Finding Optimal Bus Service Routes: Internet-Based Methodology to Serve Transit Patrons

Hassan A. Karimi[1]; Ratchata Peachavanish[2]; and Jun Peng[3]

Abstract: Internet-based transportation systems are becoming widespread due to advances in Internet, computing, and geospatial information system (GIS) technologies. An internet-based methodology that allows public transportation users to find routes and buses between pairs of origin and destination addresses is discussed in this paper. The methodology is considered as a framework where several different transportation-related objectives can be met. Of these objectives, finding the sequence of bus routes with a minimum number of bus-to-bus transfers between any two entered addresses is focused in the work presented in this paper. With the proposed methodology, users are provided with information on the bus routes (bus numbers) they should take and locations where they should get on and off the bus including all bus-to-bus transfer points. The methodology is intended to be easily accessible and available through standard Web browsers on the client machine and be interoperable with respect to data and software facilitating integration to other data and software. To demonstrate some of the features and capabilities of the methodology, a prototype of the methodology, called iBus, is also discussed.

DOI: 10.1061/(ASCE)0887-3801(2004)18:2(83)

CE Database subject headings: Geographic information systems; Buses; Route survey; Internet.

Introduction and Background

Transit planning typically deals with such issues as constraints on street networks (e.g., one-way streets), limited resources available (e.g., terminal facilities), and optimal allocation and scheduling of resources. Transit planners and managers take these and other issues into account in exploring alternative approaches for delivering bus services. However, such approaches whose ultimate goal is efficient and effective usage of transportation resources require that transit planners and managers have an understanding of specific users needs and of technologies that can potentially benefit transit activities. Moreover, they need to be familiar with the three types of information most commonly used in transit modeling and analysis, namely, location, attribute, and topology. To benefit bus riders, bus-to-bus transfer strategies are developed which improve bus networks coverage and bus resources utilization and reduce the pressure for assigning new direct bus routes.

Geospatial information system (GIS) technology is an invaluable, powerful tool available to transportation planners and engineers for solving location-, attribute-, and topology-related problems. Due to the nature of data and processing required in land-based transportation applications, transportation-specific GIS tools, GIS for transportation (GIS-T), have been developed and used in many transportation projects (e.g., see Ding 1994; Wartian and Gandhi 1994; Chou 1995; Peng et al. 1998; Thill 2000; Ozbay et al. 2001; Trepanier et al. 2002). Advances in internet technology and the continuing interest in providing GIS technology on the World Wide Web (Doyle 1997) is now making on-line transportation solutions (e.g., traffic information) possible where they are available to the public and users can visualize information. Internet-based GIS have been growing rapidly for the past several years. These systems are currently considered by many transportation organizations for providing a range of transportation services (thus internet-based GIS-T). Wireless networks is another technology which is expected to increase the demand for internet-based GIS-T where they can provide access and solutions to both desktop and mobile users. There are existing internet-based mapping and GIS software packages that support mobile location-aware devices, interconnected with servers and other mobile devices via wired and wireless networks. Most of these systems can be customized, enhanced and extended (new features and capabilities) to meet the requirements of internet-based GIS-T for transportation applications. Example commercial internet-based mapping and GIS software packages where transportation developers can develop and customize specific transportation-related applications are ArcGIS (2003) by Environmental Systems Research Institute, GeoMedia WebMap (2003) by Intergraph, and MapXtreme (2003) by MapInfo. Each of these systems has its own functionality and proprietary data formats.

Transportation users of internet-based GIS-T can benefit from lower costs since many internet-based GIS are designed and offered as services for a set of target users (Sumrada 2002). This is particularly true for internet-based GIS-T where the responsibility of hosting the server(s) is on the shoulder of transportation organizations, which tend to absorb the cost of maintenance and implementation, while the users are able to utilize the system targeted for their use without incurring any cost. Another important benefit of internet-based GIS-T is that they are not only capable of reaching a wider audience, but also are capable of accommodating users of various skill levels. This is to say that while stand-alone GIS-T are typically designed for expert users,

[1]Dept. of Information Science and Telecommunications, Univ. of Pittsburgh, 135 N. Bellefield Ave., Pittsburgh, PA 15260.

[2]Dept. of Information Science and Telecommunications, Univ. of Pittsburgh, 135 N. Bellefield Ave., Pittsburgh, PA 15260.

[3]Dept. of Information Science and Telecommunications, Univ. of Pittsburgh, 135 N. Bellefield Ave., Pittsburgh, PA 15260.

Note. Discussion open until September 1, 2004. Separate discussions must be submitted for individual papers. To extend the closing date by one month, a written request must be filed with the ASCE Managing Editor. The manuscript for this paper was submitted for review and possible publication on April 16, 2003; approved on October 10, 2003. This paper is part of the ***Journal of Computing in Civil Engineering***, Vol. 18, No. 2, April 1, 2004. ©ASCE, ISSN 0887-3801/2004/2-83–91/$18.00.

internet-based GIS-T are for both novice and expert users.

Transit bus service is a transit activity where GIS technology can help solve complex problems and where architectures based on internet can have a major impact on public transportation. An internet-based GIS-T for a specific transit bus service can be designed where it can meet a set of requested requirements and provide optimal solutions. For example, the purpose of a transit bus service could be providing optimal routes based on such criterion as travel time or distance traveled which may be different from a truck delivery service whose purpose is to provide optimal routes based on such criterion as customers served during a day operation.

The literature on routing in transportation deals with a range of problems from simple static routing to complex dynamic routing. Many existing routing techniques are based on a cost function which is minimized by selecting routes with minimal costs (Wang and Zhang 1992; Lotan 1997; Pang et al. 1999). A cost function usually has the following form:

$$f(x_i)=\sum_{i=1}^{n}(w_i x_i+\epsilon_i) \tag{1}$$

where w_i=weight; x_i=attribute of the route (time, distance, numbers of turns, familiarity, comfort); ϵ_i=random error; and n=number of optimizing attributes.

Trip planning may occur either proir to the trip, the pretrip planning or static mode, or during the trip, the real-time or dynamic mode. In either the static or dynamic mode, the optimization criterion remains unchanged while the information impacting the solution may be different. The differeces between the static and dynamic modes are that in the static mode there is no time constraint to compute routes and when traversing a computed route there is no deviation from the route while in the dynamic mode due to deviation from the computed route a new route needs to be recomputed in real time. Dijkstra's algorithm (Dijkstra 1959), a well-know algorithm for signle-sourse shortest routes, is widely used for routing applications in the static mode. Dijkstra's algorithm has the time complexity of $O(n^2)$, in the worst case scenario where n is the number of nodes in the network, which is not suitable for the dynamic mode. Alternatively, in the dynamic mode for real-time transportation applications such as vehicle routing where upon detection of deviation from the computed route, a new route between the current location of the vehicle, which is computed by such positioning sensors as the global positioning system, and the destination is computed in real time. There exist also heuristic techniques that use Dijkstra's algorithm as the base algorithm. An example of such heuristic techniques is an algorithm which reduces the number of nodes in a road network before it applies Dijkstra's algorithm (Karimi 1996). Another example heuristic technique is an algorithm which adopts a utility function to minimize the cost or maximize the probability of a selected route using Dijkstra's algorithm (Guzolek and Koch 1989; Modesti and Sciomachen 1998). There are also techniques based on a combination of heuristic and hierarchical techniques (Liu 1997; Jagadeesh et al. 2002).

Heuristic techniques have been used for on-demand vehicle routing and dispatching. An example is the *tabu* search heuristic that was developed to solve a vehicle routing problem with capacity and route length restriction (Gendreau et al. 1994, 1999) and a vehicle diversion strategy (Ichoua et al. 2000). Another example is a vehicle routing algorithm for stochastic demands that was developed to solve a capacitated vehicle routing problem (Gendreau et al. 1995). This cost-based optimization algorithm is mainly for truck collection routes that need to take into account the uncertainty of customer presence and demand. Balakrishnan (1993) has also developed a heuristic algorithm for vehicle routing with *soft time* windows to minimize the penalties incurred when the collection time windows are violated. A survey of vehicle routing heuristics can be found in Laporte et al. (2000). All these algorithms generate vehicle routes based on demands and minimize a certain set of costs by satisfying a certain set of conditions. However, computing an optimal sequence of transit bus routes that would help a bus rider to meet his/her transportation needs (traveling from an origin to a destination) is different from vehicle routing. In computing optimal routes for typical transportation applications such as pretrip planning (static routing) and vehicle routing (dynamic routing), a road network and a specific optimality criterion is used while in computing optimal routes for bus riders, the bus networks which are subsets of a road network and specific bus riders' requirements, such as minimization of bus-to-bus transfers, are used.

In this paper, an internet-based methodology which is accessible through commonly available Web browsers is discussed. Currently there are a number of transit agencies that have operational internet-based systems that passengers may query with origin and destination addresses to obtain one or more possible routings. Example of such internet-based systems are RideGuide (2003), TakeTransit (2003), TransLink (2003), and TfL (2003). In general, these systems are trip planners capable of providing a variety of transportation options to passengers. As it is stated on the Websites of some of these systems, each system, despite offering tremendous benefits to passengers, has certain inaccuracies and shortcomings. While it becomes obvious by using these systems that their functionality overlaps, it is not clear what approaches are taken to develop them. In the absence of published reports describing the development strategies in these systems, it is difficult to understand such components and technical issues as data sources and databases, computational modules, and information presentation of internet-based GIS-T. Therefore, our motivation behind presenting the proposed methodology in this paper is to bring about advanced methodologies and techniques currently available and emerging to develop internet-based GIS-T helping transportation professionals realize not only what can be performed by theses systems, but also how these systems should be designed and developed. For this same reason, we discuss the design and development of a prototype of the proposed methodology, called iBus, from scratch without customizing any existing internet-based GIS software system. It is our intention that by providing an insight into the intricacies and issues that need to be addressed when developing such systems, new and innovative internet-based GIS-T will emerge and made available to many users with diverse requirements.

The contributions of this paper include an internet-based methodology to serve the transit patrons, an on-the-fly geocoding algorithm for internet-based GIS-T, a routing module specifically designed for bus riders which takes the unique requirements and constraints of transit bus routes into consideration, and a prototype of the methodology (iBus) to realize the components, assumptions, and issues of internet-based GIS-T. The paper has the following structure. In the first section, the architecture of the methodology is discussed. The second section discusses the data requirements, followed by a discussion of the geocoding module in the third section. The fourth section is devoted to the description of the routing module and the routing algorithm. The mapping module, the client program, and the implementation of the server in the prototype are described in the fifth section and conclusions and future research are given in the last section.

J. Comput. Civ. Eng. 2004.18:83-91.

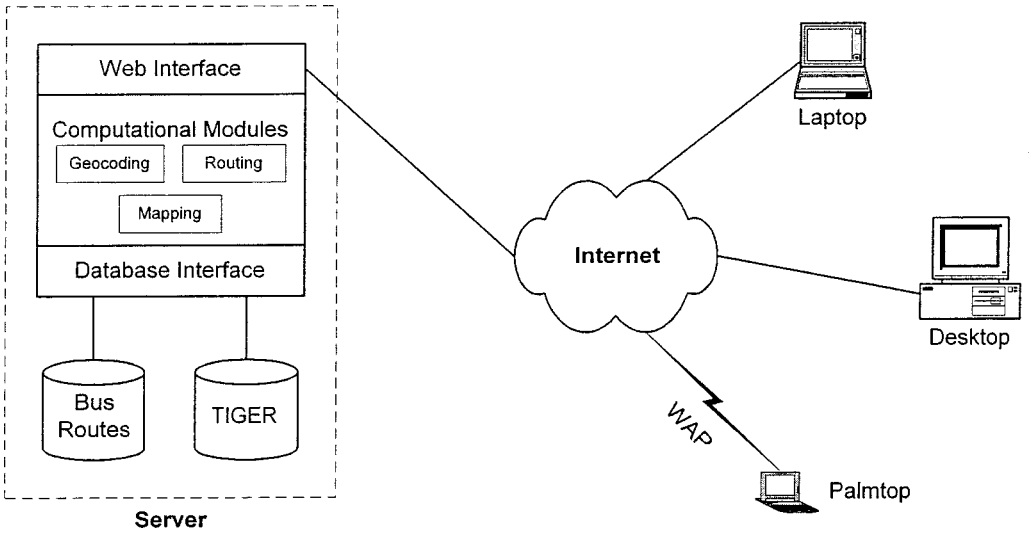


Fig. 1. Overall architecture of methodology

Architecture

The primary purpose of the methodology is to provide public transportation information to the public in an efficient and effective manner. In its current version, the methodology is developed specifically to provide bus route information with a minimum number of bus-to-bus transfers between any given pair of addresses. Riders are provided with a Web page where they can enter, through a Web browser like *NETSCAPE NAVIGATOR* or *INTERNET EXPLORER*, two addresses, one for the origin and one for the destination, in a given geographic area (e.g., city). Upon entering addresses, the rider can submit the request for the optimal bus route with a minimum bus-to-bus transfers between the pair of addresses. The methodology computes the optimal route and provides the solution by showing a map of the area with the computed route highlighted. In addition, it provides information on the location of the bus stop (nearest to the origin) where the rider would need to get on the bus, the locations of the bus stop where the rider would need to transfer, and the location of the bus stop (nearest to the destination) where the rider would need to get off the bus.

The methodology is based on a three-tier architecture, a client-side user interface, a set of server-side computational modules, and a database system. Riders can access the methodology through conventional internet-enabled desktop/laptop personal computers as well as through mobile computing devices linked to the server via wireless communication systems. Key features of the methodology include accessibility and interoperability. The methodology is intended to be easily accessible through standard Web browsers without requiring any plug-in software on the client machine. Interoperability in the methodology means integrating other data and software in a simple manner. Interoperability is an invaluable feature of the methodology as it allows the system to be part of existing and emerging transportation systems without modification of program codes or development of complex tools. To ensure interoperability in the methodology, independency on proprietary data formats and software was set as one of the design goals.

The computational modules on the server side of the methodology consist of a geocoding module, a routing module, and a mapping module. Each of these modules obtains its input data from a database, which may or may not reside on the same server. The client-side user interface is an executable program that must be loaded and run on the client's Web browser. The overall architecture of the methodology is shown in Fig. 1. The geocoding module uses data from Topologically Integrated Geographic Encoding and Referencing System (TIGER 1998) to geocode addresses (computation of coordinates for any given street address). The routing module performs two tasks: (1) it extracts bus route information from a bus route database and (2) it computes optimal bus routes using minimum number of bus-to-bus transfers as the optimality criterion. The mapping module uses the result from the routing module in conjunction with TIGER database to generate maps for display by the client-side user interface.

A prototype version of the methodology (iBus) was developed for the City of Pittsburgh and is currently being used for feedback from the users. Java and C programming languages and HTML were used in developing iBus.

Data Requirements

The required database in the methodology contains two data sets: a bus route data set and a street network data set.

Bus Route Data Set

The bus route data set contains data on the bus routes and the topology of bus networks (i.e., which buses pass through what street segments) in a given geographic area (e.g., city). For use in the methodology, the bus route data set is organized into a single table comprised of the fields shown in Table 1. Each record contains fields that can be used to identify a street segment and indicate the bus routes that pass through it. FNODE_ and TNODE_ contain identification numbers of the two end points, ''from'' and ''to'' nodes, that make up the segment. These end points are typically two adjacent street intersections, and thus they uniquely identify the location of a particular street segment relative to other street segments. ALTRAN indicates the street name of that street segment. Multiple street segments make up a named street as a typical street has many intersections passing through it. ROUTES contains the bus routes (bus numbers) that pass through the segment. Each route number is delimited by the ''|'' character. DIR indicates whether the street segment is a one-way street or a two-

J. Comput. Civ. Eng. 2004.18:83-91.

Table 1. Field Descriptions of Bus Data Set

Field name	Description	Example value
FNODE_	''from'' node of the segment	3131
TNODE_	''to'' node of the segment	3141
ALTRAN	name of the segment	Hoffman Blvd
ROUTES	bus routes	|H|55J|55B|53H
DIR	direction	1
LATF	''from'' latitude	40.443842
LONF	''from'' longitude	−80.432133
LATT	''to'' latitude	40.444821
LONT	''to'' longitude	−80.439156

way street. This field can have a value of ''1'' or ''2,'' depending on whether the street is a one-way or a two-way street, respectively LATF and LONF contain the coordinates (latitude and longitude) of the ''from'' node of the street segment. LATT and LONT contain the coordinates of the ''to'' node of the street segment.

Street Network Data Set

In the methodology the street network data set is required to: (1) geocode user-supplied origin and destination addresses, (2) build the input data required for the routing algorithm, and (3) create maps and present, along with the optimal route computed, other information that would help trip planning and decision making. The street network data set in iBus was extracted from TIGER database which contains the street network of a county or city along with such data as neighborhoods and shopping centers. It also contains specific points of interest (landmarks) such as day-care centers, hospitals, schools, parks, malls, shopping centers, and fire stations. Although TIGER database is known to have low spatial accuracy and lacks network connectivity in some areas, because it contains many useful attribute data it is enhanced by several private vendors. In iBus an enhanced version of TIGER database which has a higher spatial accuracy and completed network connectivity was used.

TIGER database for a county consists of multiple files, each called a record type. For iBus, the data in record type 1 (RT1), record type 2 (RT2), and record type (6) are used. An example of simplified RT1 record along with some of its fields is shown in Table 2. TLID, FENAME, and FETYPE identify the street segment specified in the record. CFCC indicates the class of the segment (e.g., an interstate highway or a local road). FRADDL and TOADDL specify the street address range on the left side of the street from the ''from'' node to the ''to'' node. FRADDR and TOADDR similarly indicate the range on the right side of the street. ZIPL and ZIPR indicate the zip code of the area to the left and right of the street segment, respectively. FRLONG and FRLAT identify the coordinates of the ''from'' node and TOLONG and TOLAT identify the coordinates of the ''to'' node.

Geocoding Module

Geocoding is the process of giving addresses and locations of interest geometric information (e.g., latitude and longitude coordinates). When the latitude and longitude coordinates of a given address is provided in a database, the address is said to be geocoded. Geocoded data are required to identify the precise location of point, linear, or area objects. It is because of geocoded data that GIS software packages are able to perform geometric and topological computations and provide answers to spatial queries.

Table 2. Field Descriptions of Topographically Integrated Geographic Encoding and Referencing System Street Network Data Set

Field name	Description	Example value
TLID	segment identification number	51693999
FENAME	segment street name	Edgewood
FETYPE	segment street type	Ave
CFCC	segment street class	A41
FRADDL	''from'' address, left	250
TOADDL	''to'' address, left	260
FRADDR	''from'' address, right	301
TOADDR	''to'' address, right	307
ZIPL	zip code, left	15122
ZIPR	zip code, right	15122
FRLONG	''from'' longitude	−79.860322
FRLAT	''from'' latitude	40.372607
TOLONG	''to'' longitude	−79.861370
TOLAT	''to'' latitude	40.372669

The methodology, as a special-purpose internet-based GIS-T, requires geocoded data for performing functions and since it is unknown in advance whether the addresses the user enters are geocoded or not, a geocoding module must be built in it. One of the main features of the geocoding module in the methodology is that it must be capable of on-the-fly geocoding which is not supported by all of the existing internet-based GIS. Therefore, to provide transportation developers with an insight into on-the-fly geocoding that meets the requirements of internet-based GIS-T, a geocoding algorithm is discussed. A geocoding algorithm, called LocMatch, which is suited for internet use and provides on-the-fly solutions was designed and developed for iBus. LocMatch, like other geocoding algorithms, requires a reference database containing already geocoded data for certain features (e.g., points, lines, and polygons) and an interpolation technique.

Fig. 2 shows the flowchart of LocMatch. The algorithm uses TIGER database as its reference database which contains geocoded data for several features including intersections of roads, schools, and parks. Upon entering an address, LocMatch searches to find a match (a match is defined as having prestored geocoded information on the requested address). If a match is found, the stored geocoded information (the coordinate data) is retrieved and attached to the address. If no match is found, the algorithm invokes an interpolation technique which uses the address ranges stored in RT1 and the chain nodes stored in RT2 of TIGER database to estimate the geocoded data for the given address.

LocMatch's interpolation technique uses the following address attributes in that order: ZIP code, street name, street type, street number, and street direction prefix and suffix with records stored in the reference database. It recognizes four cases upon receiving an address:

Case 1: The address is one of the end points stored in RT1. In this case, the coordinates of the end point in RT1 are assigned to the address.

Case 2: The address is within a range in RT1 with no corresponding record in RT2. In this case, the coordinates of the address are interpolated using the two end nodes of the segment.

Case 3: The address is within a range in RT1 with a corresponding record in RT2. In this case, the coordinates stored in RT2 are used to calculate the total length of the chain. Based on the assumption that addresses are uniformly distributed, the unit

J. Comput. Civ. Eng. 2004.18:83-91.

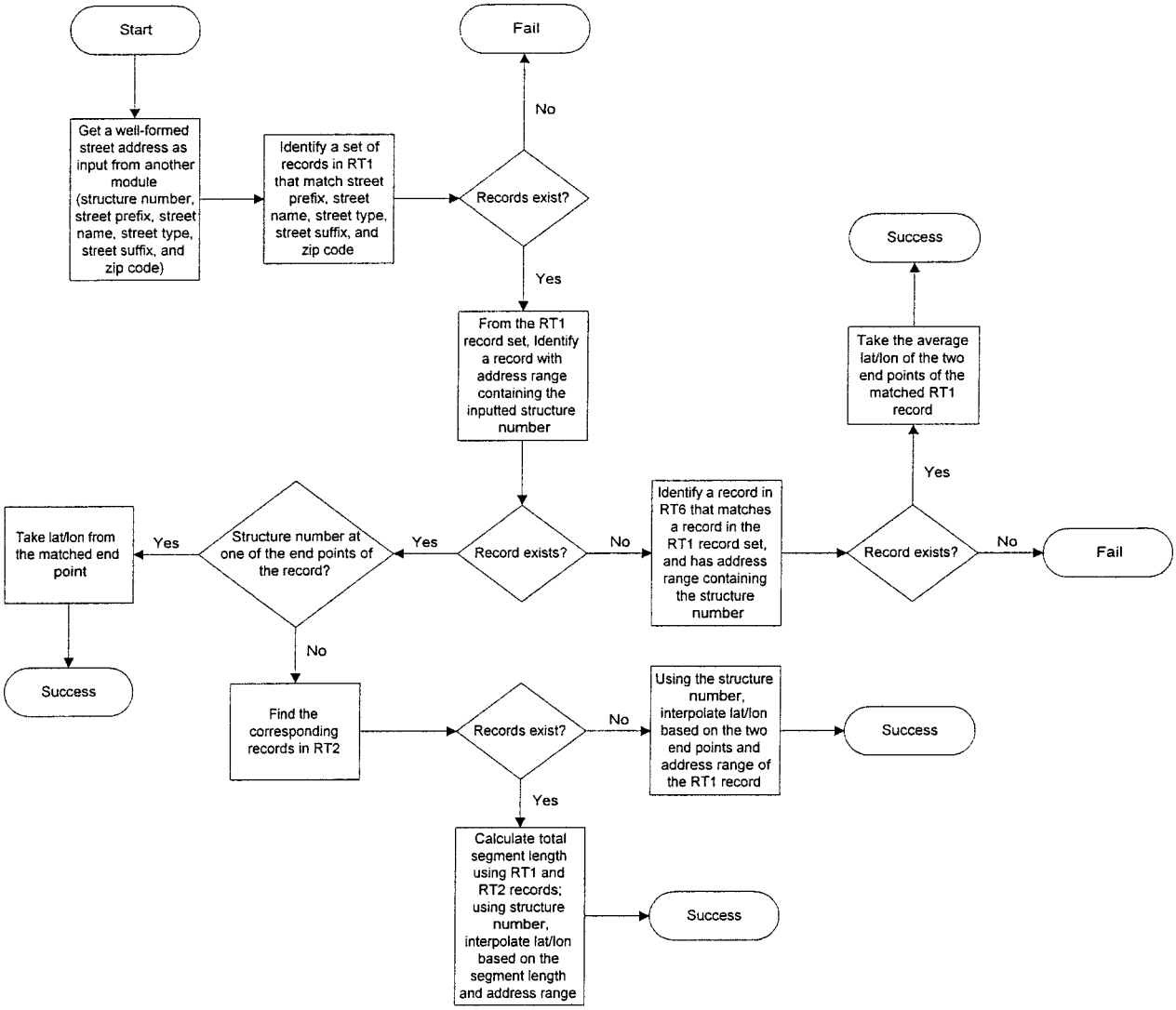


Fig. 2. LocMatch procedure

length per street segment is calculated using the total length. Using the coordinates of the two end points of the segment and the unit length per street segment, the coordinates of the address are interpolated. The following equations are used in Cases 2 and 3:

$$X = X_{\text{start}} + k_x(\text{St\_No.} - \text{Start\_No}) \tag{2}$$

$$Y = Y_{\text{start}} + k_y(\text{St\_No.} - \text{Start\_No}) \tag{3}$$

where X and Y=computed longitude and latitude of the address; $k_x = (X_{\text{start}} - X_{\text{end}})/(\text{Star\_No.} - \text{End\_No})$=longitude slope for the segment; $k_y = (Y_{\text{start}} - Y_{\text{end}})/(\text{Star\_No.} - \text{End\_No})$=latitude slope for the segment; X_{start}, X_{end}=longitudes of the start and end points of the segment; Y_{start}, Y_{end}=latitudes of the start and end points of the segment; St_No=street number to be geocoded; and Start_No, End_No=street numbers of the start and end nodes of the segment

Case 4: If the address is out of sequence, it uses data in RT6 of TIGER database to interpolate the location of the address. RT6 contains additional information on those records in RT1 which are deemed as out of sequence addresses. That is, both RT1 and RT6 contain address ranges for the same street segment, but the two address ranges are different. In this case, when an out of sequence address is encountered, LocMatch searches RT6 to see if the address falls within the range provided in RT6. If the given address falls within the address range in RT6, the coordinates of the nearest end point in RT1 to the given address are used to interpolate the coordinates of the address.

Routing Module

After addresses are validated and geocoded, they are sent to the routing module where it searches for and computes a bus number or a sequence of bus numbers if bus transfers are required. Fig. 3 illustrates the steps of the routing module in iBus. Using a buffer zone (e.g., 300 m), the bus route database is checked to find all the buses that pass near the origin and destination addresses (to find out which bus stops are within the buffer zones, the geocoded data on the origin and destination points are used). If the same bus passes through both buffer zones of the origin and destination addresses (i.e., zero transfer), it is the optimal solution. In case there is more than one bus route that offers zero transfer service, they are all presented to the rider. However, if no such route exists (i.e., at least one transfer), the module uses a cost function and Dijkstra's algorithm to calculate a minimum sequence of bus-to-bus transfers.

J. Comput. Civ. Eng. 2004.18:83-91.

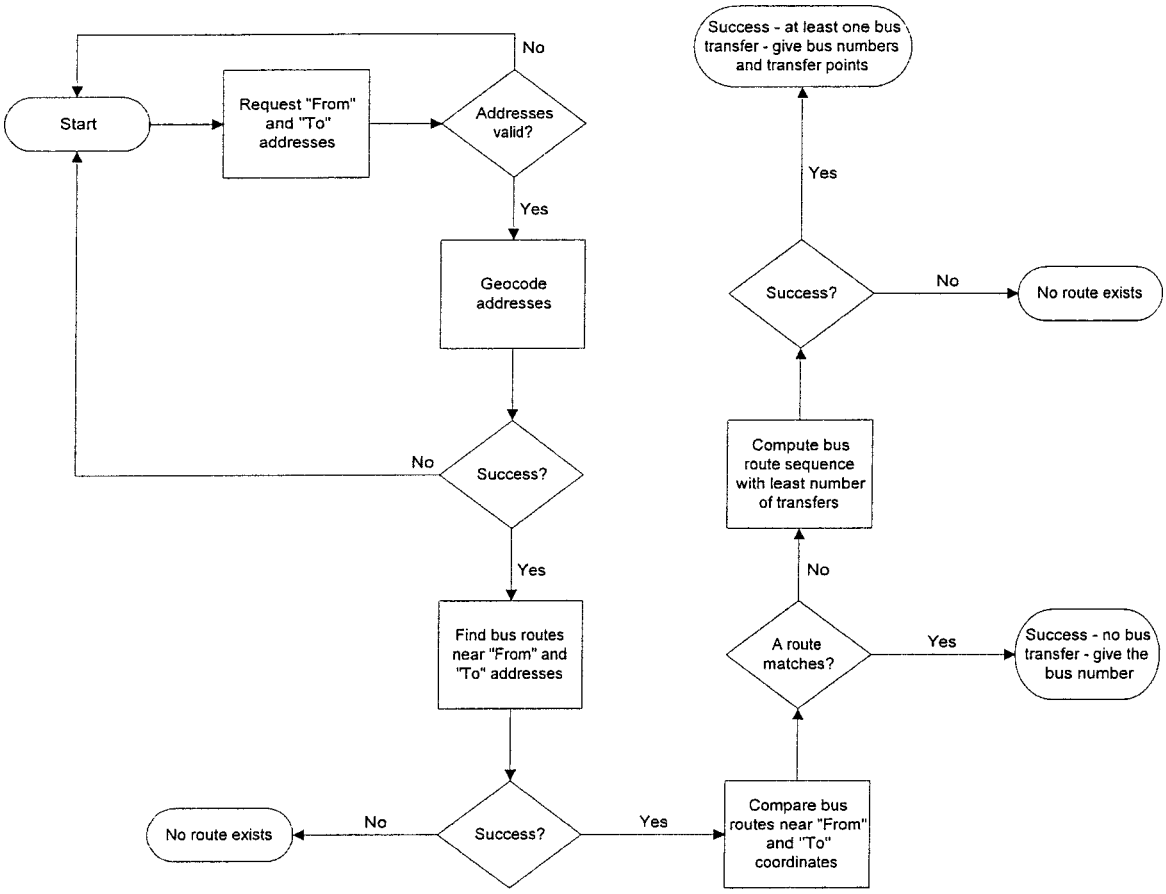


Fig. 3. Routing module procedure

Although there are other algorithms, such as Moore's algorithm (Moore 1957), which could be used, Dijkstra's algorithm was implemented in iBus. Dijkstra's algorithm guarantees to find the minimum cost route between two points in a network so long as the network is fully connected and there is no arc (edge) with negative costs. It splits the network nodes into two sets, those for which the minimum cost from the start node is known and those for which it is not known. The algorithm proceeds by transferring nodes from the second set to the first by examining the neighbors of each node in the first set. When the cost to get to the destination node is known, the algorithm can terminate. Moore's algorithm is similar to Dijkstra's but instead of considering two sets of nodes it uses a queue of nodes whose costs have yet to be finally determined. Queue members are removed from the front of the queue and their neighboring nodes are examined and added to the back of the queue to have their costs determined. When the queue is eventually empty the algorithm terminates (though the algorithm may also terminate when the next branching point is the destination node) and the minimum cost route from the start node to every other node (or the destination node) has been calculated. The cost to the destination node will be among these.

The bus routing problem in the methodology can be modeled using a graph $G=(V,E)$, where V is the set of vertices representing the bus numbers and E is the set of edges indicating the bus transfers between the connected bus numbers (vertices), with weight function $w:E\rightarrow R$, where R is a set of values (costs), mapping edges to real-valued weights. The graph represents the topology of the bus network through its edges, an edge connecting two nodes indicates that there is a transfer between the two bus numbers (represented by the two nodes). The data required to construct the graph and whether a transfer is possible between a pair of bus routes are taken from the bus route data set. An example of a graph representing three bus numbers and the transfers between them is given in Fig. 4.

The weight of a route $r=\langle v_0,v,\ldots,v_k\rangle$ is the sum of the weights of its constituent edges

$$w(r)=\sum_{i=1}^{k} w(v_{i-1},v_i) \tag{4}$$

In this first version of the methodology, each edge in the graph is given the same weight value which means all transfers between pairs of bus routes are equal (a different weight for the edges may be the total travel time, total distance, or number of bus stops). With these edges and vertices in graph G, the cost function (the least number of bus-to-bus transfers from O to D) is given by $\delta(O,D)=\min\{w(r):O\rightarrow D\}$ if there is a route from O to D. The

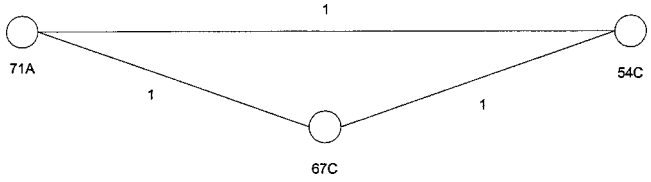


Fig. 4. Example of graph representing bus routes

J. Comput. Civ. Eng. 2004.18:83-91.

	71A	67C	54C
71A	0	1	1
67C	1	0	1
54C	1	1	0

Fig. 5. Example of adjacency matrix

least number of bus-to-bus transfers from vertex O to vertex D is then defined as any route r with weight $w(r)=\delta(O,D)$.

Minimizing the cost function [Eq. (4)] is the main objective of the routing module in the methodology which is accomplished by running Dijkstra's algorithm. In the example given in Fig. 4, there are three bus routes involved in the network: Route 71A, Route 67C, and Route 54C. The graph used to model these bus routes indicates that:

- bus 71A has at least one stop in common with bus 54C;
- bus 71A has at least one stop in common with bus 67C; and
- bus 54C has at least one stop in common with bus 67C.

If bus 71A passes near the origin and Bus 54C passes near the destination, then the rider can go from the origin to the destination by either:

1. getting on bus 71A near the origin, transferring to bus 54C where the two buses intersect, then getting off bus 54C near the destination; or
2. getting on bus 71A near the origin, transferring to bus 67C, transferring to bus 54C, and then getting off bus 54C near the destination.

With the objective of providing a solution that requires a minimum number of bus-to-bus transfers, the module may encounter situations with multiple options (e.g., six different options, each with two transfers, between the same origin and destination points). In its current version, the methodology considers such options equal and presents all of them to the rider. In future versions of the methodology, such options may be ranked based on specific requirements requested by the rider. Also in situations where two routes have more than one stop in common, the current version of the methodology takes one of them (the first one found in the database) into consideration to determine a transfer point. In future versions of the methodology, all common stops among bus routes will be taken into consideration in order to determine an optimal transfer point.

An adjacency matrix is the data structure used to store the graph of bus routes. An adjacency matrix of a graph for the methodology is an $n\times n$ (n is the number of bus routes in the graph) matrix where the entry at (i,j) is 1, i.e., $w(i,j)=1$, if there is an edge from node i to node j, otherwise the entry is 0. The adjacency matrix of the above example, involving three bus routes, is shown in Fig. 5. The adjacency matrix of the bus routes in a given area (e.g., city) is input to Dijkstra's algorithm where it is run for each possible origin–destination pair of bus routes that fall within the buffer zones (one for the origin point and one for the destination point). Suppose there are seven bus routes passing within the buffer zone of the origin address and there are eight bus routes passing within the buffer zone of the destination address, then the algorithm is run for each of the $8\times7=56$ possible options resulting in a set of sequences of bus routes to go from the origin to the destination. Upon computing all possible options, the bus route sequence with the minimum number of bus-to-bus transfers is chosen as the optimal solution.

Below are the steps of Dijkstra's algorithm for readers unfamiliar with this algorithm:

1. Initialization
   $G=\{V,E\}$ where $V=\{1,2,...,N\}$
   $S=\{1\}$ where $S=$set of vertices in the current path and vertex 1=source
   AdjTab$[i,j]$ adjacency matrix (weight from i to j)
   $D[i]$ contains the weight of the current path to vertex i
2. Search for a vertex x in $V-S$ that ensures $D[x]$ is a minimum
3. $S\leftarrow S|\{x\}$
4. For each vertex y in $V-S$, perform

$$D[y]\leftarrow \text{Min}(D[y],D[x]+\text{AdjTab}[x,y])$$

5. Repeat steps 2–4 until all vertices $(N-1)$ are processed

The performance of the methodology can be impacted by several factors. These factors include the size of the underlying bus routes network and the number of users simultaneously accessing and using the system. The main computations in the methodology are performed by the geocoding module and by Dijkstra's algorithm. Of these two, Dijkstra's algorithm is the predominant computation in the methodology and is known to have a running time of $O(V^2)$ in the worst case scenario. However, even in the worst case scenario, Dijkstra's algorithm is appropriate for the methodology because the typical number of bus routes in a city or county is not very large. Furthermore, by implementing the methodology as a public transportation tool on high-performance servers, the impact of the algorithm's performance on the overall system's performance can be reduced. Nevertheless, the performance of Dijkstra's algorithm can significantly be improved, especially if the underlying bus routes have known special characteristics. For example, when the adjacency matrix is sparse, which means that there are a few bus routes that intersect each other, or there are a few transfers between bus routes, an alternative implementation resulting in a time complexity of $O[(V+E)\log V]$ is possible. Also if the minimum priority queue is implemented with a Fibonacci heap, a running time of $O(V\log V+E)$ can be achieved.

Another alternative for improving the performance of the methodology is to employ heuristic routing techniques in the routing module (e.g., Nilsson 1980; Karimi 1996). For example, the heuristic technique suggested by Karimi (1996) can be used in the methodology if the bus network size is very large or a low-performance computing platform is considered. The premise of this technique is to run Dijkstra's algorithm on a subnetwork of the original bus network. The purpose of a subnetwork (window) is to reduce the number of nodes in the network so that a better performance can be achieved. The subnetwork can be created by taking different approaches, for example by clipping a window which contains nodes and edges within the area near the line segment connecting the origin and destination points.

iBus Prototype

A prototype of the methodology called iBus using a web-based three-tier architecture was developed. The prototype was developed based on an open system architecture strategy allowing interoperability with respect to data and software. Described below are the mapping module, the client-side user interface program, and the server implementation of the prototype.

J. Comput. Civ. Eng. 2004.18:83-91.

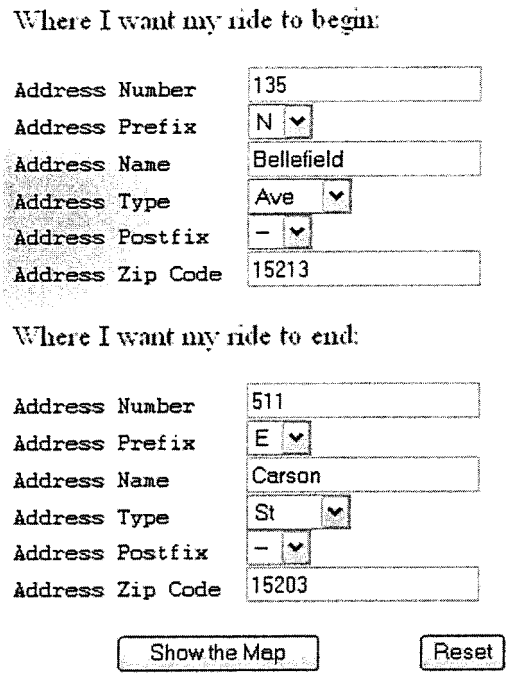
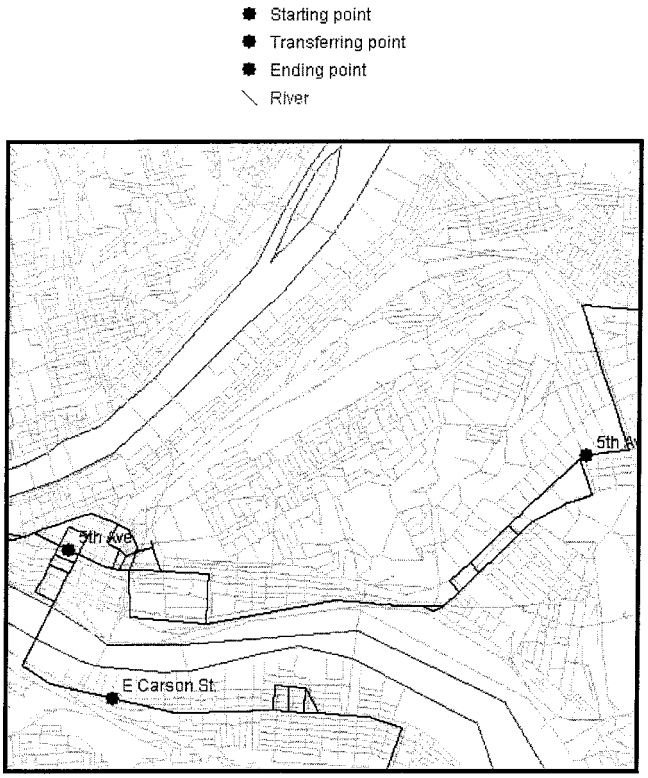


Fig. 6. Prototype iBus input applet



Board bus 77U on 5th Ave

Transfer to bus 51A on 5th Ave

Get off bus on E Carson St

Fig. 7. Sample output produced by iBus

Mapping Module

Once an optimal route is computed, the mapping module is used to display the result on a map. The mapping module uses the street network data from TIGER database to generate a map. The computed bus route, the locations where the rider is recommended to get on and off the bus, and the transfer points are superimposed on the map. The resulting image is then stored on the server and the URL address for the image is passed on to the client. The client-side user interface program simply loads the image from the server and displays the result to the rider.

Client-Side Program

To ensure flexibility in implementation and future expansion, Java applet was chosen to implement the client-side user interface. Based on the Java programming language, a Java applet is a mechanism that allows Java programs to be dynamically loaded and executed on the client computer via a Web browser. Both the *Netscape Navigator* and the *Internet Explorer* support Java applet on all major platforms. The applet in iBus is an interface for the user to: (1) enter the origin and destination addresses and (2) display the result and the corresponding map. In its current version, the prototype has limited tolerance for variations in addresses entered. Fig. 6 shows iBus applet for entering addresses. Fig. 7 shows the resulting bus route on a map.

Server Implementation

For a rapid development and for ensuring a low administrative overhead, the Windows 2000 platform was chosen to prototype the iBus server. The built-in Internet Information Services software was used as the Web server. All modules, i.e., geocoding, routing, and mapping were written in Java. TIGER database was preprocessed by a custom Java program for fast map query and rendering. The bus route data set was preprocessed and stored as a relational database file accessible via Java Database Connectivity. To simplify the implementation, a simple socket mechanism was used for communications between the client and the server.

Conclusions and Future Research

In this paper, a methodology, an internet-based GIS-T for optimizing bus riders activities, was discussed. The design options and development strategies of the methodology were described in detail in order to provide transportation developers with insights into components, assumptions, and issues of internet-based GIS-T. On the one hand, the methodology provides bus riders with trip planning related options allowing them to better schedule their activities, while on the other hand, it helps transit authorities schedule bus routes dynamically and offer bus services in an effective manner to the riders. Although a minimum number of bus-to-bus transfers was discussed as one criterion in the methodology, it is possible to customize the routing module to meet other desired criteria such as travel time. In general, the design of the methodology allows customization to meet requirements of other transit bus services and other applications such as school bus routing.

As presented in the paper, the methodology can provide bus services using different criteria and an example of which was presented for when a minimum number of bus-to-bus transfers is desired. A prototype version of the methodology, called iBus, was developed. The prototype does not utilize bus schedules in com-

J. Comput. Civ. Eng. 2004.18:83-91.

putations, as such, a bus rider may have to wait a long period of time at a transfer point, while other alternative route sequences may yield shorter travel time (even with greater number of transfers). By incorporating bus schedules into the routing module of the methodology, more practical route sequences can be computed.

In addition, the topology of bus routes strictly considers transferable bus routes to be the ones that share the same bus stops. However, a rider often makes a bus transfer on a nearby bus stop that is not located on the same street. This type of transfer was not considered in the current prototype because the two bus routes are not adjacent (connected) in the graph. Since this type of transfer can significantly reduce travel time, a buffer near the transfer points should be utilized when computing the optimal route.

Furthermore, Dijkstra's algorithm may result in a low performance when the size of the bus routes becomes very large. In this case, alternative approaches, such as heuristic techniques, may be considered. Mapping performance could also be improved by using a simpler client-side user interface which is based on HTML and the Java Server Page, especially when the client machine is not up-to-date or the bandwidth is limited.

References

ArcGIS. (2003). ''ArcGIS overview introduction.'' ⟨http://www.esri.com/software/arcgis/overview.html⟩ (March 24, 2003).

Balakrishnan, N. (1993). ''Simple heuristics for the vehicle routing problem with time windows.'' *J. Oper. Res. Soc. Am.*, 44, 279–287.

Chou, Y. H. (1995). ''Automatic bus routing and passenger geocoding with a geographic information system.'' *IEEE Proc., 1995 Vehicle Navigation and Information Systems Conf.*, Washington, D.C., 352–359.

Dijkstra, E. W. (1959). ''A note on two problems on connexion with graphs.'' *Numer. Math.*, 1, 269–271.

Ding, C. (1994). ''Impact analysis of spatial data aggregation on transportation forecasted demand: A GIS approach.'' *Proc., URISA 1994 Annual Conf.*, Milwaukee, Wis., 362–375.

Doyle, A. (1997). ''OpenGIS project document 97-009: WWW mapping framework.'' ⟨http://www.opengis.org⟩ (July 12, 2002).

Gendreau, M., Guertin, F., Potvin, J. Y., and Taillard, E. (1999). ''Parallel tabu search for real-time vehicle routing and dispatching.'' *Transp. Sci.*, 33(4), 381–390.

Gendreau, M., Hertz, A., and Laporte, G. (1994). ''A tabu search heuristic for the vehicle routing problem.'' *Manage. Sci.*, 40, 1276–1290.

Gendreau, M., Laporte, G., and Seguin, R. (1995). ''An exact algorithm for the vehicle routing problem with stochastic demands and customers.'' *Transp. Sci.*, 29, 143–155.

Guzolek, J., and Koch, E. (1989). ''Real-time route planning in road networks.'' *Proc., IEEE Vehicle Navigation and Information Systems Conf.*, Toronto, 165–169.

Ichoua, S., Gendreau, M., and Potvin, J. Y. (2000). ''Diversion issues in real-time vehicle dispatching.'' *Transp. Sci.*, 34(4), 426–438.

Jagadeesh, G. R., Srikanthan, T., and Quek, K. H. (2002). ''Combining hierarchical and heuristic techniques for high-speed route computation on road networks.'' *Computing Control, Eng. J.*, 13(3), 120–126.

Karimi, H. A. (1996). ''Real-time optimal-route computation: A heuristic approach.'' *ITS J.*, 3(2), 111–127.

Laporte, G., Gendreau, M., Potvinb, J. Y., and Semet, F. (2000). ''Classical and modern heuristics for the vehicle routing problem.'' *Int. Trans. Operational Research*, 7(4–5), 285–300.

Liu, B. (1997). ''Route finding by using knowledge about the road network.'' *IEEE Trans. Syst. Man Cybern.*, 27(4), 436–448.

Lotan, T. (1997). ''Effects of familiarity on route choice behavior in the presence of information.'' *Transp. Res., Part C: Emerg. Technol.*, 5(3–4), 225–243.

MapXtreme (2003). ''MapInfo MapXtreme for Windows-Overview.'' ⟨http://www.mapinfo.com/products/Overview.cfm?productid=1101⟩ (March 24, 2003).

Modesti, P., and Sciomachen, A. (1998). ''A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks.'' *Eur. J. Oper. Res.*, 111, 495–508.

Moore, E. F. (1957). ''The shortest path through a maze.'' *Proc., Int. Symp. on Theory of Switching*, Harvard University Press, Cambridge, Mass., 285–292.

Nilsson, N. J. (1980). *Principles of artificial intelligence*, Tioga, Palo Alto, Calif.

Ozbay, K., Karunaratne, K., and Williams, T. (2001). ''Evaluation of Garden State Parkway alternate bus routing operational field test.'' *J. Public Transp.*, 3(4), 61–83.

Pang, G. K. H., Takahashi, K., Yokota, T., and Tagenaga, H. (1999). ''Adaptive route selection for dynamic route guidance system based on fuzzy-neural approaches.'' *IEEE Trans. Veh. Technol.*, 48(6), 2028–2041.

Peng, Z. R., Groff, J. N., and Dueker, K. J. (1998). ''An enterprise GIS database design for agency-wide transit applications.'' *URISA J.*, 10(2), 46–55.

RideGuide (2003). ''It's easy as 1, 2, 3.'' ⟨http://rideguide.wmata.com/⟩ (March 24, 2003).

Sumrada, R. (2002). ''Towards distributed application of GIS technology.'' *GIS International*, 16, 40–43.

TakeTransit. (2003). ''Plan your trip.'' ⟨http://www.transitinfo.org/tripplanner/index.asp⟩ (March 24, 2003).

Transport for London (TfL). (2003). ''Transport for London.'' ⟨http://www.tfl.gov.uk/tfl/⟩ (March 24, 2003).

Thill, J. C. (2000). *Geographic information systems in transportation research*, Pergamon, New York.

Topologically Integrated Geographic Encoding and Referencing System (TIGER). (1998). ''TIGER (R)/Line files technical documentation.'' Bureau of the Census, Washington, D.C.

TransLink. (2003). ''Moving people and goods in a growing region.'' ⟨http://www.translink.bc.ca/⟩ (March 24, 2003).

Trepanier, M., Chapleau, R., and Allarad, B. (2002). ''GIS architecture for transit information website.'' *Proc., 2002 Geographic Information Systems for Transportation (GIS-T) Symp.*, Atlanta.

Wang, H., and Zhang, B. (1992). ''Route planning and navigation system for an autonomous land vehicle.'' *Proc., 3rd Int. Conf. on Vehicle Navigation and Information Systems, VNIS*, Oslo, Norway, 135–140.

Wartian, C., and Gandhi, S. (1994). ''Transportation model network interface with geographic index database system.'' *GIS/LIS 1994 Conf. Proc.*, Phoenix, 787–795.

WebMap. (2003). ''GeoMedia WebMap professional.'' ⟨http://www.intergraph.com/gis/gmwmp/⟩ (March 24, 2003).

J. Comput. Civ. Eng. 2004.18:83-91.