A ROUTING ALGORITHM FOR URBAN ADVISORY SYSTEM

Vălean Honoriu
Automation Department
Technical University of Cluj-Napoca (TUC-N)
Cluj-Napoca, Romania
Honoriu.Valean@aut.utcluj.ro

Leția Tiberiu
Automation Department
Technical University of Cluj-Napoca (TUC-N)
Cluj-Napoca, Romania
Tiberiu.Letia@aut.utcluj.ro

Aştilean Adina
Automation Department
Technical University of Cluj-Napoca (TUC-N)
Cluj-Napoca, Romania
Adina.Astilean@aut.utcluj.ro

Abstract: The routing problem in urban traffic becomes very important due to the fast growth of car number. The Urban Driving Advisory System (UDAS) as a component of a Real-Time Information and Control System (RTICS), assists drivers to follow a good path for arriving to the destination. The main function of the UDAS is to find the best way to travel from one place to another, with respect of various metrics, such as distance, travel time, etc. In the paper, a Dijkstra based routing algorithm is presented. The algorithm uses a penalty function which takes into account the actual traffic on the streets, to increase or decrease the weigth of the street graph links.

Key words: traffic control, communication systems, real-time systems, shortest path algorithms.

INTRODUCTION

Vehicle traffic control becomes important with the increase of the number of participants (i.e. cars). Vehicle traffic control is divided into urban traffic control (Leția et al. 2002a) and road traffic control (Cremer and Zhang 1993). The difficulties in solving these problems are given by the continuous (and significant) variation of the characteristics and parameters of the traffic. These depend on the period of the day and on the days (working or holiday). So, it is difficult to find one model that describes completely the entire system during a long period of time. Instead of this, a set of models that depends on the time can be used.

The continuous modification of the characteristics and parameters of the traffic implies the change of the control algorithms with those that are most appropriate for the current situation.

The activities involved by this research include the design of the information system, the traffic control system and the traffic surveillance system for a medium size town. This implies: the congestion prediction and avoidance, the control of the traffic lights, the best route advice, the dynamic route guidance, the estimation of the arrival times etc.

To get these aims, the following were achieved:

- the models of crossroads and of the traffic system (Leția et al. 2002b);
- the adaptation of the models to continuous system changes;
- the evaluation of the performances of the control algorithms (Hush 2000; Gerken 2000) to be able to choose the best algorithm and
- the best control algorithms according to different situations.

Besides the control of the colour of the traffic lights that determines the system behaviour, route planners give detailed instructions on the sequence of roads to follow to reach a particular destination. The information may be presented via mobile data services and may include:

- advisory related to the best route taking into account the minimum time, minimum distance or a combination criteria of these two;
- the expected arrival time at destination;
- warning about current jamming etc.

THE UDAS STRUCTURE

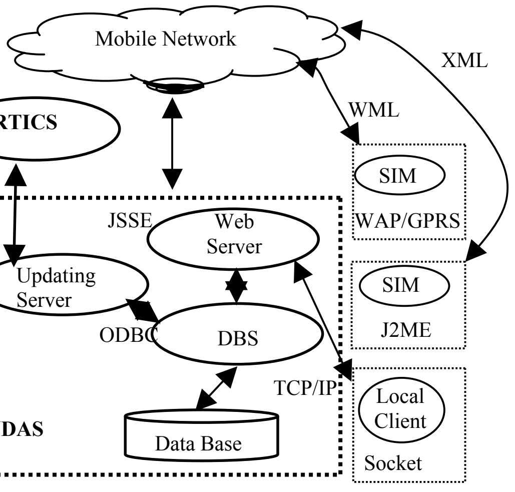


Figure 1. The software architecture of UDAS

The general structure of UDAS can be found in (Leţia et al. 2003).

The software architecture of the UDAS is represented in the Fig. 1. The abbreviations used in the figure have the following meanings:

- RTICS – Real Time Information and Control System;
- UDAS – Urban Advisory System;
- ODBC – Open Database Connectivity;
- DBS – Database Server;
- JSSE – Java Secure Socket Extension;
- SIM – Subscriber Identity Module;
- J2ME – Java 2 Platform Micro Edition;

More details of the UDAS communication structure can be found in (Aştilean et al. 2002; Aştilean et al. 2003). The notions are used according to (Bettstetter 1999;Flickenger 2000; Muratore 2000)

THE SHORTEST PATH PROBLEM

The computation of shortest path over a network has been the target of many research efforts (Dijkstra, 1959; Goldberg and Radzik, 1993;Glover, Klingman and Phillips, 1985; Chabini, 1997). These efforts have resulted in a number of different algorithms and a considerable amount of empirical findings with respect to performance (Chabini 1997; Zhan and Noon 1998). To solve the shortest path problem, there are a lot of requirements that should be taken into account for choosing the better algorithm:

- the optimisation may be performed with respect to various constraints, usually distance, travel time, fuel consumption, etc. ;
- traditional algorithms may impose an unacceptable long computational time when applied to realistic road networks;
- additional constraints may be imposed, because most people prefer to travel on main roads;

Many the suggested solutions to the shortest path problem during the last four decades are based on the Dijkstra algorithm and its variants . However, in a practical context, these algorithms are not necessarily finding a best solution on a road network. If the algorithm minimizes the distance, Dijkstra{'}s algorithm guarantees to find the best route. But, if the requirement is to minimize the travel time, the algorithm fails, because it cannot know how much time will have elapsed since the start of the travel when any particular link is traversed. Another problem of the algorithm is the following: the best solution from the point of view of the distance may be a very poor solution from the point of view of the time and the best solution from the point of view of the time may be a again, a very poor solution from the distance point of view.

Instead of finding the best solution considering the distance or the best solution considering the travel time, the paper proposes a Dijkstra algorithm which finds the best solution on distance and time, that takes into account the actual context in the street network.

THE PROPOSED ALGORITHM

For defining the shortest path problem, the following requirements are made:

- the solution must be provided off-line, because it is accessible via SMS and also, is uncomfortable for the drivers to change the route during the travel;
- the solution must take into account not only the actual street map stored in the database, but also the actual context on the streets: closed roads, congestions, one-way restrictions, bottle-necks, etc.;
- the solution must act as traffic formatter, for avoiding the congestion and discharging
- the control system implemented on the Real-Time Information and Control System (RTICS) (Leţia et al. 2003);

The street network is represented as a graph. Each vertex on the graph is oriented, denoted by *(i,j)*, e.g. the link is starting form node *i* and ending in node *j*. Each node *(i,j)* is label with a label d_{ij}, representing the cost of the link. A two-ways street is represented by two links, *(i,j)* and *(j,i)*, where only in particular cases $d_{ij}=d_{ji}$. A simple crossroad is represented by a graph consisting in 5 nodes and 6 links, and so on (Fig. 2).

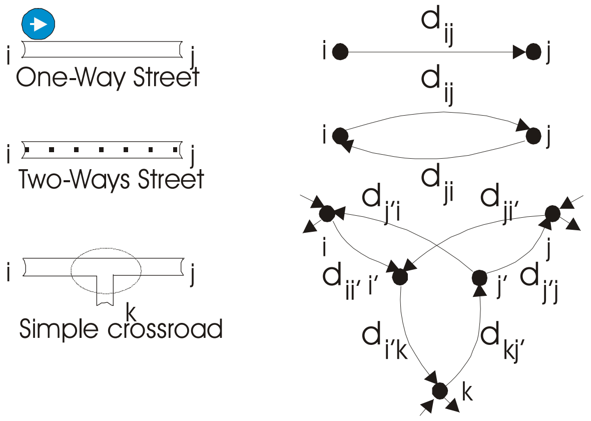


Figure 2. Different graphs related to the streets

The choice to consider one-to-one shortest path computation is motivated by the fact that the most of the travellers have different starting and ending points and in the case of time-dependent networks (networks in which edge weights vary in time) the shortest path tree will be different for each traveller.

The algorithm consists in the following steps:

- build the associated graph from the geographic information stored in the database;
- label the links with the costs $d_{ij}=\delta_{ij}$, where δ_{ij} is the distance (in km.) between two nodes;
- for each link *(j,i)*, compute a penalty function p_{ij} given by

$$p_{ij} = \max(\delta_{ij}) - \frac{\max(\delta_{ij})}{1+e^{-0.2(\upsilon_{ij}-25)}} \qquad (1)$$

  where $max(\delta_{ij})$ is the length of the longest street and v_{ij} is the actual average speed reported by RTICS for the correspondent street, at the moment. The accepted maximum speed on the streets is 50 Km/h. A sygmoid penalty function has been choose because of the smoothness of the shape;
- compute the new label of the links as

$$d_{ij} = \delta_{ij} + p_{ij} \qquad (2)$$

  If on a street the average speed is very low (i.e. a congestion is imminent), the power of the link for the correspondent link will be significantly increased;
- find the shortest path using a variant of Dijkstra's algorithm (Morris 1998):
  - initialise d and pi;
  - set S to empty;
  - while there are still vertices in V-S
    - sort the vertices in V-S according to the current best estimate of their distance from the source;
    - add the closest vertex in V-S, to S;
    - relax all the vertices still in V-S connected to the last vertex added in S;

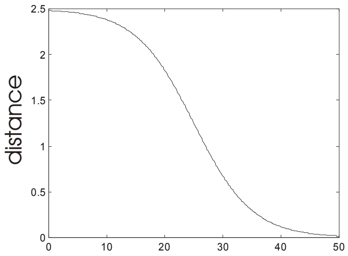


Figure 3. The penalty function shape

In Fig. 3 is represented the penalty function shape for $max(\delta_{ij})$=2.5 Km and a maximum allowed speed of 50 Km/h.

When UDAS receive a request, it opens a new thread for computing the path. The UDAS will execute the following set of instructions:

```
build_graph(Graph g)
 for each link (i,j) in ( g )
    $d_{ij}=\delta_{ij}$;
    compute_penalty_function($p_{ij}$);
    $d_{ij}=d_{ij}+p_{ij}$;
```

First, the UDAS thread build the graph *g=G(E,V,L)*, where *E=Edges(g)* is the set of edges, *V=Vertices(g)* is the set of vertices and *L=Links(g)* is the set of links. Then computes the weights d_{ij} for all the links *(i,j)* from L, taking into account the geographic information stored in database and the computed penalty functions. Now, the shortest path can be computed (Morris 1998):

```
shortest_paths( Graph g, Node s )
   initialise_single_source( g, s )
   S = { 0 };       /* Make S empty */
   Q= Vertices( g ); /* Put the vertices in a PQ */
   while not Empty(Q)
      j= ExtractCheapest( Q );
      AddNode( S, i ); /* Add i to S */
      for each vertex i in Adjacent( j )
         relax( j, i, $d_{jk}$ );

initialise_single_source( Graph g, Node s )
  for each vertex i in Vertices( g )
     g.sd[i] = infinity;
     g.pi[i] = null;
     g.sd[s] = 0;

  relax( Node j, Node i, double $d_{ij}$ )
   if sd[i] > sd[j] + $d_{ij}$ then
     sd[i] = sd[j] + $d_{ij}$;
     pi[i] := j;
```

The used notation have the following meanings:

- S – the set of vertices whose shortest paths from the source have already been determined;
- sd - array of best estimates of shortest path to each vertex;
- pi - an array of predecessors for each vertex;

The *initialise_single_source()*function sets up the graph so that each node has no predecessor (pi[i] = null) and the estimates of the cost (distance) of each node from the source (sd[i]) are infinite, except for the source node itself (sd[s] = 0). The *relax()* function checks whether the current best estimate of the shortest distance to i (sd[i]) can be improved by going through j (*i.e.* by making u the predecessor of i).

CASE STUDY

For testing the routing algorithm, the urban area map presented in Fig. 4 was taken into account. The names of the streets are given by numbers and letters. A_i means the ith Avenue, B_j the jth Boulevard and S_k the kth Street. As shown in (Aştilean et al. 2003), customers send their request and receive the answers using SMS. Because the origin and destination points cannot be localized precisely (as an exactly marked point on the urban map), a request has to be formulated indicated the nearest crossroads for departure (the circle marked point) and for arrival (the square marked point). Consequently, the two end points, are represented as the starting and the ending point on the graph. The entire monitoring and control system acts based on the coded indicatives of the crossroads and streets, the corresponding conversion being performed in each case. To calculate the minimum costs, the RTICS determine the average speeds and periodically transmits the information to the UDAS. The main part of the considered urban area corresponds to a real map.

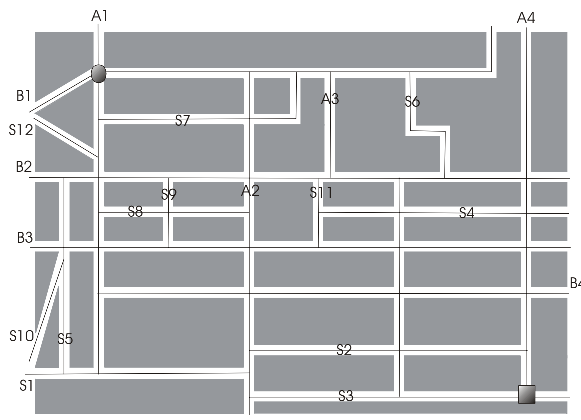
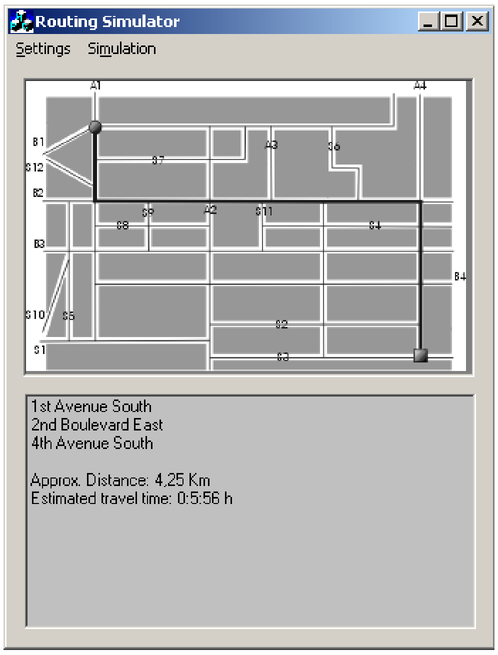


Figure 4. The urban map

The simulation program generates a random number of cars on the streets, with a random behaviour (travel direction, speed). The time for green and red lights on the crossroads is fixed, and crossroads are considered as FIFO queues. The extraction of a car from the queue is done with a constant rate. Two examples of routing are presented (Fig. 5 and Fig. 6), corresponding to the same starting and ending points, but at two different moments.

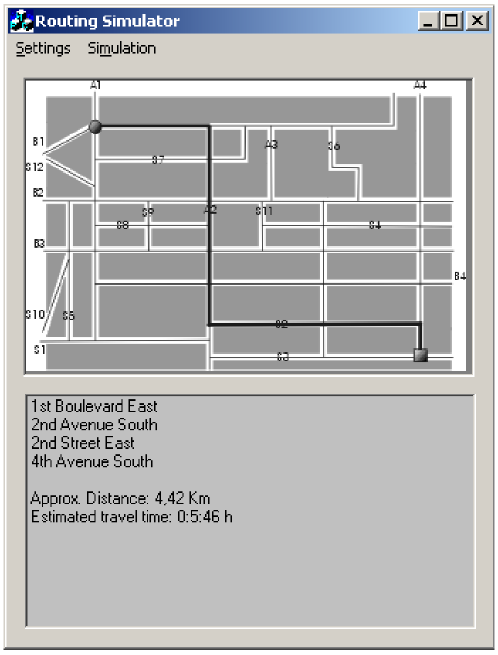


1st Avenue South
2nd Boulevard East
4th Avenue South

Approx. Distance: 4,25 Km
Estimated travel time: 0:5:56 h

Figure 5. Indicated route for simulation 1

As can be seen, the two routes are not identical. In the first case, the travel distance is shorter, but the time is longer, probably due to a smaller average speed. In the second case, the distance is a little longer, but the time is shorter. The algorithm will generate all the time, a suboptimal solution, but the best solution for the actual context on the map.



1st Boulevard East
2nd Avenue South
2nd Street East
4th Avenue South

Approx. Distance: 4,42 Km
Estimated travel time: 0:5:46 h

Figure 6. Indicated route for simulation 1

The average time for finding the best path for the given graph on a P4/1700MHz computer was about 3 ms.

CONCLUSIONS

The paper presents a solution for the problem of urban vehicle routing, which transforms an on-line routing problem in an off-line problem.

The algorithm used in the paper is a Dijkstra algorithm because of the simplicity of implementation. The distances marked on the links of the graph are not constant. They are computed by taking into account not only the length of the correspondent streets, but also the time-restrictions (as penalty functions) due to the high traffic or congestions, via the average speed reported on each street.

The algorithm gives a suboptimal solution for the shortest path problem and acts as primary traffic formatter, for avoiding the congestions.

The penalty function presented in the paper could be applied not only to the Dijkstra algorithm but also to other shortest path algorithms.

REFERENCES

Aştilean, A., Avram, C., Leţia, T.S., Hulea, M., Vălean, H. (2002) Using Mobile Data Services for Urban Driving Advisory Systems. *Mobile Open Society through Wireless Telecommunications Conference, Technology for Mobile Society*, pp. 4, Warsaw, Poland.

Aştilean, A., Leţia, T.S., Avram, C. (2003) Information and communication strategies for Urban Driving Advisory System. *Proc. of 14th International Conference on Control Systems and Computer Science (CSCS-14)*, pp. 523-528, Bucharest, Romania.

Bettstetter, C., Vögel, H. J., Eberspächer, J. (1999). GSM Phase 2+ General Packet Radio Service GPRS: Architecture Protocols, and Air Interface, *IEEE Communications Surveys*.

Chabini, I. (1997). Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Records*.

Cremer, M., Zhang X. (1993). System Architectures for Complex Road Traffic Information and Control Systems. *Proc. of IFAC 12th World Congress*, Sydney, 1993, Vol. 9, pp.229-232.

Fidge, C.J. Real-Time Schedulability Test for Preemptive Multitasking. Real-Time Systems, 14, p. 61-93, 1998.

Flickenger, R. (2000) Building Wireless Community Networks, *O'Relly & Associates, Inc*.

Gerken J. (2000) A Practical Approach to Managing Intersection Traffic Data for Large Scale Studies, *Mid–Continent Transportation Symposium*.

Husch, D. (2000). "Intersection Capacity Utilisation", www.trafficware.com.

Letia, T. S., Astilean, A., Avram, C., Hulea, M., Valean, H. (2002a). Urban traffic control system, *2002 4th International Workshop on Computer and Information Technology, University of Patras, Greece*, September.

Letia, T. S., Astilean, A., Hulea, M., Avram, C. and Valean, H. (2002b). Object street traffic model. *Proceedings of 2002 IEEE-TTTC International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca*, *Romania,* Vol. 1, pp. 170-175.

Letia, T. S., Hulea, M., Avram, C. and Valean, H. (2003). Real-time traffic information and control system. *27th IFAC/IFIP/IEEE Workshop on Real/Time Programming WRTP'03*, pp. 205-210, Poland.

Morris J (1998). Djikstra's algorithm morris@ee.uwa.edu.au

Muratore, F. (2000). UMTS Mobile Communications for the Future, *John Wiley & Sons, LTD*.

Zhan, B.Y.F. and Noon C.E. (1998). Shortest path algorithms: An evaluation using real road networks. *Transportation Science*.