Navigation and Multimodal Transportation with EasyTransport

Maria Fragouli and Alex Delis, *University of Athens*

Today's intelligent navigation and transportation systems influence the way travelers access information to plan trips efficiently. By and large, such systems derive their navigational suggestions out of precompiled data regarding direct road connections recorded in their databases. They typically facilitate passenger navigation within predefined geographic regions, provide location tracing and direction functionalities over cellular networks[1] or the Web (www.ecitysoftware.com/3DCities.html, www.euroave.com/maps), and deliver accurate but not necessarily optimal transition paths (www.mapsonus.com).

The EasyTransport system helps travelers plan the most efficient route and choose the best modes of transportation in urban or large-scale geographic areas. Users can adapt the generated outcomes through an interactive interface.

This article proposes EasyTransport, an intelligent navigation and multimodal transportation guidance system that focuses on near-optimal trajectory planning and informed decision making. EasyTransport introduces a number of novel features. At its core lies a breadth-first-search–based algorithm that dynamically produces shortest paths. Because the algorithm runs in an iterative and spatially uniform manner, it can scale up to encompass large geographic areas. It uses heuristics to minimize potentially large search spaces and thus prevent unproductive computational steps. See the "Related Work in Intelligent Transportation Systems" sidebar to learn what other researchers have done in this area.

Design considerations

EasyTransport features a dual mode of operation depending on the type of information the user intends to extract. In the first mode, *Navigation*, the system directs a user's transition from a starting point to his or her destination. It traces the shortest paths, typically multiple if any, that connect them. In the *Transportation* mode, EasyTransport combines all available transport options along each path generated during Navigation. These options can include all means of public transportation such as buses, trains, and even planes and boats—supporting the notion of multimodal transportation. The two modes of system operation are interdependent: Navigation must precede Transportation.

The system also lets users adapt the presentation of generated answers to more effectively address their fast delivery and easy exploration. EasyTransport manages large volumes of data related to each geographic area as well as to the multiple transport alternatives involved. Finally, the system seeks to offer an interactive, high-performance navigation and transportation guidance tool that delivers its results over the Web and might also function with geographic information systems.

Outline of operation

During a session, EasyTransport functions within a selected geographic area, inherently restricting a transition's range. The system presents this working area as an on-demand-loaded map—the focus of the Java-based user interface designed to deliver EasyTransport's services through a Web browser. The system considers this working area a *grid* consisting of *cells*. A cell represents the system's basic processing unit and defines its operational granularity. Moreover, ordinary grid cells differ from *dead-cells*, which represent areas lacking transportation means. Figure 1 demonstrates the segmentation of a geographic area into these two sets of cells comprising the working grid.

A user indicates via the interface a transition's *input points*, referred to as S (start) and D (destination), and the system derives the paths connecting them. Each path from a cell consists of a sequence of neighboring cells that can be reached after every primitive, allow-

1541-1672/05/$20.00 © 2005 IEEE
Published by the IEEE Computer Society

Related Work in Intelligent Transportation Systems

There is a significant body of supporting work on navigation and transportation systems that enables travelers' navigation and mobility planning.

RETINA[1] is a real-time navigation system providing instantaneous traffic information as well as path-searching and direction functions to mobile clients. Its underlying path-finding mechanism, based on static road connections, calculates the shortest path connecting the traveler's current position to a destination.

Available Web-based guides (www.ecitysoftware.com/3DCities.html, www.euroave.com/maps) enable traveler location tracing, facilitate passenger navigation within predefined geographical scopes, and offer location-based services with limited user interactivity.

Maponus (www.mapsonus.com) provides not only map and geographic information system services but also trip-planning facilities, suggesting accurate but not necessarily optimal transition paths.

Mybus[2] is an intelligent transportation system that helps bus riders make informed decisions about their unimodal travel options. It predicts bus departures at specific locations throughout a transit region and delivers that information to Web browsers and cell phones using real-time data flowing from an automatic vehicle location system.

Gerd Kramer proposed an architecture for an automatic road transportation system consisting of a network of traffic surveillance and road condition sensors as well as a centralized information system that provides interactive services to drivers.[3] Other projects that incorporate graph-theoretic algorithms to develop navigation applications work with limited search spaces and usually trace queried paths by extensively visiting nodes.

Michael Hallett, in his 1997 work "BFS in the Underground of Barcelona, Spain," applies the breadth-first search algorithm to a limited set of transition points corresponding to the subway stations in Barcelona, Spain, while Sartaj Sahni uses the BFS algorithm to trace paths between any two points on a 2D space, or grid.[4]

References

1. K.Y. Lam et al., "RETINA: A REal-time TraffIc NAvigation System," *Proc. 2001 ACM SIGMOD Conf.*, ACM Press, 2001.
2. S.D. MacLean and D.J. Dailey, "MyBus: Helping Bus Riders Make Informed Decisions," *IEEE Intelligent Systems*, vol. 16, no. 1, 2001, pp. 84–87.
3. G. Kramer, "Envisioning a Radar-Based Automatic Road Transportation System," *IEEE Intelligent Systems*, vol. 16, no. 3, 2001, pp. 75–77.
4. S. Sahni, *Wire Routing*, Univ. of Florida, 2000; www.cise.ufl.edu/~sahni/dsaaj/JavaVersions/applications/wire_routing/wire_routing.htm.

able movement has been explored in all eight possible directions, as Figure 1 also outlines. However, the number of possible neighbors might be smaller due to the existence of dead-cells, which carry no useful information and are simply ignored. Every cell is defined by its spatial coordinates on the grid. The movement toward a neighboring cell happens by advancing or decrementing one or both of the cell's coordinates. A cell's size depends on the density of available transportation means within the particular transition area. Smaller cell sizes demonstrate higher availability of transport options. The consideration of all areas as grids regardless of cell size ensures their uniform treatment by the system's engine.

All system operations take place on a per-cell basis. Once the path-planning phase concludes, the system can draw each suggested route on the map element of its interface in the form of edges connecting constituent successive cells. If the system is in Transportation mode, it retrieves information regarding available transportation means for every cell along a selected route; these means are graphically rendered on the map. From this point on, the system must retrieve a rather substantial volume of data to adequately satisfy a traveler's query. Clearly, the longer the route, the more data there is to retrieve, process, and render. The system is set up in advance for every region available to expand the search, more like a "one view fits all" fashion with regard to grids, cell sizes, and transportation alternatives. All this data resides in the system's back-end database. Figure 2 presents an abstraction of the overall system architecture as we've described it so far.

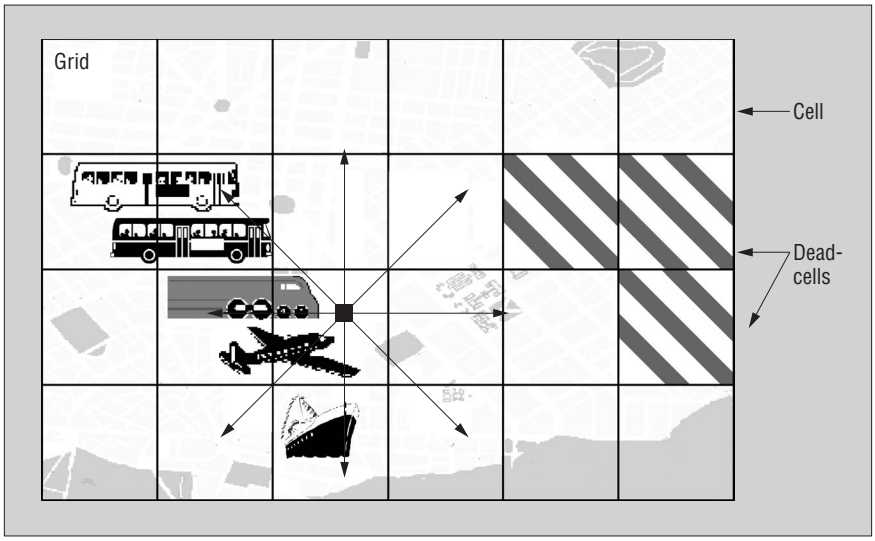

Figure 1. A geographic area represented as a grid of ordinary cells and dead-cells to facilitate Navigation. Multimodal Transportation entails diverse transport options.

Algorithmic issues

In our implementation, we view both the grid and neighboring cells as elements of a highly connective graph of vertices, occasionally featuring a dense formation of cycles. Given S and D points, we seek an algorithmic approach that guarantees delivery of near-optimal solutions, if such solutions exist, by expanding only those

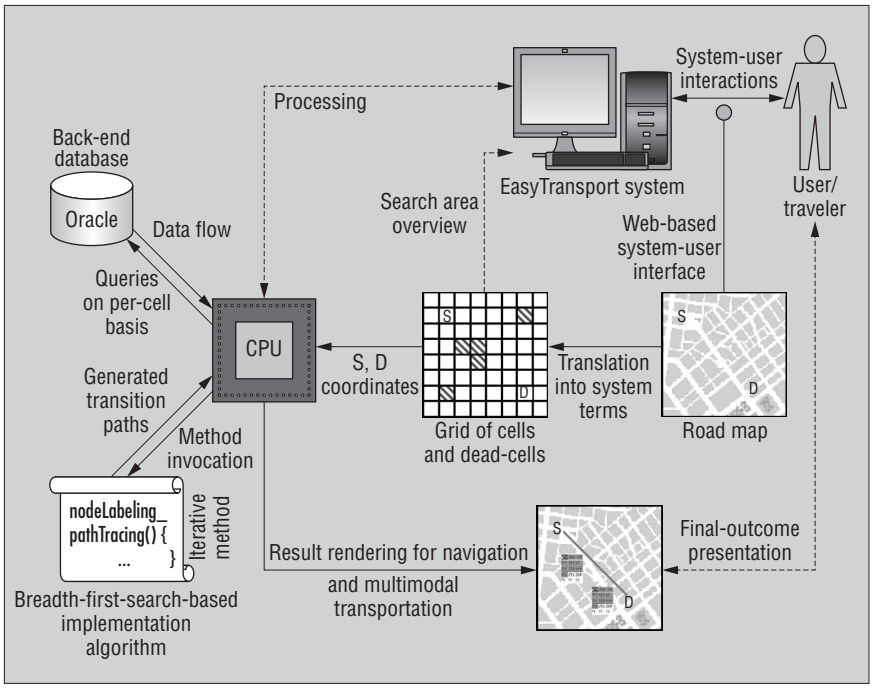

Figure 2. An abstraction of the EasyTransport system architecture.

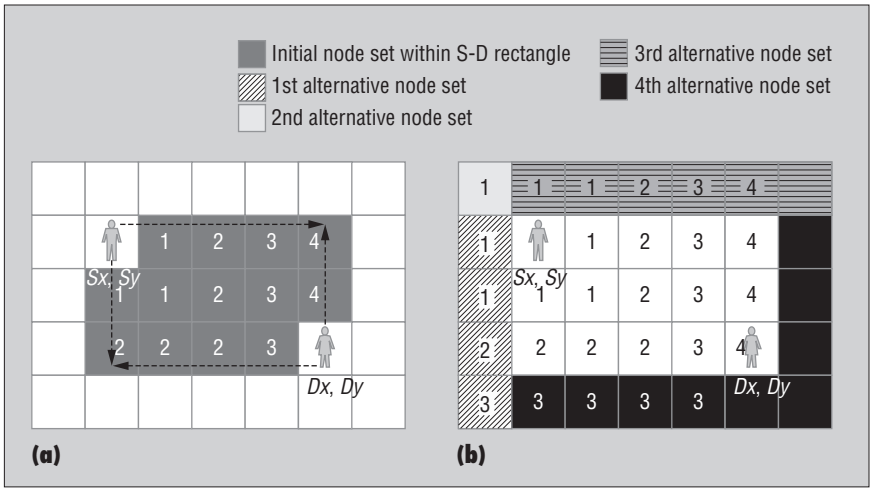

Figure 3. Sets of nodes the algorithm successively works with: (a) nodes in the S-D rectangle; (b) additional node sets incrementally considered to expand the search.

nodes that are necessary. The traditional brute-force search could certainly produce the requested answers but would also incur a prohibitive cost.[2] By assuming that each transition to a neighboring node "costs" a single unit and by applying heuristics, our approach results in estimating the minimum transition cost to reach node D. While searching for a path, our iterative algorithm labels eligible nodes with cost values. Nodes that overestimate the transition to the goal cost will eventually be pruned.

Given the preset cell granularity, our algorithm almost certainly yields multiple paths. Although an *A** solution[3] would also trace the shortest path, it assumes a unique answer, and, more importantly, its admissibility property is too restrictive because it requires excessively long times to differentiate among almost equivalent candidates.[4] The use of *IDA** (Interactive Deepening-A* search solution) might offer multiple paths, but *duplicates* (previously considered nodes) are permitted to reexpand, possibly leading to expensive backtracking in the search area.[5] Finally, adopting a breadth-first-like search algorithm[2] (as opposed to a depth-first-like one) to examine all the current attached nodes fits nicely with our specific application. This is because BF-like approaches build their results uniformly by first visiting all eligible nodes located within k units from the start, then those located $k + 1$ units away, and so on.

As EasyTransport's engine examines candidate cells for deriving paths, two concurrent operations take place: it assigns estimated heuristic transition cost values to the nodes in the search space (see Figure 3), and it traces the paths that answer to the submitted transition query. Because all paths must comply with the shortest-path criterion, we conjecture that candidate routes are probably located within the conceptual rectangle outlined by the coordinates of the S and D nodes, as Figure 3a depicts. By focusing on the nodes enclosed in the S-D rectangle, we reduce the algorithm's operation range and enable the faster generation of sought paths.

Clearly, this assumption represents an optimistic search effort that might not work well in light of dead-cells dominating the area within the S-D rectangle. In this case, our algorithm gradually considers additional sets of nodes and can ultimately perform an exhaustive expansion of all nodes if necessary. Figure 3b illustrates the sets of nodes we incrementally consider in our approach. More specifically, for every such set, our engine labels nodes only if their transition cost is less than or equal to the one already estimated en route to D. In the worst-case scenario, our engine is forced to work with all these sets outside the boundaries of the initially designated rectangle. However, the probability of returning positive results in an urban terrain after examining the additional node sets mentioned is rather slim, as the number of dead-cells within such an area is significantly limited. Each time the system considers a new set of nodes, the entire attempt to trace shortest paths repeats. The algorithm proceeds while preserving earlier assigned cost values. If all search possibilities are exhausted in the entire problem space, the algorithm fails to render a plausible outcome.

The algorithm

Figure 4 presents the algorithm outlining the core iterative node-labeling and path-tracing mechanisms in EasyTransport's engine.

Structures and variables

We use a number of variables and data struc-

tures to represent the input points' coordinates, the working grid, and the set of dead-cells. During each iteration, the variable **pathLength** represents the length of the shortest path connecting **S** with the node currently under examination. As the algorithm moves toward **D**, it gradually advances **pathLength**, which in its final assignment represents the length of the shortest path(s) found. We use a **queue** structure to temporarily store the coordinates of the nodes visited at each iteration and the nodes next to expand; the **queue** also ensures that the search proceeds in a BF-like fashion. The algorithm returns a **pathNodes** vector containing all visited nodes, the combination of which will produce the shortest paths.

```
input: Coordinates of S & D nodes
output: Structure (Vector) pathNodes containing ordered sets of path-nodes
...

begin
while (D is not reached) do
    for (all eight possible neighbors) do
        if (current neighbor within grid and node-set boundaries
            and not dead and not yet visited) then
                assign it with a label equal to that of its parent plus one;
                insert it into queue so that it is later expanded;
                insert it into pathNodes along with its label;
                adjust pathLength to reflect current transition cost;
        end if
    end for
    if (queue is empty and D is not reached) then
        return null;  {no shortest path exists between S, D}
    else
        if (D is reached) then
            break;  {exit while loop}
        end if
    else
        get a node out of queue;
        set current node equal to the one just out-of-the-queue;
    end if
end while
return pathNodes;
end
...
```

Figure 4. Outline of the EasyTransport algorithm for iterative node labeling and path tracing.

Node-labeling mechanism

The algorithm loops as long as node **D** is not reached. While at a node, all eight neighboring nodes are candidates for a visit, unless one of them is node **D** itself. If they fall within the current range of operation—that is, within framework and node set boundaries—and are not dead, they are then examined to identify duplicates. If a node is visited for the first time, it's labeled and inserted into the **queue** so that it is later expanded. Moreover, the value of **pathLength** is accordingly adjusted. The execution proceeds until all nodes in the examined node set successively in and out of the **queue** are assigned with a cost value. Upon termination, the method has computed the end-value of the **pathLength** variable, which is the shortest path to reach **D**.

Path-tracing mechanism

As nodes are labeled, paths are traced at the same time. The algorithm almost certainly results in multiple paths, given the adopted cell granularity. As the algorithm visits and assigns cost values to nodes, it inserts these nodes—that is, their coordinates—along with their corresponding labels into the vector **pathNodes**. The BFS-based mode of operation warrants that when the method terminates, **pathNodes** will be populated with node sets ordered according to their **pathLength** assignments. To generate the sought paths, we combine all nodes in each of these sets, provided that the nodes with consecutive cost values are also grid neighbors. When no more nodes are left for expansion in **queue** and node **D** has not yet been reached, the method reports its inability to deliver the requested shortest-path answers.

Figure 5 depicts the outcome of the algorithm's iterative operation on a grid segment of a transition area corresponding to approximately 3.00 km^2, consisting of 21 cells. Transition to neighboring cells proceeds clockwise and only to the ones not yet visited. This example uses a scale of 1:13,000 with regard to the real-area representation and is only illustrative as to the multiple answers produced while operating in Navigation mode.

Transportation mode

When EasyTransport operates as a transportation guide to facilitate travel planning, it retrieves, processes, and delivers data pertinent to the available transportation options along a designated route consisting of a sequence of cells. The system maintains information for every cell regarding the transportation means traversing the specific geographic area. Once it has derived a path, the system retrieves this information on a per-cell basis and renders it as part of the interface. Figure 6 shows how EasyTransport presents transportation options to the user (for the randomly selected case of Path 7 in Figure 5), taking into consideration the transition's direction. The representation chosen corresponds to the perception that real passengers are accustomed to in reference to itineraries and local naming conventions used for transportation means.

Engine and scalability issues

To make our system scalable, we pursued several issues that enable fast responses to user requests regardless of the geographic area considered.

Using key-nodes

When segments of the geographic region that the engine works with offer few transport options, our algorithm doesn't need to work extensively on the whole set of constituent cells. Instead, we designate a number of nodes in the grid as *key-nodes* for navigation. Key-nodes are spread throughout the grid, preferably in a uniform fashion and within a specific distance, and act as *pivots* (or *landmarks*) within a specific geographic area. Their selection is greatly affected by the existence of dead-cells in their vicinity and the scarcity of transportation means. Each key-node knows about the key-nodes that are considered its peers. Consequently, key-nodes are all webbed in a graph as depicted in Figure 7, where dashed lines indicate neighboring key-nodes. They are further associated with sequences of grid cells that constitute the paths connecting them.

EasyTransport's core algorithm can be employed on top of key-nodes. Initially,

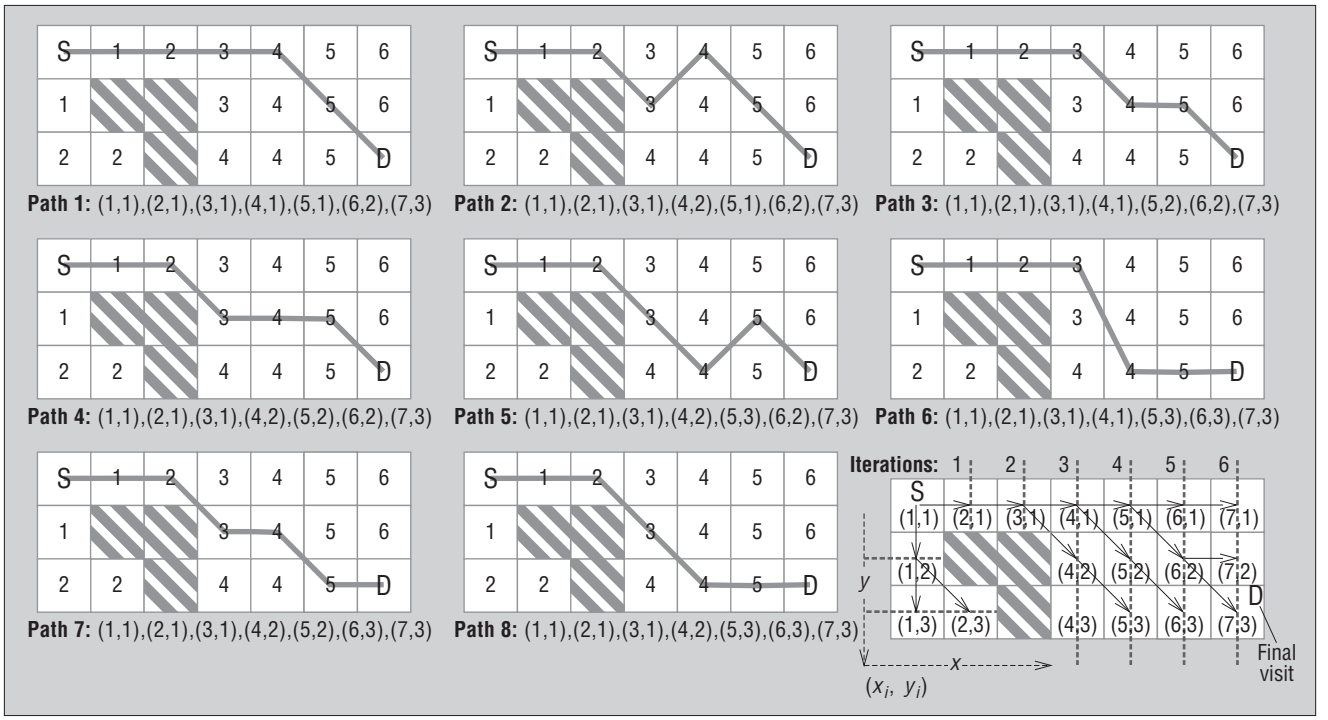

Figure 5. A sample path set generated to answer a user's Navigation query for a limited area of 21 cells.

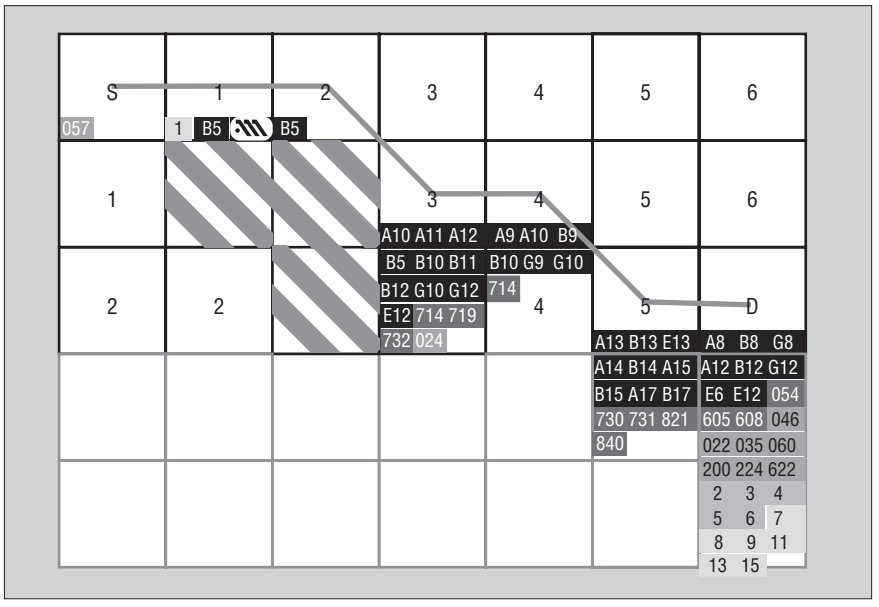

Figure 6. Display of multimodal output when the Transportation mode is on.

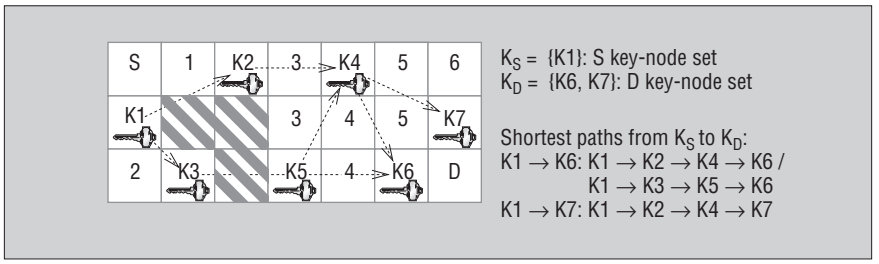

Figure 7. Key-nodes and their neighbors, all webbed in a transition graph.

EasyTransport's engine begins searching from node S and tries to locate its closest key-nodes. For instance, in Figure 7, key-node K1 is designated as the follow-up node to S. In the same spirit, the set {K6, K7} offers the closest key-node follow-up set from node D. The algorithm now divides the search area into sets of key-nodes, similar to the baseline approach, and works with them incrementally until one or more paths (if any), consisting of key-nodes alone this time, are returned. For the sample grid area in Figure 7, the algorithm needs to consider only eight nodes (for the path combination of the two key-node sets {K1} —> {K6, K7}) out of the whole set of (ordinary) nodes located within the conceptual S-D rectangle (18 nodes). It generates the end-paths using predefined path information—that is, the sequences of ordinary grid nodes associated with the respective key-nodes comprising the key-node paths.

Whether we can reduce the number of iterative path-finding steps substantially or not is a function of the density with which grid nodes are designated as key ones. Essentially, there's a trade-off between the number of key-nodes selected in a search area and the total volume of information residing in them. The fewer key-nodes selected, the greater the volume of information we must maintain and process for each one of them. Should the number of key-nodes approach that of grid

nodes, the current approach's derived benefits would greatly diminish. EasyTransport's engine can employ both cell-based and key-node-based approaches and specialize its operation depending on the area instances it is to work with. In areas offering many transportation options, the engine uses the baseline (cell-based) approach, whereas in regions that feature nondense transport hubs, single-route suggestions, or unimodal transportation, the key-node approach is preferable.

Determining cell size

The system's overall response time is a function of the number of cells comprising the grid. Smaller grids incur shorter execution times because fewer cells are expanded during the Navigation mode. On the other hand, smaller cell sizes better distribute information and reduce the overall data processing per unit when operating in Transportation mode.

The most important factor affecting the selection of cell size appears to be the morphology of the search space with respect to the transport density that can be observed within its various segments in real-world settings. Evidently, a high density of transportation means requires finer segmentation of the search area into smaller-sized cells compared to areas with little or no diversity of transport options. In our prototype, the default cell size is set to 0.15 km^2, which corresponds approximately to an area of seven to eight city blocks.

Back-end storage support

For EasyTransport to function effectively, voluminous and well-designed data must be readily available at all times. When working with large-scale terrains, loading all the necessary information in main memory would exceed permissible storage limits. Instead, we opt for using a database. In our Oracle implementation, we relate Entity-Relationship Model entities on a per-cell basis. So, the pair of coordinates for a cell is the deciding factor for retrieving requested data. In general, a cell is associated with

- the grid it's part of,
- a value representing the distance that a cell spans,
- a set of available transportation means,
- a traffic density characterization with regard to the traffic situation at specific time intervals, and
- the approximate time required for specific means of transportation to traverse the area of transition.

To be able to indicate a selection as preferable to other alternatives, the database also maintains information concerning descriptive and graphical itineraries, departure schedules, and characterizations of the stored means of transportation. When we need to support a utility not already provided in the current design as a consequence of a newly posed user requirement, we can augment our current database schema to make additional information resources available. Moreover, retrieving data produces no substantial overhead for the system's core path-finding engine because all the processing occurs after the requested paths are generated.

> By allowing cell size diversity throughout a working area, we can efficiently facilitate the system's operation in large-scale geographic regions.

Enabling large-scale operation

Allowing key-nodes and varied cell sizes means that our search space can have multiple grids. However, once the system enters a particular grid, it operates within its prespecified cell size. The final output results from combining the separate sets of answers generated for each such grid that is part of the geographic area of focus.

The different granularities might also be reflected in the result delivered to the user for both the Navigation and Transportation modes. The same geographic area can potentially participate in multiple searches depending on the type of launched query. This is the case when requesting transportation alternatives for traveling between two cities and then between any two suburbs within these metropolitan areas. Initially, when an area covers a rather wide scale, the system offers only those transportation alternatives that are appropriate for such long-haul trips. For example, if a user query requests possible ways to travel from Rome to London, the system would use large-sized cells to yield paths connecting all known airports serving the two capitals. If the user asks for additional transportation suggestions—for instance, how to get to the airport from a specific district in Rome or from Heathrow Airport to a specific London suburb, the system must consider grids of varying size in sequence. In other words, EasyTransport addresses the case of restricting the system's operational scale to only a certain fraction of a previously indicated wide search area, requesting more detailed answers for just that part of it. By allowing this cell size diversity throughout a working area, we can efficiently facilitate the system's operation in large-scale geographic regions.

The system implements this scalability feature by distinguishing grids with different-sized cells and triggering certain actions when switching among them. This essentially comes down to initializing the algorithm's variables with values corresponding to the newly adopted grid while preserving the outcomes of previously considered grids. The combination of all the grids' path answers will eventually produce the final outcome.

Another important issue is automatic grid selection. We implemented this mechanism by defining in advance the specific grids and respective cell sizes that correspond to all possible users' transition area selections; taken together, these specifications reflect the focus scale of the algorithm's operation. The automatic grid selection also helps render the results so that they are displayed effectively on the user screen.

System utilities and user interface

To increase the interface's usability and thus its acceptance by users, we built it so that it can display multiple pieces of information. After logging on, the user selects one of the two operation modes. The system can easily switch back and forth between Navigation and Transportation by enabling or disallowing the display of the available transportation means along a selected transition route.

Several other services enhance the system's overall functionality and intelligence. When the Navigation mode is on, the following features are enabled:

- The system can contribute one or more intermediate points for a transition, restricting the set of answers to the subset of paths that pass through the whole sequence of input points. Intermediate points are considered in the order they are indicated, and the engine operates in pair-

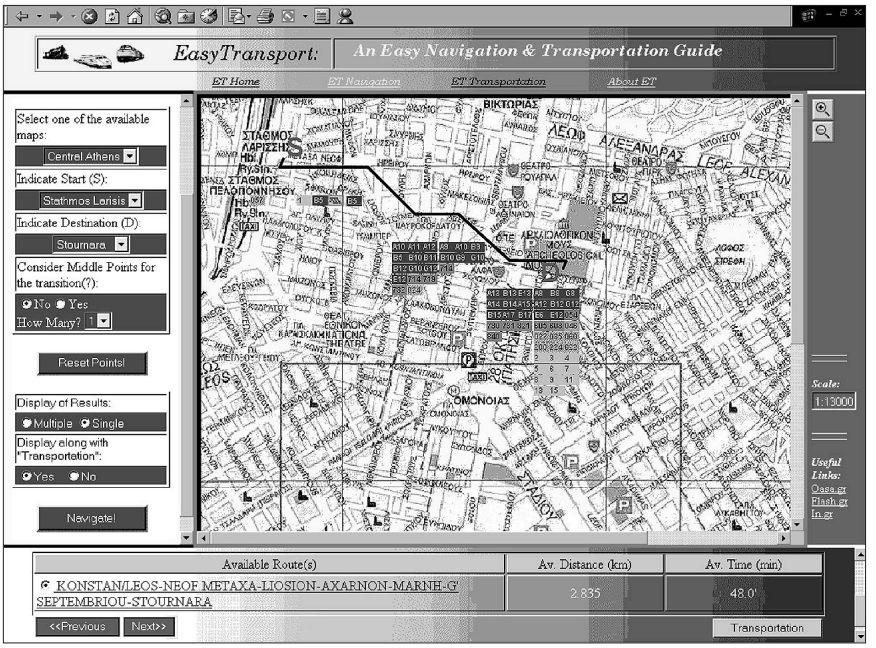

Figure 8. EasyTransport's frame-based interface, featuring delivery of a multimodal answer to a transition query.

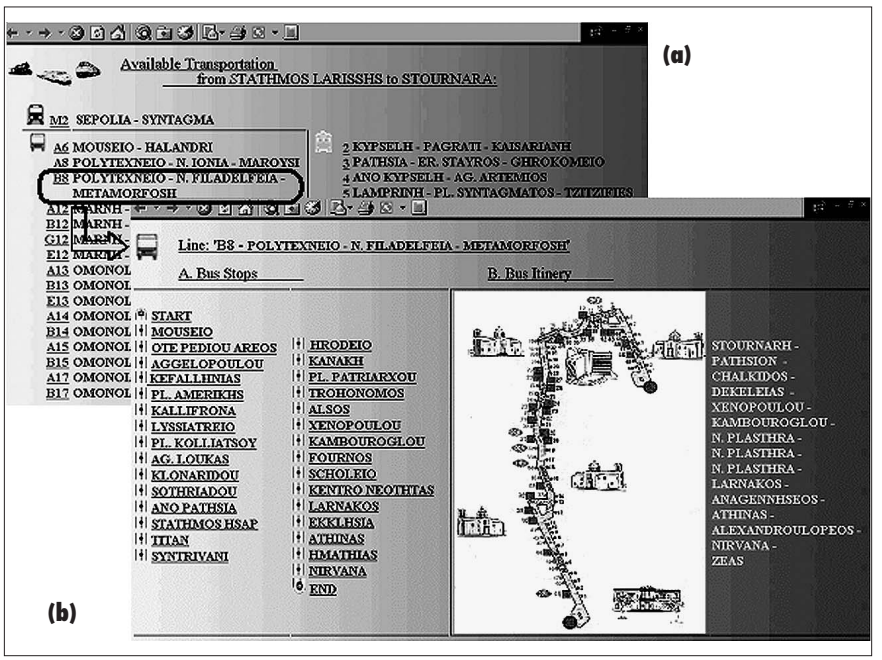

Figure 9. Additional information regarding (a) multimodal transportation for a selected route and (b) a specific means of transportation (bus stops and itinerary).

wise fashion regarding the entire sequence of input points.

- For each suggested route, the system computes and displays a distance approximation. This is only with regard to the real-world settings of the corresponding areas, because the system relies on the assumption of a uniform distance between two cells and normally allows no variations as to the length of paths.
- The system characterizes routes as *hot* (rendered in red onscreen) with regard to typically observed and recorded traffic conditions at certain times of the day or year. Other equivalent alternatives with comparably less road traffic can then supersede these routes.

When the Transportation mode is on, the enabled features include the ability to

- estimate the time needed to traverse the transition area based on the traffic density recorded by the system and the availability and traveling features of the various means of transportation for a selected route;
- present additional transportation information such as transportation authorities, itineraries, departure schedules, useful telephone numbers, landmarks, and nearby resources; and
- suggest the "best" route for a transition, because it either requires the fewest transfers or uses transportation means that are always preferable, such as a metro rail system.

Regardless of the system's operation mode, users may request that EasyTransport make its answers available in batch fashion. Thus, users can view the generated routes one at a time or in groups—for instance, in tens—easily navigating through the routes and searching for the most favorable one.

Figure 8 depicts a screen shot of the system after a user has selected the Transportation mode, the single-view display mode, and the option to indicate no intermediate (Middle) points for the transition. This follows the users' Navigation query and the system's generation of alternative path sets, shown in Figure 5. The system identifies each suggested route uniquely by referring to the locations it entails along with overall distance and required-time estimations. When the user selects one of the routes (Path 7 in Figure 5), the system renders the available transportation for each section (conceptual cell) comprising it. Such sections are transparent to users but distinct in the system for its operation.

The user can further retrieve information regarding all available transportation means for the selected route, as presented by the screenshot in Figure 9a, while Figure 9b depicts an information display pertinent to the schedule and anticipated stops of a specific bus selection.

Requested shortest-route answers in conjunction with transportation alternatives allow EasyTransport to propose the best choices feasible for a trip. It's finally up to users to plot their preferred mobility scenario

once they examine all system-generated route suggestions along with the available means of transportation.

Unlike most contemporary transportation systems, EasyTransport's core objective is to give travelers multiple transportation options along a route. Having investigated EasyTransport's operation within the boundaries of an urban area and considering all available public transportation, the answers we generated met our expectations in terms of response time, accuracy, and multiplicity of the system's outlined trajectories and transportation alternatives. In the future, we plan to deploy the system over multiple geographic areas exhibiting various degrees of transportation availability. We will incorporate real-time transportation features regarding time factors and traffic situations, and we'll develop interfaces for mobile-computing units or phones attached to the system. ▪

Acknowledgments

We are grateful to the reviewers for their detailed comments. Alex Delis was partially supported by the US National Science Foundation grant IRI-9733642. A preliminary version of this work appeared elsewhere.[6]

References

1. K.Y. Lam et al., "RETINA: A REal-time TraffIc Navigation System," *Proc. 2001 ACM SIGMOD Conf.,* ACM Press, 2001, p. 615.
2. T.H. Cormen et al., *Introduction to Algorithms*, 2nd ed., McGraw-Hill, 2001.
3. P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Systems Science and Cybernetics* (SSC-4), vol. 4, no. 2, 1968, pp. 100–107.
4. R. Dechter and J. Pearl, "Generalized Best-First Search Strategies and the Optimality of A*," *J. ACM*, vol. 32, no. 3, 1985, pp. 505–536.
5. R.E. Korf, M. Reid, and S. Edelkamp, "Time Complexity of Iterative-Deepening-A*," *Artificial Intelligence*, vol. 129, 2001, pp. 199–218.
6. M. Fragouli and A. Delis, "EasyTransport: An Effective Navigation and Transportation Guide for Wide Geographic Areas," *Proc. 14th IEEE Int'l Conf. Tools with Artificial Intelligence* (ICTAI 2002), IEEE CS Press, 2002, pp. 107–113.

The Authors

Maria Fragouli is an IT professional. Her research interests include Web-accessible information systems, Internet applications, and algorithms for intelligent Web navigation. She received her MS in computer science from the University of Athens. Contact her at the Dept. of Informatics and Telecommunications, Univ. of Athens, GR 15771, Athens, Greece; mfrag@di.uoa.gr.

Alex Delis is an associate professor with the faculty of Informatics at the University of Athens. His research interests are in distributed computing, networked databases, and intelligent systems. He received his PhD in computer science from the University of Maryland, College Park, and is a member of the IEEE Computer Society and ACM. Contact him at the Dept. of Informatics and Telecommunications, Univ. of Athens, GR 15771, Athens, Greece; ad@di.uoa.gr.