Real-Time Route Planning in Road Networks

John Guzolek, Senior Software Engineer
Edward Koch, Senior Software Engineer
NAVIGATION TECHNOLOGIES CORPORATION
740 East Arques Avenue
Sunnyvale, California 94086
Tel: 408/737-3200
Fax: 408/736-3734

Abstract

Real-time route planning forms an integral part of systems ranging from in-vehicle route guidance to dispatching and off-line direction generation. A route planning subsystem is responsible for determining optimum routes between a user specified origin and destination, which can be explicated into a set of directions that may be used for route guidance.

This paper presents the methodologies used in a Route Planning subsystem that has the ability to:

- Determine the best routes in the road network using different optimization criteria.
- Incorporate a wide range of information from the road network database into the route planning. This includes the possibility of using real-time traffic information.
- Plan a strategic route in a fast and efficient manner.
- Perform real-time tactical navigation in conjunction with in-vehicle route guidance when the driver deviates from the strategically planned route.

Much of the capability and functionality of the route planning subsystem is derived from a database which was specially designed to facilitate the path planning problem. This database is unique in both the breadth of information it contains and the way in which it is structured.

Route Planning Context

In a variety of contexts, it is useful to be able to determine an optimal (or at least highly satisfactory) driving route between two locations. Some of these contexts are:

- In-vehicle Route Guidance Systems, which combine route determination with direct real-time guidance to the driver on pursuing the route.
- Out-of-vehicle Route Finding Systems, which provide the user with the ability to plan one or more routes on paper, think through itineraries, etc. Such systems can provide the capability to browse destination alternatives (e.g., restaurants) before selecting an actual destination.
- Applications which are more interested in the characteristics or consequences of a route than the route itself. For example, a route planning system can be used to help determine the statistical distribution of travel times to a school from a selection of locations within the school district.

All of these contexts contain a variety of functionality surrounding the route planning itself, including user interface, system coordination, database access, memory management, etc. (see Figure 1). For in-vehicle applications, the route guidance system must also accurately track the position of the vehicle, so it can present the next driving maneuver at the appropriate time and so it can determine whether the driver has deviated from the prescribed route, which might trigger a re-routing requirement.

In this paper, we will make some references to this surrounding functionality, but primarily will focus on the route planning process itself with emphasis on in-vehicle applications. The description of route planning presented here corresponds to the work performed at Navigation Technologies Corporation (NavTech) in successfully developing:

- Complete self-contained route generating systems.
- Software for operational in-vehicle route guidance prototypes.

To the authors' knowledge, NavTech is the only organization which has successfully produced a metropolitan-area-sized database with the detail and accuracy suitable for route planning and guidance.

Route Planning Overview

A Route Planner's main responsibility is to determine an "optimum" route between some origin and destination. Optimum in this sense refers to the best route as measured against some criteria. The following criteria can be selected among by the user and used for determining optimum routes:

- Minimum time.
- Minimum distance.
- Fewest turns (i.e., maneuvers or instructions).
- Avoiding or encouraging freeways.

For in-vehicle applications there are two similar, but distinct scenarios for doing route planning. The first is done prior to taking a trip where a path is planned from the vehicle's fixed current location to some specified destination. This is termed "strategic" route planning.

The second scenario arises if an en-route vehicle deviates from the previously planned route (e.g., makes a wrong turn or misses a freeway exit). In this case the driver may need a new set of directions to enable him to get to his destination and thus path planning is again required. This is termed "tactical" route planning and differs from strategic on two points. First, it is under a much tighter time constraint since the vehicle may still be moving. Second, the user may simply want the quickest path back to the original route since he may already be familiar with that set of directions.

CH2789-6/89/0000-0165$1.00©1989IEEE

Toyota Petition '628 Exhibit 1022
Page 1 of 5

The problem is to find an optimum route under the constraints that the route:

- Must be determined in a reasonable amount of time.
- Must be done using a fixed amount of memory.
- Must be done using a database that may exist on a relatively slow secondary storage media such as a magnetic or optical disk.

The Route Planning subsystem discussed in this paper contains three major components to solve this problem:

- A database with sufficient breadth of information to allow path planning to be performed.
- A modified A* graph search procedure for searching the road network to determine the optimum route.
- A database structure and interface which enables the A* algorithm to efficiently access all necessary data.

Database Content

The database contains the road network of the area for which route planning is required, but to route effectively and allow the user to specify destinations efficiently, the database also includes [2]:

- Road classification (highways, residential roads, etc.)
- Effective time-of-day travel speed.
- Turn restrictions at intersections and one-way restrictions.
- Points-of-interest and business information.
- Existence of car-pool lanes.
- Roadway topography: position and distance.
- Address ranges for roads.
- Signals; turn and driving restrictions.
- Speed limits and other impedance factors (center dividers, etc.)
- Non-road transportation systems (rail, bus, etc.)
- Building facility characteristics (e.g., parking)

Road classifications distinguish freeways at the highest level from dirt roads or side streets at the lowest level. Also included in the database would be items which are specified and maintained by the user, such as:

- Detours.
- Locations to be used for Route Planning (e.g., the user's home).

Generating an Optimal Route

The goal of the path planning module depicted in Figure 1 is to provide an "optimal" route based on current location, desired destination(s), optimization criteria, and any detour information. The network database is assumed to be represented by a fully connected graph.

The primary tool for finding the "best route" is a well known AI graph search technique called A* which was introduced by Nilsson[1]. A* is a general graph search algorithm that is guaranteed to find the "optimum" route through a graph if one exists. In general, the A* algorithm performs a directed, breadth first search from source(s) to destination(s). For detailed description, definition of terms, proofs, and the implementation of the A* algorithm, the reader is referred to [1]. For the purpose and scope of this paper, the following terms will be defined:

- A node in the search graph corresponds to an intersection of streets in the road network.
- A segment in a the search graph corresponds to a street in the road network. Segments are directed arcs (unidirectional). A two-way street would be represented by two separate arcs (segments).
- A path is a directed sequence of one or more segments joining two nodes.
- Every node in the search path has a cost 'f' associated with it and is defined by: $f(n) = g(n) + h(n)$

The term g(n) is the "known" cost to get from the source to node 'n'. The term h(n) is the heuristic estimate (intelligent guess) of the cost of the path from node 'n' to the destination. If the heuristic cost h(n) is a lower bound on all paths from node 'n' to the goal, then it is called an "admissible" heuristic and the search algorithm is guaranteed to find an optimal path from the source to the destination. In this case the algorithm is called A*. In other words, the heuristic h(n) must be more optimistic than the actual cost from the node 'n' to the destination. A typical admissible heuristic for route planning is the "Euclidian" (straight line) heuristic. An over-optimistic heuristic will still work and give an optimal result but will produce computationally longer searches than a "better" heuristic. When the heuristic cost, h(n), is zero for all nodes, the A* algorithm degenerates to what is known as a Dijkstra search.

Adaptation of the A* algorithm to path planning in a road network

In theory, the A* algorithm alone solves the route planning problem quite handily. This would also be so in practice if one did not have to worry about the two important physical limitations of the route planning system:

- Limited fixed size memory
- Slow secondary memory access

When implemented with proper heuristics, the A* algorithm has been shown to produce the optimum path while considering the fewest number of nodes in the graph. Nonetheless it still fills up memory very quickly, since once a node is examined, it must stay around in memory until the search concludes. For this reason, we must implement some method of "pruning" the search tree.

In addition, the A* algorithm associates costs only with segment arcs and assumes no cost between choosing particular descendents of a node. The road network graph, however, requires taking into account not only the characteristics of the segments traversed, but also the characteristics of the

Toyota Petition '628 Exhibit 1022
Page 2 of 5

nodes crossed. In particular, given multiple paths to and through a node, the cost of passing through the node will differ among the paths. For example, a "cheaper" path to a node which would then require a left turn to be taken at the node may be more costly overall than a more expensive arriving path that travels straight ahead through the node.

The A* algorithm has been modified in order to optimize its performance within the physical resource limits of the host computer by:

- Tailoring the optimization 'g' cost criteria
- Heuristic 'h' function variations
- Search graph pruning methods

Optimization Criteria

Optimization criteria are implemented by run-time path planning adjustments to the 'g' cost in the A* search. (The 'g' cost is the known cost from the source to the current node). The 'g' cost for a particular iteration of A* includes *weighted* combinations of:

- Segment speed limit
- Segment distance
- Time of day traffic conditions
- Traffic signal and stop sign delays
- Type of segment surface (street, artery, freeway, ramp, etc.)
- Turning delays at an intersection

The previously mentioned optimization criteria: minimum time, minimum distance, fewest turns (instructions), and avoid freeways, may all be implemented by the appropriate relative weighting of the above terms.

Heuristic Variations

The heuristic function h(n) has a large impact on the search time required to plan a route. Generally, the "Euclidian" heuristic gives an optimum path since the heuristic is admissible. There are occasions, however, to use other heuristics:

- Dijkstra heuristic: $h(n) = 0$

  Using no heuristic allows more nodes to be searched in a given time since there is no overhead associated with computing the h(n) heuristic value. Dijkstra searches are useful for generating short routes or navigating to non-specific destinations (e.g., generate a path to the nearest highway).

- "Near" admissible heuristics

  These heuristics are more heavily directed than the "Euclidian" heuristic in an attempt to further minimize the number of nodes searched while still generating "near" optimal routes. Near admissible heuristics are not guaranteed to generate optimal routes, but in most cases do. In the remaining cases the routes are "near" optimal.

Search Tree Pruning

In order to reduce the amount of time required to determine a route and work within the limited memory constraints, search graph pruning techniques are employed. This is especially useful on very long routes. "Pruning" simply means ignoring some path possibilities while the search is in progress. Technically, this implies that the solution path generated may not be optimal or, in the worst case, that a solution may not be found. Pruning, therefore, must be done cautiously, and only on those paths deemed to have a very low probability of being optimal.

Generally, the road network graph is well connected and contains a very large number of alternate paths for a given source and destination pair. Pruning the lowest probability paths allows the search to proceed much faster while still producing "near optimal" paths.

Pruning may be implemented in several ways:

- Near admissible heuristics
- Staged searches
- Neglecting low probability paths

Near admissible heuristics, described earlier, effectively prune low probability path alternatives as the search progresses, allowing the search to run deeper in the search graph without running out of memory.

A "staged" search breaks the search into a serial list of sub-goals. For instance, a very long distance may be searched faster without running out of memory if we generate the path in three steps, by identifying an appropriate intermediary point or route:

1. Find a path to the intermediary.
2. Find a path from the intermediary to the final destination.
3. "Sew" together the partial paths to generate the total path.

Pruning by neglecting low probability paths during the search is very effective on paths traveling longer distances on the highway network. Paths are neglected by deleting part of the list of nodes to be explored when memory limitations are reached and also by suppressing the insertion of certain path possibilities into the list.

Tactical vs. Strategic Route Planning

Tactical route planning is the process of re-figuring a route for a vehicle which has deviated from the previously planned route. Tactical route planning is simply a special case of the strategic route planning algorithm, which includes two special considerations: First, since the vehicle may be in motion, the system must either predict an origin based on current vehicle position, heading, and speed; or use an area around the vehicle as an origin, rather than a point. Second, the destination goals for navigation are considered to be any node on the original strategically planned route (including the destination, of course). Thus, tactical route generation either finds a path back to the original route or an entirely new route to the original destination.

Detour Handling

The graphical image of the road network may change from day to day due to new road building, highway maintenance, etc. Therefore, it is very likely that some parts of the route planning system's static image of the road network may differ from the "real" road network on a given day. (For instance, a one mile stretch of highway may be closed for repairs today). To

Toyota Petition '628 Exhibit 1022
Page 3 of 5

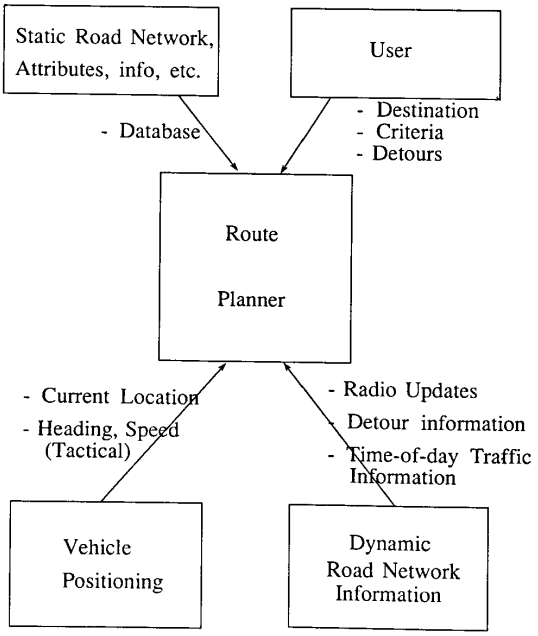


Figure 1. Route Planning System, Virtual Input Diagram

prevent the route planning system from generating routes through inaccessible segments, up to date "detour" information should be used when available. Detours may be input to the system by the driver, or received via radio broadcast if the vehicle is so equipped. In either case, detours are managed by the database access interface and thus are transparent to the route planning system. Handling detours at the database access interface makes detours appear to the route planning system as if they were part of the static road network.

Database Structure

The database must be optimized to facilitate geographic access of arbitrary rectangular regions. Efficient data access to node / segment descendant and ancestor information is also essential for route planning.

Physically the database is organized into a multidimensional binary search tree (k-d tree) [3] which uses the notion of parcels, zones and regions. Parcels are the lowest logical level and define a geographical area that contains a certain number of road segments (always less than some maximum). Zones are the next highest entity and are simply a collection of parcels and likewise regions are a collection of zones.

The organization of parcels, zones, and regions allows separate geographic regions to be "sewn" together in a fairly easy and transparent manner. This allows the database to be split and physically exist over numerous physical storage volumes. Typically the geographic area covered on one disk will overlap the geographic area on another disk.

The structure of a k-d tree allows for quick and easy access of data that is specified only by geographic latitude and longitude positions. Essentially given any latitude and longitude the k-d tree is searched down through the regions, zones, and parcels until the parcel containing the information is found.

A parcel is the actual block of information that gets swapped into memory from the secondary storage media. The size and placement of parcels on the disk are adjusted to optimize the access of data from the disk and minimize the amount of information which must be read. In addition the database access routines use a parcel swapping algorithm which tries to minimize the possibility of swapping out a parcel that may contain information the graph search procedure may need in the future.

The Route Planning subsystem interfaces with the database through a layer of general purpose access routines. The physical database structure and parcel management are completely hidden. The nodes and segments contained in the parcels are referenced in the application software by using a virtual node identifier. The following database access functionality is implemented:

- Locate nodes in the database by geographical position.
- Retrieve all nodes and segments which are considered "successor" to a particular node or segment. This is used to traverse the road network.
- Retrieve all information about a given node.
- Retrieve all information about a given segment.

Toyota Petition '628 Exhibit 1022
Page 4 of 5

- Retrieve all attribute information about a road or segment. (Addresses, road names, signs, etc.)
- Retrieve all segments within the bounds of a specified region.

There are numerous other access routines which are used to access road names and points of interest etc. These are not used directly by the Route Planning subsystem, but are used to help the user specify the origin and destination of Route Planning and also to generate directions for the user once the route has been determined.

Conclusions

Accurate, real-time route planning is not only feasible - it is now successfully being done. It requires good software and an outstanding navigable database, both of which are currently being provided by Navigation Technologies. NavTech has incorporated the database and route planning software into prototype in-vehicle route guidance systems, and into self-standing out-of-vehicle systems.

NavTech's commercial DriverGuide product is currently in use in about 70 locations throughout the San Francisco Bay Area, offering detailed printed driving directions between any two addresses in the Bay Area. NavTech has also developed a PC-based product (Personal DriverGuide) which provides similar services on the purchaser's IBM PC/AT or equivalent. The database and all necessary software are readily installed on the PC's hard disk.

Acknowledgments

The authors are happy to acknowledge the assistance of Richard Weiland (SEI Information Technology) in the preparation and refinement of this paper.

References

[1] Nilsson, N. J., Principals of Artificial Intelligence, Palo Alto, CA, Tioga Publishing Co., 1980, Chapter 2.

[2] Cass, R. A., "Digital Databases for Vehicle Navigation a Review of the State of the Art," Proceedings of 20th International Symposium on Automotive Technology & Automation, pp 1241-1253, Florence Italy, May 1989.

[3] Bently, J. L., "Multidimensional Binary Search Trees in Database Applications," IEEE Transactions on Software Engineering, vol. SE-5, no.4, July 1979, pp. 333-340.

Toyota Petition '628 Exhibit 1022
Page 5 of 5