„Transportation and Telecommunication in the 3rd Millenium“
10th Anniversary of the Foundation of the Faculty Transportation Sciences
Prague, May 26 – 27, 2003

DYNAMIC VEHICLE ROUTING USING AN ABC-ALGORITHM

Ronald Kroon, Leon Rothkrantz *

***Abstract**: In the past years the application of agent algorithms based on the natural behaviour of ants have shown to be successful in routing data through communication networks. Using the trail-laying abilities of ants the mobile agents are able to create well performing routing tables. In this paper an Ant Based Control algorithm is applied to the routing of road traffic trough a city. The algorithm is tested in a simulation environment that makes it possible to show the effect in different cities and circumstances. The agents do not move through a real city, but use a model of a city map. This model is supplemented with actual data from the traffic in the city. This enables the agents to divert traffic from congested routes, which improves travelling-times.*

Key words: Routing algorithms, mobile agents, ant-based algorithms, distributed routing

1. Introduction

Road traffic is getting busier and busier each year. Everyone is familiar with traffic congestion on highways and 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. These systems display the route to be followed when the user has entered his destination. 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.

The routing system uses a routing algorithm based on earlier versions of Ant Based Control-algorithms. Exact routing algorithms like Dijkstra’s algorithm only apply to central routing. And ant-based algorithms have proven to be superior to other distributed routing algorithms in [1,2]. In [2] an ant-based algorithm was used for routing and load balancing in a telephony network. In [3] the algorithm is applied to packet switched networks with basic ideas taken from [1]. And now we will apply a variant of the algorithm to a traffic network in a city.

2. Theory

This section presents a short introduction to an important aspect of the behaviour of ants and the basic ideas of ant based control.

* Ronald. Kroon, Leon .J.M Rothkrantz
Delft University of Technology, Knowledge Based Systems, Mekelweg 4, 2628 CD Delft,
The Netherlands, E-mail: L.J.M.Rothkrantz@cs.tudelft.nl

2.1 Emergent behaviour of ants

Insects like ants perceive only a very local piece of the world they live in. But it is no coincidence that they do find their way back to a food source or their nest. When a group of such animals interact they can exhibit a higher-level behaviour. This behaviour is most often called emergent behaviour. Instead of a central controlling authority ants interact with their environment to achieve common goals. The resulting behaviour of an ant colony can be very complex.

The emergent behaviour is enabled by stigmergy. Stigmergy is a way for entities to communicate indirectly with each other through the environment. An ant colony uses this for finding the shortest route from their nest to a food source and back. Ants only react to local stimuli from their environment, but they can change some of those local stimuli. Such a modification will influence future actions of other ants at that location.

The ants lay pheromone, a kind of hormone, as a mutual signalling system. When looking for food, the ants follow the pheromone trails with a probability proportional to the strength of the trail. They do not necessary follow the track with strongest pheromone trail. There will often be a certain amount of error (or noise). The strength of the trail sensed by an ant depends on the original strength and the time elapsed since the pheromone was laid. This is because the pheromone diffuses in time. Several ants can travel the same route, resulting in a pheromone trail laid by different ants at different times. The pheromone trail sensed by ants is therefore a composite one. The probability that an ant chooses a particular route depends on the concentration of the pheromones. A stronger pheromone trail increases the chance an ant chooses the route belonging to that pheromone trail. This mechanism makes the ants bias towards the shortest paths. It works for the following three reasons:

- Shorter routes will be completed earlier than longer routes and thus attract other ants *earlier*.
- The ant density will be bigger at shorter routes when the ants choose the alternatives equally likely, and more ants produce *more* pheromone.
- Ants travelling shorter routes will arrive earlier. This causes the pheromone to be *stronger*, because less of the pheromone trail has diffused.

This strengthening process may continue until there is no more pheromone on the longer paths.

2.2 Ant-based control for network management

We can use the idea of emergent behaviour of natural ants to build routing tables in any network. We will apply it in a traffic network in a city, i.e. the composition of the roads and their intersections. This network is represented by a directed graph. Each node in the graph corresponds to an intersection. The links between them are the roads. Mobile agents, whose behaviour is modelled on the trail-laying abilities of natural ants, replace the ants. The agents move across the network between randomly chosen pairs of nodes. As they 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. Each node in the network has a probability table for every possible final destination. The tables have entries for each neighbouring node that can be reached via one connecting link. The probabilities influence the agent's 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 no global information on 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.

Agents are launched at each node with regular time intervals with a random destination node. They travel around the network using the probabilities in the probability tables. The probabilities per destination are all filled with equal values for all nodes before the process begins.

3. Design

This section explains the design of the routing system.

3.1 Dynamic data

We want to route the traffic dynamically through a city. Therefore we need dynamic data about the state of the traffic in the city. This can be gathered 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 us 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. 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.

3.2 Architecture

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. The driver enters the address where he wants to go and expects a routing system to tell him where to go. Besides the destination the routing system needs to know the location where the vehicle is at the moment. Therefore the vehicle sends a request to a satellite of the GPS (Global Positioning System). This is shown by arrow A in figure 1. GPS is a system that can determine a position of the sender with an accuracy of a few meters. So the GPS-satellite answers the vehicle with its current position (arrow B). This position is measured in latitude/longitude co-ordinates. In the vehicle these co-ordinates 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 D). Arrow E 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 it 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. Table 1 shows a detailed enumeration of the information that is send along the indicated arrows.

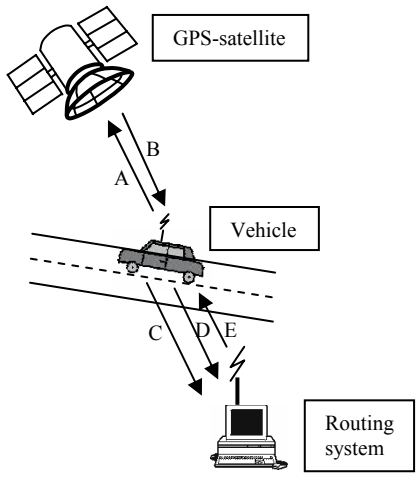


Fig. 1 *Communication of the vehicle*

Table 1 Communicated data between the different objects

Arrow	From	To	Data
A	Vehicle	GPS-satellite	REQUEST_POSITION
B	GPS-satellite	Vehicle	ANSWER_POSITION, latitude/longitude co-ordinates
C	Vehicle	Routing system	UPDATE, previous time/position, covered road A, covered road B, covered road C, …, current time/position
D	Vehicle	Routing system	REQUEST_ROUTE, current position, destination
E	Routing system	Vehicle	ANSWER_ROUTE, road A, road B, road C, …

3.2.1 Communication in time

Now we will give an impression of how the communication works in time. Firstly, a REQUEST_POSITION (arrow A) will be send with some regular time interval, for example every minute. Directly after that an ANSWER_POSITION (arrow B) will be sent back to the vehicle. The information that is sent with an UPDATE (arrow C) is most valuable as soon as a new position is known. The fact is that the UPDATE must be sent together with a current position, and the longer it is delayed the older the data is. Therefore an UPDATE will always be sent directly after an ANSWER_POSITION. This does not alter the fact that an UPDATE does not have to be sent after every ANSWER_POSITION. This frequency can for example be lower to reduce communication. Finally a REQUEST_ROUTE (arrow D) and an ANSWER_ROUTE (arrow E) will also always succeed an ANSWER_POSITION. When a new route is requested and acquired, the old route is overwritten. Clearly a REQUEST_ROUTE needs a current position. An old value for the position could cause a vehicle to be travelling on a road that is no longer in the list of its new route. The interval by which a REQUEST_ROUTE is sent can differ from the former messages. After the first route is acquired it will mostly be valid for the rest of the travelling-time. So the frequency can be lower than the frequency of the other messages. Figure 2 clarifies again which messages succeed each other.

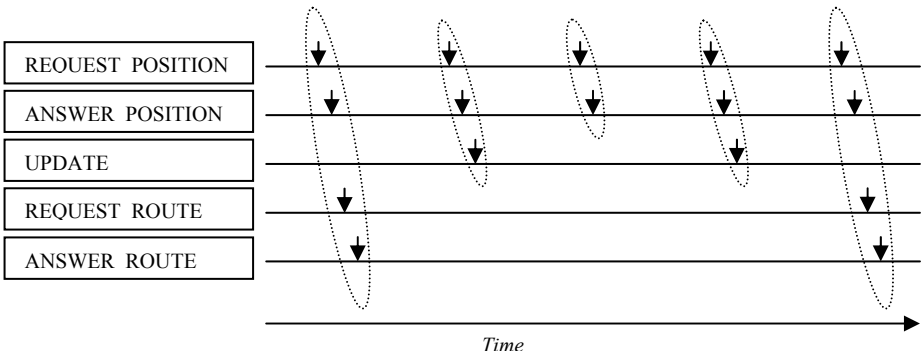


Fig. 2 *Succeeding messages*

3.2.2 Distributed routing system

The use of the ABC-algorithm allows the routing system to be distributed. This means that the computation is done on several computer systems that are mutually connected via a network. Distribution of computational power gives some advantages above a central routing system.

Firstly what normally has to be done by one computer system is now done by several computer systems, which increases the speed and the memory space. Secondly, when properly implemented the failure of one of the systems does not have to imply a total break down of the routing system. This does however involve some extra communication necessary for information that is not available on the concerning computer system. It will eventually depend on the amount of information that needs to be communicated between the computer systems whether the actual speed will be higher than with a central routing system. Another possible disadvantage is that the traditional routing algorithms cannot be used. Those routing algorithms allow for perfect routing, i.e. giving the best routes possible. The ABC-algorithm only approaches the best routes. But the results of this research will have to show that the ABC-algorithm is sufficiently accurate to route the traffic.

3.3 Routing problem

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*. The relations are shown in figure 3. The function of the *route finding system* will be clear: we are building a system to route vehicles. The reason why we need the *timetable updating system* is the following. The *route finding system* needs information about the 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 travelling-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 reason to keep a history will become clear later in section 3.3.1. 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.

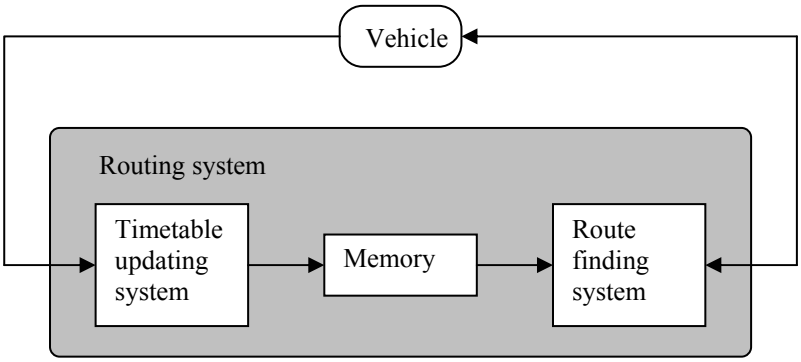


Fig. 3 *Design of the routing system*

3.3.1 Timetable updating system

This subsystem could receive its information about the traffic network in the city from different sources. This could for example be directly from sensors in the road-surface. But the main sources are the vehicles themselves. They provide the system with information about the path they followed and the time it took them to cover it. With this information the timetable updating system computes the travelling-times for every part of the road. The travelling-times are placed in the timetable. This timetable can be seen as a two-dimensional matrix with all intersections of the traffic network along both axes. When one can go from one intersection

directly to another, there will be an entry in the table that represents an estimate for the time to cover that road. The intersections that cannot reach eachother unless via another intersection will have no entry in the table. Figure 4 is an example of a traffic network and table 2 shows its timetable.

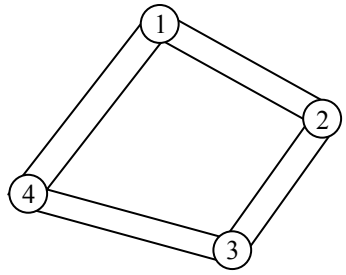


Fig. 4 *A simple traffic network*

Table 2 *The timetable matching figure 4*

From: \ To:	Intersection 1	Intersection 2	Intersection 3	Intersection 4
Intersection 1		25 sec		33 sec
Intersection 2	24 sec		18 sec	
Intersection 3		18 sec		20 sec
Intersection 4	31 sec		23 sec	

We will now explain how we represent a traffic network in our model. Figure 5 shows a simplified part of a city map. Figure 6 shows the internal representation of that map. As one can see there is a forward and a backward link for every part of the road. This represents that the traffic can move in both directions. Furthermore one can notice that at every point where a driver must choose between more than one road, the road is divided into separate links.

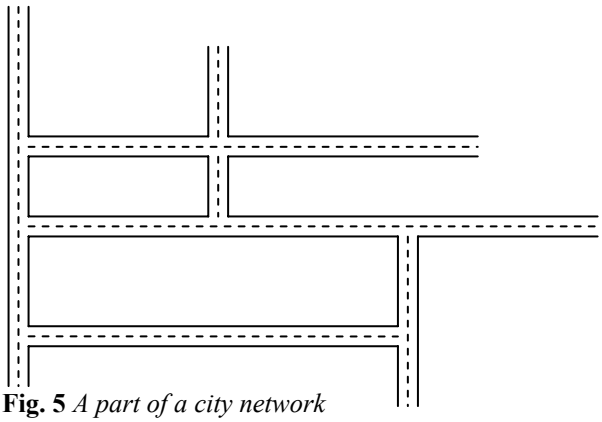


Fig. 5 *A part of a city network*

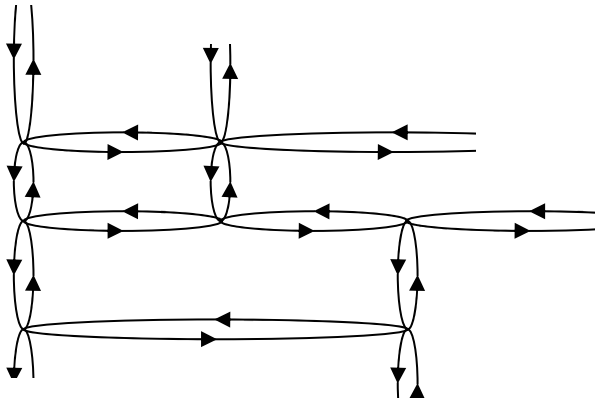


Fig. 6 *Internal representation*

The timetable updating system computes the time for every part of the covered road by using dynamic information from the vehicles and the static information about the network. Therefore we need the total of the covered road of a vehicle since the last update of that vehicle to compute an estimate for the time to cover the separate links:

$$\left.\begin{aligned} D &= \sum_l d_l \\ M_l &= \frac{d_l}{D}\left(t_2 - t_1\right) \end{aligned}\right\} \quad (1)$$

d_l is the covered distance on link l. D is the covered road for the update of this vehicle. t_1 and t_2 are the times of the updates. M_l is the measurement of the time for link l.

On quiet roads it is very well conceivable that there will be no vehicles that send route information for a long time. When there are no updates for a certain part of a road we still want to know an estimate for the time it takes to travel that way. For that purpose we can use the length of the road, the maximum allowed speed and some correction factor. The length and speed yield a time estimate for the road. That time can be adjusted a little with the correction factor to represent that the average speed will be a bit higher (or lower) than the

maximum allowed speed. This yields a default value for the travel time of the road. When we look at a road with a length of 200 meter where the average speed is 50 km/h (= 14 m/s), the default value will be 14 seconds. When at some time a vehicle covers that road in 20 seconds we want the entry in the timetable to be adjusted so that it represents a value of about 20 seconds. This way the routing algorithm will less likely use the road for other vehicles. But half a day later it is useless to know that there has ever been a car that was delayed on that road. The situation has changed many times since then. In fact information older than an hour is usually obsolete. So what we really want is that new information gives an impulse to adjust the time, but the effect should gradually diminish. And after for example an hour the effect of the information should be faded out completely. If closely after the first a second car provides the system with new information, then we should compute a weighed average. The information of the first car counts less than the information of the second, because it is older. And when the information of both vehicles gets a little older the default value should become more important. There are several ways to accomplish this, but we will use the following function because it is intuitively useful and easy to adjust.

$$\left.\begin{aligned} T(t) &= \frac{D + \sum_k W(t - S_k) \cdot M_k}{1 + \sum_k W(t - S_k)} \\ &\textit{with} \\ W(t) &= w \cdot f^{t^2} \quad \forall\ 0 \le t < h \\ W(t) &= 0 \qquad \textit{else} \end{aligned}\right\} \quad (2)$$

Here $T(t)$ is the value in the timetable at time t, D is de default value in the timetable, M_k is the result of the k^{th} measurement acquired from the information of a vehicle, S_k is the start time at which the information is added. $W(t)$ is the weight of a measurement at time t, w is the weight of the measurement at the start time, f is a factor that diminishes the weight in time, h is the time that a measurement has any effect on the outcome.

To explain this function: as long as there are no results of a measurement, only the default value D influences the value in the timetable. When, at time S_1, the first measurement is received, that value M_1 gets a weight of w (for example 25). Then the value for the timetable is a weighed average:

$$\frac{1 \cdot D + 25 \cdot M_1}{26}$$

The weight of M_1 fades in time when f is positive and less than one, for example 0.99. With the used function $W(t)$ the result of a measurement on the value of the timetable will be shaped like a half bell. This is shown in figure 7.

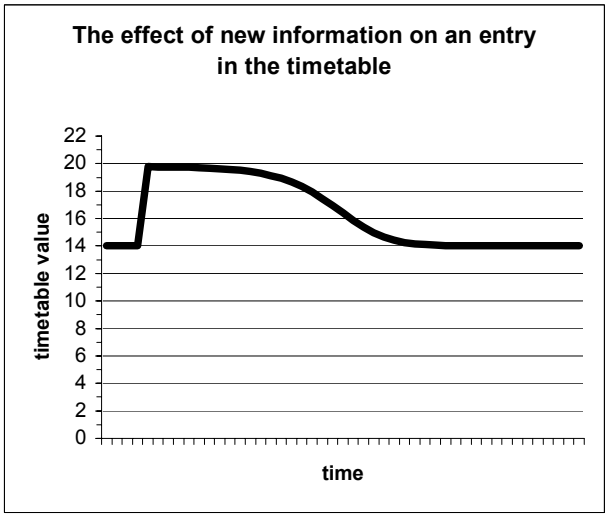


Fig. 7 *An update gives an impulse to a value in the timetable*

fter h time a measurement does not have any effect anymore. So all measurements must be kept in memory for h time. The function takes care to average measurements when there are more than one available for a certain road. When a new measurement becomes

available the values should be recomputed. But the values also have to be recomputed at regular intervals because of the fading effect.

3.3.2 Route finding system

This system uses the earlier mentioned Ant-Based Control algorithm (ABC-algorithm). This algorithm makes use of forward and backward agents. The forward agents collect the data and the backward agents update the corresponding probability tables in the associated direction. The algorithm consists of the following steps:

- At regular time intervals from every network node **s**, a forward agent is launched with a random destination **d**: F_{sd}. This agent has a memory that is updated with new information at every node **k** that it visits. The identifier **k** of the visited node and the time it took the agent to get from the previous node to this node (according to the timetable) is added to the memory. This results in a list of (**k**, $\mathbf{t}_k$)-pairs in the memory of the agent. Note that the agent can move faster than the time in the timetable.
- Each travelling agent selects the link to the next node using the probabilities in the probability table. The probabilities for the nodes that have already been visited by this agent are filtered out for this agent. Then a copy of the remaining probabilities is made for this agent and these probabilities are normalized to 1. Only this agent uses this temporary probability distribution to choose a next node, so the probability table is not updated yet.
- If an agent has no other option than going back to a previously visited node, the arising cycle is deleted from the memory of the agent.
- When the destination node **d** is reached, the agent F_{sd} generates a backward B_{ds}. The forward agent transfers all its memory to the backward agent and then destroys itself.
- The backward agent travels from destination node **d** to the source node **s** along the same path as the forward agent, but in the opposite direction. It uses its memory instead of the probability tables to find its way.
- The backward agent with previous node **f** updates the probability table in the current node **k**. The probability p_{df} associated with node **f** and destination node **d** is incremented. The other probabilities, associated with the same destination node **d** but another neighbouring node are decremented. The used formulas are given below.

The probability of the entry corresponding to the node **f** from which the backward agent has just arrived is increased using the following formula:

$$P_{new,f} = \frac{P_{old,f} + \Delta P}{1 + \Delta P} \tag{3}$$

Here, $P_{new,i}$ is the new probability, $P_{old,i}$ the old probability and ΔP the probability increase. ΔP should be inversely proportional to the age of the forward agent. The formula we use is:

$$\Delta P = \frac{a}{t} + b \tag{4}$$

Where a and b are constants and t is the trip-time of the forward agent from this node to the destination node. This trip-time is the sum of the trip-times from this node to the destination node of the forward agent. We do not take into account that the conditions of the traffic network can change from the moment that the node is visited by the forward agent and the updating of the backward agent.

The other entries in the probability table with the same destination but other neighbouring nodes are decreased using the formula:

$$P_{new,i} = \frac{P_{old,i}}{1+\Delta P}, \quad \forall\, i \neq f \tag{5}$$

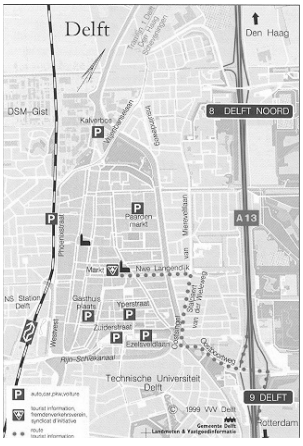
These formulas ensure that the sum of the probabilities per destination remains 1. Probabilities can only decrease if another probability increases. Probabilities can approach zero if other probabilities are increased much more often. This is not very desirable, because in time it may appear that the choice associated with that probability is the best at that time, but the agents will not detect it because they hardly ever take that route. This problem can be solved after analogy with the natural ants; they do not always use the pheromone trail as their guide, but sometimes just explore new routes. Therefore we introduce an exploration probability as a minimum value for each probability. An example could be 0.05 divided by the number of next nodes. After setting this minimum, the probabilities per destination are normalized to one again. This ensures that none of the entries in the probability table will approach zero.

For a given value of ΔP, the absolute and relative increase of P_{new} is much larger for small values of P_{old} than for large values of P_{old}. This results in a weighted change of probabilities. The formulas were taken from[3]. The probability tables are initialised with equal values in such a way that all the probabilities for one destination sum up to 1. The first agents do not have any information about routes, let alone the quality of the routes. The performance of the routing system will therefore be very bad and cannot be evaluated properly.

The quality of the routes found by the agents improves with time. At first the agents will find many cycles, but the number of cycles decreases as the probability tables are filled with information that is more accurate. Appearing congestion causes further adjustments. Finally the vehicles will be routed according to the highest probabilities in the tables. They do not have to explore other routes. They just want the best route.

4. Implementation

To test the proposed routing system a simulation environment is being developed. The development environment used is Borland Delphi 5. As well as the routing system also a simulation of traffic in a city is build. The traffic simulation should provide the dynamic information about the state of the roads in the city. And it also will use the routing system to route a certain fraction of the vehicles through the city. This simulation environment is still unfinished. Figure 8 and 9 give an impression of how a map of a city is used as a model in the simulation environment.



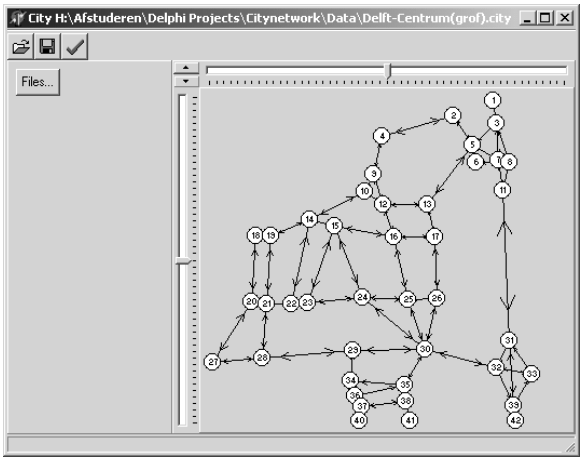


Fig. 8 Map of centre of Delft ***Fig. 9*** Visualisation of the map by the simulation environment

References

[1] G. Di Caro and M. Dorigo. *AntNet: distributed stigmergetic control for communication networks*. Journal of Artificial Intelligence Research (JAIR), Volume 9, pages 317-365.

[2] R. Schoonderwoerd, O. Holland, J. Bruten, L.J.M. Rothkrantz. *Load balancing in telecommunication networks*, Adaptive Behaviour, 5, 2 , 1997.

[3] L.J.M. Rothkrantz, J.C. Wojdel, A. Wojdel, H. Knibbe. *Ant based routing algorithms*, Neural Network World, Volume 10, 2000, pages 455-462.