Adaptive Routing for Road Traffic

John Fawcett and Peter Robinson
University of Cambridge

An integrated system uses road congestion information to guide routing. GPS tracks vehicles, and the GSM short message service maintains communications with a central planning service.

Many emerging information appliances assist motorists as they travel on today's congested roads:

- Route planners give detailed instructions on the sequence of roads to follow to reach a particular destination. The information may be presented via a display mounted on the dashboard or as audio using the speakers attached to the vehicle's entertainment system.
- Roadside monitors identify congested roads and transmit reports to wireless receivers in vehicles. Cars equipped with computers can report their own speed to improve the quality of this information. The central system can broadcast all the information to provide users with a global perspective.
- The Global Positioning System (GPS) can be used to make vehicles aware of their current location.
- Mobile phones enable communication while traveling.

However, these services do not link to each other, and using them while driving can be dangerously distracting.

This article reports on an integrated system that uses congestion information to guide routing, both in advance and while in transit. It offers two novel features:

- Historic information about congestion is collected and retained for use when planning routes.
- GPS tracks vehicles while they undertake journeys, and the Global System for Mobile Communications (GSM) Short Message Service (SMS) maintains communications between a moving vehicle and a central planning service to suggest revised routes avoiding congestion.

What is congestion?

We need a useful definition of congestion. What offends drivers?

There are many possible answers, such as the number of vehicles per mile on a road divided by the number the road was designed to bear, or the speed of traffic compared to the speed limit for that class of road. Further options include the length of queues of traffic traveling at similar speeds and measurements of traffic density compared to a predetermined set of values. In practice, most drivers want to know how much longer a journey is likely to take under the prevailing conditions, compared with traveling on an empty road.

We based the provision of routes discussed in this article on minimizing the time taken to complete a journey. Consequently, the definition of congestion used here is the ratio of the current speed of traffic to the limit for that class of road. We can use this ratio, combined with the lengths of the relevant road sections, to calculate delays.

Available technologies

Several new technologies that have recently become generally available can help solve the problem.

Computer programs to calculate routes for motorists have been available for several years. Microsoft AutoRoute (http://www.microsoft.com/AutoRoute/) comes in versions with route data for both Great Britain and Europe, and Vicinity Corporation's MapBlast (http://www.mapblast.com/) offers a similar online service in the United States.

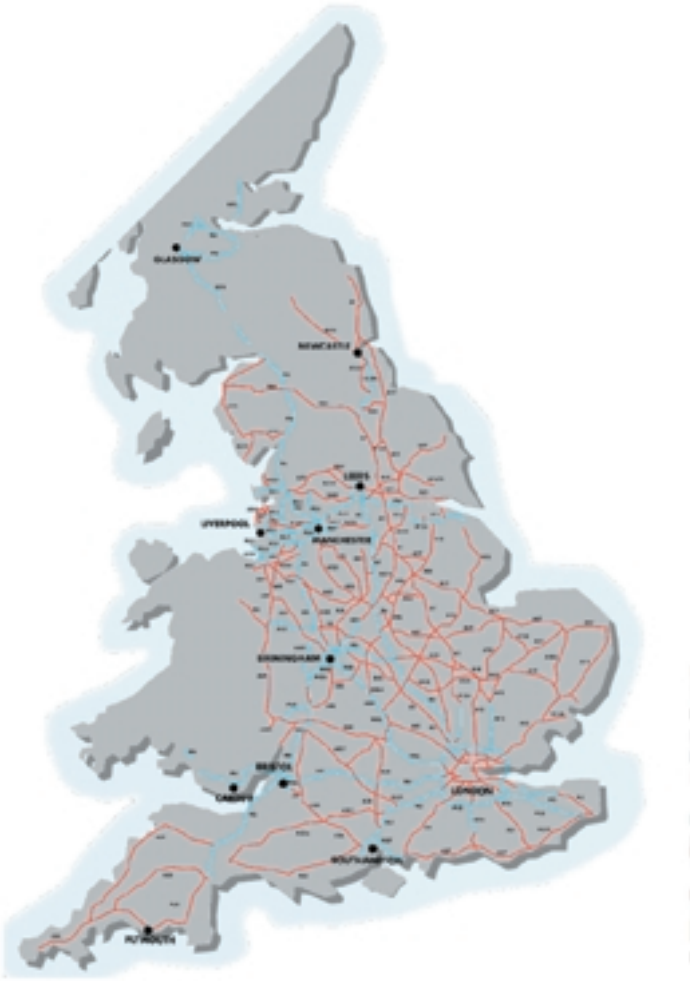
However, both programs assume the capacity of roads remains static. The algorithms calculate a vehicle's speed solely on the basis of the class of road on which it is traveling (one speed for motorways or freeways, another for A-roads or main roads, and so on). No attention is paid to variations in the traffic density with time.

A possible solution relies on the data offered by the Trafficmaster system. According to the Highways Agency, Department of the Environment, Transport, and the Regions (http://www.highways.gov.uk/), there are 10,458 km of trunk roads in the UK. Trafficmaster

0272-1716/00/$10.00 © 2000 IEEE

(http://www.trafficmaster.co.uk/) has installed sensors every few kilometers that measure the speed of vehicles moving on each stretch of road (see Figure 1). Trafficmaster uses two techniques:

- On major roads with separate carriageways, sensors mounted on bridges and using infrared transceivers capture the speed of individual cars.
- On smaller roads, cameras mounted on roadside poles identify the central four characters in vehicles' registration numbers and timestamp the sightings. The results from adjacent sensors when correlated give the vehicle's average speed while traveling between them.



1 Trafficmaster sensor coverage in the UK.

The information returns over a very high frequency (VHF) packet radio system to a control center for rebroadcast to users.

Trafficmaster offers two main types of display. Subscribers can see maps of any part of the road network with annotations indicating areas of congestion. An alternative free service gives warnings of congestion within about 20 km of the receiver, together with an indication of the affected directions of travel. The first service indicates the speed of traffic on roads experiencing congestion; the second simply reports the duration of the anticipated delay.

The information is also available to anyone equipped with a cellular telephone. When the cell phone user calls the appropriate number, information from the cellular network determines the caller's position. Then the Trafficmaster system reports the current state of traffic on nearby roads using synthesized speech, including both the speed and anticipated delay. The latter is surprisingly accurate.

Two further technologies also prove relevant.

The GPS[1] uses a constellation of satellites in low earth orbit (LEO) to provide a positioning system with accuracy of a few tens of meters using relatively inexpensive ground receivers. Auxiliary services such as differential correction and dead reckoning improve the accuracy to a few centimeters.

GSM-SMS provides reliable delivery of text messages up to 160 characters in length between mobile phones. This has been packaged to provide a convenient way for mobile workers to interact with computers at their bases.[2]

Associated work

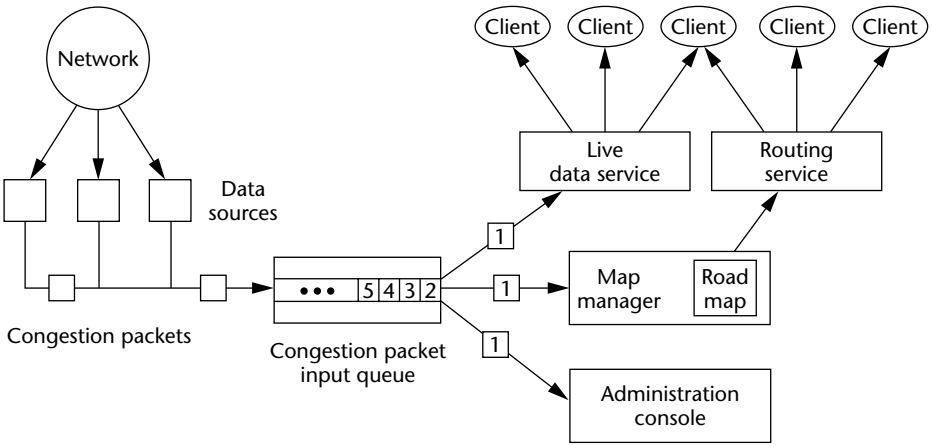
The idea of using traffic monitoring and computer support to alleviate the problems of congestion is not new. Several significant results have been achieved.

The Royal Automobile Club (RAC, http://www.rac.co.uk/) offers a route-planning service that takes account of the currently congested roads as reported by the Trafficmaster system. The service uses the current congestion information at the time of planning the route. It does not account for the fact that conditions may differ when the journey actually starts or, indeed, may change during the course of a journey.

Similar monitoring systems are appearing elsewhere in Europe and in the United States. The Calendar+ system developed by Schaffer and Alway at the University of Washington (http://www.cs.washington.edu/) combines routes from MapBlast with congestion information from the Washington State Department of Transportation (Puget Sound Traffic Conditions, http://www.wsdot.wa.gov/PugetSoundTraffic/) online reports to advise travelers of likely delays. However, this neither uses historical data to predict delays nor suggests alternative routes to avoid them.

Cameron et al.[3] described a simulator that models traffic flows through urban areas. This uses a massively parallel computer to simulate the behavior of a large number of vehicles in real time. It is intended for use in designing road networks rather than optimizing their use after construction.

Bar-Noy and Schieber[4] discussed the problem of



2 Overall structure of the system.

planning a route through a road network susceptible to blockages. Their mobile agent only becomes aware of blockages slightly before encountering them, so routes have to be sufficiently intelligently planned to incorporate flexibility for bypassing blockages as they arise. The approach presented in this article takes a broader view of the network and so can plan more radical alternative routes.

Awerbuch et al.[5] discussed possible trade-offs between the size of the data held at each node in a graph and the time taken to perform a routing calculation. This subtlety isn't really necessary when traversing graphs with the complexity of a road network.

Mitchell and Papadimitriou[6] described the problem of finding a shortest path through a plane divided into sectors where different speeds can be achieved. Their algorithm would support the calculation of routes for off-road driving, but that's a different problem.

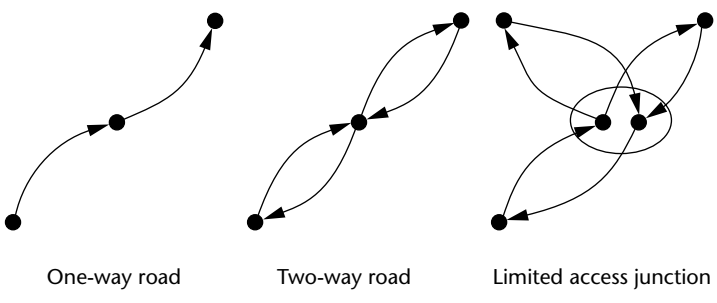
Aims

The project described in this article investigated possible ways to integrate congestion information from the Trafficmaster system with a route planner in such a way that the recommended route would reflect the congestion anticipated at the (future) time when the journey would be undertaken. Moreover, the traveler should be notified if congestion at the time of travel makes an alternative route preferable.

System structure

The overall system consists of several modules, as shown in Figure 2:

- The system collects congestion information from a variety of different network services. Converters for any electronic data source can be written independently and the software can be asked to dynamically add (or remove) data sources from the system without restarting.
- These data are collated into a single queue and copies distributed to registered data recipients, including, in particular, a map manager that accumulates historic information and a live data service that informs clients of unanticipated congestion.
- The map manager annotates a detailed road map with congestion information for different times of the day and days of the week. The optimum time granularity—determined to be 15 minutes—adequately captures trends in traffic congestion. Quadratic interpolation used between 15-minute aggregated samples estimates the instantaneous congestion at intermediate times.
- The routing service uses the connectivity and capacity information in the map manager to find an optimal route between a pair of points for a given time of departure, in terms of minimizing the time taken to make a journey. The optimization criterion could easily be changed to minimizing fuel consumption, emissions, or a trade-off among several.
- The live data service monitors the current positions of clients and current congestion along their anticipated routes, and advises them of revised journey times and revised routes to avoid congestion.
- An administration console allows an operator to monitor and control the system, for example, to introduce new data sources dynamically.



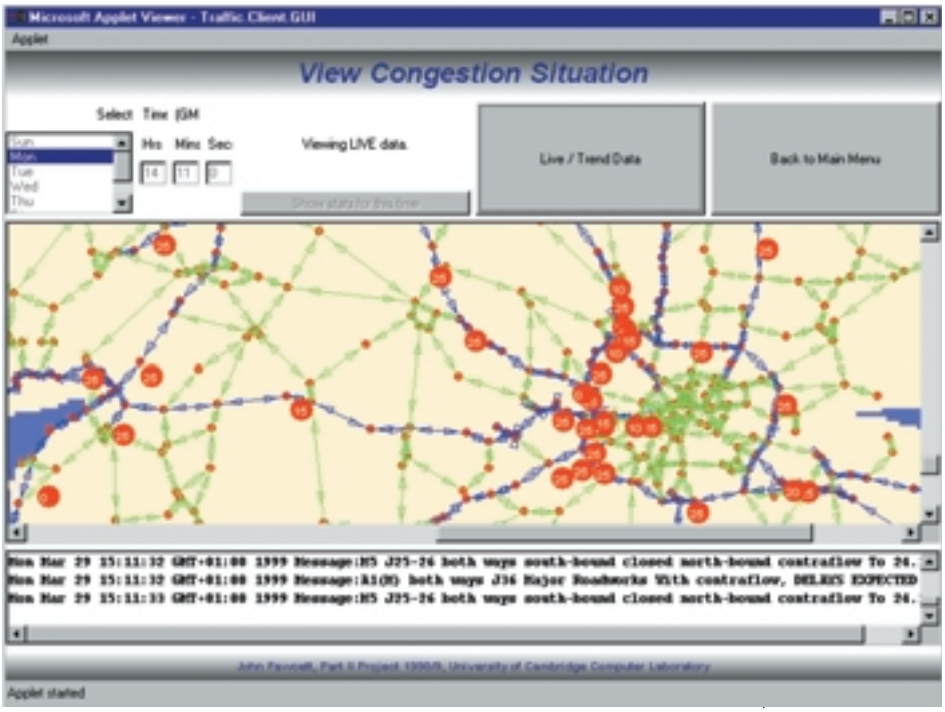
3 Representing different types of road and junction.

Data collection

Trafficmaster made the raw sensor data from its entire network available to the AT&T Research Laboratory in Cambridge using a special VHF radio receiver. This information is then distributed to nominated users over standard transport control protocol/Internet protocol (TCP/IP) socket connections. Reports are delivered as lines of text giving a time stamp, the Ordnance

Survey grid coordinates (OSGB) of the sensor, the name of the sensor's location, the direction of traffic flow, and the traffic speed, together with some other housekeeping details. These are interspersed with text messages reporting road restrictions for repairs or because of accidents. This collection of information was massaged into a standard form by a Trafficmaster data source filter and the resulting packets inserted into the processing queue.

Adaptors for other data sources could be written similarly. For example, the British Broadcasting Corporation (BBC) broadcasts traffic information in data subcarriers of its radio and television services and through its Web pages, and these data could be fed into the system as well.



4 Live congestion information.

Map manager

We represented the road network using a directed graph with each edge depicting a one-way road and each node corresponding to a junction. Two-way roads were represented as a pair of edges, one in each direction. This model permits easy modeling of one-way roads and limited access junctions (see Figure 3). We didn't model the exact routes of roads between junctions, assuming them to be straight lines. Given the frequency of junctions, assuming that roads run straight between them doesn't introduce significant error into the calculations.

Obviously the level of congestion on roads changes with the time of day. This leads to a specific requirement regarding the algorithm used to calculate routes. Weights (or costs) attached to the edges in the directed graph represent the speed of traffic on the corresponding roads (in the direction of the edge) and have to vary with the time of day.

This raises an important question: What temporal granularity should we use to record congestion information? If the timing is too precise, the size of the stored data will be too large, and if the timing is too coarse, the predicted route times will be inaccurate. After collecting congestion reports for a couple of months, we decided to record road capacities for 15-minute intervals for 24 hours each day and 7 days each week. Weekends differ sufficiently from weekdays to make this necessary, but seasonal variations through the year are generally not worth recording. Consequently, every edge in the road map carries 572 bytes of speed information.

The system revises the speed information for an edge at the current time-of-week every time it receives a new congestion report by calculating a running average:

$$revised = \alpha \times old + (1 - \alpha) \times new$$

This effectively calculates a geometrically weighted average of all the past values. Empirical evidence suggested a value of 0.5 for α to give a good balance between rapid response to new reports and long-term stability. (Note that duplicate reports are suppressed, since they would bias the accumulated statistics. Also, reports are generated even in the absence of congestion; otherwise, the expected speed on roads would decrease monotonically.)

The entire map database including revised congestion averages is check-pointed to disk at regular intervals and archives are kept.

Routing service

The routing service takes any two points in the map and an intended start time, and calculates the quickest route between the two points given the anticipated congestion from historic information. The resulting itinerary lists roads, junctions, distances, and estimated times, and is optimal with respect to minimizing the time taken.

Live data service

The live data service serves two functions. First, it provides the data needed by graphical user interfaces to display a map of the road system overlaid with current congestion information (see Figure 4).

The red circles indicate the location of traffic reports, and the white numbers are the traffic speeds in miles per hour at these sites. The text area at the bottom logs human-readable reports delivered by data sources such as the Trafficmaster system.

Secondly, information from the live data service is used to advise clients already on the road of alterations to their itineraries recommended in light of recently received congestion information.

The final route produced by Lee's algorithm is optimal in the sense that starting at a time specified by the user and taking into account the time when each section of road would be driven, the output route offered the quickest path to the destination.

Routing algorithms

Routing is well studied in computer science,[7] but some of the standard algorithms turn out to be lacking or inappropriate for this application in one or more respects.

Breadth-first and priority-first searches

Breadth-first search and priority-first search would work and might be expected to find an optimal route, but they require substantial computational effort because they explore inappropriate parts of the graph in addition to the relevant parts. It's not clear what heuristics should be used with priority-first search to find good routes in a reasonable time. Some of the early commercial route planners were notoriously ineffective in this respect and would, for example, lead the driver through miles of twisting country lanes rather than heading a few miles directly away from the destination to reach a motorway junction.

Dijkstra's algorithm

Dijkstra's algorithm guarantees to find the best route (with respect to minimizing the total cost of edges in the route) between two nodes, if a route actually exists. The algorithm considers edges with the smallest cost first, but this means it cannot know how much time will have elapsed since the start of the journey when any particular link is traversed. This application cannot determine the weight of an edge until it knows the time of travel. It can evaluate the weights only at an estimated time, and the start time of the journey provides as good an approximation as any.

A better estimation of the time when an edge will be traversed can be derived by interpolation from the straight-line distance between the edge and the start and an estimate of the total journey time. However, this improves matters only slightly. The algorithm guarantees to terminate in all circumstances. The asymptotic cost is $O(v^2)$ for a graph with v vertices.

Lee's algorithm

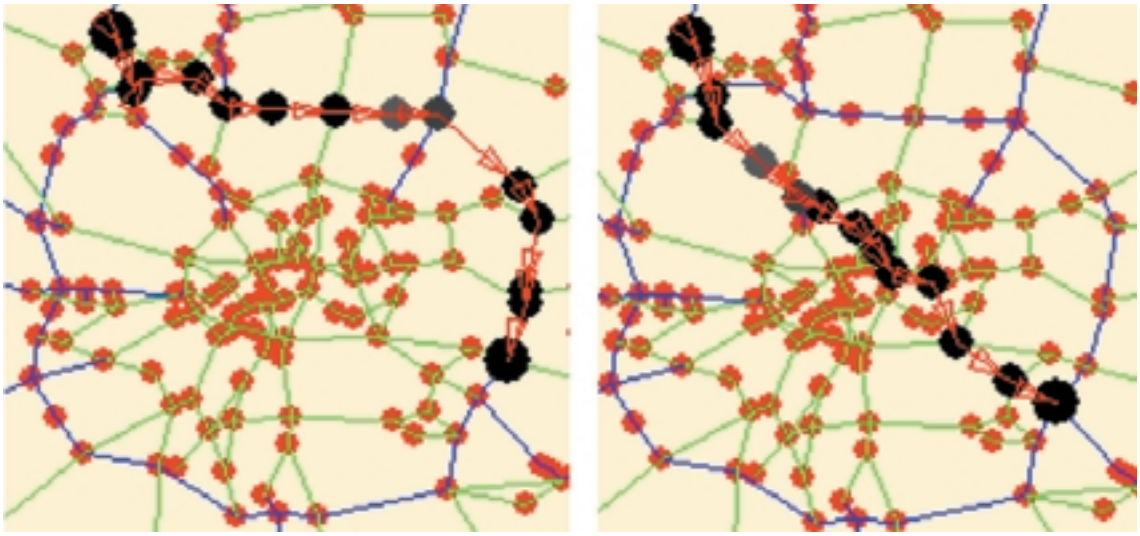
Originally used for track routing on printed circuits and integrated circuits, Lee's algorithm[8] suits changing road capacity more naturally than Dijkstra's. This algorithm also finds the best route with respect to optimizing some metric, as long as such a route exists. The method simulates an inkblot spreading out over a piece of paper, centered on the start point. The area covered represents vertices already explored. The real time is known upon traversal of each edge, so the algorithm can calculate the correct value of the delay (because the delays depend on time and day). When the destination vertex is reached, the algorithm traces the route back to the start.

When used for track routing in very large scale integrated (VLSI) circuit design applications, a Lee router explores the surfaces of a circuit board by making equal-sized steps in all permissible directions, beginning with the start node. The search takes a breadth-first approach and is initialized by putting the start node in a queue of nodes to explore. Items in the queue are tagged with the distance (number of steps) already spent in reaching that point.

At each computation step the router examines the queue. An empty queue means no route exists. Otherwise, the algorithm extracts and examines the head of the queue. If the head of the queue is the required destination node, then a route has been found. Using a simple induction can prove that the path taken by the router is the quickest route from the start. If the head of the queue is not the destination node, then the router considers equal-sized steps in all (valid) directions to reach further nodes. These nodes move to the end of the queue, with their distance-so-far indicators incremented by one. However, a node already in the queue or already explored is not added again, since a shorter route to that location has already been found. If the quickest path to the destination does indeed go through that location, there can be no need to go via the new intermediate point, as that route would take at least as long as the previously discovered one.

In the road-traffic routing application, the step sizes are unequal because roads have differing lengths and levels of congestion, which change the time taken to traverse them. The distance-so-far attached to items in the queue equals the total time spent in reaching that location from the start. As the router explores nodes and finds them not to be the destination node, it considers new locations. It marks these new locations as reachable in a time equal to the time taken to reach the explored node plus the time taken to drive along the joining road at the relevant time-of-day (computed from the user-supplied start date and time of the journey and the time-so-far into the route). The newly reached nodes must be inserted into the queue in increasing order of arrival time. For this reason, the router uses a time-ordered list to store the list of vertices to explore, rather than a simple queue. Duplication in the time-ordered list needs careful attention: If a node being added is already in the queue, then only the entry with the lower time is kept. This may involve removing an existing queue item. Otherwise, the algorithm proceeds as described above. It also guarantees to find the shortest route (in terms of time), as long as one exists.

Adaptive traffic routing needs a further modification because the cost of an edge may change between the exploration and trace-back phases, which can cause the tracer to find no routes with the expected cost. Merging



5 Recommended routes for the same journey at similar times on different days.

the trace-back and exploration phases fixes this—the explorer builds a tree as it runs, and when it finishes, the optimum route is the unique path through the tree from the destination leaf to the root (the start vertex). A tree is built rather than a general graph because the algorithm does not explore cycles.

This algorithm's cost is difficult to express. It is best described as polynomial in the number of edges. The worst case only occurs when there is no route between the start and destination vertices, and so is unlikely to occur in practical road maps, especially if ferry crossings are included in the map. The average cost is less than that of Dijkstra's algorithm because it stops considering routes when they are longer than the best route so far from the source to the destination. Lee's algorithm can be seen as taking a local view, whereas Dijkstra's takes a global view, considering all edges at once, which wastes time.

Soukup's algorithm

Based on Lee's algorithm, Soukup's routing algorithm uses heuristics to limit the search and prioritize the order of vertex exploration. These modifications mean that the algorithm cannot guarantee to find the optimum route or even to find a route if one exists. We expected that the running time of a Lee router on a graph of the sparseness of a road map would be only a fraction of a second, so optimizations motivated by limiting the search space to improve performance don't apply.

Bellman-Ford routing algorithm

Bellman-Ford[7] is a distance-vector routing scheme used by the early Arpanet routers and implementations of Routing Information Protocol (RIP, for DECnet and Novell IPX packet routing). Each node maintains a table that lists the quickest route to every other node. At regular intervals, the nodes broadcast their tables to their neighbors, which update their tables accordingly and propagate the information to their neighbors, and so on. Care has to be taken with cycles. Calculating a route between two nodes is an $O(\log n)$ operation for a graph with n nodes—it only involves a look-up in the table held by the start node. Using an appropriate data structure (for example, a heap, red-black tree, B-tree, or a hash table), this can be very efficient.

The disadvantage for adaptive traffic routing is that the amount of storage required for all the tables is vast and increases as the square of the number of nodes in the graph. While routes can be provided with very little computation, maintaining the tables carries a large expense. This arduous housekeeping must be done even when no users are requesting routes.

Fast Lee routing

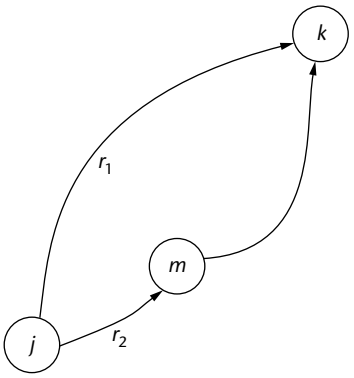
On balance, we used Lee's algorithm because it gives satisfactory results in a reasonable time without adding undue complexity. The final route produced is optimal in the sense that starting at a time specified by the user and taking into account the time when each section of road would be driven, the output route offered the quickest path to the destination.

Performance improves by representing the edges in Lee's inkblot as events in a time-ordered list, in a similar way to how an event-based simulator models signal changes propagating through a piece of hardware. This allows the algorithm to develop the perimeter of the inkblot extremely quickly, avoiding parts of the graph already passed or still some distance away. Finding a route from one end of the United Kingdom to the other takes an imperceptible fraction of a second on a modest PC.

The images in Figure 5 show the routes offered by the system to cross London at similar times on Sunday and Monday mornings. On Sunday, the program recommends using the M25 orbital motorway, but given the known congestion of the M25 on a Monday with commuters, it suggests a direct route through the center of London instead. Actually, this scenario is unrealistic, reflecting the absence of congestion sensors in central London, but it nevertheless demonstrates the principle.

The sitting-still problem

A problem still arises from the quantization of the changing speeds on individual edges in the graph. This



6 The sitting-still problem.

will affect any algorithm that attempts to find an optimal route.

Suppose a route being explored has reached junction j at time τ, from which there are two routes to some vertex k (possibly via one or more intermediates), as shown in Figure 6. If both routes are heavily congested at time τ, but r_1 slightly less so than r_2, then r_1 will be selected. However, in a few minutes the congestion may have cleared considerably on r_2, and it may be that waiting those few minutes and then taking r_2 results in a quicker route overall. Of course, it is never sensible to "sit still" at junction j—the driver should take the r_2 branch immediately, as this will always give the same or better time to reach k.

The solution adopted is to reduce the granularity of speed data samples in the edges by interpolating between the values for consecutive time slots. When estimating the time it would take to drive from one road junction to a neighboring junction at a specific time of the week, the distance between the junctions is divided by the traffic speed at that time. Quadratic interpolation used within each time slot improves the estimates of speed.

Real-time routing

The second goal of this project was to calculate revised routes in the light of actual congestion information received at the time of travel. This uses two further pieces of technology.

The Navstar GPS and the GSM-SMS extend the functionality of the routing software. We envisage that soon cars, vans, and trucks will be fitted with GPS receivers and GSM telephones (or some similar wireless communications technology). With a Car-PC (Kontron Embedded Computers, http://www.kontron.com) or similar device installed, lightweight code can be executed on the move, and the full potential of the GPS and GSM-SMS hardware can be realized.

At regular intervals (in either time or distance covered) the equipment in the car determines its location using the GPS receiver and sends the coordinates, along with an identifier, by GSM-SMS to an SMS server back at base. The identifier is looked up in a table of previously recommended routes to determine where this driver is heading. The routing service is then called to calculate a revised route from the current location of the vehicle to its known destination. This route is compared with that given to the driver before the journey started.

If, in light of accidents for example, the best route has changed, the server returns the revised route to the driver by GSM-SMS. Alternatively, the best route may not change, but the road speeds might, making the estimated time of arrival significantly different. In this case, the server advises the driver to follow current directions but that the estimated time of arrival has increased (or decreased) by the appropriate amount. The server also describes any likely congestion ahead.

The additional service is implemented through the SMS server at the AT&T Research Laboratory in Cambridge.[2] This effectively presents a standard command-line interface on the computer running the routing services through GSM-SMS. Commands are sent as short messages and responses returned similarly.

In this case, the command carries arguments specifying the driver's current position and the identity of the route:

```
GSMRelay 544606 262498
  robinson_route_42
```

The response indicates anticipated congestion:

```
20MpH on M25 J27-J28
20MpH on M1 J9-J8
25MpH on M1 J12-J11
15MpH on M6 J7-J6
20MpH on M6 J10-J9
20MpH on M6 J10a-J10
```

The advisory function checks each active route for new congestion and generates text when appropriate, relaying it to the driver as soon as possible. To simplify the presentation of information to the user, and to improve the user interface, a combined functionality information appliance could be created by passing the congestion advice as speech, either generated on the server and relayed as a voice message to the mobile phone, or forwarded as a command directly to a speech synthesizer in the car.

The routing algorithm is sufficiently fast that a single server can support several hundred clients whose positions are monitored every couple of minutes. A more elaborate scheme would use congestion reports as indices back into currently active journeys and only reconsider directly affected users, but this isn't necessary in practice. As the computing power available in a standard car increases, the whole system could be distributed and recalculations performed locally in the vehicle.

Conclusions

Whilst not a commercial product, the software has demonstrated that adaptive computerized route generation could help drivers cope with the growing congestion on UK roads. Although it doesn't tackle the causes of congestion, the software could prove useful in minimizing the delays resulting from queues of traffic.

As the network of sensors expands, the routing advice will improve with little extra cost. The extra data should

enhance accuracy through better modeling of the real world. At the same time the cost of running Lee's algorithm shouldn't increase significantly, so the software should continue to calculate routes quickly.

Commercial organizations conceivably might use software similar to that described here. Many companies already have database facilities available to help their drivers find delivery addresses and to help the management team assess performance of their employees. Adding route planning facilities to the database could increase productivity and profits for the company by reducing wasted time and money when drivers sit in traffic. This would also help to cut vehicle pollution, which plagues many inner-city areas.

The software is implemented in Java. This gives it the ability to run on Web browsers in homes and offices, and also in hotel rooms and boardrooms via WebTVs. The system shows how computing, communications, and location information combined in an information appliance can provide valuable facilities in a genuinely ubiquitous computing environment. ■

Acknowledgments

We are very grateful to Trafficmaster UK and AT&T Research Laboratories, Cambridge, UK for making the congestion information available.

References

1. I. Getting, "The Global Positioning System," *IEEE Spectrum*, Vol. 30, No. 12, Dec. 1993, pp. 36-38, 43-47.
2. F.M. Stajano and A.H. Jones, "The Thinnest of Clients: Controlling it all via Cellphone," *ACM Mobile Computing and Communications Review*, Vol. 2, No. 4, Oct. 1998, pp. 46-53.
3. G. Cameron, B.J.N. Wylie, and D. McArthur, "Paramics—Moving Vehicles on the Connection Machine," *Conf. on High Performance Networking and Computing*, Washington, Nov. 1994, pp. 14-18.
4. A. Bar-Noy and B. Schieber, "The Canadian Traveler Problem," *Symp. on Discrete Algorithms*, ACM Press, New York, Jan. 1991, p. 261.
5. B. Awerbuch et al., "Compact Distributed Data Structures for Adaptive Routing," *ACM Symp. on the Theory of Computing*, ACM Press, New York, May 1989, pp. 479-489.
6. J.S.B. Mitchell and C.H. Papadimitriou, "The Weighted Region Problem," *J. ACM*, Vol. 38, No. 1, 1991, pp. 18-73.
7. R. Sedgewick, *Algorithms*, 2nd edition, Addison-Wesley, Reading, Mass., 1988.
8. C.Y. Lee, "An Algorithm for Path Connectivity and its Applications," *IRE Trans. on Electronic Computers*, Vol. 10, No. 3, Sept. 1961, pp. 346-365.

John Fawcett *is a PhD student at the Laboratory for Communications Engineering at the University of Cambridge in England and a member of Churchill College. He is part of the Sentient Computing group. He previously studied for a degree in the Computer Laboratory at the University of Cambridge in England. His current research focuses on applications for mobile users, includes services that may be delivered to vehicles over wireless communications media, and extends to additional facilities that can be brought about by having large numbers of equipped vehicles.*

Peter Robinson *is a lecturer in the Computer Laboratory at the University of Cambridge in England, where he is part of the Rainbow Group working on computer graphics and interaction. He is also a Fellow, Praelector, and Director of Studies in Computer Science at Gonville & Caius College, where he previously studied for a first degree in mathematics and a PhD in computer science. His research interests are in the general area of applied computer science. The main focus for this is human-computer interaction. He also works on electronic design automation and, in particular, on support for self-timed circuits. He is a Chartered Engineer and a Fellow of the British Computer Society.*

Readers may contact Robinson at the University of Cambridge, Computer Laboratory, New Museums Site, Pembroke St., Cambridge, England CB2 3QG, e-mail pr@cl.cam.ac.uk.