Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

TRAVEL TIME PREDICTION FOR DYNAMIC ROUTING USING ANT BASED CONTROL

Bogdan Tatomir
Leon J.M. Rothkrantz
Adriana C. Suson

Delft University of Technology
Mekelweg 4, Delft, 2628 CD
THE NETHERLANDS

ABSTRACT

Currently most car drivers use static routing devices based on the shortest distance between start and end position. But the shortest route can differ from the shortest route in time. To compute alternative routes it is necessary to have good prediction models of expected congestions and a fast algorithm to compute the shortest path while being able to react to dynamic changes in the network caused by special incidents. In this paper we present a dynamic routing system based on Ant Based Control (ABC). Starting from historical traffic data, ants are used to compute and predict the travel times along the road segments. They are finding the fastest routes not only looking to the past and present traffic conditions but also trying to anticipate and avoid future congestions.

1 INTRODUCTION

At this moment the capacity of the highways is not sufficient to transport all car drivers without delay. Especially in the rush hours there are enormous traffic jams. In case of special events such as traffic accidents car drivers can be delayed by hours. The losses in time and money are enormous, so the problem of traffic congestion has a high priority.

To increase the capacity of the road network is not an option. The construction of new freeways in some areas is blocked by ecological, financial or political reasons. On the short term the only solution is to use the road network in an optimal way. Usually, there are many alternatives to travel from A to B. In case one route is blocked or delayed car drivers should be informed about alternative routes. Nowadays, information about traffic jams is broadcasted via news on radio and TV, or via mobilephones, but normally only big congestions are reported and no alterative routes are suggested. Most current route planner devices have TMC (Traffic Message Channel) integrated but their information is updated only every 30 minutes which is not so fast. For extra payment special services provide personalized information which gets updated every few minutes. But all these services use the situation of the roads at a given moment and are unable to take the future into account. If all the cars are advised to take the same alternative route, this one will also get congested.

To efficiently route car drivers, besides a detailed information of the road network, it is necessary to know the travel speed along road segments as a function of time. In many countries travel speed on the motorways is measured by special devices as: cameras, wires in the road and tracking car devices. Combining this data with historical data, travel time predictions can be computed. In this project we have access to the data of ANWB, an organization involved in traffic management.

In this paper we describe a system for providing dynamic routing information to car drivers. It uses Ant Based Control algorithm which is able, based on current and historical traffic information, to avoid congestions by predicting the future load on the roads. The outline of the paper is as follows. In the next section we present and discuss related work. The following section contains a description of the data and of the model used by our approach for time prediction. In Section 4 is presented the new ABC algorithm we use for dynamic routing. The experiments and results are described in Section 5. We end the paper with conclusions and future work.

978-1-4244-5771-7/09/$26.00 ©2009 IEEE

2 RELATED WORK

Various shortest path algorithms are available for computing the optimal route. The most popular algorithm is Dijkstra's algorithm that has a runtime complexity of $O(n^2)$, where n is the number of nodes in the network. Many variations to the Dijkstra's algorithm such as bidirectional search and binary heap implementation have been proposed to improve its response time. The A* algorithm (Chabini and Lan 2002), which is widely used in vehicle navigation, is an improved version of the Dijkstras algorithm. It makes use of an appropriate heuristic function to search the most promising nodes first thereby reducing the computation time.

Static routing algorithms like Dijkstra's algorithm only apply to central routing. To minimize the traveling time we need a decentralized routing algorithm which is able to adapt to the dynamic changes that take place in the traffic network. We found the answer in the real life behaviour of ants. Despite they are very simple insects, together, in a colony, they show what is called emergent behaviour and are able to accomplish complex tasks. They lay a special sort of volatile hormone, or pheromone, to create a signaling system between them. Using pheromone trails ants are able to find the shortest path from their nest to a food source and vice-versa.

Ant-based algorithms have successfully been applied for dynamic routing in different types of problems. First they were used for packet routing in networks ((Schoonderwoerd, Holland, Bruten, and L.J.M.Rothkrantz 1996) and (Di Caro and Dorigo 1998)) and in wireless networks ((Gunes, Sorges, and Bouazizi 2002) and (Di Caro, Ducatelle, and Gambardella 2005)). Recently, ant colony optimization was applied to the vehicle routing problems with time-dependent travel times ((Ichoua, Gendreau, and Potvin 2003) and (Donati, Montemanni, Casagrande, Rizzoli, and Gambardella 2008)).

In (Tatomir and Rothkrantz 2004) a dynamic vehicle routing system was introduced which uses the Ant Based Control algorithm (ABC-algorithm) for car navigation in a city. But the algorithm proved to be suitable for small networks and showed scalability problems on big networks as the one of the streets of a city. This problem was solved in (Tatomir and Rothkrantz 2006) where the H-ABC, a scalable ant colony optimization algorithm was presented. Similar to current navigation systems the previous algorithms look only at the past and present and no future is considered. Some ideas of using ABC for travel time prediction are presented in (Ando, Masutani, Sasaki, Iwasaki, Fukazawa, and Honiden 2005) and (Weyns, Holvoet, and Helleboogh 2007).

Many other different traffic prediction approaches can be found in the recent literature. Jagadeesh et al. (Jagadeesh, Srikanthan, and Quek 2002) present an efficient hierarchical routing algorithm that finds a near-optimal route and evaluate it on a large city road network. In (Rice and van Zwet 2004) is described a method to predict the time that will be needed to traverse a given section of a freeway when the departure is at a given time in the future. The prediction is done on the basis of the current traffic situation in combination with historical data. In (Wu, Ho, and Lee 2004) support vector regression (SVR) is applied for travel-time prediction. The authors of (Kim, Lewis, and White 2005a) developed a decision-making procedures for determining the optimal driver attendance time, optimal departure times and optimal routing policies under time-varying traffic flows based on a Markov decision process formulation. The same authors in (Kim, Lewis, and White 2005b) model the dynamic route determination problem as a Markov decision process (MDP) and present procedures for identifying traffic data having no decision-making value.

3 TRAVEL TIME MODELING

We consider a traffic network of highways composed of roads segments and intersections (see Figure 1). The most straightforward approach to model the problem would be to construct a graph. As with other routing problems, Dijkstra's shortest path algorithm could be used to find the shortest path between origin and destination in this graph. However, in the case of dynamic route planning, representing the problem through a graph $G = (N, E)$ is not as straightforward as with a 'normal' routing problem, when the nodes $(N = 1, 2, \ldots, n)$ represent the junctions and the links $(E = (e_j)_{j \leq m})$ represent the roads between them with fixed travel times.

When trying to represent the dynamic route planning problem, two major issues have to be dealt with. Firstly, the dynamic aspect of the data has to be taken into account. The travel time between the different junctions changes in time, and somehow these changes have to be taken into account and incorporated in the graph. When, for example, travelling from Amsterdam to Delft in the morning rush hour, a departure with only 5 minutes later, can affect the travel time by more than 20 minutes, since major congestion may have occurred along the route during these 5 minutes. For example, an accident might have happened or a sudden peak in cars that want to access the highway may have occurred.

To model the dynamic travel time t_{ij} along the road segments we add vertical time axis and discretize the time into intervals $(I_0, I_1, \ldots, I_n)$, where $I_k = [start_k, end_k]$. At each time intervals we make a parallel copy of the (x,y) plane but with different travel times $t_{ij}(I_k)$.

An example of a network constructed this way is shown in Figure 2. The original graph consisting of nodes A to G is repeated for each time interval. The edges between the nodes A to G (the lowest graph at I_0) represent which nodes are connected to each other. For clarity these edges have been kept in the different layers of the graph to show these connections. The dotted links that intersect the different layers are the actual road connections. For each layer and for all nodes, outgoing links are constructed according to the travel time at that moment. It should be noticed that in Figure 2 not all the links are shown, since that would have resulted in a cluttered figure.

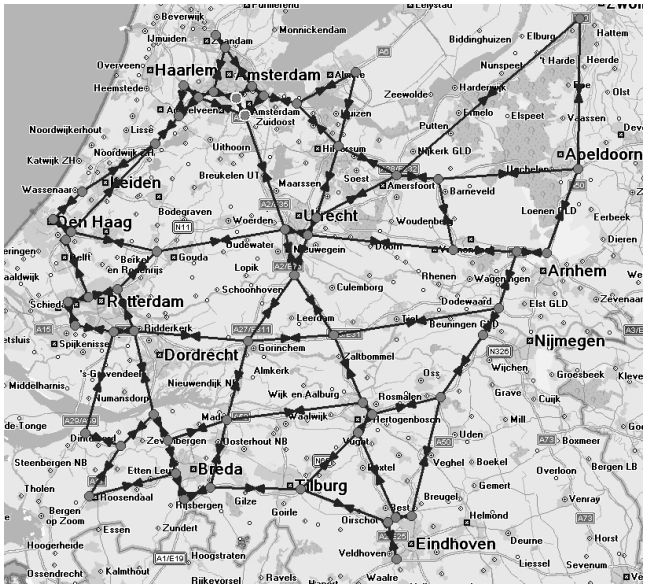

Figure 1: Map of important highways in the Netherlands

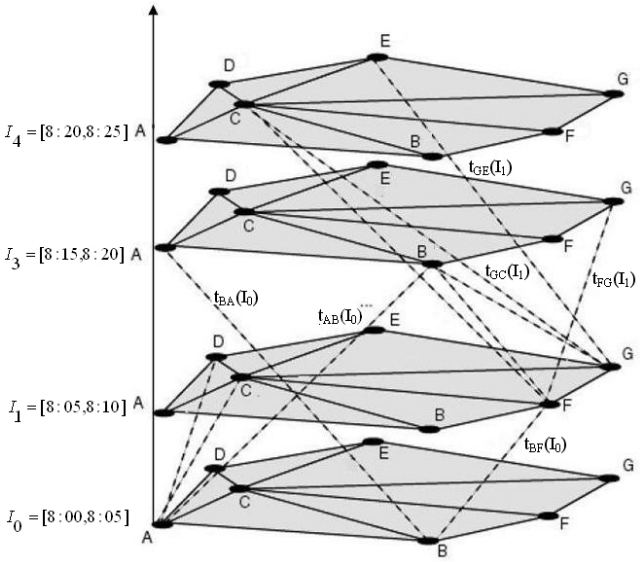

Figure 2: A 3D time dependent model of a road network

3.1 Travel Time Prediction

To predict travel time, historical data can be used, especially when the prediction horizon is moved further away in time. Apparently the load on the freeway network is almost the same on compatible days (same day of the week). We used historical data recorded from the ANWB. In Figure 3 you can see how the actual travelling speed on a road evolved in one morning. Unfortunately, this approach has one considerable drawback: when unexpected events, like accidents, happen a travel time prediction based on historical data will be completely wrong. Somehow, the actual state of the network has to be taken into account, especially when predicting travel times in the near feature. In the following section an Ant Based Control algorithm is used to approximate the travelling speed between two nodes. It will try to predict how many cars are expected to travel between nodes i and j in the time interval I_k. Based on the speed/density relation of the traffic (Figure 4) we can make an estimate of the expected travelling time on that segment during I_k.

We use the following formula to compute the average speed when travelling between A and B in the time interval I_k:

$$v_{AB}(I_k) = \tau h_{AB}(I_k) + (1-\tau) f_{AB}(I_k) \tag{1}$$

where $h_{AB}(I_k)$ is the travel speed based on historical data in the interval I_k and $f_{AB}(I_k)$ is the predicted travel speed. We chose τ=0.5 to balance the past information with the future prediction.

Next we present the algorithm used to compute the travelling time $t_{ij}(I_k)$ between i and j when starting in interval I_k.

3.1.1 *GetTravelTime(i, j, t_i)*

{n - the time intervals horizon}
{i - current node, j - successor node of i}

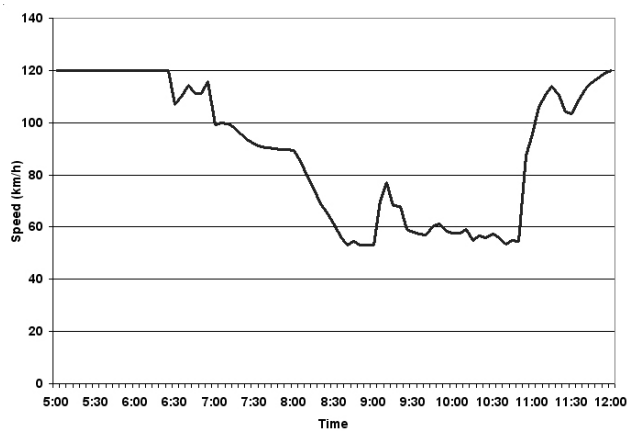

Figure 3: Travel speed between Deil and Tiel as a function of time

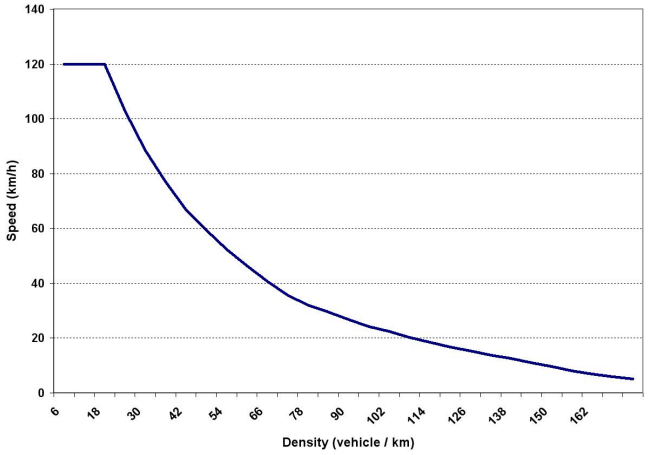

Figure 4: Traffic speed/density relation

$\{t_i$ - departure time from node $i\}$
$L \leftarrow d(i,j)$ $\{d(i,j)$ - distance between i and $j\}$
$t \leftarrow t_i$
$k \leftarrow 0$ {start the iteration with the first time interval of that road}
while $k \leq n$ and $L > 0$ **do**
  if $t \in I_k$ **then**
    $v_{ij}(I_k) \leftarrow \tau(h_{ij}(I_k)) + (1-\tau)(f_{ij}(I_k))$ {compute the average speed} {check if the remained distance can be traversed in this time interval}
    if $L \leq v_{ij}(I_k)(I_k.end - t)$ **then**
      $t \leftarrow t + \frac{L}{v_{ij}(I_k)}$
      $L \leftarrow 0$
    else
      $L \leftarrow L - v_{ij}(I_k)(I_k.end - t)$
      $t \leftarrow I_k.end$
    end if
  end if
  $k \leftarrow k+1$
end while
return $t - t_i$

4 PREDICTIVE ANT BASED CONTROL ALGORITHM

4.1 Predictive Model

We developed a variant of ABC to predict travel speeds. In the classical ABC, the ants were travelling between random generated pairs of source and destination. In our case most of the ants will be created because of the presence of a car on the roads. They will have as source the starting node of the car or, a more probable situation, the next highway intersection the car is driving to. The destination will be the same one as for the related car. Some of the ants which travel between random nodes. This is useful especially at the beginning of the simulation for training the routing tables. Also in case the traffic flow is very low, we want to maintain the routing tables up to date.

The mentioned routing tables are also modified. Every node will keep now multiple layers of probability tables, one for each time interval. $P_{dn}(I_k)$ represents the probability to go to destination d via neighbour n when you are in node i at time $t \in I_k$. Besides probability tables every node i has the following structures:

- $\mu_d(I_k)$: stores the average travel time from node i to destination d when starting at node i in the interval I_k.

- $\delta_n(I_k)$: stores the cars expected to go during I_k to next node n. It contains pairs (x, t_x) where x is the car id plate and t_x is the time stamp when the corresponding ant visited the node. Each time a new ant generated for car x comes to a node, it updates the value of t_x. A timer checks periodically if $\delta_n(I_k)$ is up-to-date and removes the old records (old t_x - no update was received for car x).
- $\rho_n(I_k)$ is the expected traffic density on the road (i,n) during I_k. It is based on the size of $\delta_n(I_k)$ and is used to compute the predicted travel speed $f_{in}(I_k)$.

We are now going to describe some of the features of the dynamic ABC algorithm. Its characteristics make it suitable for both central and distributed implementations.

4.2 Forward Ants

The forward ants behave similar with the forward ants in *AntNet* (Di Caro and Dorigo 1998). They keep a memory about the visited nodes and the estimated time to reach them. As mentioned before they are created periodically for each car in the network. Still, at every source node s in the network the random forward ants $F_{sd}(t_0)$ are generated. Not only their destination d is chosen but also the starting time t_0 is selected as the current time or close in the future (5-15 min away). In this way the ants not only determine the shortest paths for the present but they are training the routing tables in advance for the near future.

At every node i, the selection of the next node n, to move to, is done according to the probabilities P_d. If a cycle is detected ant is deleted.

Lets consider t_{si} the estimated arrival time at node i coming from source s. Once arrived at node i and before going to the next neighbour n, the ant calculates the estimated travelling time between i and n when the departure is in the interval I_k. t_{sn} represents the estimated time necessary for a car to travel from intersection s to intersection n.

$$t_{sn} \leftarrow t_{si} + GetTravelTime(i, n, t_{si}) \tag{2}$$

A forward ant reaches its destination when it arrives at node d. At this moment the ant $F_{sd}(t_{sd})$ finishes its trip. It transfers all of its memory to a new generated backward ant B_{ds} and dies.

4.3 Backward Ants

A backward ant takes the same path as that of its corresponding forward ant, but in the opposite direction. Its role is to update the routing tables along the path based on the travel time estimations collected by the the forward ant. At every step it pops its stack to know the next hop node.

Once arrived at node i the backward ant $B_{ds}(t_{si})$ first selects the time interval $t_{si} \in I_k$ for which the routing table should be updated. The updates are done for all the nodes j between i and d. This is done in 5 steps:

1. Get estimate traveling time from node i to node j when starting in the interval I_k at node i:

$$t_{ij} = t_{sj} - t_{si} \tag{3}$$

2. Update the average travelling time $\mu_j(I_k)$:

$$\mu_j(I_k) = \mu_j(I_k) + \eta(t_{ij} - \mu_j(I_k)) \tag{4}$$

   We used $\eta = 0.1$. It represents the impact a measurement will have in time over the average value.
3. Compute the reinforcement r to be used to update the routing table. This is a function of the time t_{ij} and its mean value $\mu_j(I_k)$.

$$r = \begin{cases} \frac{t_{ij}}{c\mu_j(I_k)};\ \frac{t_{ij}}{c\mu_j(I_k)} < 1,\ c = 1.1 > 1 \\ \\ r = 1 \text{ otherwhise} \end{cases} \tag{5}$$

4. Update the routing table for destintion j:

$$P'_{jn}(I_k) = P_{jn}(I_k) + (1-r)(1-P_{jn}(I_k)) \text{ for the link } i \rightarrow n \tag{6}$$

$$\text{otherwhise } P'_{jl}(I_k) = P_{jl}(I_k) - (1-r)P_{jl}(I_k), \text{ for } l \neq n \tag{7}$$

5. If $r < 1$ and F_{sd} was generated for a car x, update $\delta_n(I_k)$ with the pair (x, t_{now}). If is the first time the car x is expected in node i, a new record is created. Otherwise the old record (x, t_{old}) is replaced by the new one. Be aware that t_{old} and t_{now} are system times and not calculated by ants.

When the source node s is reached again, the ant B_{ds} is deleted.
Next we present the pseudocode of the algorithm.

4.3.1 ABC Algorithm

```
{i - current node, d - destination node, s - source node}
{n - successor node of i, p - predecessor node of i}
for all Nodes do
  if time to generate an ant at node s then
    for all now and next k time intervals do
      Create F_sd(t_k)
    end for
  end if
  for all forward ants F_sd(t_si) received at node i do
    if cycle detected then
      Remove F_sd(t_si)
    else
      if i = d {destination reached} then
        Create B_ds(t_sd)
        p ← GetPrev(F_sd(t_sd)) {select previous node}
        Move B_ds(t_sd) to p
        Remove F_sd(t_sd)
      else
        n ← GetNext(F_sd(t_si))
        t_sn ← t_si + GetTravelTime(i,n,t_si) {compute the travelling time}
        F_sd ← (n, t_sn) {add the new information on the stack}
        Move F_sd(t_sn) to n
      end if
    end if
  end for
  for all backward ants B_ds received at node i do
    UpdateRoutingTables(i, i ← d) {update the node information}
    if d ≠ i {destination not reached} then
      n ← GetNext(B_ds(t_si) {select next node to go}
      Move B_ds(t_sn) to n
    else
      Remove B_ds(t_si)
    end if
  end for
end for
```

4.4 Tests and Results

To test our concepts we modeled a part of the Dutch highway network (see Figure 1). For this we used the Quintiq software which has an integrated GIS system. The network we built consists of 58 nodes (highway intersections) and 84 bidirectional roads. Each road is characterized by the number of lanes and the maximum allowed speed. Some of the real roads are composed of multiple segments with different speed limits and number of lanes. In this cases we choose an average speed limit and a fixed number of lanes. For historical data we used data collected from the ANWB website. Because of the big amount of data, we focused only on the morning period between 5:00 and 12:00. In all the nodes we created routing tables for every 10 min interval.

We populated the roads with random traffic which was generated between the intersections. We used four types of vehicles. 25% of them were guided using the ABC dynamic routing. 25% received traffic update every 30 minutes like the common car navigators are working. Then 25% of the cars received traffic update every 10 minutes like a special traffic information service. The other cars were routed along the shortest paths using Dijkstra's algorithm. By a route we considered any pair of nodes (source, destination). For each of the 3306 possible routes we compared the average travelling time between the four types of vehicles with the departure in the same time interval. Our first question was if the dynamic routing makes sense in general case and not only when accidents happen. In The Netherlands at rush ours almost all the roads show delays and the possible alternatives to a route might also be congested. Also, we were interested if the current systems that get traffic update every 30 minutes are not already efficient. We wanted to know which is the impact of a higher information update frequency and if it is any space left for our algorithm to improve?

In Figure 5 we display for the three 'smart' strategies the improvement compared to the static routes. In average for 50% of the routes, faster alternatives were found by the routing systems. We considered that a route was improved when, at a specific point in time, the selected alternative was faster then the initial route. Also, we recorded only the best score difference obtained during the simulation time. The number of deteriorated routes was very low and was depending of the granularity of the time intervals used by the algorithms.

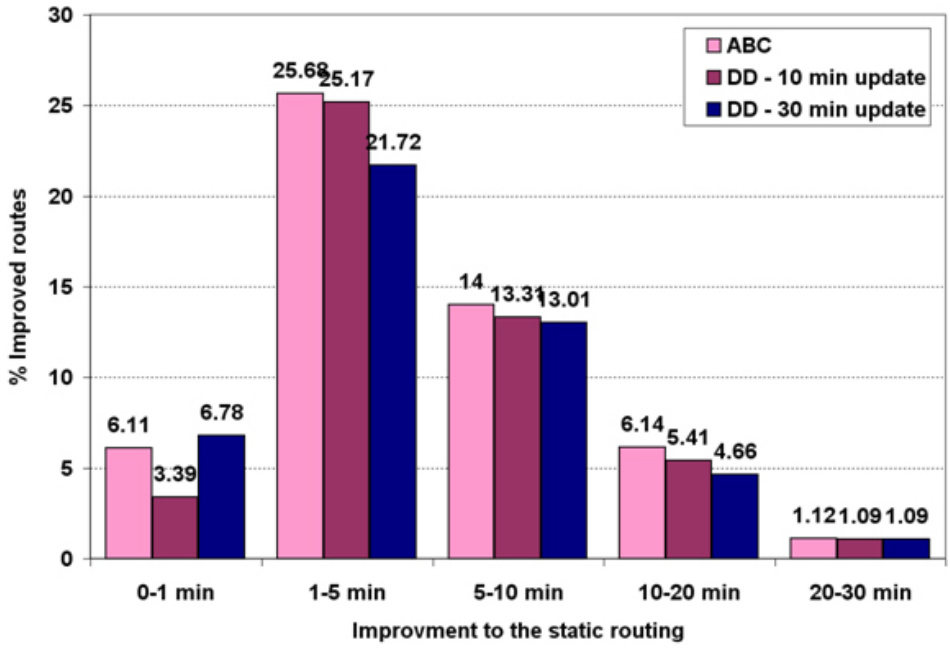

Figure 5: The difference between dynamic and static routes

The best score was obtained using ABC algorithm, with 53% of the routes improved. For 6.11% of the cases, an insignificant improvement of less than one minute, is gained compared to the static path. 26.68% of routes suggested by the ABC were up to 5 minutes faster than the static ones. An improvement of 5 to 10 minutes was noticed in 14% of cases and of 10 to 20 minutes in 6.14% of the situations. For 1.12% of the routes the improvement was up to 30 minutes. If we

consider that The Netherlands is a small country and most of the selected routes are shorter than 150 km, then using the dynamic routing system is a real benefit.

Let us focus now on two particular cases. First the A2 motorway, which connects Eindhoven to Amsterdam. It is the most congested corridor in The Netherlands. Here we selected for a detailed discussion the route between the junctions De Hoght, situated at west of Eindhoven, and Muiderberg, positioned east of Amsterdam (Figure 6). The shortest path has a length of 104 km. There are two main alternatives available. One is to go via Tilburg and follow A27 to Utrecht (130 km). The second, also 130 km long and presented in the figure, is via Nijmegen and Arnhem using A50. For the segment between Utrecht and Amsterdam, the main options are via A2 direction ring Amsterdam, or via A27 and A1.

In Figure 7 is shown how the travelling time between these two locations has been oscillating during the morning. Between 7:40 and 10:40 the ABC algorithm suggested different faster routes than the static one (Dijkstra). Around 10:00 the ABC suggestion via Nijmegen was in average 15 min faster despite the additional 26 kilometers. The results of the guidance using the current traffic updates were somewhere in between. Increasing the frequency of receiving the traffic information from every 30 minutes to every 10 minutes, didn't show much difference. It can also be observed that for a while, around 9:50 and 11:00, these suggestions were actually slower than the static route. This is because, the alternatives were calculated only using the actual state of the congestion, and the future evolution of the traffic jam was not considered. This was not the case for the routes calculated by the ABC algorithm.

Figure 6: The shortest and the fastest route from De Hoght to Muiderberg at 9:30

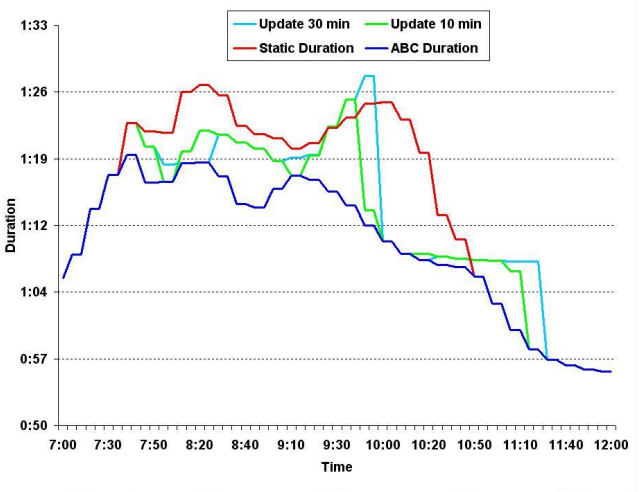

Figure 7: Traveling time from De Hoght to Muiderberg

For the second particular case we chose as starting point the motorway intersection Empel situated on the ring of s'Hertogenbosch not far away from the Quintiq office (Figure 8). The destination is Prince Clausplein intersection. It is situated nord from Delft close to The Hague. The shortest way (82 km) is via Utrecht following A2 and A12. But again, many alternatives are available, like going via Waalwijk on A59 then A16 to Rotterdam or via Gorinchem on 15. Also Rotterdam can be passed via east or western ring. The suggestion in the figure is 98 km long.

On this route the ABC suggested better paths from 7:00 to 9:40 (Figure 9). At 7:50 was recorded the highest difference from the static route: 13 minutes and 39 second. Again we can observe that using the current navigation is not always efficient and sometimes the suggested path proves to be longer in time than the direct route.

5 CONCLUSION

In this paper we discussed the problem of dynamic routing. We adapted the ABC algorithm and we were able to compute the shortest path in time taking into account travel time predictions along road segments. We combined it with historical data

Figure 8: The shortest and the fastest route from Empel to Prins Clausplain at 8:30

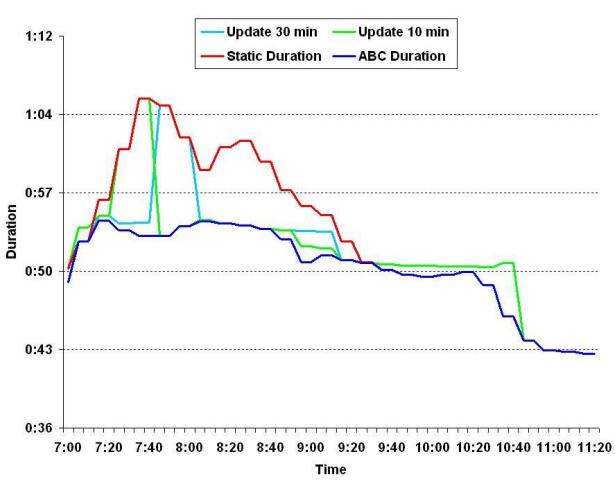

Figure 9: Traveling time from Empel to Prins Clausplain

from ANWB containing speed measurements along the freeways in the Netherlands. In the rush hours, we see a drop of the maximum speed on most of the roads, so our main question was weather we can reduce the travel time by choosing alternative routes, omitting congestions. It was already proved ((Tatomir and Rothkrantz 2004) and (Tatomir and Rothkrantz 2006)) that in case of incidents such as traffic accidents, it makes sense to choose an alternative route. In this paper we demonstrated that dynamic routing results in a reduction of the travel time in general not only in case of accidents. Combining historical information with future prediction, the ABC algorithm is more reliable than just using the current traffic information. In the future we want to analyze how good our travel time prediction is, and to study the influence of different parameters in this feature of the ABC algorithm.

REFERENCES

Ando, Y., O. Masutani, H. Sasaki, H. Iwasaki, Y. Fukazawa, and S. Honiden. 2005. Pheromone model: Application to traffic congestion prediction. In *Engineering Self-Organising Systems*, 182–196.

Chabini, I., and S. Lan. 2002. Adaptations of the a* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. *IEEE Transactions on Intelligent Transportation Systems* 3:60–74.

Di Caro, G., and M. Dorigo. 1998. Antnet: distributed stigmergetic control for communication networks,. *Journal of Articial Intelligence Research (JAIR)* 9:317–365.

Di Caro, G., F. Ducatelle, and L. M. Gambardella. 2005. Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking* 16 (5).

Donati, A., R. Montemanni, N. Casagrande, A. Rizzoli, and L. M. Gambardella. 2008. Time dependent vehicle routine problem with a multi ant colony system. *European Journal of Operational Research* 185(3):1174–1191.

Gunes, M., U. Sorges, and I. Bouazizi. 2002. Ara - the ant-colony based routing algorithm for manets. In *Proceedings of the 2002 International Conference on Parallel Processing Workshops*.

Ichoua, S., M. Gendreau, and J. Potvin. 2003. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research* (144): 379–396.

Jagadeesh, G., T. Srikanthan, and K. H. Quek. 2002. Heuristic techniques for accelerating hierarchical routing on road networks. *IEEE Transactions on Intelligent Transportation Systems* 3 (4): 301–309.

Kim, S., M. Lewis, and C. White. 2005a. Optimal vehicle routing with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems* 6 (2): 178–188.

Kim, S., M. Lewis, and C. White. 2005b. State space reduction for nonstationary stochastic shortest path problems with real-time trac information,. *IEEE Transactions on Intelligent Transportation Systems* 6 (3): 273–284.

Rice, J., and E. van Zwet. 2004. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems* 5 (3): 200–207.

Schoonderwoerd, R., O. Holland, J. Bruten, and L.J.M.Rothkrantz. 1996. Ant-based load balancing in telecommunications networks. *Adaptive Behavior* 2:169–207.

Tatomir, B., and L. Rothkrantz. 2004. Dynamic traffic routing using ant based control. *International Conference on Systems, Man and Cybernetics IEEE SMC*:3970–3975.

Tatomir, B., and L. Rothkrantz. 2006. Hierarchical routing in traffic using swarm intelligence. *The 9th International IEEE Conference on Intelligent Transportation systems*:228–235.

Weyns, D., T. Holvoet, and A. Helleboogh. 2007. Anticipatory vehicle routing using delegate multi-agent systems. *The 10th International IEEE Conference on Intelligent Transportation systems*:87–93.

Wu, C. H., J. Ho, and D. Lee. 2004. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems* 5 (4): 276–281.

AUTHOR BIOGRAPHIES

BOGDAN TATOMIR is a PhD candidate in computer science at the Department of Mediamatica at the Delft University of Technology. Since 2007 he is working as a consultant in the Logistics department at Quintiq, a Dutch company that provides advanced planning and scheduling solutions. His research interests are in the areas of dynamic routing and ant colony optimization. His email address for these proceedings is <bogdan.tatomir@quintiq.com>.

LEON J.M. ROTHKRANTZ is an Associate Professor in the Man-Machine-Interaction Group at Delft University of Technology since 1992, and a Professor at Netherlands Defence Academy since 2008. His current research focuses on crisis management, dynamic routing and multimodal communication including: speech recognition, speech generation, facial expression analysis and synthesis, natural dialogue management and human affective feedback recognition. His email address for these proceedings is <l.j.m.rothkrantz@tudelft.nl>.

ADRIANA C. SUSON is a master student with a Shell scholarship at the Department of Mediamatica at Delft University of Technology. She received the BSc. degree in computer science from the Faculty of Mathematics and Computer Science at Transilvania University of Brasov, Romania, in 2008. Her email address for these proceedings is <ac.suson@student.tudelft.nl>.