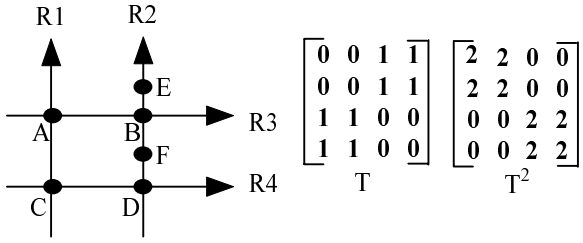

Figure 3. A portion of a transportation network and its connectivity matrices

The values of $T^k_{i,j}$ in the k^{th} power of T are related to the numbers of ways that we can transfer between routes i and j via k transfers, assuming that we compute powers of matrices with the following standard formula.

$$T^{k+1}_{i,j} = \sum_l T^k_{i,l} * T_{l,j} \tag{1}$$

In Figure 3, $T^2_{1,2} = 2$ indicates that we have two ways to transfer from R1 to R2 by two transfers. The first method is to transfer from R1 to R4 at C and from R4 to R2 at D, and the second method is to transfer from R1 to R3 at A and from R3 to R4 at B. However, we need to be careful in interpreting the numbers. $T^2_{1,2} = 2$ does *not* mean that we may travel from any locations served by $R1$ to those served by $R2$ by two transfers. Although we may go to E from C by transferring at A and B, we cannot go to F from C via $R1$, $R3$, and then $R2$. Furthermore, notice that we cannot transfer from $R2$ to $R1$ at all, but we have $T^2_{2,1} = 2$.

The key is that being able to transfer from a route X to another route Y does not guarantee that we may travel from all locations served by X to all locations served by Y. Computing the powers of connectivity matrices with Equation (1) ignores the service directions of the routes and relative locations of stops on the routes. Consequently, numbers in a connectivity matrix may exaggerate the number of ways that one may travel from one location to another.

We can make the information encoded in T^2 precise by calculating T^2 with an improved mechanism. In reality, there is exactly one way to transfer from route i to j via route r for each unique pair of stops $s \in CS(i, r)$ and $t \in CS(r, j)$ such that the following condition holds.

$$K(r, s) < K(r, t) \tag{2}$$

If the inequality reverses, we would have to travel on route r in its opposite service direction to transfer from i to j, which is not allowed in practice. Applying (2), we will obtain the correct T^2 for Figure 3.

Notice that we need to check the feasibility of travel plans even if we compute connectivity matrices precisely. There are three possible ways to transfer from route i to j via r in the following figure, i.e., transferring at location pairs (B, G), (B, H), and (D, H). Not all of them may constitute feasible travel plans for all needs to travel from locations served by i to locations served by j. If we are to travel from A to H, all three alternatives will work. Only one of these

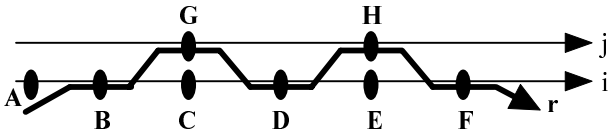

Figure 4. A difficult example for computing $T^2_{i,j}$

alternatives is right for traveling from C to H because we cannot travel backwards from C to B on i, as is implied by $K(r, \mathsf{B}) < K(r, \mathsf{C})$. Hence, when the connectivity matrix T^2 suggests that we may transfer from one service route to another, we need to double check the K values of the origin, the destination, and the transfer locations to make sure the travel plan is feasible.

In fact, we can show that $T^k_{i,j} = \alpha$ implies that there are *at most* α ways to transfer from stops served by route i to stops served by j by exactly k transfers. Taking advantage of this observation, we can prove the following property that is useful for planning service routes in designing service routes for a public transportation system [10].

Property 1 *Let T represent the connectivity matrix for a public transportation system. Assume that there is no service route running from a location x to another location y. If there exists an integer k such that $\sum_{l=1}^{k} T^l_{i,j} = 0$ for all $i \in SR(x)$ and $j \in SR(y)$, then traveling from x to y requires no less than k transfers.*

If we do not want people to transfer more than twice to travel between locations, we must at least make such a k smaller than 3 for all location pairs served by the public transportation system. Let $M_{i,j} = \sum_{l=1}^{k} T^l_{i,j}$. If there are some very large $M_{i,j}$ in M, there may be many ways to travel from x to y. In this case, we may consider to shift the service routes so that we can balance services between other location pairs without seriously degrading current services.

The connectivity matrices capture the route constraint on how one may travel from the origin to the destination. Using this information, the search algorithm can reduce the amount of search work that would be needed otherwise. Overestimating ways of transferring between routes may offset the improvement in efficiency, but this will not make our algorithms slower than traditional uninformed search algorithms. We will apply the connectivity matrices and *connecting-route* functions to path-planning problems.

CONNECTING-ROUTE FUNCTIONS

For a pair of routes, i and j, our algorithms employ connecting-route functions $CR_1(i, j)$ that relates i and j to route numbers of the routes that provide ways to transfer from i to j.

$CR_1(i,j)$ represents the route numbers of the routes that directly run from some stops on route i to some stops on route j. When $T^2_{i,j} > 0$, the algorithm checks $CR_1(i,j)$ for detail information about how to transfer from route i to j. We can compute $CR_1(i,j)$ at system design time with the following formula:

$$\{r | T_{i,r} * T_{r,j} > 0 \text{ and } \min_{s \in CS(i,r)} K(r,s) < \max_{t \in CS(r,j)} K(r,t)\}. \quad (3)$$

Applying (3) to the example shown in Figure 3, $CR_1(1,2)$ will be set to $\{3,4\}$, and both $CR_1(1,1)$ and $CR_1(2,1)$ will be set to empty sets. Notice that the second condition in (3) ensures that $CR_1(i,j)$ does carry correct information. If we drop this condition, we would obtain $CR_1(1,1) = \{3,4\}$ and $CR_1(2,1) = \{3,4\}$. It is easy to verify that $CR_1(1,1) = \{3,4\}$ suggests impractical travel plans and that $CR_1(2,1) = \{3,4\}$ suggests impossible travel plans.

A natural extension of $CR_1(i,j)$ is $CR_2(i,j)$ that relates routes i and j to a pair of routes $(r1, r2)$ that will connect some stops on i to some stops on j. This function and $T^3_{i,j}$ are very useful for algorithms that are designed to compute travel plans that require three transfers as we explain shortly.

FINDING CANDIDATE PATHS

We present the algorithm for finding traveling plans that require no more than two transfers next. In the algorithm, we have assumed that $i \in SR(O)$ and $j \in SR(D)$.

Algorithm 1 *PathPlanning (Route information, Origin O, Destination D)*

1. *If $O = D$, there is no need to commute.*
2. Direct connection. *For any route $r \in SR(O) \cap SR(D) \neq \emptyset$, if $K(r,O) < K(r,D)$, we can go from O to D by r.*
3. One transfer. *If $T_{i,j} \geq 1$, $K(i,O) < K(i,s)$ and $K(j,s) < K(j,D)$ for a stop $s \in CS(i,j)$, we can take i at O, transfer from i to j at s, and get to D by j.*
4. Two transfers. *If $T^2_{i,j} \geq 1$ and there exist stops $s \in CS(i,r)$ and $t \in CS(r,j)$ for a route $r \in CR_1(i,j)$ such that*
   (a) $K(i,O) < K(i,s)$,
   (b) $K(r,s) < K(r,t)$, and
   (c) $K(j,t) < K(j,D)$,

   then we can take i at O, transfer from i to r at s, transfer from r to j at t, and get to D by j.

The first step checks if the origin and the destination are the same location, and is included for completeness of the algorithm. The second step examines if both the origin and the destination are served by a service route r. If yes, we have to examine if this route runs from the origin to the destination. Work conduct at this step may be carried out for all location pairs at the system design time to improve the efficiency of the algorithm.

The third step looks for travel plans that require one transfer from a route i to a route j. If $T_{i,j}$ is positive, then we may transfer from route i to j. We must then check the location of the transfer stop s, and make sure that it is possible to travel from O to s by route i and from s to D by route j by comparing the involved K values.

The fourth step looks for travel plans that require two transfers. In fact, there is a general procedure for searching for plans that require k transfers. We make sure that $T^k_{i,j}$ is positive, look into $CR_{k-1}(i,j)$ for possible connecting routes, and examine if there are locations where we can transfer.

The algorithm relies on the K values of stops for checking the feasibility of travel plans. The assignment of K values implicitly assumes that all service routes are directed and acyclic. There is no need to compare K values of two stops on a route r when the path of r forms a loop, since we can go between any stops on such a route.

An example will make the algorithm easier to understand. We have computed T^2 for the network shown in Figure 3. The fact that $T^2_{1,2} = 2$ indicates that we can transfer from route $R1$ to $R2$ by two transfers in two different ways. Also, the facts that $\mathsf{A} \in CS(R1, R3)$, that $\mathsf{B} \in CS(R3, R2)$, and that $R3 \in CR_1(R1, R2)$ indicate that we may transfer from R1 to R2 via A and B. The algorithm determines that we can travel from C to E via A and B after checking the involved K values. The algorithm will also find that we cannot travel from C to F via A and B because $K(R2, \mathsf{B}) \not< K(R2, \mathsf{F})$.

Since we can extend the algorithm to find travel plans that require more than three transfers, the algorithm is *complete* in the sense that it can find all plans suitable for the desired trip. In addition, because the algorithm starts its search for travel plans from those that require lesser transfers, the algorithm is also *optimal* in the sense that it can find the travel plans that require the least number of transfers.

Property 2 PathPlanning *is both complete and optimal for searching travel plans that require the least number of transfers.*

IMPLEMENTATION ISSUES

Whether one needs to employ the aforementioned methods for computing connectivity matrices and connecting-route functions in implementing the path-planning algorithm depends on the requirements of the system. If the route database is changing frequently and if travel plans recommended by the algorithm must reflect such changes immediately after the raw data is modified, a precise update of involved functions and matrices may not be desirable for complex public transportation systems. Doing so may take too much time. Employing approximation techniques to quickly update the functions should better meet the need for on-line systems. For such special cases, it may be better for the system to adopt approximation strategies during the day, and conduct a precise data update during the night.

It is possible for the algorithm to sidestep the exact methods, while the algorithm still computes the best travel plans at a slower speed for each individual query [10]. For instance, there exists a simple method for approximating $T^2_{i,j}$. The following condition ensures that there is at least one way to transfer from route i to j via r.

$$\min_{s \in CS(i,r)} K(r, s) < \max_{t \in CS(r,j)} K(r, t) \tag{4}$$

If we add the quantity $T_{i,r} * T_{r,j}$ to the summation for computing $T^2_{i,j}$ when (4) holds, we will compute an exact T^2 for the example in Figure 3. We can make up artificial examples, such as the one shown in Figure 4, to illustrate that T^2 computed with the restriction of (4) may exaggerate the number of ways for transferring between routes. For this particular case, $\min_{s \in CS(i,r)} K(r, s) = K(r, \mathsf{B})$ and $\max_{t \in CS(r,j)} = K(r, \mathsf{H})$, so (4) holds. As a result, we would add the quantity $T_{i,r} * T_{r,j} = 3 * 2 = 6$ to $T^2_{i,j}$, even though there are only three ways to transfer from i to j via r. Values computed with this method may be larger than their exact values. As a result, *PathPlanning* may conduct more search than necessary, although *PathPlanning* will find the best plan eventually. Nevertheless (4) can be effective in some special cases.

Property 3 *With* (4)*, we compute $T^2_{i,j}$ precisely if no routes in the public transportation system overlap with each other at two or more separately continuous segments.*

Proof. *Assume that route r overlaps with i and j at only one continuous segment. Given* (4)*, we have a generic situation shown in the following figure. Hence, we can transfer from i to r at any stop $s \in CS(r, i)$ and from r to j at any stop $t \in CS(r, j)$.* ■

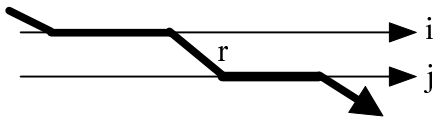

Figure 5. Applying (4) can lead to exact $T^2_{i,j}$ for some cases.

ROUTE PLANNING WITH HUBS

In a metropolis, there are local business centers where a lot of service routes concentrate. These *hubs* provide a great chance for people to switch between different routes. For instance, we may prefer travel plans that transfer at hubs at steps 3 to 4 in *PathPlanning*. In a preliminary report [9], we outlined an algorithm that compute travel plans with hubs, and we provide a clearer algorithm here.

We promote a stop to a hub in the route-information database based on relative importance of stops. Bus stops served by more than 15 routes are selected as hubs. Stops of subway and traditional train systems are automatically considered as hubs. To simplify the task of route planning, we assign at least two hubs on every service route. If a route has less than two hubs, we will designate selected stops served by the route as hubs. This extra selection is based on the total number of routes that serve the stops, and we select the stop served by the most number of routes as a hub. At this moment, our system has about 170 hubs that are selected from about 2500 ordinary stops.

Technically, the route map has two levels in the database. The base level contains all stops, and the upper level only hubs. The algorithm needs to know ways for connecting hubs. We can solve this bootstrap problem with human knowledge or a very simple path planner, since typically there are direct or one-transfer routes that serve major hubs. The base level will be used for finding travel plans for trips that require zero and one transfer. For other cases, the

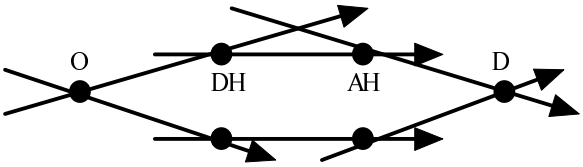

Figure 6. Travel through hubs when there is no simple plan.

algorithm looks for travel plans with hubs, and Figure 6 illustrates the idea. When we cannot travel from the origin to the destination directly or with only one transfer, we seek hubs, DH and AH, that are guaranteed to lead us from O to D. To implement the idea, the algorithm finds ways to travel from the origin to its *nearest departing hubs* and ways to travel from the *nearest arriving hubs* to the destination. We can then use the information for connecting hubs to create complete travel plans. A stop s and any of its nearest hubs, h, must be served by a common route $r \in SR(s)$. A nearest departing hub and a nearest arriving hub of s on a route r is respectively the hub h that minimizes positive $K(r, h) - K(r, s)$ and $K(r, s) - K(r, h)$. The algorithm will use the set of all nearest departing hubs, denoted *NDHS*(s), and the set of all nearest arriving hubs, denoted *NAHS*(s), of a stop s. In preparation of the route database, we ensure that *NDHS*(s) and *NAHS*(s) are not empty for any stop s. To this end, we may need to promote terminal stops of some service routes to hubs. The algorithm follows.

Algorithm 2 *PathPlanning2(Route information, Origin O, Destination D)*

1.–3. Same as those in PathPlanning

4. Transfer via hubs*: Recommend a path from O to* $s \in$ *NDHS*(O)*, from* s *to* $t \in$ *NDHS*(D)*, and from* t *to D .*

The last step subsumes the functionality of all steps for finding travel plans that require more than one transfer in *PathPlanning*. Since *NAHS*(s) and *NDHS*(s) are not empty for any stop s, there is at least one hub for going to D and one hub for leaving O. Given that our database will contain information for traveling between any hub pairs, we can find at least one travel plan for any desired trip. Therefore this algorithm is complete. However, this algorithm is not optimal as the algorithm is not guaranteed to find the travel plans that require the least transfers.

PRIORITIZING CANDIDATE PATHS

A path-planning algorithm needs to prioritize travel plans when there are multiple ways for the desired trip. Common factors for comparing travel plans include number of transfers, monetary costs, expected travel time, and seat availability. Different categories of travelers may weigh one factor more than others [2]. For instance, people who are not familiar with the metropolis may strongly prefer travel plans with the least transfers for minimizing the burden of locating bus stops.

We can categorize the preferences into two types, depending on the relationship between the preferences and time. Many *time-independent* factors, such as the monetary costs of taking buses, do not change with time when we consider a duration of shorter than few hours. In contrast, finding the best travel plan that carries *time-dependent* costs require us to model the change of time rather precisely when we compare the merits of different travel plans. Researchers have established an array of path-planning algorithms under these different constraints.

Our algorithms, both *PathPlanning* and *PathPlanning2*, can be modified to incorporate a mechanism for ordering travel plans based on traveler's preferences. It is easy to suspend our algorithms whenever they find travel plans at any step, thereby favoring travel plans with smaller numbers of transfers. Heuristics can be applied to prioritize travel plans found at the executed steps in our algorithms. For instance, one may prefer subway service to bus service for the same trip. If the average travel times between stops are available, we can compute the expected travel times for travel plans found at a step, and prioritize alternative travel plans accordingly.

To this end, we may employ our *PathPlanning* algorithm as an embedded function in a standard shortest-path algorithm. This algorithm consults *PathPlanning* for next stops to explore during the search process. Since *PathPlanning* can distinguish what next stops can lead the travelers from the origin to the destination, it only returns these viable intermediate stops to the standard shortest-path algorithm. Through the cooperation of a standard shortest-path algorithm and *PathPlanning*, we can avoid exploring partial travel plans that appear to be highly preferred but do not lead to the destination.

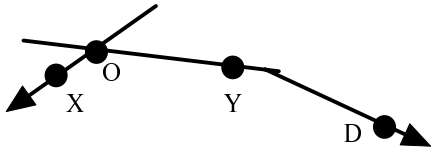

Figure 7. Proximity may lead to futile search of travel plans.

Figure 7 illustrates a possible such scenario. A standard shortest-path algorithm may explore the way to X that is near the origin O because X appears to be a promising intermediate step initially. However, X does not belong to any route that will lead to the destination D. Therefore

a good path planner should avoid considering X as an intermediate stop, and our *PathPlanning* algorithm can perform as a good consultant to standard path-planning algorithm in this aspect.

PathPlanning3 is a skeleton for the cooperation between *PathPlanning* and standard path planner. If the preferences are not time-dependent, we can integrate our algorithms with any shortest-path algorithm, such as the Dijkstra's algorithm [1], to find the best travel plan.

Algorithm 3 *PathPlanning3(Route information, Origin O, Destination D)*

1. Attempt to find simple solutions*:*
   - *(a) Call PathPlanning for travel plans that require no more than one transfer.*
   - *(b) Use a traditional shortest-path algorithm to select the best travel plans from those found in 1(a).*
2. Find more complex solutions*:*
   - *(a) Call PathPlanning for travel plans that require two or three transfers.*
   - *(b) Use a traditional shortest-path algorithm to select the best travel plans from those found in 2(a).*

When users' preferences are time-dependent, this two-level architecture is still applicable. We just need to replace traditional shortest-path algorithms because they may not find time-dependent shortest paths [5]. Our algorithms can still serve as the consultant, and we can employ algorithms that are capable of computing time-dependent shortest paths. For time-dependent constant costs, Kaufman and Smith identify a condition under which the Dijkstra's algorithm remains applicable. This *consistency* condition requires that leaving a place later will not make one to arrive at another place earlier [7]. This condition holds pretty well with public transportation vehicles, although this condition might be violated in some special conditions in general transportation problems. Wellman et al. extend the idea to cope with time-dependent probabilistic link-travel times [14, 15]. Their *stochastic consistency* condition dictates that the probability of arriving at the next location by any specific time will not be increased by leaving the current location later. Under this condition, standard shortest-path algorithms, including the A* algorithm, remain applicable for path-planning problems. If we do want to compute the fastest travel plans when the travel costs are time-dependent and probabilistic in very complex transportation networks, the computational load may be too high for the system to respond in a timely manner. Liu and Wellman discuss techniques for computing bounds of travel-time distributions under the time constraint [11].

USER INTERFACE AND APPLICATIONS

The outermost crust of our frame work shown in Figure 1 is a multi-media, multi-lingual, and multi-platform user interface. We offer output in audio, graphical, and text formats to help users gain better idea about what the system recommends them to do. We believe that the system has to provide a multi-lingual user interface so that we can help foreign travelers. We also hope to distribute the route information across platforms. In particular, we hope to make route information accessible from mobile computing devices. To meet this goal, the output of our algorithms can be converted into the WML format [3].

Realistic bus-information provision systems must have a good user interface to deal with naming problems of bus stops. Although we may provide a graphic user interface, we cannot assume all users will locate their origins and destinations on a digital map. Any user-friendly system must prepare to accept queries that use text input. Such a demand requires our system to manage the mapping between text inputs for location names and stop names gracefully.

Popular street names such as Main Street may be used as stop names in multiple cities in the metropolis. Therefore, we need to cope with the problem of different stops carrying the same stop names. We solve such homographic problems by annotating extra information with the stop names. When encountering an ambiguous query, our system asks for clarification.

In addition, the user interface needs to cope with synonym problems. It is possible for bus stops that surround a big site, such as the Darling Harbour in Sydney, to have different names. When users ask about how to go to the Darling Harbour, a good system should be able to return a travel plan that terminates at a stop that is geographically close to the Harbour, although the stop might not be named as Darling Harbour.

We can incorporate other information into our system. Parking information may be valuable for non-local people who need to come into town. Integrating information about parking and public transportation system may help these non-local people to park at places where they can continue their trips on public vehicles. Information about tourist spots and special events can be interesting to both local and non-local people. Our system can direct people to take public transportation vehicles for these non-recurrent trips. The channels for accessing the services of our system may include the Internet, wireless mobile devices, and information kiosks installed at subway stations. By making the bus information as easily accessible as possible, we maximize the contribution of this information system to the community.

CONCLUSIONS

The popularization of the Internet and wireless communication devices offers new channels for promoting the public transportation systems. By providing easier access to travel-related information, we may help people to adopt public vehicles more easily than ever. We take advantage of recent progress in computer and communication technologies in constructing this route-information management and provision system. This system helps service providers to manage and monitor operation data about their services so that they can plan and provide better services to customers. We also propose path-planning algorithms for finding bus connection. This service could be very valuable for non-recurrent trips. Our system shows the travel plans in HTML, WML, graphic, and audio formats to meet requirements in different application contexts.

Planning explicitly with the route constraint and hubs provides us a chance to find travel plans more efficiently than planning at the stop level. Connectivity matrices and related functions capture the route constraint by encoding the possibilities of transferring among routes. With this information, the *PathPlanning* algorithm focuses on related routes to search for feasible travel plans. As a result, this algorithm can foresee which search direction may lead to the solution, thereby offering better search efficiency than traditional search algorithms. Categorizing stops into hub and regular-stop classes allows us to tackle more complex queries efficiently. This hierarchical structure is similar to the *hierarchical encoded map views* technique used for finding shortest path in large areas [6]. We have implemented *PathPlanning2* for a field test at <http://iris.cs.ntou.edu.tw>. Results collected from field tests indicate that this algorithm can compute satisfactory travel plans within a couple of seconds.

ACKNOWLEDGMENT

This work was supported in part by Grants NSC-89-2213-E-004-007 and NSC-89-2515-S-019-001 from the National Science Council of Taiwan. The authors thank Po-Nien Chen and Chih-Yao Yang for their active participation in implementing the system. The authors would

also like to thank the anonymous reviewers for their valuable comments on revision of this paper.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin (1993) *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall.

[2] S. Bae (1995) An advanced public transportation systems application: Feasibility study of bus passenger information systems operational test in the town of Blacksburg. *Proceedings of the Sixth Vehicle Navigation and Information Systems Conference*, pp. 408–413.

[3] T.-Y. Chen and T.-C. Tsai (2001) Bus information systems and techniques for converting HTML documents to the WML format. Technical report, Departmetn of Computer Science, National Chengchi University.

[4] P. J. Elkins (1993) Service management systems for public transport–the German approach. *Proceedings of the IEE Colloquium on Vehicle Location and Fleet Management Systems*, pp. 401–410.

[5] R. W. Hall. (1986) The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20(3):182–188.

[6] N. Jing, Y.-W. Huang, and E. A. Rundensteiner (1998) Hierarchical encoded path views for path query processing: An optimal model and its perfromance evaluation. *IEEE Transaction on Knowledge and Data Engineering*, 10(3):409–432.

[7] D. E. Kaufman and R. L. Smith (1993) Fastest paths in time-dependent networks for intelligent vehicle-highway systems applications. *IVHS Journal*, 1(1):1–11.

[8] N. Koga (1999) Public transportation priority system using optical bus detectors. *Proceedings of the Second International IEEE Conference on Intelligent Transportation Systems*, pp. 135–138.

[9] C.-L. Liu, T.-W. Pai, and C.-T. Chang (2000) IRIS: Integrated route information service for multimodal public transportation systems. *Proceeings of Taiwan's International Conference & Exhibition on Intelligent Transportation Systems 2000*, pp. 186–196.

[10] C.-L. Liu, T.-W. Pai, C.-T. Chang, and C.-M. Hsieh (2001) Path-planning algorithms for public transportation systems. *Proceesings of the Fourth International IEEE Conference on Intelligent Transportation Systems*.

[11] C.-L. Liu and M. P. Wellman (1999) Using stochastic-dominance relationships for bounding travel times in stochastic networks. *Proceedings of the Second International IEEE Conference on Intelligent Transportation Systems*, pp. 55–60.

[12] H. Ohdake (1999) On construction of a public transportation priority system. *Proceedings of the Second International IEEE Conference on Intelligent Transportation Systems*, pp. 550–555.

[13] S. Russell and P. Norvig (1995) *Artificial Intelligence: A Modern Approach*. Prentice Hall.

[14] M. P. Wellman, M. Ford, and K. Larson (1995) Path planning under time-dependent uncertainty. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 532–539.

[15] P. R. Wurman and M. P. Wellman (1996) Optimal factory scheduling using stochastic dominance A*. *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 554–559.