# Dynamic traffic routing using Ant Based Control*

Bogdan Tatomir
Faculty of Electrical Engineering,
Mathematics and Informatics
Delft University of Technology
Delft, The Netherlands
b.tatomir@ewi.tudelft.nl

Leon Rothkrantz
Faculty of Electrical Engineering,
Mathematics and Informatics
Delft University of Technology
Delft, The Netherlands
l.j.m.rothkrantz@ewi.tudelft.nl

**Abstract** – *This paper tackles the problem of dynamic routing for traffic in a city. We built a simulation environment with a routing system that guides the vehicles through the city using the fastest way, taking into account the load on the roads. A version of the AntNet algorithm is applied for routing the traffic. The artificial ants are moving in a virtual street network where each node corresponds to a crossroad. Each node has a routing table used for guiding the cars. This model is supplemented with actual data from the traffic by the cars themselves. This enables the agents to divert the traffic from congested routes and improves the traveling time. The simulation environment makes possible to see the effect in different cities or accident circumstances, especially in crisis situations, when, because of a disaster, multiple roads become unavailable or are heavily congested, and most of the drivers are disoriented.*

## 1 Introduction

Road traffic is getting busier and busier each year. Everyone is familiar with traffic congestion on highways and the ones in the city. And everyone will admit that it is a problem that affects us both economically as well as mentally. Furthermore finding your way in an unknown city can be very difficult even with a map. Navigation systems like CARiN can help in such cases. After the user has entered his destination, these systems display the route to be followed. The latest versions are also able to use congestion information to avoid trouble spots. But such information is only available for highways and not in a city. This paper addresses the dynamic routing of traffic in a city. We want to set up a routing system for motor vehicles that guides them through the city using the shortest way in time, taking into account the load on the roads. Furthermore we want the routing system to be distributed, for more robustness and load distribution. Exact routing algorithms like Dijkstras algorithm only apply to central routing. Ant-based algorithms have proven to be superior to other distributed routing algorithms in [1], [2], [5]. In [5] an ant-based algorithm was used for routing and load balancing in a telephony network. In [4] the algorithm is applied to packet switched networks with basic ideas taken from [5]. In this paper we will apply a variant of the AntNet algorithm described in [2], to a traffic network in a city. In [3] we assumed a similar problem with another variant of ABC presenting preliminary test results.

*0-7803-8566-7/04/$20.00 © 2004 IEEE.

## 2 Dynamic data

To route the traffic dynamically through a city we need dynamic data about the state of the traffic in the city. This can for example be directly from sensors in the road-surface. Such sensors can count vehicles and measure the speed of the vehicles. That information can be used to compute the time it takes to cover a part of the road. Another source can be the traffic information services. They can inform the system about congestion, diversions of the road, roadblocks and perhaps open bridges. And finally the vehicles themselves can provide the system with information about the path they followed and the time it took them to cover it. The current technology enables to fix the position of a vehicle with an accuracy of a few meters. That position can be communicated to the system along with the covered route. For our routing system we will at first only use the latter type of information as dynamic data. But of course the model is open for additional types of dynamic data. The information from the vehicles is handled by a separate part of the routing system, called the timetable updating system (see figure 1). This subsystem takes care that the information is processed for use by the ant-based algorithm. This way one vehicle drives a certain route and sends its performance to the routing system. Another vehicle is able to use that information to choose the shortest route.

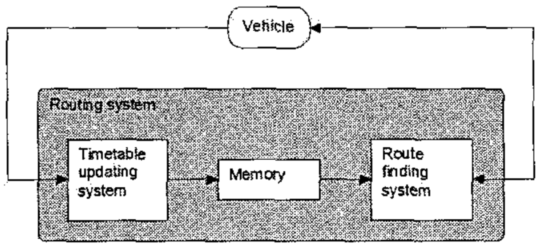

Figure 1: Design of the Routing system

## 3 Communication with the system

We will now explain the structure of the system from the viewpoint of the vehicle and its driver. A vehicle is driving through a city and it wants to know the way it should follow. The driver enters the address where he wants to go and expects the routing system to guide him. Besides the destination the routing system needs to know the location where the vehicle is at the moment. The current position is obtained by using the GPS (Global Positioning System). This is shown by arrow A in figure 2. The position is measured in latitude/longitude coordinates. For the vehicle these coordinates are translated in a position on a certain road with the aid of a digital map of the city. Now the vehicle has enough information to request the routing system what route to follow. The vehicle sends its position and its desired destination along with the request for the route to the routing system (arrow B). Arrow D is the answer from the routing system that contains the route that the vehicle should follow. These steps are pretty obvious, but we have skipped arrow C. This arrow indicates that the vehicle provides the routing system with information about the route is has followed since the previous time. The information consists of (1) the location and time at the moment of the previous update, (2) the location and time at this moment and (3) the route that the vehicle has followed in between these times and locations. Figure 3 shows a detailed enumeration of the information that is send along the indicated arrows.

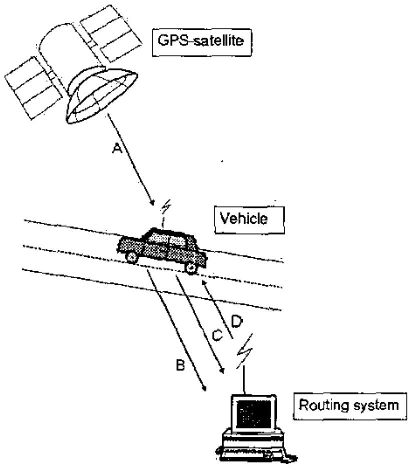

Figure 2: Communication of the vehicle

## 4 Timetable updating system

The most important problem of this research is solved by the timetable updating system and the route finding system. These two subsystems together form the routing system. If the function of the route finding system is clear, routing the vehicles, let's argue now for the use of the timetable updating system. The route finding system needs information about

| | From | To | Data |
|---|---|---|---|
| Arrow A | GPS-satellite | Vehicle | SEND_POSITION, latitude/longitude coordinates |
| Arrow B | Vehicle | Routing system | UPDATE, previous time/position, covered road A, covered road B, covered road C, ..., current time/position |
| Arrow C | Vehicle | Routing system | REQUEST_ROUTE, current position, destination |
| Arrow D | Routing system | Vehicle | ANSWER_ROUTE, road A, road B, road C, ... |

Figure 3: Communication data

the current state of the network. A static route finding system could use a fixed set of data, but we will use a dynamic route finding system that needs dynamic data. Those data are provided by the timetable updating system. That information can be for example the load of the parts of the network. But a more direct and, therefore, more practical type of information, is the time it takes to cover a road. Vehicles send information about their covered route to the timetable updating system. From that information this system computes the traveling-times for all roads and stores it in the timetable in the memory. Besides the timetable also a history of measurements is stored in the memory. The route finding system uses the information in the timetable to compute the shortest routes for the vehicles. When a vehicle requests route information, the route finding system sends this information back to the vehicle.

Now, let's see how the update information is used to compute the travel time for every link of the covered route. Because only the start time and the end time of the routes is given in the update information, only average travel times can be computed for every link. First of all, the total time to cover the route without delay at intersections or on the links, needs to be known. The formula for this is:

$$T = \sum \frac{d_l}{S_l} \qquad (1)$$

With the optimal time calculated with formula (1) the travel time for every single link can be computed by using the following formula:

$$M_l = \frac{L_l}{S_l \cdot T} \cdot t \qquad (2)$$

1. $d_l$ is the covered distance on link $l$ in meters
2. $L_l$ is the length of link $l$ in meter
3. $S_l$ is the allowed speed on link $l$ in meter per second
4. $T$ is the total time in seconds that is needed to cover the route without delay
5. $t$ is the total time of the covered route in seconds. $t = t_2 - t_1$ with $t_1$ start time and $t_2$ end time
6. $M_l$ is the measurement of the travel time for link $l$ in seconds

## 5 Route finding system

This system uses the AntNet algorithm [2]. The idea of emergent behavior of natural ants can be used to build routing tables in any network. We will apply it in a model of a traffic network in a city, represented by the roads and their intersections. This network is a directed graph. Each node in the graph corresponds to an intersection. The links between them are the roads. Routing is determined through complex interactions of network exploration agents. The behavior of these agents is modelled on the trail-laying abilities of natural ants. The agents move across this virtual network between randomly chosen pairs of nodes. They are divided into two classes, the forward ants and the backward ants. The idea behind this sub-division of agents is to allow the backward ants to utilize the useful information gathered by the forward ants on their trip from source to destination.

As the forward ants move, pheromone is deposited as a function of the time of their journey. That time is influenced by the congestion encountered on their journey. They select their path at each intermediate node according to the distribution of the simulated pheromone at each node. Every single node in the network has a probability table for all possible final destinations. The tables have entries for each neighbouring node that can be reached via one connecting link. The probabilities influence the agents selection of the next node in their journey to the destination node. The probability of the agents choosing a certain next node is the same as the probability in the table. The probability tables only contain local information and not global information about the best routes. Each time an agent visits a node the next step in the route is determined. This process is repeated until the agent reaches its destination. Thus, the entire route from a source node to a destination node is not determined beforehand.

Once the destination is reached, a backward ants inherit the data collected by the forward ant and use it to update the routing tables of the nodes. Based on this principle, no node routing updates are performed by the forward ants, whose only purpose in life is to report network delay conditions to the backward ants.

Besides probability tables, a node $i$ keeps a second data-structure, which its main task is to follow the traffic fluctuations in the network. It is given by an array $M_i(\mu_d, \sigma_d^2, W_d)$ that represents a sample means $\mu_d$ and variance $\sigma_d^2$ computed over the packet's delay from node $i$ to all the nodes $d$ in the network, and an observation window $W_d$ where the last packet's best trip time towards destination $d$ are stored.

Our network model differs from the packet switch network for which AntNet was designed. There are no packets running in our network. It also has no buffers in nodes, and an infinite bandwidth is available on the links. The links still have a virtual delay provided from the time tables. The delay represents the necessary time for a car to cross the link.

We adapted AntNet for our traffic network. The algorithm works as follows:

1. The mobile agents $F_{s \longrightarrow d}$ are launched at regular time intervals from every network node $s$.

2. Each ant keeps a memory about its path (visited nodes). When an ant arrives in a node $i$, coming from node $j$, it memorizes the identifier of the visited node $i$ and the virtual delay of the link ( the trip time necessary for a car to travel from intersection $j$ to intersection $i$). These data are pushed onto the memory stack $S_{s \longrightarrow d}(i)$.

3. When ant comes in the node $i$, it has to select a next node $n$ to move to. The selection is done according with the probabilities $P_d$.

4. If a cycle is detected, that is, if an ant is forced to return to an already visited node, the cycle's nodes are popped from the ant's stack and all the memory about them is destroyed.

5. When the destination node $d$ is reached, the agent $F_{s \longrightarrow d}$ generates another agent (backward ant) $B_{d \longrightarrow s}$, transfers to it all of its memory, and dies.

6. The backward ant takes the same path as that of its corresponding forward ant, but in the opposite direction. At each node $i$ along the path it pops its stack $S_{s \longrightarrow d}(i)$ to know the next hop node.

7. In every node $i$, the backward ant $B_{d \longrightarrow s}$ updates the data structures of the node: the local traffic statistics and the routing table for all possible paths (these are via all neighbours of the node $i$) with the destination node $d$, and, if $T_{i \longrightarrow d} < I_{sup}(d)$, also the sub-paths of $s \longrightarrow d$.

$$I_{sup}(d) = \mu_d + z\frac{\sigma_d}{\sqrt{|W_d|}} \quad (3)$$

$$z = \frac{1}{\sqrt{1-\gamma}}, \quad \gamma \in [0, 1) \quad (4)$$

A reinforcement value $r$ is computed. This is a function of the time $T_{i \longrightarrow d}$ the ant computed and the local stochastic model of traffic in the node.

$$r = c_1 \frac{I_{inf}(sd)}{T_{i \longrightarrow d}} + c_2 T \quad (5)$$

$$T = \frac{I_{sup}(d) - I_{inf}(d)}{(I_{sup}(d) - I_{inf}(d)) + (T_{i \longrightarrow d} - I_{inf}(d))} \quad (6)$$

where $I_{inf}(d) = min\{W_d\}$, the best value in the observation window $W_d$.

Then, the reinforcement is transformed and the new value is used to update the probabilities:

$$r = \frac{1 + e^a}{1 + e^{\frac{a}{r}}} \quad (7)$$

$$P'_{dj} = P_{dj} + r(1 - P_{dj}) \quad \text{when } j = n \text{ next node} \quad (8)$$

$$P'_{nd} = P_{nd} - rP_{nd}, \text{ for } j \neq n \quad (9)$$

8. When the destination node $d$ is reached, the agent $B_{d \longrightarrow s}$ dies.

In the mapping function for $r$ we skipped the $|N_i|$ factor present in the AntNet. This is because we want to have small values for $r$ and less fluctuation in the probability tables. In the packet switch networks one of the aim of the routing algorithm is to use the bandwith capacity in optimal way. The packets are sent to the destination not necessary following the best route.

This can't be accepted in the car traffic environment. We can't send the cars on longer paths just for keeping away the congestion on the roads. They are not packets. The traffic jams should be avoided, but also the drivers want to reach their destinations in the shortest possible time. So, the Routing system computes the optimal path for the vehicles choosing, all the time as next intersection, the one with the higher entry in the probability table.

## 6 Experiments

As a proof of concept we run experiments in two different simple traffic networks simulating two different situations. In the first experiment, as displayed in Figure 4. The cars are generated on the middle of the roads and their destination is the middle of another link. For the vehicles going from the roads between intersection 1 and 2 and intersection 9 and 10 there are two reasonable options. They can take the northern path via intersections 3, 4 and 8, or they can take the southern path via intersection 6. The length of both routes is different. The southern is shorter and more attractive from this point of view. So all vehicles from 1/2 to 9/10 and vice versa will initially take the southern route.

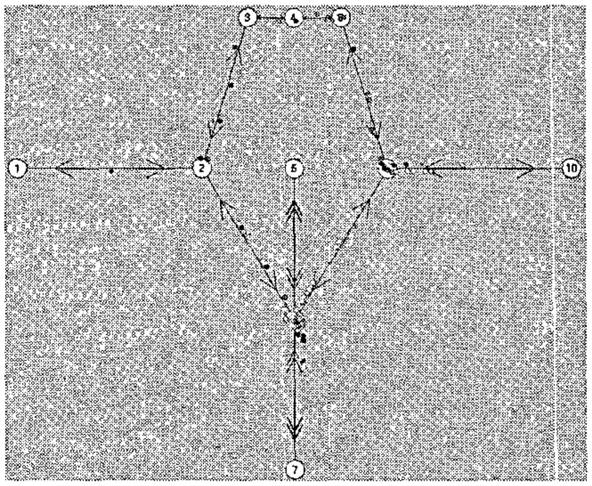

Figure 4: Test1: Node 6 is a crowded intersection with traffic lights

Nevertheless intersection 6 of the southern route is a point where many vehicles from different roads join and cross each others path. Therefore it is controlled by traffic lights. These traffic lights make the crossing safer and the priority for vehicles from different roads is distributed more fairly. On the other hand the traffic might perceive a considerable delay at this intersection because of the heavy load and the traffic lights. This will cause the northern path to be more attractive for vehicles from intersection 1/2 to 9/10 and vice versa. So we expect the vehicles that follow the advice of the Routing system to drive via the northern roads, where there are no delaying intersections. This should result in faster routes for these vehicles as opposed to the vehicles that do not use the Routing system.

In the second experiment (see Figure 5), we simulate an accident on the lane between nodes 11 and 7. The speed on this lane is set to 15km/h. For the other links, including the lane between nodes 7 and 11 the speed is 50km/h. The traffic is generated from intersection 1/2 to 15/16 and vice versa. The default route for a car driving from 15/6 to 1/2 will be: 15-11-7-3-2. We expect that a queue will form on this lane and will grow affecting also the lane between nodes 15 and 11. Because all the roads have the same length, two alternative routes are available for the vehicles guided by the Routing system: (15-14-10-6-2), (15-11-10-6-2). We expect these vehicles to follow one of them.

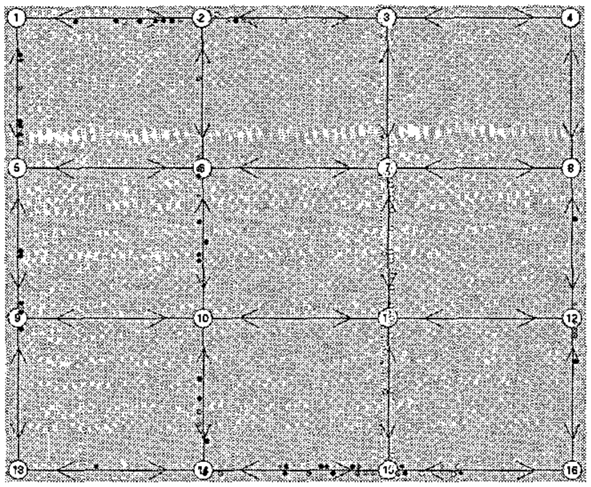

Figure 5: Test2: Accident on the lane between nodes 11 and 7

## 7 Results

First we run the experiments with the routing system disabled. The average time needed for the cars to reach their destination is displayed in figure 6 for the first network and in figure 7 for the second network. All the cars had the same behavior, following the predefined paths. In the first network big queues were formed on the lanes between 1-2, 2-6, 5-6, 7-6, 9-6 and 10-9. In the second example the congestion grow from 11-7, to 15-7 and 16-7. 1 second is 5 time steps, so the simulated period is of 2000s, more than 30 minutes.

The value at the end of the test is the average of all measured travel times since the start of the experiment until the end.

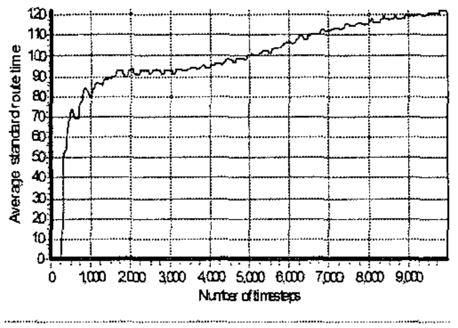

Figure 6: Test1: The average trip time without smart routing

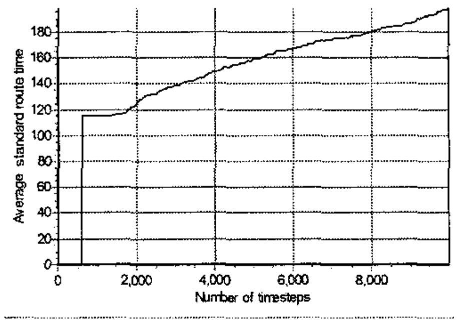

Figure 7: Test2: The average trip time without smart routing

We enabled the Routing system and run the tests again. The first network example provided the following graphs (Figure 8 and Figure 9). These graphs show the differences in the average travel time of standard and smart vehicles. The standard vehicles do not use the Routing system, the smart vehicles do. When we zoom in on the graphs, we see that the (rounded) value for the standard vehicles is 87 seconds and the value for the smart vehicles is 80 seconds. So, the overall profit for the smart vehicles in this case is 8% on average as opposed to the standard vehicles, which do not use the Routing system. Comparing with the results in Figure 6, the conclusion is that the Routing system improves the travelling times not only for the cars that use it. Indirectly, also the other cars have benefit from it. The responsiveness of the routing system is fast. After only 80s from the start, the smart cars change their route from the south path, via node 6, to the north path via nodes 3, 4 and 8. It is a big improvement compared with the 250s obtained using in the same situation the ABC algorithm version described in [3].

In the last test we simulated, in the second network, an accident on the lane between nodes 11 and 7. This was done decreasing, after 1000 time steps, the speed on this lane from 50km/h to 15km/h (see Figures 10 and 11). The response of the routing system was very fast. Because two alternative

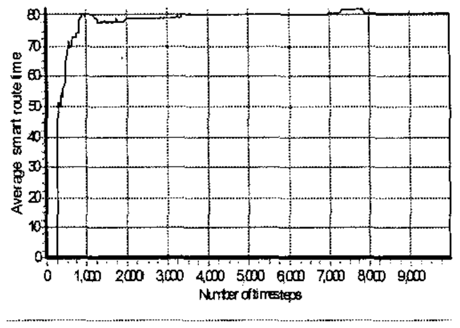

Figure 8: Test1: The average trip time for the cars using the Routing system

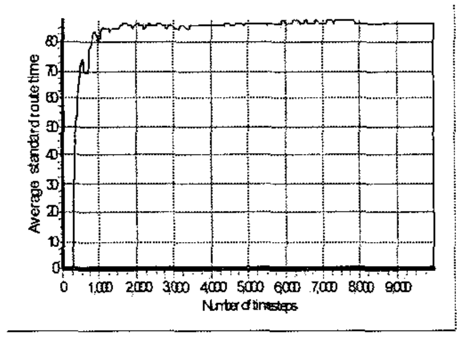

Figure 9: Test1: The average trip time for the cars using the standard path

routes are available, with the same length, some of the smart cars are routed even before the incident, on different routes. All of them are soon leaving the default path. But this leads to new congestion in the intersections 2 and 15 after around 4000 time steps. The answer from the Routing system is to guide the cars to the destination via the nodes 1 and 16, so using longer routes. The average trip time is shrinking constantly for both type of cars. After 2000s we notice 144s for the smart ones and 166s for the cars that don't use the system, which means 12.5% difference. This test proves the high performance of the algorithm, that solves the two congestions. It guides the cars on a longer path in distance, but faster in time, even when two shorter alternative roads are available. The algorithm presented in [3], in the same situation, uses only the two short paths via nodes 2 and 15, resulting in big congestions in these intersections.

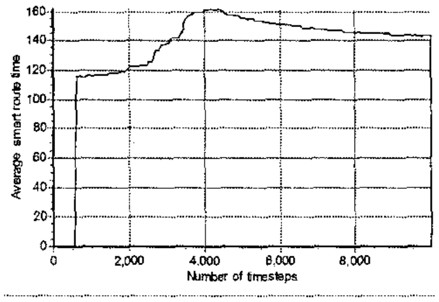

Figure 10: Test2: The average trip time for the cars using the Routing system

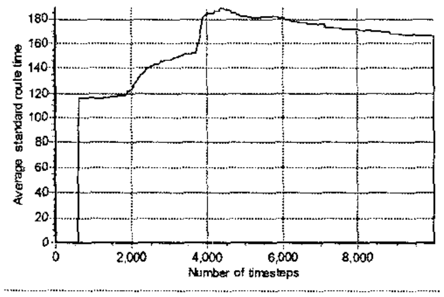

Figure 11: Test2: The average trip time for the cars using the standard path

## 8 Conclusions and future work

The ant algorithm described in the paper, proved to be reliable for dynamic routing of the traffic on the streets network of a city. The small networks used in the tests were built just for proving the performance of the algorithm. Nevertheless, we tested the system with networks for whole cities like Delft (42 nodes), the centrum of Rotterdam (110 nodes) and a detailed map of Beverwijk (523 nodes). In the close future we want to make the Routing system distributed on more levels. On the the base level each Routing system will route the traffic for one city. At a higher level we will have another Routing system for the motorway network.

At DECIS Lab Delft, we are involved in a project concerning Multi-agent based intelligent network decision support systems in a chaotic open world (COMBINED). We are planing to expand the system to a simulation environment for a crisis situation in a city. We are planing to apply such an algorithm, like the one presented in the paper, for evacuation from a contaminated city area. The vehicles will have more than one possible destinations ( every exit point from the dangerous area ).

## References

[1] G. Di Caro, M. Dorigo. AntNet: A Mobile Agents Approach to Adaptive Routing, Technical Report 97-12, IRIDIA, Universit Libre de Bruxelles (1997)

[2] G. Di Caro and M. Dorigo. AntNet: distributed stigmergetic control for communication networks. Journal of Artificial Intelligence Research (JAIR), 9, (1998), 317–365

[3] R. Kroon, L.J.M Rothkrantz. Dynamic vehicle routing using an ABC-algorithm, Transportation and Telecommunication in the 3rd Millenium, Prague, May 26 27, (2003)

[4] L.J.M.Rothkrantz, J.C. Wojdel, A. Wojdel, H. Knibbe. Ant based routing algorithms, Neural Network World, Vol. 10, No.3, July (2000)

[5] R. Schoonderwoerd, O. Holland, J. Bruten, L.J.M. Rothkrantz. Load Balancing in Telecommunication Networks, Adaptive Behaviour, vol. 5 no. 2, (1997)