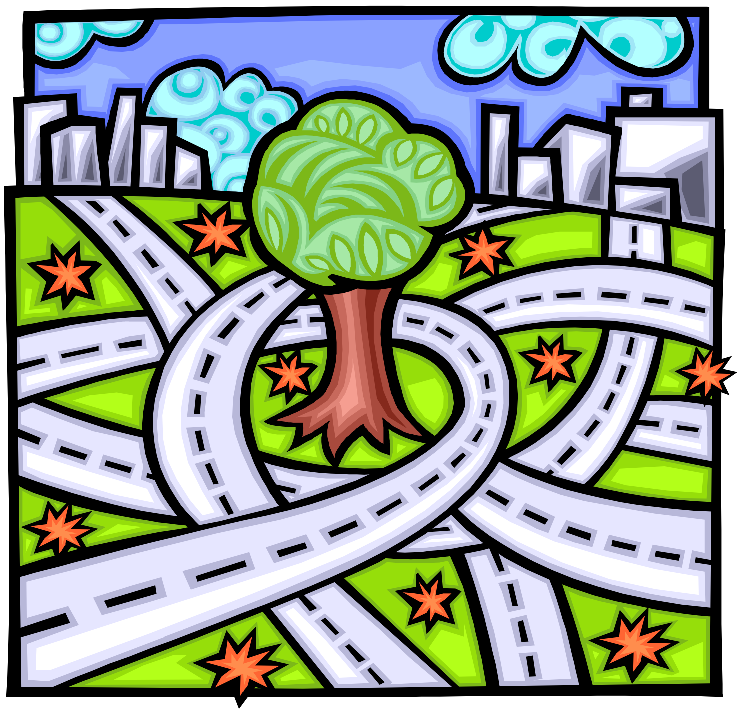
Delft University of Technology

A Simulation System for Hierarchical Routing Using Ant Based Control

By **Rui Li**



Supervisor: Dr. Drs. L.J.M. Rothkrantz
Delft University of Technology, the Netherlands
Faculty of Information Technology and Systems
Knowledge Based Systems Group

Graduation committee:

Dr. Drs. L.J.M. Rothkrantz	*(Delft University of Technology)*
Dr. Kees van der Meer	*(Delft University of Technology)*
Prof. Dr. Ir. E.J.H. Kerckhoffs	*(Delft University of Technology)*
Prof. Dr. H. Koppelaar	*(Delft University of Technology)*

Rui Li (superlr2000@hotmail.com)

Master's thesis, July 2004
A simulation system for hierarchical routing using Ant Based Control

Delft University of Technology, the Netherlands
Faculty of Information Technology and Systems
Knowledge Based Systems Group

Keywords: Static routing system, Dynamic routing system, Intelligent agents, Navigation system, Ant Based Control, Hierarchical topology, Hierarchical routing system.

Acknowledgements

I am very lucky to have such a great opportunity to finish my master study in Delft University of Technology (TU Delft), especially to do my master's thesis at the excellent group -- Knowledge Based Systems Group. The people in this group are so friendly and accommodating. Whenever I encountered a difficulty, I could always turn to them for help. Without their help, it would be much harder to finish my thesis work.

First of all, I would like to express profound gratitude to my master thesis supervisor, Dr. Drs. L.J.M Rothkrantz, for his careful supervision, useful advices throughout this master research work. In my opinion, he is very nice person and well-credited scientist. I really enjoyed our weekly discussions that were both comfortable and inspiring. His earnest and patient guidance enabled me to complete my work successfully.

I am deeply grateful to B. Tatomir, an excellent PhD student in KBS group. He gave me lot of valuable suggestions for this research. And he also gave me lot of help on Delphi programming.

Finally I would like to thank my parents and friends for their love and support throughout my life. I will never forget this precious period in Delft.

Summary

In modern society, traffic congestion is a tough problem for many countries all over the world. Governments employ many kinds of approaches to mitigate the traffic jam. They build broader roads, establish more reasonable traffic rules and engage more managers.

Dynamic routing is an efficient way to reduce traffic congestions. Based on different routing problems, variant routing algorithms are developed. The Ant Based Control algorithm (ABC-algorithm) is a promising approach among them. This algorithm is developed from the principles that ants use to find food in the nature, and it is especially suited to find solutions to difficult discrete optimization problems where the dynamic data changes very fast. It introduces intelligent agents (artificial ants) to explore the traffic network and find the optimal route in time.

But when traffic networks become more and more complex, the Ant Based Control routing shows poor performance. To improve the efficiency of a routing system based on ABC-algorithm, we propose hierarchical routing. The hierarchical routing is mainly devised to reduce memory requirements of simulations over very large topologies. In this approach, a complex traffic network is broken down into several layers of networks: one abstract level network and several detailed level networks. The hierarchical routing system therefore consists of some distributed routing systems where each of them is responsible for one network of the hierarchical network. And each distributed routing system needs less information to perform routing.

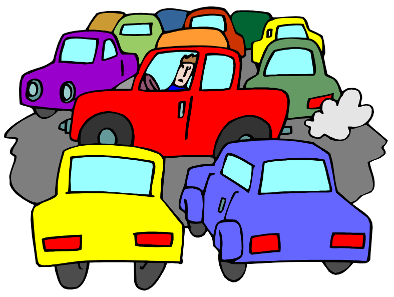
We develop a software prototype – a hierarchical routing simulation system. The system is based on the Ant Based Control algorithm. We deliver some experiments on the system, and get excellent results presenting high stability, efficiency and robustness. Thus we conclude that the system is capable of routing vehicles in real time, even in very complex traffic network.

TABLE OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

Chapter 1 Introduction



1.1 Problem setting

Traffic Congestion has become part of daily life in many cities as traffic continues to increase on a relatively unchanging traffic city network. Reasons for congestion could be crashes, atrocious weather, emergency events and other temporary disruptions to the transportation system. To people, congestion means loss of time, missing of opportunities, and waste of personal resources. To the society, congestion means lower work productivity, delivery delays, and increased investment. Many kinds of approaches are adopted by the government to mitigate the traffic congestion: building more roads, making more reasonable transportation rules, arranging more transportation managers. But for a variety of political, financial, and environmental reasons, some of them don't work, such as building more roads. Normally roads building need large sums of money. Furthermore growing rate of roads doesn't come up with the amazing increasing amount of vehicles in modern cities. Like Chinese capital Beijing which is confronted with prominent traffic congestion problem with automobiles increasing by 10 percent annually and roads being extended by only 2 percent.

Figure 1: Traffic congestion

How to relief the traffic congestion based on current transportation situation without vast money costing and numerous manpower requiring? Since one of the reasons for congestion happening is that road capacity is not optimally used. Optimal routing system may direct government to the right way for problem resolving. By collecting all route information along the road, people can exploit the traffic network capacity in an optimal way. Then vehicle could avoid congested roads and choose alternative ways. If most vehicle drivers can choose their way by using such kind of optimal routing system, the traffic congestion will be reduced to some degree.

1.2 Routing system

Nowadays, many companies offer navigation service. According to the input data, navigation system can be divided into two categories: static and dynamic. In a static environment the travel time along all traffic links of a network are fixed and do not change. The shortest routes for all pairs of nodes need to be computed only once and need not be updated, because the travel time does not change. On the contrary, in a dynamic environment, the travel time of the links changes over time. And then it is impossible to compute the shortest routes only once because of travel time updating. The detailed information about these two types' routing systems will be given in following sections.

1.2.1 Static routing system

Static routing systems are independent of the current state of the users and the transportation network. Static routing is based on expected rather than actual roads state. They are widely used by many companies nowadays. But the obvious drawback of such routing is that they can not avoid routing a car into congestion on the road network because of lacking real time information. This kind of routing system can only be considered as assistant for finding the way between source and destination, but not to find the most optimal way in time. Dijkstra's algorithm is a centralized routing algorithm for such static environment with guaranteed shortest paths. Centralized router means that all data is available at one location. So it is designed for non-distributed networks and is not suitable for routing in distributed networks.

The ANWB navigation system belongs to static routing systems. It can provide users routes based on static information and additional it can inform about some possible congestions. The following figure show the service provided by ANWB.

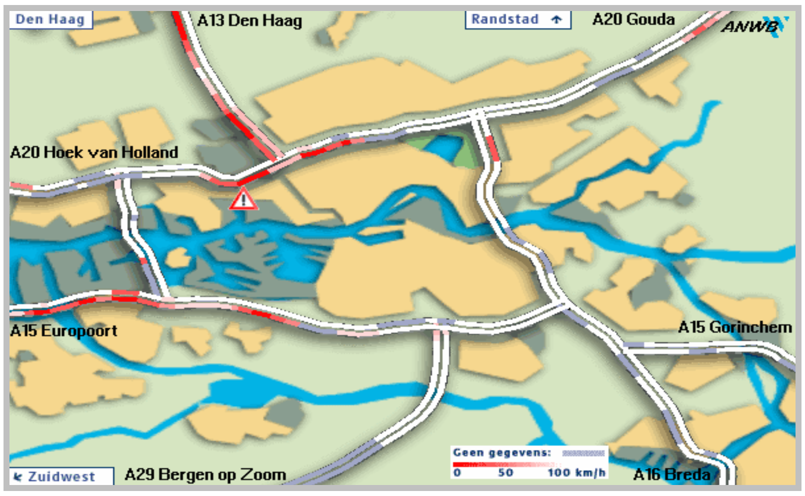


Figure 2: Traffic information provided by ANWB

1.2.2 Dynamic routing system

Dynamic Routing systems select routes based on the current state information for the transportation network. The state information can be predicted or measured but the route will change depending on the available state information at the time of the routing request. The dynamic routing could be illustrated with 3-D picture in figure-3. This picture displays four layers at four different times with an interval of one minute between neighbored layers. The dotted links that intersect the different layers are the actual travel times to go from the node of the lower layer, where the link starts, to the node of the upper layer, where the node ends. The travel time of source and destination nodes change over time. In practice, dynamic information about the traffic network is collected through three ways: traffic monitoring systems, emergency services and motorists' calls.

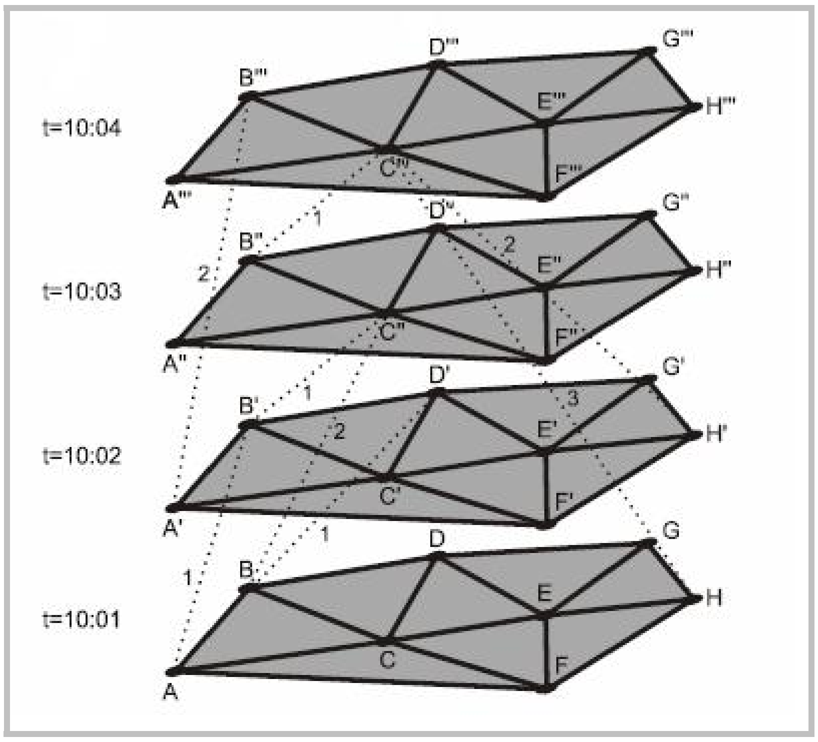


Figure 3: 3-D picture of the dynamic routing

The Tegaron Scout from the Germany Company Tegaron Telemetric GmbH and VDO Dayton MS 5500 SD are applicable navigation routing systems that use dynamic data. According to the users' request, the navigation system computes the best route and sends it back. Such a routing system always provides the route with the shortest travel time and advises the optimal route when faster roads are available.

Figure 4: VDO Dayton MS 5500 SD

Figure 5: Tegaron Scout

1.3 Hierarchical routing system

One ideal routing system must give optimal result as soon as possible. In practice, the traffic network could be very large and complex. So the amount of information that must be propagated increases and the routing calculation becomes increasingly expensive. Hierarchical routing is an approach that hides information from far-away nodes, reducing the amount of information a given router needs to perform routing. Because our hierarchical routing is based on Ant Based Control algorithm, let's review this algorithm firstly.

1.3.1 Ant Based Control routing

In dynamical routing system, Ant Based Control algorithm (ABC-algorithm) is a promising approach. It is proposed by [Schoonderwoerd] for routing and load balancing in circuit switched telecommunications networks. This algorithm developed from the principles that ants use to find food in nature. And it is especially suited to find solutions to difficult discrete optimization problems where the dynamic data changes very fast. More about the ABC-algorithm is explained in chapter 2.

Based on ABC-algorithm, traffic simulation of dynamical routing system has been developed by [Kroon]. This simulation is made up with several subsystems: control center, timetable updating system, route finding system.

- The control center is responsible for parameters setting, traffic network loading and cooperation between traffic network and routing system.
- Timetable updating system receives the dynamic information about traffic network in the city from different sources. This dynamic information could be travel time of the covered route by vehicle, congestion of the certain road, diversions of the road and roadblocks.
- Route finding system is the central part of the simulation. It is responsible for the route request from the individual vehicle. Based on the situation of the traffic network, routing system computes the optimal way from source to the destination.

1.3.2 Hierarchical routing system

Current dynamical routing systems, like ABC-routing system, are only suitable for small city network. They can not deal with more complicated situations: larger city network and connective city networks through motorways. The hierarchical routing reduces the amount of information a route request needs to compute by dividing the complicated networks into several levels. The advantage of hierarchical routing is that each distributed routing system needs less information to perform routing. According to the hierarchical routing theory, one complicated problem could be split up into some simpler and easier ones. Connective city networks can be separated into some independent networks connected by motorways. In a similar way, one larger, complicated city network, which contains main streets, could be split up into several smaller networks connected by these main streets.

Then, how the hierarchical routing system computes the route from source within one city network to a destination located in another city network? The solution is derived from the normal human’s choose procedure. The usual way for people doing this is to find a route, which is the fastest and easiest route from the environs of the source city to the environs of the destination city. To resolve this problem, routing system needs two levels traffic network, abstract level and detailed level. Abstract level consist only cities on the road map and the motorways between them. It is responsible for computing the route from source city to the destination city through the motorway. Detailed level contains streets located in source and destination cities. Using this, routing system could compute the route within the city.

1.4 Project description

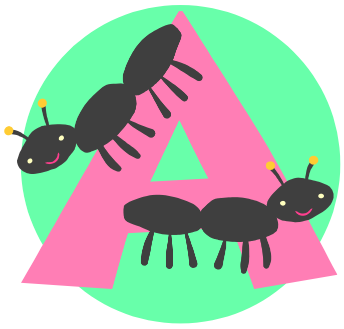
The different phases done in this project are enumerated below.

1. Study of dynamic routing systems and routing algorithms.
2. Study of traffic simulation.
3. Study and design of a model for hierarchical traffic network.
4. Study and design of a model for a hierarchical routing system.
5. Implementation of hierarchical traffic simulation.
6. Testing the hierarchical traffic simulation.
7. Implementation of hierarchical routing system based on ABC-algorithm.
8. Testing the hierarchical routing system.
9. Experimenting with the routing system.

1.5 Report structure

This thesis has the following structure. Chapter 2 gives some background information about the Ant Based Control routing system and the hierarchical routing system. Chapter 3 the Ant Based Control design and results of hierarchical routing design are presented, including description on the hierarchical timetable updating system and routing finding system in detail. Chapter 4 shows technical details of the implementation. Chapter 5 presents some experiments designed for hierarchical routing system comparing with Ant Based Control routing methods. In chapter 6 the conclusions from the experimental results are presented and some recommendations are provided.

Chapter 2 Theories



2.1 Swarm intelligence

Insects that live in colonies, such as ants, bees and wasps, have fascinated scientists for a long time. Each insect in a colony seems to have its own agenda, and yet the group as a whole appears to be highly organized. Apparently the seamless integration of all individual activities does not require any supervision. In fact, scientists who study the behavior of social insects have found that cooperation at the colony level is largely self-organized: in numerous situations the coordination arises from interactions among individuals. Although these interactions might be simple (one ant merely following the trail left by another), together they can solve difficult problems (finding the shortest route among countless possible paths to a food source). This collective behavior that emerges from a group of social insects brought new scientific area -- swarm intelligence. Swarm Intelligence [SI] is the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge. SI provides a basis with which it is possible to explore collective (or distributed) problem solving without centralized control or the provision of a global model.

2.2 Ant Colony Optimization

Based on SI research, study on the foraging of ants has led to a novel method for rerouting network traffic in busy telecommunications systems or traffic transportation systems. In nature, real ants are capable of finding shortest path from a food source to the nest without using visual cues. Also, they are capable of adapting to changes in the environment, for example finding a new shortest path once the old one is no longer feasible due to a new obstacle. It is well-known that the main means used by ants to form and maintain the line is a pheromone trail. Ants deposit a certain amount of pheromone while walking, and each ant probabilistically prefers to follow a direction rich in pheromone rather than a poorer one. This elementary behavior of real ants explains how ants can find the shortest path from nest to the food source.

The following picture will help us to understand the foraging behavior of ants and how they can manage to find the shortest path between the nest and a food source using simple local decisions. At the beginning, showing in figure-6, there is no pheromone on route A and route B, so ants will choose the road randomly. In this situation, one ant chooses route A, the other ant chooses route B. On the way to the food, ants will lay a pheromone trail, which diffuses slowly. Because the route B is shorter than A, the ant choosing B will reach the food in a short time than another ant along A. Figure-7 illustrates such situation. When an ant finds food, it will return to nest with this information. The ant on route B will follow the original way and go back to the nest. Obviously, the ant on route B should return to the nest earlier than ant on the route A. At this moment, the third ant will leave for the food; it will choose the route based on the intensity of pheromone. The route B will be taken because of the stronger pheromone (passed twice by one ant). This situation is shown by figure-8.

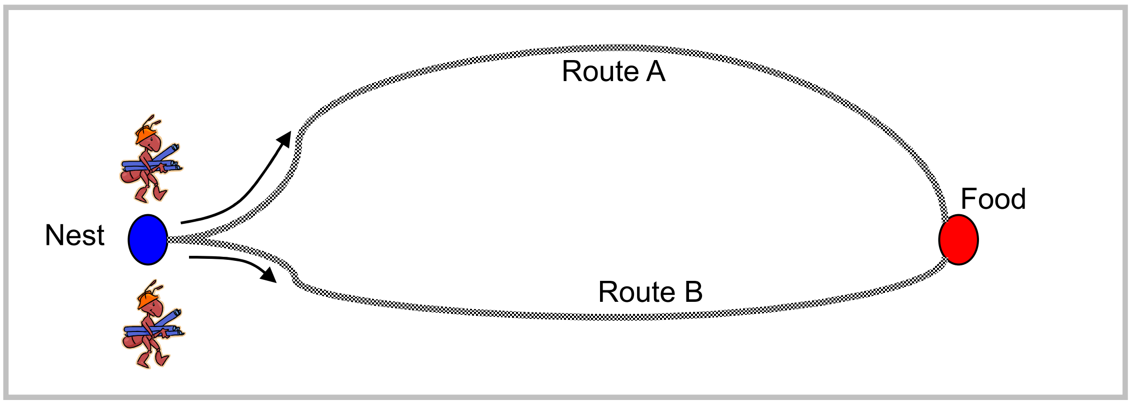


Figure 6: Ants choose route randomly at beginning

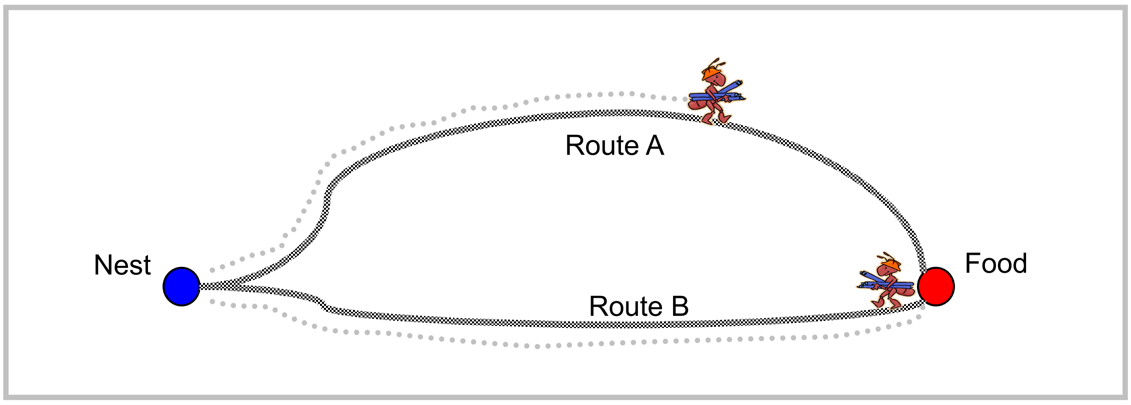


Figure 7: An ant found food and returned to the nest

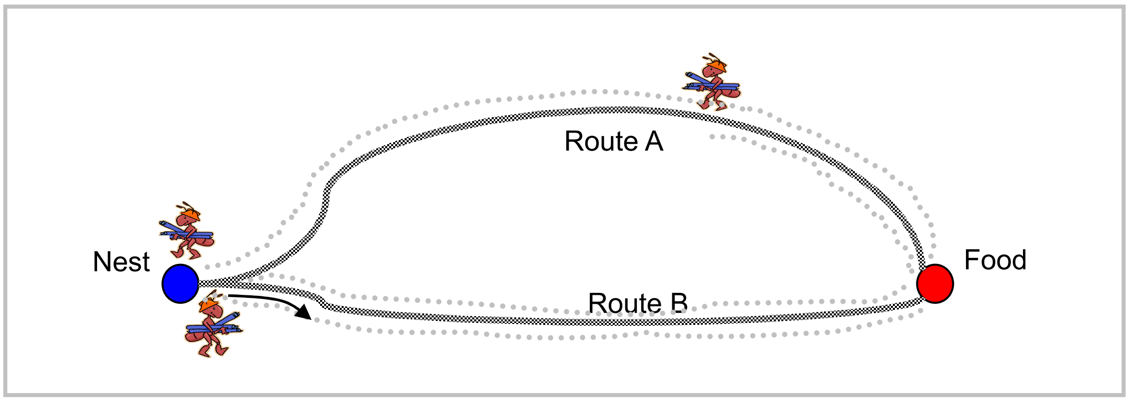


Figure 8: The third ant chooses route based on pheromone intensity

Consequently, more and more ants will choose the route B, and pheromone intensity will become stronger and stronger. Figure-9 shows this situation clearly.

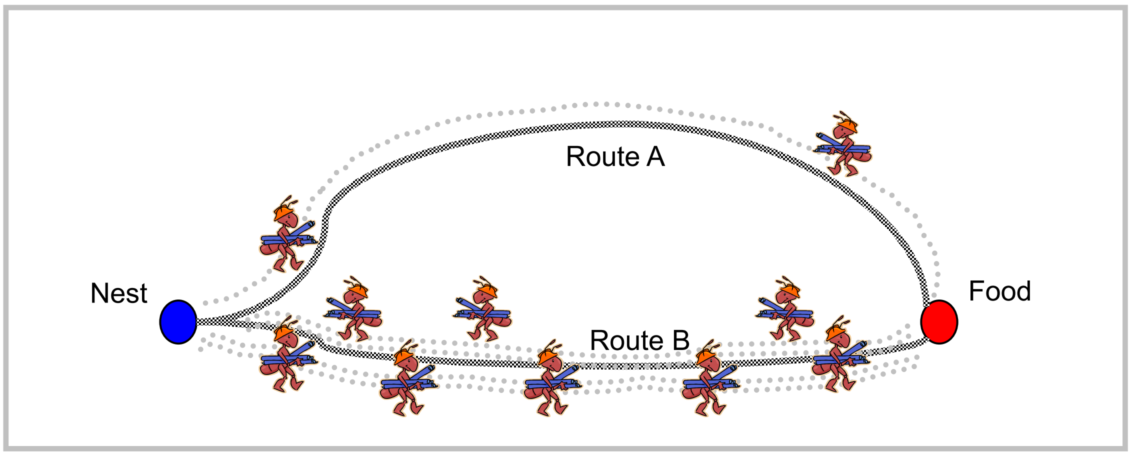


Figure 9: The route situation after some time

In practice, the environment of the route will change in real time. For example, route B is broken because of some unexpected reasons. According to the discussion above, most ants will choose a route based on the strength of the pheromone. In such kind of situation, how ants find another optimal way? Fortunately, the pheromone will evaporate as time goes on, and some ants will choose routes besides B by chance. These ants will be responsible for the exploration of other routes. It makes sure that a colony of ants will always find an optimal route from nest to food source in a changing environment. Following figure-10 shows such situation. When route B is broken, ants colony have chance to find alternative route C to the food.

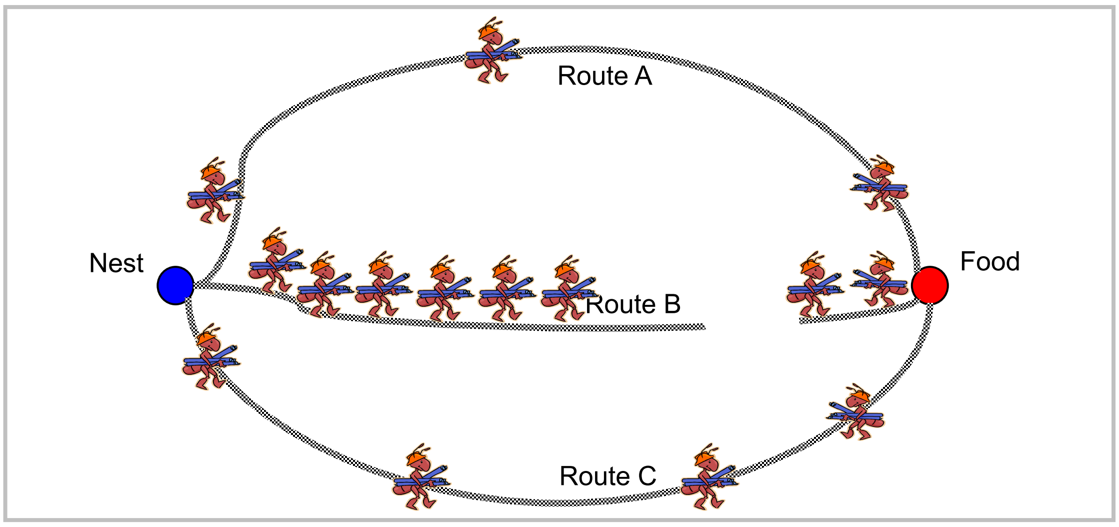


Figure 10: Exploring other routes when environment change

2.2.1 Ant Colony Optimization algorithms

Based on studies in SI, especially in ant colony optimization, several efficient algorithms are developed for different problems. These problems could be distinguished into two classes: one is static combinatorial optimization problems, and the other is dynamic ones. Static problems are those in which the characteristics of the problem are given once and for all when the problem is defined, and do not change while the problem is being solved. A example of such problems is the classic traveling salesman problem, in which city locations and their relative distances are part of the problem definition and do not change at run-time. On the contrary, dynamic problems are defined as a function of some quantities whose value is set by the dynamics of an underlying system. The problem changes therefore at run-time and the optimization algorithm must be capable of adapting online to the changing environment. The example is network routing. Table-1 lists the available implementations of ant colony optimization algorithms.

Table 1: ACO algorithms and applications

Problem name		Authors	Year	Algorithm name
Static Problems	Traveling salesman	Dorigo, Maniezzo & Colorni Gambardella & Dorigo Dorigo & Gambardella Stutzle & Hoos Bullnheimer Hartl & Strauss	1991 1995 1996 1997 1997	AS Ant-Q ACS & ACS-3-opt *MM*AS AS*rank*
	Quadratic assignment	Maniezzo, Colorni & Dorigo Gambardella, Taillard & Dorigo Stutzle & Hoos Maniezzo & Colorni Maniezzo	1994 1997 1998 1998 1998	AS-QAP HAS-QAP*a* *MM*AS-QAP AS-QAP*b* ANTS-QAP
	Job-shop Scheduling	Colorni, Dorigo & Maniezzo	1994	AS-JSP
	Vehicle routing	Bullnheimer, Hartl & Strauss Gambardella, Taillard & Agazzi	1996 1999	AS-VRP HAS-VRP
	Sequential ordering	Gambardella & Dorigo	1997	HAS-SOP
	Graph coloring	Costa & Hertz	1997	ANTCOL
	Shortest common super sequence	Michel & Middendorf	1998	AS-SCS
Dynamic Problems	Connection-oriented network routing	Schoonderwoerd, Holland, Bruten & Rothkrantz White, Pagurek & Oppacher Di Caro & Dorigo Bonabeau, Henaux, Gu´erin, Snyers, Kuntz & Th´eraulaz	1996 1998 1998 1998	ABC ASGA AntNet-FS ABC-smart ants
	Connection-less network routing	Di Caro & Dorigo Subramanian, Druschel & Chen Heusse, Gu´erin, Snyers & Kuntz van der Put & Rothkrantz	1997 1997 1998 1998	AntNet & AntNet-FA Regular ants CAF ABC-backward

The following section will explain how to design a real system in order to imitate the collective behavior of the ants. Cooperation is the most important component for designing such a system. One choice is to assign the computational resources to a set of artificial ants that communicate indirectly by stigmergy. Artificial ants abstract some similarities from the real ants in nature. Additional, to make system more efficient artificial ants are own some unique attributes different with real ants.

- ***Similarities with real ants:***

  a) ***Colony of cooperating individuals.*** Similar with real ant colonies, ACO algorithms are composed of a population of concurrent and asynchronous entities globally cooperating to find a good solution to the task under consideration.
  b) ***Pheromone trail and stigmergy.*** Artificial ants change some numeric information - artificial pheromone trail - locally stored in the problem's state they visited. This information denotes the performance of the ant in past or at present. And this information could be accessed by any other ant. The evaporation mechanism ensures that artificial ants have chance to search towards new direction without being over-constrained by past information.
  c) ***Shortest path searching and local moves.*** Artificial ants move step-by-step from the original state to the destination state through 'neighboring state'.
  d) ***Stochastic and myopic state transition policy.*** Artificial ants' police make use of local information only and they don't predict future states.

- ***Unique attributes different with real ants:***

  a) Artificial ants move in the discrete environment. That is, from one discrete state to another discrete state.
  b) Artificial ants have an internal state; with it they could remember the past actions.
  c) Artificial ants deposit an amount of pheromone according to the quality of the solution found.
  d) Artificial ants timing in pheromone laying is the problem dependent.

2.2.2 Ant Based Control algorithm

2.2.2.1 Basic idea

Ant based control algorithm is proposed by [Schoonderwoerd] for routing and load balancing in circuit switched telecommunications networks. This kind of network is modeled by a graph *G = (N, A)*, where each node *i* has the same functionalities as crossbar switch limited connectivity (capacity) and links have infinite capacity (that is, they can carry a potentially infinite number of connections). The basic idea about the algorithm is the following. With algorithm running, artificial ants are continuously generated at any node in the network and are assigned random destination nodes. On their way to the respective destination node ants move around in the network and lay their "pheromone trails". They do this by updating the so-called in routing tables the routers (pheromone tables). Every node has a route table for every possible destination in the network and the destinations' neighbor nodes. Thus a node with *k* neighbors in a network with *n* nodes has a route table with *(n-1)* rows, where each row corresponds to a destination node, and has *k* entries where each entry corresponds to a neighbor node. The route table contains probabilities (representing the strength of pheromone), which get regularly updated as soon as an ant reaches a node. Updating the probabilities thus represents pheromone laying.

In this project, the ABC algorithm will be used for the real traffic network routing. The traffic network is made up with roads and intersections, and it could be represented by a directed graph (as figure-11 shows).

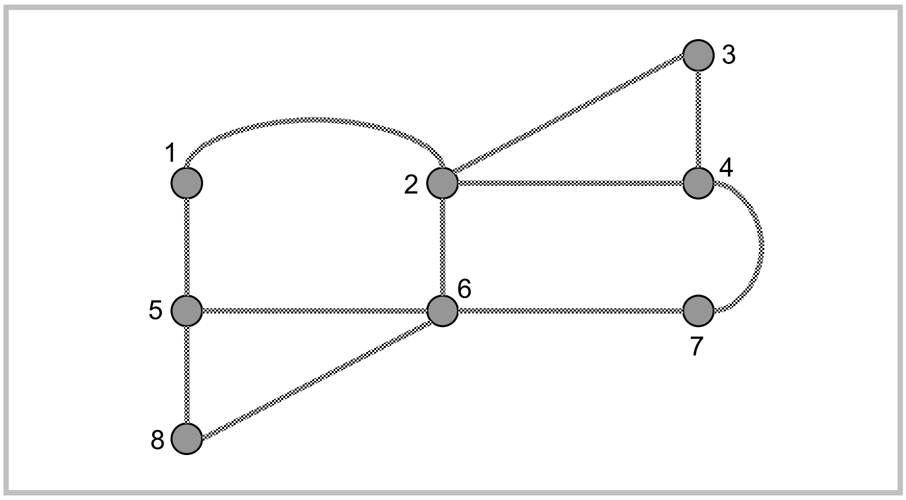


Figure 11: Real traffic network

Based on the description above, every node in this directed graph will have one route table with 7 rows, where each row corresponds to a destination node except for itself. The number of entries for every node will be different according to the situation of neighboring nodes. For example, the route table of node 2 will contain 7 rows and 4 entries; and the route table of node 7 will contain 7 rows and 2 entries. The table-2 will show the detailed structure of the route table for node 6.

Table 2: Route table of intersection 6

	Next / Dest	Neighbor nodes (Entries)				SUM
		Node 2	Node 5	Node 7	Node 8	
Destination nodes (Rows)	Node 1	0.0994023	0.8522169	0.0241903	0.0241903	1
	Node 2	0.9302326	0.0232558	0.0232558	0.0232558	1
	Node 3	0.9302326	0.0232558	0.0232558	0.0232558	1
	Node 4	0.0232558	0.0232558	0.9302326	0.0232558	1
	Node 5	0.0232558	0.9302326	0.0232558	0.0232558	1
	Node 7	0.0232558	0.0232558	0.9302326	0.0232558	1
	Node 8	0.0232558	0.0232558	0.0232558	0.9302325	1

From table-2, we can see the detailed structure of route table for intersection 6. This route table shows that, if a vehicle request route from intersection 6 to intersection 3, according to the probability value (probability with underline) in the route table it will choose neighbor node 2 to move. As artificial ants at every step have good recent information about their trip from the source node to their current node, the entries in the route tables are updated referring to the source node. Thus ants directly influence those ants traveling towards their source node and only indirectly those traveling in their same direction.

- *Probability updating*

The route table plays the very important role in the ABC algorithm; it let ants communicate with each other. Now we will explain how the probability *P* in the route table is updated. The entry corresponding to the node from which the ant just came is increased, and at the same time, all other entries are decreased to guarantee the sum of every column equal to 1(see the right sum column in table-2) . Following formulas will be used to compute the new probability in the route table. We will give detailed description about these formulas in following section.

$$P_{new} = \frac{P_{old} + \Delta P}{1 + \Delta P} \qquad (1)$$

$$P'_{new} = \frac{P_{old}}{1 + \Delta P} \qquad (2)$$

$$\Delta P = \frac{A}{t} + B \qquad (3)$$

- P_{new} and P'_{new} are the new probability.
- P^{old} is the old probability in the routing table.
- ΔP is the probability increase.

2.2.2.2 Algorithm

The ABC-algorithm is derived from the pheromone following of ants that optimize the route to a food source. The natural stigmergy pheromone is replaced by artificial stigmergy which can be modeled by computers. For the ABC-algorithm, the artificial stigmergy is realized by route tables. The detailed information about ABC-algorithm will be described in the following section.

- ***Intelligent agents (artificial ants)***

In computer system, intelligent agents are introduced to replace the natural ants. To compute and update the probabilities in the route table, ABC-algorithm develops two kinds of intelligent agents: the forward agents and the backward agents. All forward agents have the same structure and all backward agents have the same structure. These agents move inside the traffic network by hopping at every time step from a node to the next node along the existing links. And they move much faster than the vehicles could do in the real traffic network. They communicate with each other in an indirect way by concurrently reading and writing the route tables on the way. They receive percepts from the environment and they can do certain actions. Condition-action rules are used to define what action should be taken under certain situation. [Henrik Dibowski] gives the PAGE description of both agent types using table-3.

Table 3: PAGE description of forward and backward agents

Agent Type	Percepts	Actions	Goals	Environment
Forward agent	▪ ID of current node ▪ IDs of the neighbored nodes and links that lead to them ▪ routing table of current node	▪ update memory ▪ remove cycle from memory ▪ determine and go to next node ▪ transform to backward agent	▪ go to destination node ▪ store the route and the travel times	network consisting of nodes and directed, weighted links connecting the nodes, only local information available at every node (routing tables)
Backward agent	▪ ID of current node ▪ IDs of the neighbored nodes and links that lead to them	▪ update routing table ▪ go to next node ▪ kill the agent	▪ go back to the source node along the stored route of the forward agent ▪ update the routing tables	Similar with forward agent

Table 4: Condition-action rules of the agents

Agent Type	Condition-action rules
Forward agent	▪ IF current node already exists in memory THEN remove cycle from memory ▪ IF current node ≠destination node THEN update memory AND determine and go to next node ▪ IF current node = destination node THEN update memory AND transform to backward agent
Backward agent	▪ IF current node ≠source node THEN update routing table AND go to next node ▪ IF current node = source node THEN update routing table AND kill the agent

- ***Algorithm description***

The Ant Based Control 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 could be described as follows:

I. At regular time intervals, a forward agent F_{sd} is launched from every node s of the network with a random destination node d. The task of the forward agent is to discover an optimal route from the source node s to the destination node d.

II. At every visited node k on the way to the destination node, a forward agent executes following actions:
   a) Check its memory whether node k has already been visited. If visited then a cycle in this forward agent's route exists and this cycle is deleted from the memory.
   b) Update its memory through adding a new (k, t_k) pair to the memory. k is identifier of the visited node and t_k is the time it will take a vehicle to travel the link from last visited node to this node under the current traffic situation.
   c) If node k is not the destination node then the forward agent determines the next node to go by using the probabilities in the row of the route table of node k which represents the destination node d. A random number between 0 and 1 is generated for every link to a neighbored node according to the magnitude of the probabilities. The node where the agent just came from is filtered out to avoid that the agent directly goes back to that node. Also links that are disabled are filtered out. With the generated probabilities the next link is randomly selected. Then a copy of the remaining probabilities is made for this agent and these probabilities are normalized to 1. The forward agent goes to the next node along that link.
   d) If node k is the destination node then the forward agent F_{sd} generates a backward B_{ds} . The forward agent F_{sd} transfers all its memory to the backward agent B_{ds} and then destroys itself. As the result, the backward agent inherits the whole route information from the forward agent. The task of the backward agent is to go back to the source node s along the same path as the forward agent but in the opposite direction and to update the route tables on this path.

III. At every visited node k on the way back to the source node a backward agent B_{ds} execute the following:
   a) Update the route table of node k by using the travel times stored in its memory.
   b) If node k is not the source node then backward agent uses its memory to determine the next link of the path back to the source node. The backward agent goes to the next node along that link.
   c) If node k is the source node then the backward agent is killed.

- ***Route table updating***

At every visited node k on the way back to the source node s, a backward agent updates some of the probabilities in the routing table of node k by using the travel information in its memory which was collected and transformed by the forward agent. Following figure shows example route for a forward agent.

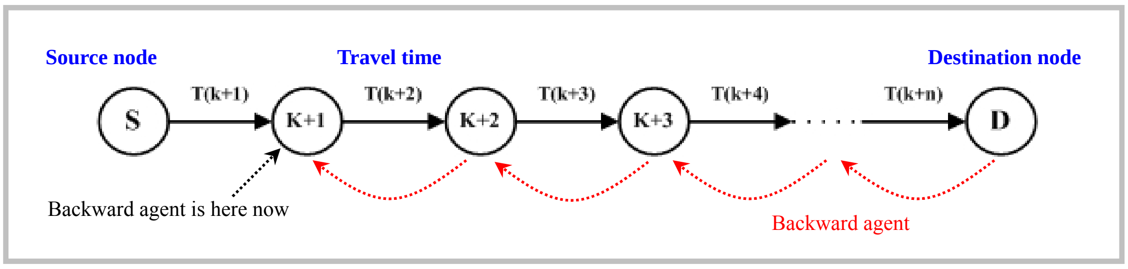


Figure 12: Example route from source node to destination node

Node S represents the source node, node D represents the destination node. Nodes $k+1, k+2\cdots$ are the visited nodes on the way from S to D. T_{k+1} represents the travel time between S and $k+1$, T_{k+2} represents the travel time from $k+1$ to $k+2$ and so on. A forward agent will collect information (travel time) during its moving from source node S to the destination. When the destination node D is reached, the forward agent activates the backward agent and transfers the whole route information to it. The backward agent will move from destination to source node ($D \to S$). At every visited node on the backward path, the backward agent will update the probabilities in the routing table using information inherited from the forward agent.

We assume that the backward agent now reaches node $k+1$, and then it will update the probabilities in routing table of this node $k+1$ through travel time $T_{k+2}, T_{k+3}\cdots T_{k+n}$. In Ant Based Control algorithm, the backward agent not only updates the probability to reach the destination node D from node $k+1$ along node $k+2$, but also updates the probabilities to reach every other node ($k+3, k+4, \ldots, k+n$)on the sub-path to the destination node D along node $k+2$. Table-5 shows the routing table structure of the node $k+1$.

Table 5: Route table of node k+1

Next / Dest	Node k+2	Node n+1	Node n+2	……
Node 1	$P_{1,k+2}$	$P_{1,n+1}$	$P_{1,n+2}$	……
……	……	……	……	……
Node k	$P_{k,k+2}$	$P_{k,n+1}$	$P_{k,n+2}$	……
Node k+2	……	……	……	……
Node k+3	……	……	……	……
……	……	……	……	……
Node D	$P_{D,k+2}$	$P_{D,n+1}$	$P_{D,n+2}$	……

The probability $P_{destination,nextnode}$ represents the probability from current node to destination through this nextnode. Based on routing table above, $P_{D,k+2}$ represents the probability from $k+1$ to D choosing $k+2$ as the next node. Table-6 will show some probabilities that are updated by the backward agent at node $k+1$ and corresponding travel time that are used.

Table 6: The probabilities that are updated and corresponding travel time

Probabilities to be updated	Travel time to be used
$P_{k+2,k+2}$	t_{k+2}
$P_{k+3,k+2}$	$t_{k+2}+t_{k+3}$
$P_{k+4,k+2}$	$t_{k+2}+t_{k+3}+t_{k+4}$
……	……
$P_{D,k+2}$	$t_{k+2}+t_{k+3}+t_{k+4}+\ldots+t_{k+n}$

Let's remember the formulas that we described in the section ***2.2.2.1***:

$$P_{new}=\frac{P_{old}+\Delta P}{1+\Delta P} \qquad (1)$$

$$P'_{new}=\frac{P_{old}}{1+\Delta P} \qquad (2)$$

Sign ΔP is the probability increase. In ABC-algorithm, ΔP is computed using

$\Delta P=\frac{A}{t}+B$	▪ A, B are constants, proper values are A = 0.8, B= 0.01. ▪ t is the travel time of the ant agent from node $k+1$ to the destination.

This formula ensures that the probability increase ΔP is inversely proportional to the travel time of the forward agent: the higher the time t, the lower the probability increase ΔP, and vice verse. As the result, the good path (with short travel time) receives a strong update. For a given values of ΔP, the absolute and relative increase of $P_{D,k+2}$ is much larger for small values of $P^{old}_{D,k+2}$ than for large values of $P^{old}_{D,k+2}$. That is low probabilities go up very fast to adapt to the new traffic situation whereas probabilities which are already high are increased only a little bit. Based on this example path, the formulas that we used to compute the probabilities are:

Updating formula for node $k+1$:

$$P^{new}_{D,k+2}=\frac{P^{old}_{D,k+2}+\Delta P}{1+\Delta P}$$

Node $k+1$ may have other several neighboring nodes ($n+1, n+2 \ldots$). To ensure the sum of the probabilities per destination node remains 1, probabilities of neighboring nodes except node $k+2$ will be updated accordingly. Updating for other neighboring nodes will use the following formula:

Updating formula for other neighboring nodes:

$$P^{new}_{D,i}=\frac{P^{old}_{D,i}+\Delta P}{1+\Delta P} \quad (i=n+1, n+2\cdots)$$

We use table-7 to show the change after the probabilities updating.

Table 7: Probabilities change after updating

Next / Dest	Node k+2	Node n+1	Node n+2	……	SUM
……	……	……	……	……	1
Node D	$P_{D,k+2} \uparrow$	$P_{D,n+1} \downarrow$	$P_{D,n+2} \downarrow$	……	1

This means that probabilities can only decrease if another probability in the same row is increased. Probability can approach zero if other probabilities in the same row of the routing table are increased more often or much stronger. 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. To overcome this disadvantage, an exploration probability as a minimum value for each probability is introduced for ensuring that no probability in the routing table reaches zero. Whenever a probability is smaller than the exploration probability this probability is set to the exploration probability. Normally, such exploration probability could be 0.05. Thus agents have chances to explore the new route like nature ants do.

2.3 Hierarchical routing

As described in the introduction, if the traffic network is complicated, conventional ABC routing system shows poor performance. To improve the efficiency of a routing system based on ABC-algorithm, hierarchical routing is proposed. Hierarchical routing was mainly devised to reduce memory requirements of simulations over very large topologies. A topology is broken down into several layers of hierarchy, thus downsizing the routing table. The routing table size is reduced from, for flat routing, to about *log n* for hierarchical routing. However some overhead costs results as number of hierarchy levels are increased. Optimal results were found for 3 level of hierarchy and the current implementation supports up to a maximum of 3 levels of hierarchical routing. In our hierarchical routing system, we will use 2 levels topology, abstract level and detailed level.

To be able to use hierarchical routing for the simulations, we need to define the hierarchy of the topology as well as provide the nodes with hierarchical addressing. In conventional routing (Ant Based Control routing), every node in the network knows about every other node in the topology, thus routing table size increases. On the contrary, for hierarchical routing, each node knows only about those nodes in its level. For all other destinations outside its level it forwards the information to the border router of its level. Thus the routing table size gets smaller. Through hierarchical topology, one complicated network could be divided into several small and simple networks. Our developed hierarchical routing system is based on these basic ideas. The detailed information about hierarchical definition will be described in following sections.

2.3.1 Network layout

In practice, the traffic network can be considered as a model of a geographical map of real cities. This kind of traffic network may consist of only one city. Furthermore, the traffic network is possibly made up of several cities and these cities are connected by motorways. Figure-13 will show some examples of traffic networks.

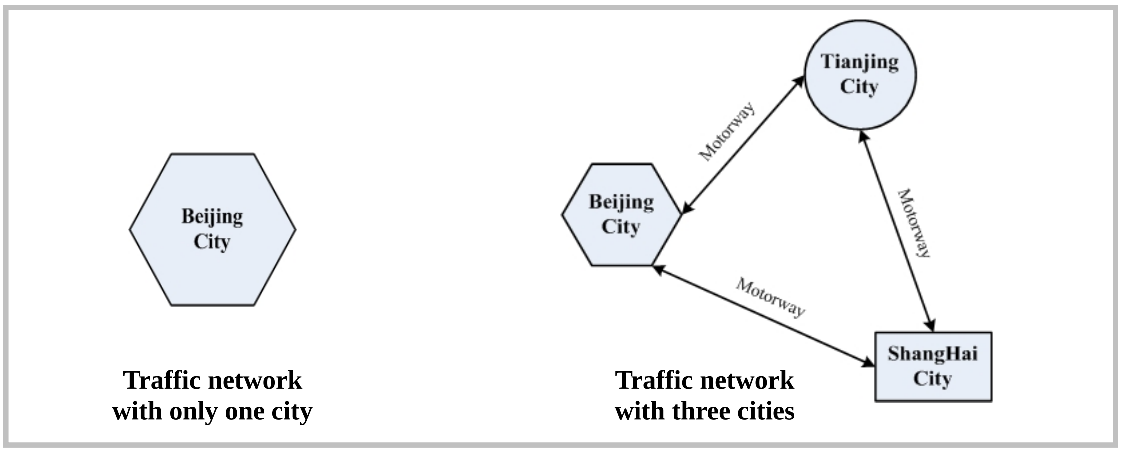


Figure 13: The traffic network with cities

In our hierarchical routing system, each city will be treated as one city network and we define this city network as sector. Inside each sector there can be a further kind of partition. The sector consists of nodes and links. These links represent streets in the city and the nodes represent intersections. Additionally, each city sector will own its unique characteristics. Some city sectors may be surrounded by motorways (rings); some may have no motorways. Figure-14 shows one detailed traffic network with three city sectors.

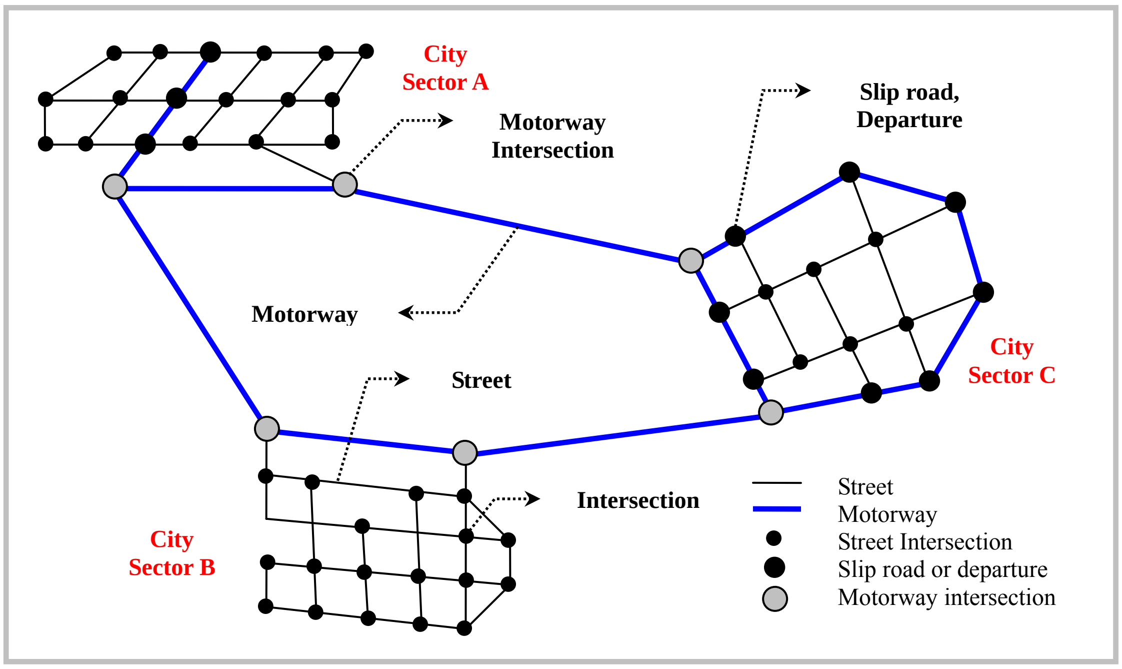


Figure 14: Traffic network with detailed structure

From figure-14 we can see that, city sector C is surrounded by motorway, city sector B has no motorway around it, and there is one motorway in the middle of the city sector A. These three city sectors connect through motorways. Each node and link has some hierarchical attributes based on their functionality in the hierarchical traffic network. We will explain how these hierarchical attributes are defined in the following section.

2.3.1.1 Hierarchical attributes

In the figure-14, the nodes are divided into several types: *Intersection, slip road, departure and motorway intersection*. The links are also divided into some types: *street and motorway*. We use table-8 to explain the rules for how these hierarchical attributes are assigned to nodes and links.

Table 8: Hierarchical attributes definition rules

Hierarchical attributes		Definition
Links	Street	▪ It only locates within the city sector ▪ It is normally narrow and with slow speed
	Motorway	▪ It is around the city or in the middle of the city ▪ It is normally broad and with high speed ▪ It connects a city sector to the others
Nodes	Intersection	▪ It only locates on the street
	Slip road	▪ It only locates on the motorway ▪ It connects street and motorway with single direction (S→M)
	Departure	▪ It only locates on the motorway ▪ It connects motorway and street with single direction (M→S)
	Motorway intersection	▪ It locates on the motorway ▪ It connects one motorway to the others

We introduce *slip road* and *departure* because the direction of the street. If the street is bidirectional, there will be no difference between the *slip road* and *departure*. In our hierarchical routing system, we treat these two kinds of nodes equally.

2.3.2 Network hierarchy

In hierarchical view, the traffic network could be slipped up into several less complex networks. The final hierarchical network consists of two levels: the abstract level and the detailed level. On the abstract level there is only one network, the motorway network, which contains all motorways, slip roads, departures and motorway intersections. And the detailed level is made up of several independent city networks; every city network contains all streets, intersections and motorways belonging to this sector. Based on traffic network in figure-14, the abstract level network and the detailed level network are shown in the following figures (figure-15 and figure-16).

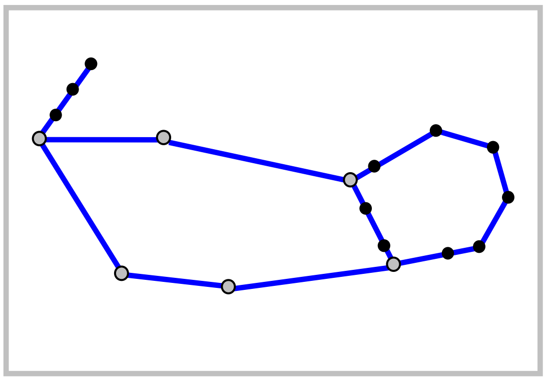


Figure 15: Abstract network

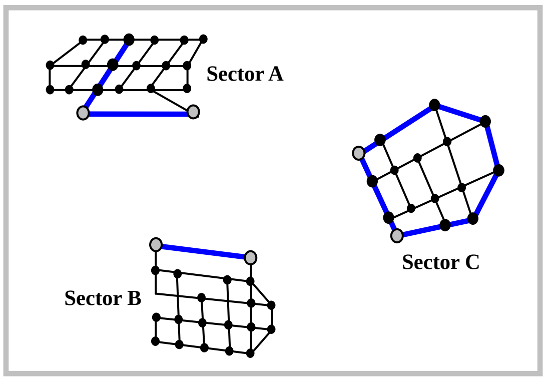


Figure 16: Detailed networks

2.3.3 Hierarchical network representation

For routing system, it is essential that all nodes and links of the network have to be identifiable. In Ant Based Control routing program, nodes and links already could be distinguished by using natural numbers. At the same time, the normal types of nodes are defined either, they are *normal, traffic light, roundabout* (defined by [Kroon]).

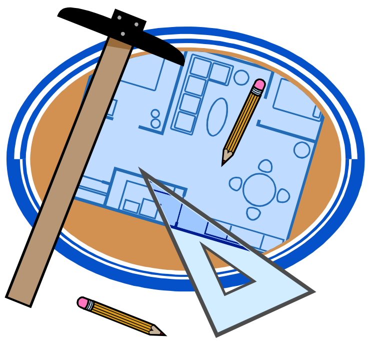
But hierarchical routing system needs more attributes of nodes and links. If a node is defined to be a *slip road, departure or motorway intersection*, it will locate on the motorway. And if a node is defined to be an intersection it will locate on the inner street of a city network. For each link the kind of link must be given, whether it is a motorway or a street. But the most important information needed for hierarchical routing system is the sector number to which each of the nodes and links belongs to. So the sectors must be numbered with nature numbers and this number must be added to each node and link. The intersections and the streets within a city network are always the part of exactly one sector, because they are situated inside a sector. The slip road, departure, and motorway intersection, which surrounded a sector, also belongs to this sector.

Given all this information a routing system is able to create a hierarchical model of the traffic network. It means by using this information the routing system is able to create an internal representation of the network for the abstract level and internal representation of the detailed level networks for each single sector like displayed in figure-15 and figure-16. We use table-9 to show the comparison between conventional ABC routing network attributes and hierarchical routing attributes.

Table 9: Comparison between two types' routing attributes

Items	Conventional ABC routing	Hierarchical routing
Nodes	▪ Normal, traffic light, roundabout	▪ Normal, traffic light, roundabout ▪ Intersection, slip road, departure, and motorway intersection ▪ Sector number
Links	▪ Speed, lanes, propriety and so on	▪ Speed, lanes, propriety and so on ▪ Street, motorway ▪ Sector number

Chapter 3 Detailed Design



3.1 Kroon's design

As we mentioned in the former chapter, our hierarchical routing system is based on the Kroon's simulation (dynamic routing system). So, the first thing we do is to study Kroon's program in detail and then propose our hierarchical routing system prototype.

3.1.1 Global design

Suppose a vehicle driver wants to be informed how to get from position A to position B in the shortest time. To request information from routing system, the driver must know about the location where the vehicle is at the moment and the destination where the vehicle wants to go, and then the request message is sent to the routing system. Using certain routing algorithm (Ant Based Control algorithm), the routing system computes one optimal path and sends it back to the driver. In Kroon's program, all communication between the vehicle and the routing system can be done with packet switched communication. The following figure will show such kind of communication.

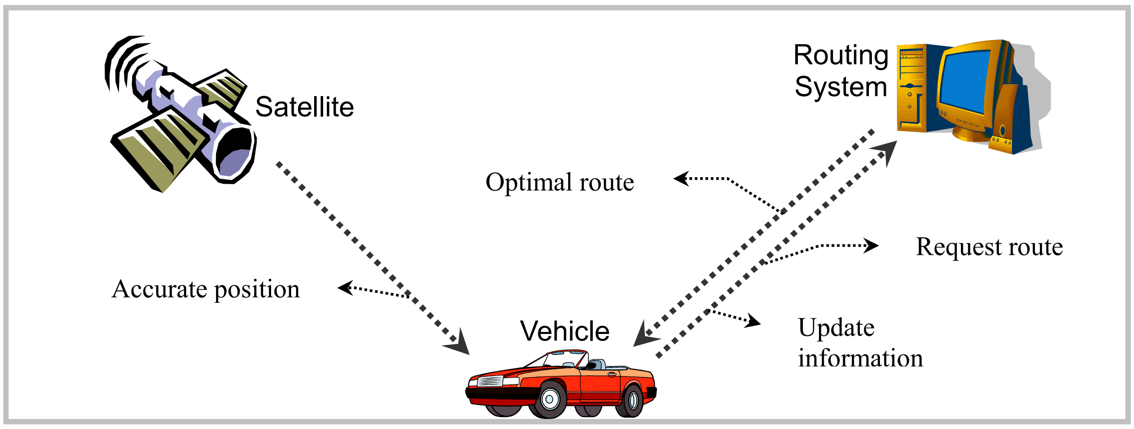


Figure 17: Communication between objects

From the figure above we notice that, a vehicle will send update information to the routing system while requesting the route. This is the dynamic information we introduced at the beginning of the thesis, such as travel time of certain covered path, sudden broke of route and so on.

The obvious property of the Ant Based Control algorithm is distribution. This means that the computation is done on several computer systems which are mutually connected via network. Such a distributed architecture shows some advantages in practice, like faster speed, more stability. In Kroon's design, all the vehicles will run on a single computer system, and as a result, connection through server is adopted. That is the vehicle connect to a central server and this server sends the request to the appropriate computer (routing system). The figure-18 displays connection through server.

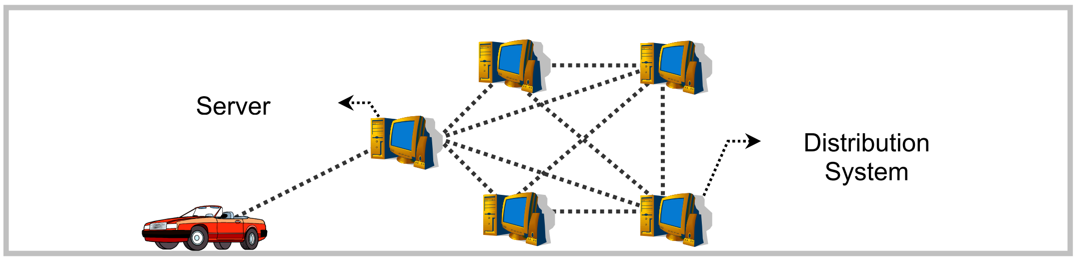


Figure 18: Connection through server

3.1.2 System design

The complete simulation system of Kroon is made up of two parts, city program and routing system. The main application is the city program. This is where the simulation environment, the city traffic, will run. In this environment vehicles will drive through the road network. The movement of vehicles is influenced by other vehicles and their surroundings. While driving around, the vehicles may send information to the routing system. There the information will be handled by the timetable updating system. This system processes the information and stores it in the timetable. The route finding system uses the information in the timetable to construct optimal route, which can be used by the vehicles of the city traffic. Furthermore there is the control center. This is the place where all the information is gathered together from the city traffic as well as from the routing system. Moreover everything that is adjustable can be adjusted here.

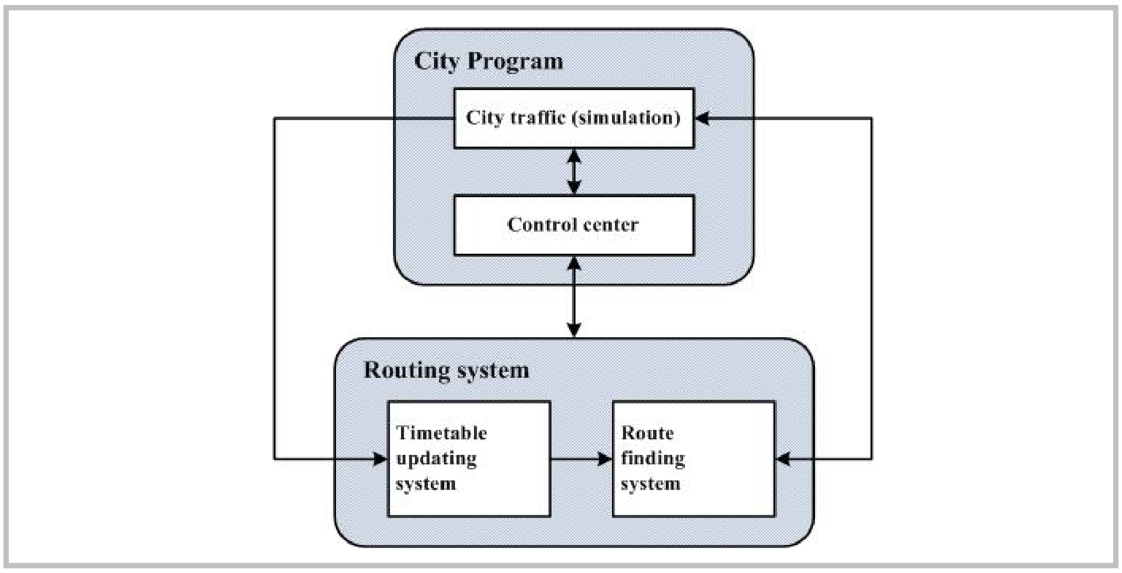


Figure 19: Kroon's system design

3.1.3 Routing system

Routing system could be considered as the most important part of the Kroon's simulation. It contains the Timetable updating system and Route finding system. The Timetable updating system is responsible for providing the routing system the dynamic information about the network. Through this information, the routing system knows about the state of the real traffic network and then could compute the optimal route. The Timetable is a kind of memory which contains dynamic data about the state of the network measured in time that is needed to cover each single link from its source to its destination. It is a two-dimensional matrix with all nodes of the network graph along the axes. In this table an entry exists for each link of the network. One entry represents the time estimates for the vehicles to cover the corresponding road. The figure-20 and table-10 show an example of a traffic network and a corresponding timetable.

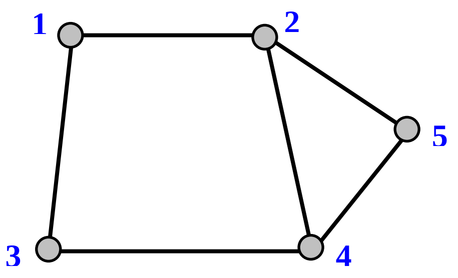


Figure 20: Example traffic network

Table 10: Example timetable for graph

To / Form	Node 1	Node 2	Node 3	Node 4	Node 5
Node 1	--	9s	11s	--	--
Node 2	10s	--	--	8s	7s
Node 3	12s	--	--	15s	--
Node 4	--	10s	14s	--	6s
Node 5	--	8s	--	7s	--

From graph and timetable above we can see that time entry exists only between two connected nodes, like node 1 and 2.

3.1.3.1 Timetable updating system

The timetable updating system could receive information about the traffic network in the city from different sources, such as sensors in the road-surface. Because of the high investment of these sensors, vehicles themselves are the main source to collect information. In Kroon's simulation, vehicles provide the system with information about the last part of the path they followed and the time it took them to cover it. With this information the timetable updating system computes the travel times for every link in the traffic network and then stores travel times in the timetable.

- ***Computing the travel times***

Let’s make clearer about the way for computing the travel times. First of all, the total of the covered road since the last update needs to be calculated. The formula for this is:

$$D = \sum_l d_l$$

- d_l is the covered distance on link l in meters
- D is the total road covered by this vehicle since the last update

With the covered distance calculated with formula above, the travel time for every single link can be computed by using the following formula:

$$M_l = \frac{L_l}{D} \cdot t$$

- L_l is the length of link l in meter
- 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
- M_l is the measurement of the travel time for link l in seconds

3.1.3.2 Route finding system

Route finding system receives requests for routes form individual motorists. For each motorist the shortest route in time will be calculated and send back to the motorist. The route finding system can be divided into two modules as can be seen in figure-21, the route optimization module and the route computation module.

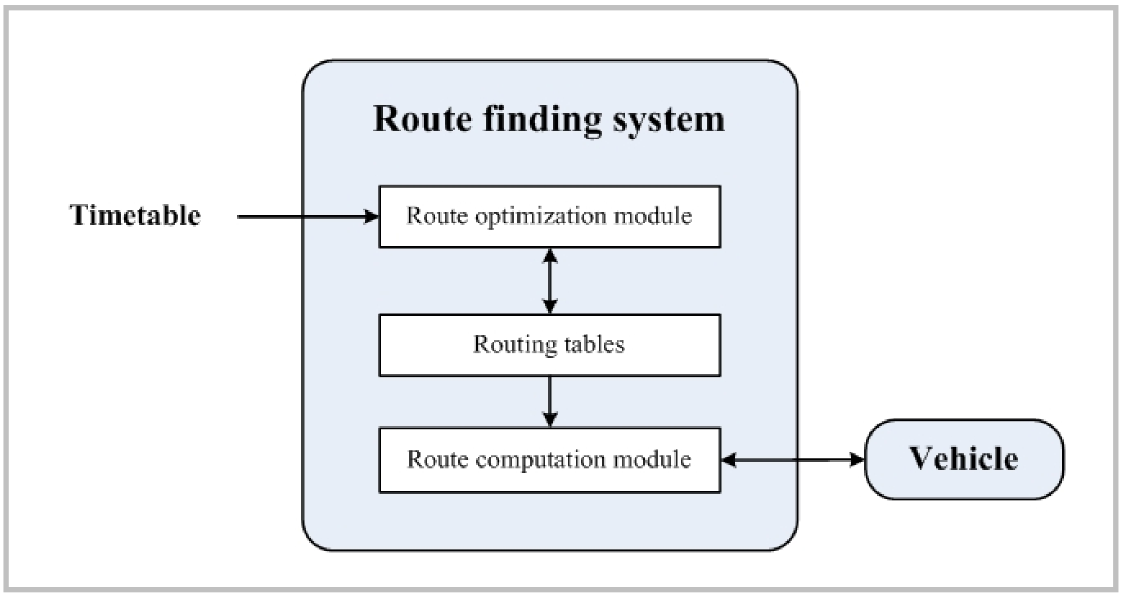


Figure 21: Structure of route finding system

The route optimization module is responsible for optimizing the routes from every node of the network to every other node. Therefore it uses the dynamic data in the timetable of the routing system. For the optimization the Ant Based Control algorithm is used. The second module, the route computation module, is the part of the route finding system that computes the shortest routes in time for vehicles from a source node to a destination node. Vehicles connect with this module to request a route. The route computation module uses the information in the routing tables, which is provided by the route optimization module, to compute the routes. After the routes are computed they are sent back to the vehicles.

3.2 Hierarchical system design

Based on conventional Ant Based Control routing system, we make some adjustments to satisfy the requirements of hierarchical routing system. Such improvements mainly focused on routing system because of the hierarchical network division.

3.2.1 Global design

The hierarchical routing system consists of several parts. These are the Global routing system and a Sector routing system for each sector that contains a real city network. So if there are N sectors in a city network then the hierarchical routing system will consist of N sector route finding systems. Every part of hierarchical routing system consists of the same components as shown in figure-22, namely a Timetable updating system (TUS) and a Route finding system (RFS). Besides all routing systems have their own timetable which stores the dynamic data of the traffic network. The global route finding system is responsible for routing vehicles from a source sector to a destination sector (remote sector) along the shortest motorway in time. The sector routing finding system is responsible for routing vehicles from a source to a destination within the same city sector.

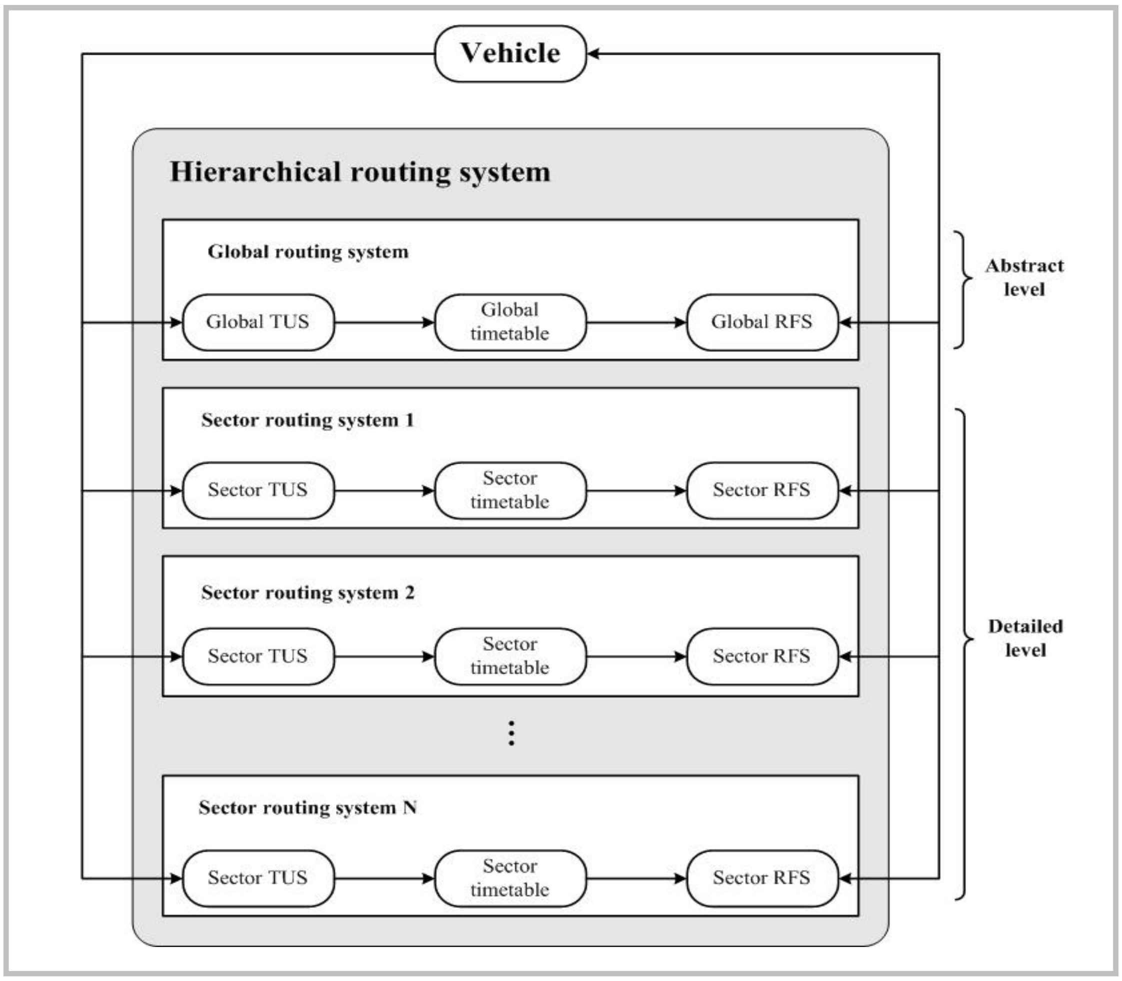


Figure 22: Hierarchical routing system

3.2.1.1 Cooperation between global and sector routing system

- ***Global routing system***

The global routing system is responsible for the abstract level of the traffic network. It computes the shortest routes in time for the vehicles from a node of the motorway to another node of the motorway along the motorway only. Therefore it uses the global timetable which holds the dynamic data of the abstract level of the network. The global routing system only works with the internal representation of the abstract level of the traffic network and not with the networks of the detailed level. We can see from the figure-23, if a vehicle wants go from point A (in sector 2) to D (in sector 3), the global routing system will receive the request and compute a optimal route from A to D along the motorway between sector 2 and sector 3.

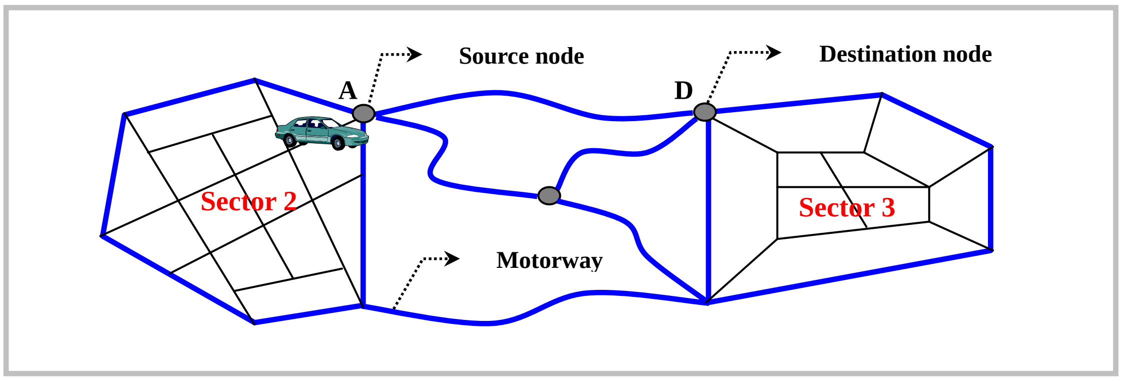


Figure 23: Responsibility of global routing system

- ***Sector routing system***

A sector routing system is responsible for one sector of the detailed level which contains a city network (see figure-16). It only works with the internal representation of the city network of that sector. Additionally a sector routing system needs the information for all other nodes of the traffic network to what sector they belong and what type of node they are. A sector timetable holds the dynamic data for all streets within the corresponding sector. For mitigating the influence of the communication delay, the time estimates of the surrounding motorway links should also be stored locally in the sector timetables, what allows a fast access to these values. If a vehicle wants go from A to D, the sector routing system will receive the request and compute a optimal route from A to D within the sector 2.

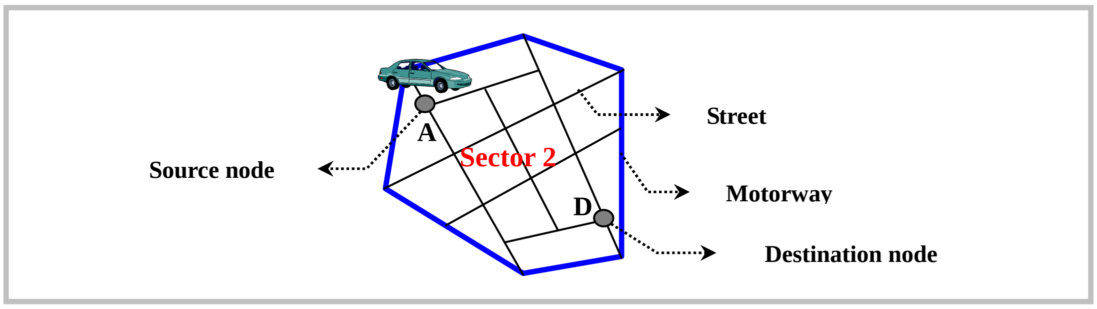


Figure 24: Responsibility of sector routing system

- ***Cooperation***

In most situations, sector routing system and global routing system must cooperate together to compute the optimal route for a vehicle. For example, compute a route from a street within one sector to a destination street in another distant sector. The figure-25 shows such a case, a vehicle wants go from A to D. In practice, a driver will reach the nearest motorway intersection to leave the source sector firstly, and then it goes to destination sector along the motorway. After arriving at the destination sector, the driver will choose the proper route to get to final destination. In this situation, the different routing systems must cooperate to compute the shortest route in time.

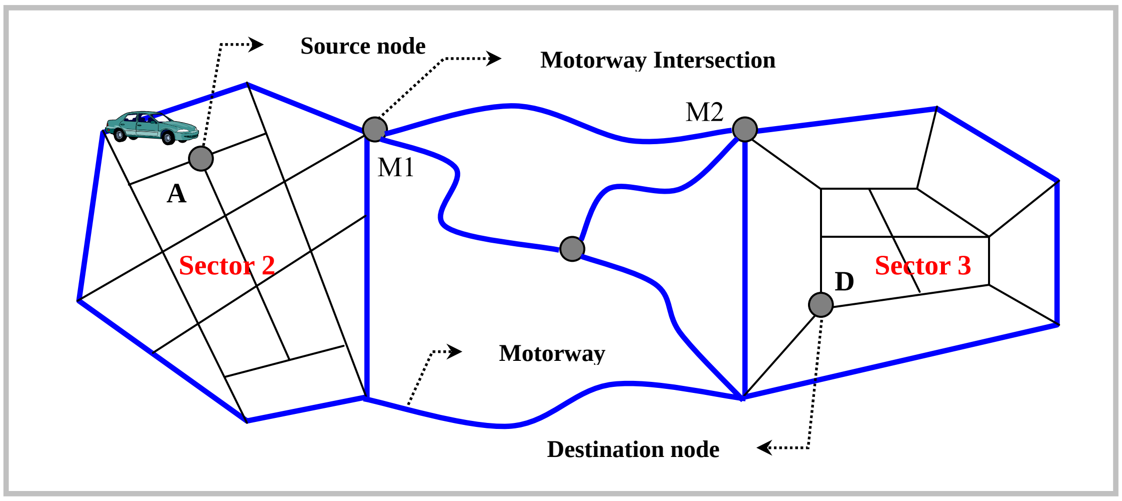


Figure 25: Cooperation between sector and global routing system

Based on traffic network above (Source node $A \rightarrow$ Destination node D), we explain the route computing procedure in detail.

Step 1: *compute the most suitable motorway intersection for the source node A. In this step, the sector 2 routing system will navigate the vehicle from A to such a kind of motorway intersection as soon as possible.*

Step 2: *we assume that the motorway intersection M_1 is the most suitable motorway intersection according to the sector 2 routing system. As a result, the sector 2 routing system will navigate vehicle from A to M_1. When M_1 is reached, the global routing system will work to compute the optimal route from M_1 to destination sector (sector 3) along the fastest motorway. In this example traffic network, we assume the most top motorway between M_1 and sector 3 is the fastest way, then the global routing system will let the vehicle move from M_1 to M_2 along this way.*

Step 3: *when the vehicle arrives at sector 3 (reach M_2), the sector 3 routing system will be responsible for computing the optimal route from M_2 to the final destination D.*

Following figure-26 will show such cooperation step by step.

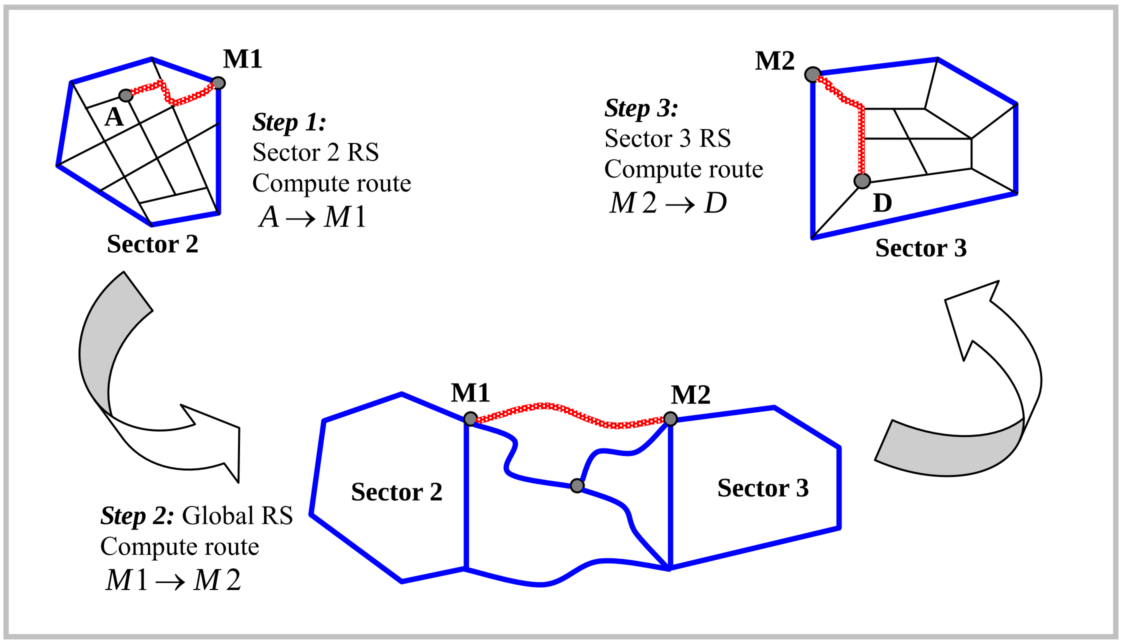


Figure 26: Routing between two different sectors

3.2.1.2 Hierarchical communication

For the communication between the routing systems the internet or a wide area network is used. A connection must exist from every routing system to every other routing system to work as a common system. Figure-24 shows such communication.

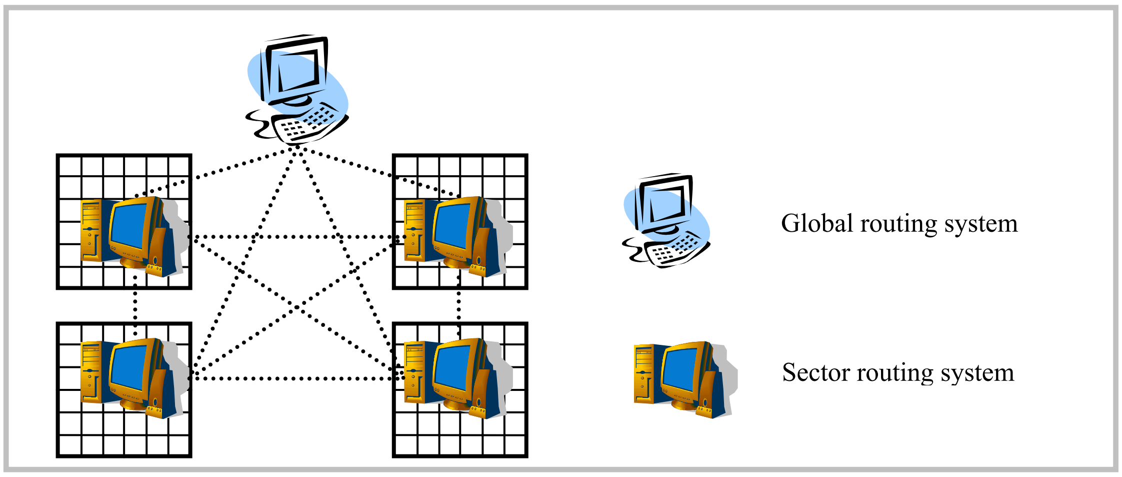


Figure 27: Communication between global and sector routing systems

The sector routing systems which are responsible for a sector are situated within or at least close to this sector. The global routing system is centralized because it is responsible for the whole motorway network. The following rules define with which part of the routing system vehicles must connect:

I. A vehicle on a street inside a sector which wants to request a route must connect with the sector route finding system of that sector.
II. A vehicle on a street inside a sector which wants to send update information must connect with sector timetable updating system of that sector.
III. A vehicle on a motorway which wants to request a route must connect with the global route finding system.
IV. A vehicle on a motorway which wants to send update information must connect with the global timetable updating system.

The figure-28 shows these connections in detail.

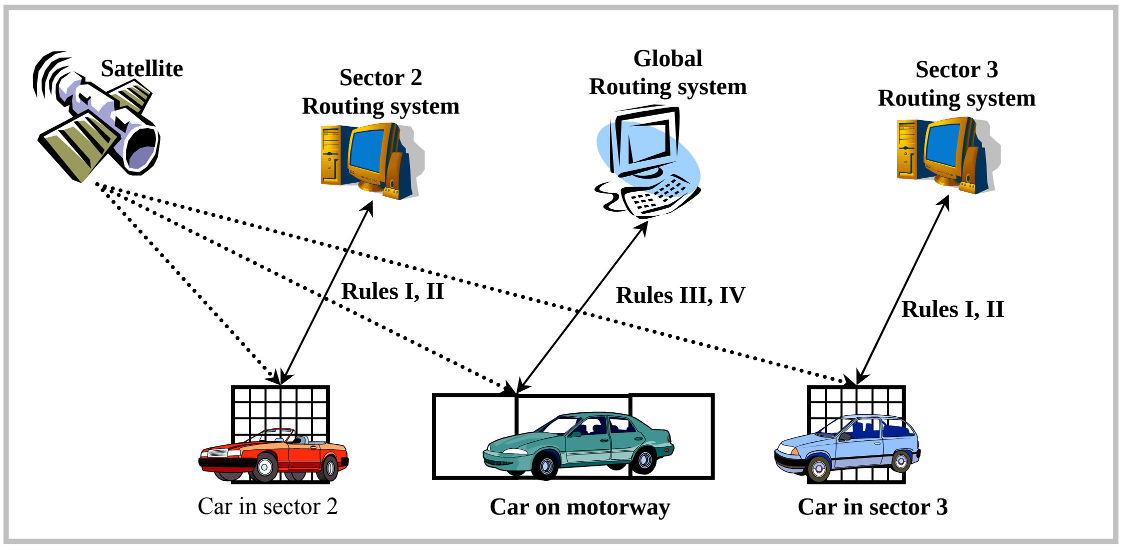


Figure 28: Hierarchical routing system connections

3.2.2 Timetable updating system

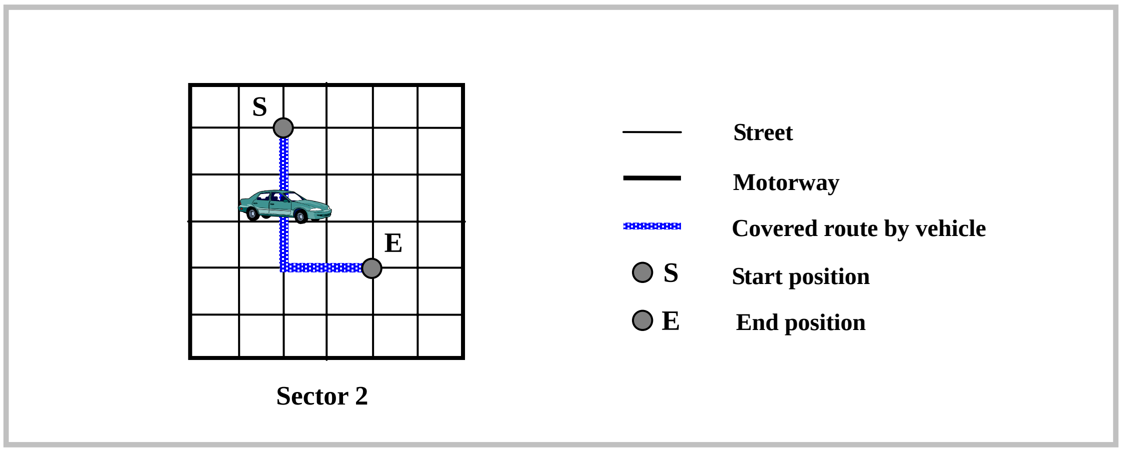
According to the hierarchical routing requirements, we must make some adjustments on conventional timetable updating system.

3.2.2.1 Update information

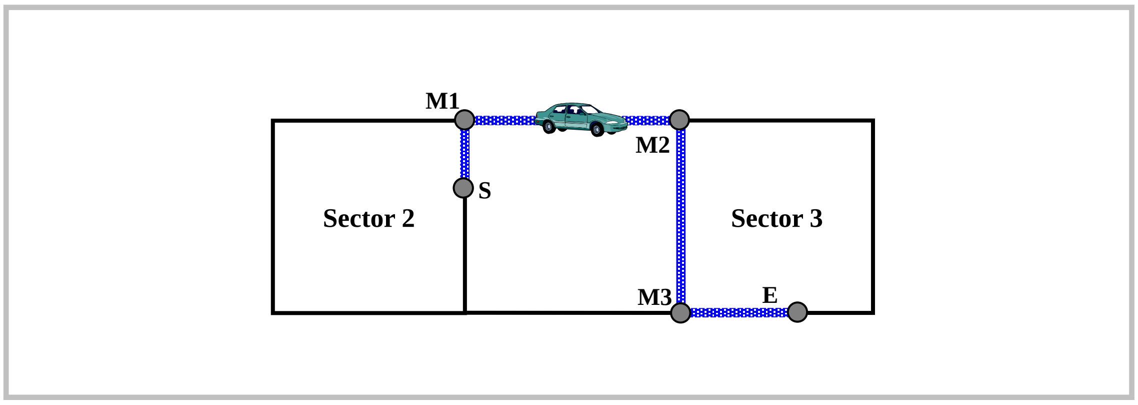
As we described in the former chapter, vehicles send update information at regular intervals about the last part of the path they followed and the time it took them to cover it. With the update information of the vehicles a timetable updating system can compute the travel times for every covered link and use these times to update the corresponding time estimates in the timetable.

The update information, which vehicles send to a timetable updating system, contains the location and time at the moment of the previous update, the location and time at the moment of the update and all the links the vehicle has covered between both positions. Besides, the vehicles must provide the times when the motorway was entered and left (a new city network was entered). Because in the hierarchical routing system, timetable updating system is divided into two categories: global timetable updating system and sector timetable updating system. The vehicles must connect with the corresponding timetable updating system using connection rules defined in section ***3.2.1.2***. According to the route, which vehicle covered, there exist several cases. We will present detailed explanations for these different situations.

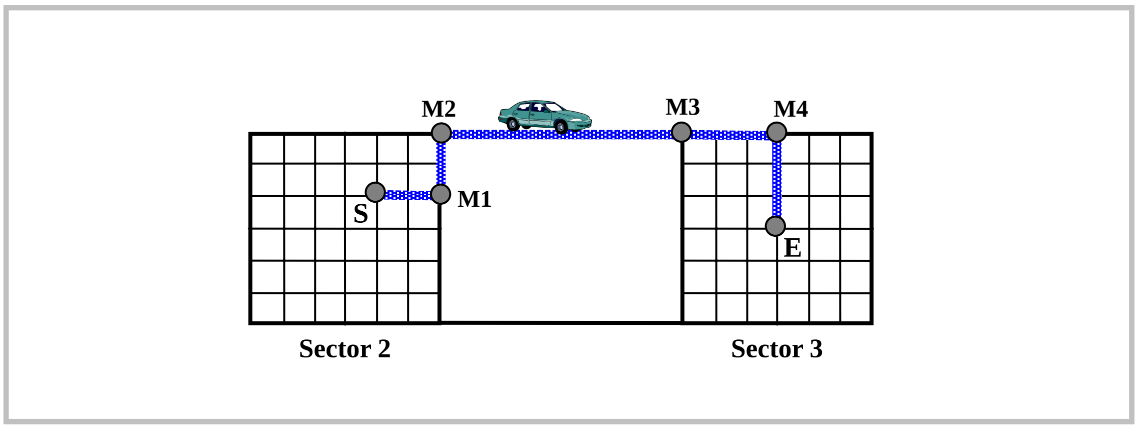
Case 1: covered route located within one sector



Case 2: covered route located on motorway



Case 3: covered route starts from an intersection to a distant intersection



3.2.2.2 Splitting the update information

Update information must be split by the timetable updating system which receives the information. Based on three cases displayed above, let's see how to split the update information.

Table 11: Update information split

	Update Information Split situations
Case 1	Covered route located within sector, update information is split for the following timetable updating systems: ▪ Sector 2 ($S \to E$)
Case 2	Covered route located on the motorway, involved two sectors. Update information is split for the following timetable updating systems: ▪ Sector 2 ($S \to M1$) ▪ Global ($S \to E$) ▪ Sector 3 ($M2 \to E$)
Case 3	Covered route located not only within sector, but also on the motorway. Update information is split for the following timetable updating systems: ▪ Sector 2 ($S \to M2$) ▪ Global ($M1 \to M4$) ▪ Sector 3 ($M3 \to E$)

After the update information has been split, corresponding timetable updating system could compute the travel time. We have given computing formula, used in Kroon's simulation, for travel time in section ***3.1.3.1***. Thinking about the accuracy, we could make some improvements on the original formula as follows.

$$T = \sum_l \frac{d_l}{S_l} \qquad M_l = \frac{L_l}{S_l \cdot T} \cdot t$$

- d_l is the covered distance on link l in meters
- L_l is the length of link l in meter
- S_l is the allowed speed on link l in meter per second
- T is the total time in seconds that is needed to cover the route without delay
- 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
- M_l is the measurement of the travel time for link l in seconds

The traveled times computed by the formula above are only average values and can strongly deviate from the actual travel times of the vehicles. For example one of the streets can be congested and others are congestion-free. Then the delay of the congestion on one of the links is distributed over the computed travel times of all links of the covered route although only one street is congested. And this results in falsified travel times. This can be avoided if the vehicles send update information at every intersection. So really the time is given which is needed to cover one link of the network. But a drawback is the amount of data and communication required between the vehicles and the timetable updating systems. To reduce the communication a longer update interval is desirable. Therefore a compromise must be found for a suitable update interval to avoid too much communication on one hand and to ensure accurate and actual travel times on the other hand.

3.2.3 Route finding system

As the most important part of the simulation, the route finding system must be adjusted based on requirements of hierarchical view.

3.2.3.1 Virtual nodes

For every sector that contains a city network a virtual node is introduced. Virtual nodes are used on the abstract level and the detailed level of the network. A virtual node is the union of all nodes that belong to the same sector. So it can be understood as abstraction from all the nodes of the sector. It represents one sector of the detailed level of the traffic network which contains a city network. Each virtual node will have an entry in the data structures of every node at both levels of the hierarchy (abstract and detailed). Whenever a vehicle is routed from an intersection of one sector to an intersection of a distant sector, this destination sector is represented by its virtual node. So vehicles are always routed to the virtual node until that the sector is reached.

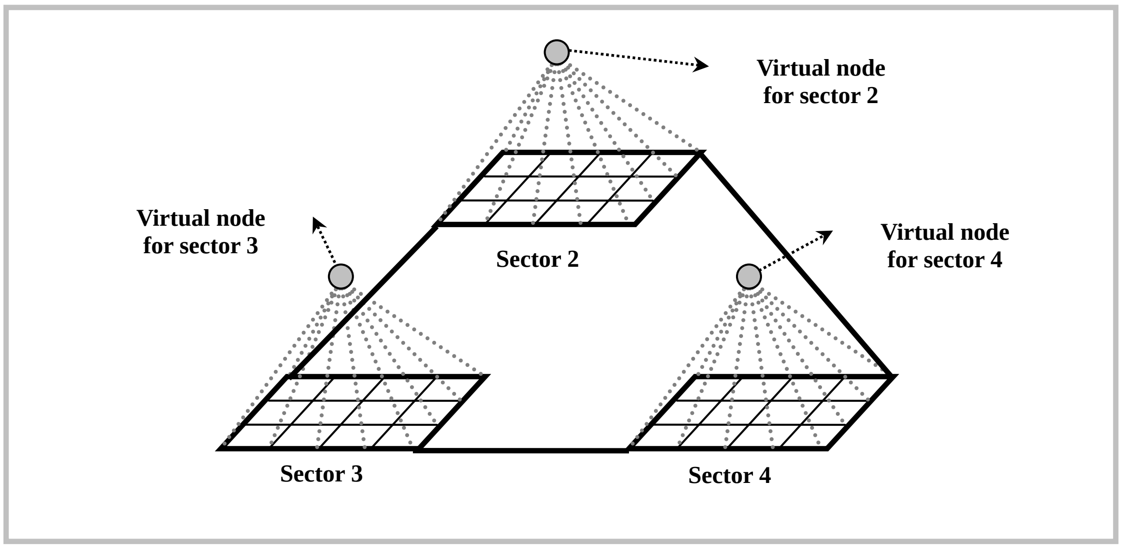


Figure 29: Virtual node for city network

3.2.3.2 Hierarchical routing table

The routing table structure for every node in the hierarchical traffic network will be changed because of introduction of virtual nodes. In hierarchical view, nodes could be defined according to the figure-14. Routing table structure of a node located on the abstract level network is different from a node located on the detailed level network. The table-12 shows the routing table structure of the node i, which belongs to the detailed level network. The destination nodes are divided into two types: the nodes belong to the same sector as the node i, and the virtual nodes representing the different sectors with node i. The table-13 represents the routing table structure of the node j, which belongs to the abstract level network. Its destinations also can be divided into two types: the nodes only located on the motorway, and the virtual nodes representing the different sectors with node j.

Table 12: Routing table for detailed level node i in the sector 2

Node i (detailed level)		Neighboring nodes			
		Next Node 1	Next Node 2	……	Next Node N
Nodes belong to the same sector	Node 1	$P_{1,1}$	$P_{1,2}$	……	$P_{1,N}$
	Node 2	……	……	……	……
	……	……	……	……	……
	Node N	$P_{N,1}$	$P_{N,2}$	……	$P_{N,N}$
Virtual nodes for different sectors	Virtual node 3	$P_{V_3,1}$	$P_{V_3,2}$	……	$P_{V_3,N}$
	Virtual node 4	……	……	……	……
	……	……	……	……	……
	Virtual node N	$P_{V_n,1}$	$P_{V_n,2}$	……	$P_{V_n,N}$

Table 13: Routing table for abstract level node j in the sector 2

Node j (abstract level)		Neighboring nodes			
		Next Node 1	Next Node 2	……	Next Node N
Nodes located on the motorway	Node 1	$P_{1,1}$	$P_{1,2}$	……	$P_{1,N}$
	Node 2	……	……	……	……
	……	……	……	……	……
	Node N	$P_{N,1}$	$P_{N,2}$	……	$P_{N,N}$
Virtual nodes for different sectors	Virtual node 3	$P_{V_3,1}$	$P_{V_3,2}$	……	$P_{V_3,N}$
	Virtual node 4	……	……	……	……
	……	……	……	……	……
	Virtual node N	$P_{V_n,1}$	$P_{V_n,2}$	……	$P_{V_n,N}$

In a hierarchical traffic network, a node, which is located on the motorway, will have both an abstract level routing table and a detailed level routing table. A node, which is located within a sector, will only have a detailed routing table. The figure-30 shows one hierarchical traffic network with some nodes. The table-14 gives the necessary type of routing table for every node in this traffic network.

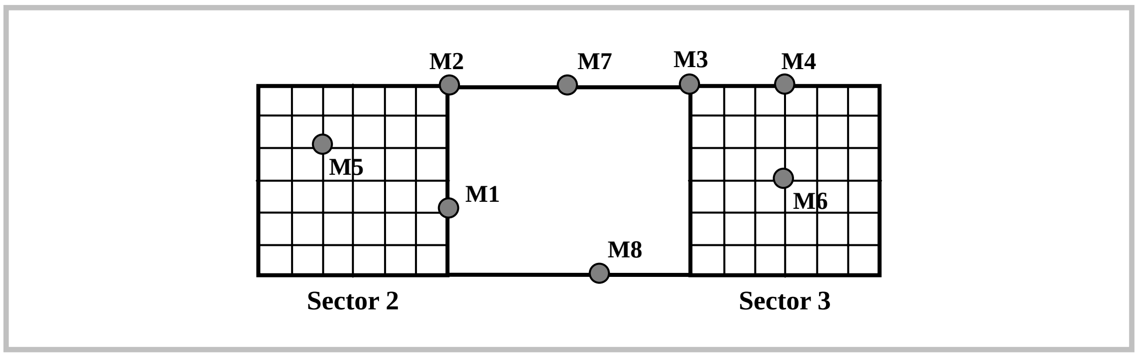


Figure 30: Nodes on the traffic network

Table 14: Routing table classification

Nodes ID	Routing table data structure
M1, M2, M3, M4	▪ Abstract level routing table ▪ Detailed level routing table
M5, M6	▪ Detailed level routing table
M7, M8	▪ Abstract level routing table

3.2.3.3 Hierarchical intelligent agents

In the hierarchical routing system, besides forward and backward agent classification, intelligent agents could be divided into abstract level agents and detailed level agents. Abstract level agents only move among the abstract level traffic network (figure-15), and detailed level agents only move among the detailed level traffic network (figure-16). The PAGE descriptions for intelligent agents are shown by the table-15 and table-16. We only list the difference between Kroon's definition and hierarchical routing.

Table 15: PAGE for abstract level agent

Agent Type		Percepts	Actions	Goals	Environment
Forward agent	Abstract level	IDs of the new entered sectors	determine and go to the next node along the motorway	store the route and the travel time for every visited sector	abstract level network
	Detailed level	Similar with Kroon's definition	determine and go to the next node within a sector	Get the virtual travel time to distant sector	Detailed level network (a city sector)

Backward agent	Abstract level	Similar with Kroon's definition	- Update the virtual part of a routing table - Send the update information to sector routing system	Update the virtual part of a routing table	abstract level network
	Detailed level	Similar with Kroon's definition	Update the virtual part of a routing table	Update the virtual part of a routing table	Detailed level network (a city sector)

3.2.3.4 Hierarchical algorithm

The Ant Based Control algorithm has to be changed for a hierarchical traffic network. The algorithm for detailed level agent is different from the abstract level.

1. ***The algorithm for the detailed level agent***

1) The forward agents $F_{s \to d}$ are launched at regular time intervals from every sector node S and with random destination. This random destination should be another node in the same sector with node S .
2) Each forward agent keeps a memory about its path (visited nodes). When an agent arrives in a node i , coming from node j , it memories the identifier of the visited node i and the virtual delay of the link (the trip time necessary for a vehicle to travel from intersection j to intersection i). These data are pushed onto the memory stack $S_{s \to d(i)}$.
3) When a forward agent 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 in the routing table of node i .
4) If a cycle is detected, that is, if a forward agent is forced to return to an already visited node, the cycle's nodes are popped from the agent's stack and all the memory about them is destroyed.
5) When the destination node d is reached, the forward agent $F_{s \to d}$ generates backward agent $B_{d \to s}$, transfers to it all of its memory, and dies.
6) If destination node d is on the motorway (d is not an intersection within a sector), when forward agent reached d , it gets the travel times from d to other different sector n and pushes them onto the memory stack $S_{d \to \mathrm{sector}(n)}$.
7) The backward agent takes the same path as that of its corresponding forward agent, but in the opposite direction. At each node i along the path it pops its stack $S_{s \to d(i)}$ to know the next hop node.
8) In every node i , the backward agent $B_{d \to s}$ updates the data structures of the node: the local traffic statistics and the routing table for all possible paths (these are via all neighbors of the node i) with the destination node d , and also the sub-paths of $s \to d$.
9) In every node i , the backward agent $B_{d \to s}$ updates the data structures of the

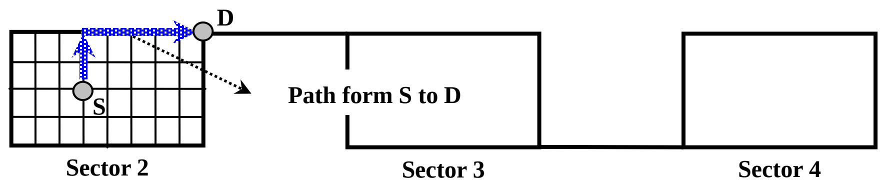
node: the local traffic statistics and the routing table for all possible paths (theses are via all neighbors of the node i) with the destination remote sector n. The probabilities are updated with a reinforcement value. This is a function of the time $T_{i\to d}$ ($T_{i\to \text{sector}(n)}$) the agent computed and local stochastic model of traffic in the node.

$P'_{dj} = P_{dj} + r(1 - P_{dj})$ And $P'_{\text{sector},j} = P_{\text{sector},j} + r(1 - P_{\text{sector},j})$

When $j = n$ the next node chosen by the agent.

$P'_{nd} = P_{nd} - rP_{nd}$ And $P'_{n,\text{sector}} = P_{n,\text{sector}} - rP_{n,\text{sector}}$ for $j \neq n$

10) When the source node s is reached again, the agent $B_{d\to s}$ dies.



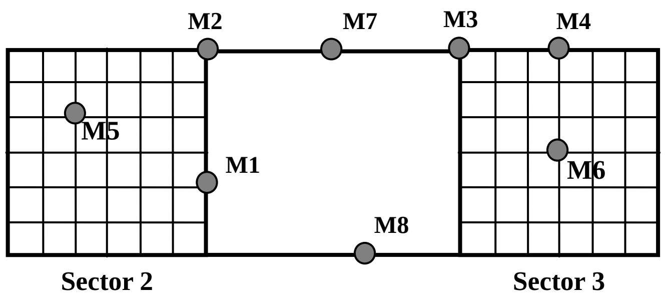
Forward agent move from S to D along the path above:

- When the agent arrives at node D, it will request travel times from D to sector 3 and sector 4. And the travel times are put into the memory stack of the agent.
- When backward agent reached node along the path, it will update the virtual part of the routing table of that node using memory.
- When backward agent reach source S, it dies.

Figure 31: Algorithm for detailed level agent

2. The algorithm for the abstract level agent

The nodes except for intersections are present on both networks, local and global. The global routing system copies from the node at the abstract level, the values μ_v for the virtual nodes v, and send them to the sector copy of the node. To avoid extra communication, this is done just in case a value μ_v changes with more than 5% in the node at the abstract level. For the global routing system the algorithm is slightly different from the sector routing system. The routing tables of the nodes have entries only for the nodes on the abstract level. In our definition, these nodes are *slip road, departure, motorway intersection*. The figure-32 shows the classification of nodes in detail.



- M1 ~ M6 belong to the detailed level networks. M1, M2 and M5 belong to sector 2; M3, M4 and M6 belong to sector 3.
- M1 ~ M4, M7 and M8 belong to the abstract level network.
- Detailed level agent of sector 2 will be generated at node M1, M2, and M5; detailed level agent of sector 3 will be generated at node M3, M4 and M6.
- Abstract level agent will be generated at node M1 ~ M4, M7 and M8.

Figure 32: Classification of nodes

1) The forward agents $F_{s \to d}$ are launched at regular time intervals from every node S (located on motorway), and with random destination. This random destination should be another node on the motorway.
2) Each forward agent keeps a memory about its path (visited nodes). When an agent arrives in a node i, coming from node j, it memories the identifier of the visited node i and the virtual delay of the link (the trip time necessary for a vehicle to travel from intersection j to intersection i). These data are pushed onto the memory stack $S_{s \to d(i)}$.
3) At the same time, each forward agent keeps a memory about its visited sectors. When an agent arrives in a new sector n, coming from node j, it memories the identifier of the visited sector n. These data are also pushed onto the memory stack $S_{s \to \sec tor(i)}$.
4) When forward agent 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 in the routing table of node i.
5) If a cycle is detected, that is, if a forward agent is forced to return to an already visited node, the cycle's nodes are popped from the agent's stack and all the memory about them is destroyed.
6) When the destination node d is reached, the forward agent $F_{s \to d}$ generates backward agent $B_{d \to s}$, transfers to it all of its memory, and dies.
7) The backward agent takes the same path as that of its corresponding forward agent, but in the opposite direction. At each node i along the path it pops its stack $S_{s \to d(i)}$ to know the next hop node.
8) In every node i, the backward agent $B_{d \to s}$ updates the data structures of the node: the local traffic statistics and the routing table for all possible paths (these are via all neighbors of the node i) with the destination node d, and also

the sub-paths of $s \to d$.

$P'_{dj} = P_{dj} + r(1 - P_{dj})$ When $j = n$ the next node chosen by the agent.

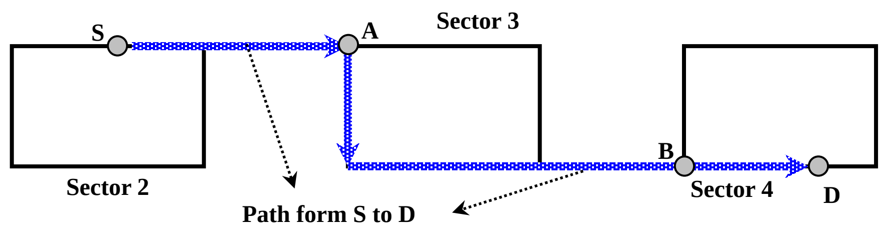
$P'_{nd} = P_{nd} - rP_{nd}$ For $j \neq n$

9) When the source node s is reached again, the backward agent $B_{d \to s}$ updates virtual part (if any) of the routing table of the node: the local traffic statistics and the routing table for all possible paths with the destination remote sector n. The probabilities are updated with a reinforcement value. This is a function of the time $T_{s \to \mathrm{sector}(n)}$, the agent computed and local stochastic model of traffic in the node.

$P'_{\mathrm{sector},j} = P_{\mathrm{sector},j} + r(1 - P_{\mathrm{sector},j})$ When $j = n$ the next node chosen by the agent.

$P'_{n,\mathrm{sector}} = P_{n,\mathrm{sector}} - rP_{n,\mathrm{sector}}$ For $j \neq n$.

If probabilities changes with more than 5%, then global routing system will send message to corresponding sector routing system for synchronization. And then backward agent $B_{d \to s}$ dies.



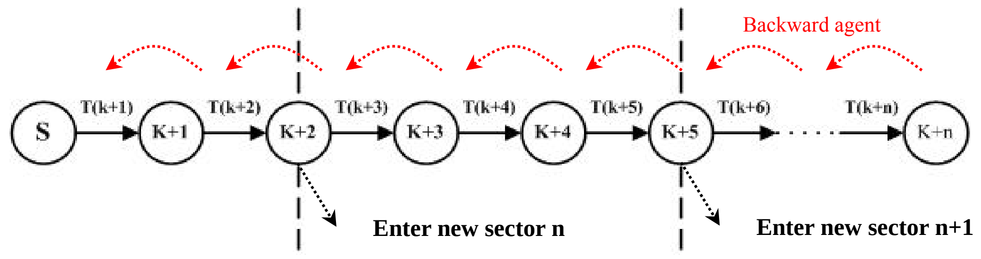
Forward agent move from S to D along the path above:

- When agent arrives in A and B, it will memorize sector 3 and sector 4 as visited sector. The travel times from S to sector 3 and sector 4 are put onto the memory stack of the agent.
- When backward agent reached source node S, it will update the virtual part of the routing table of S using memory.
- If value changes with more than 5%, global routing system will send message to the corresponding sector routing system. In this example, the message will be sent to sector 2 routing system.
- After that, backward agent dies.

Figure 33: Algorithm for abstract level agent

3.2.3.5 Hierarchical routing table updating

Now we will explain how routing table of a node in the traffic network is updated. We must remember that in hierarchical view, backward agent not only update the normal part of the routing table, but also update the virtual part of the routing table. Let's see following figure similar with figure-12.



This is the whole path for abstract level forward agent from source S to the destination $k+n$; the agent will enter several sectors along the route. Forward agent will memorize the times of entering new sectors. When backward agent returns to the source node S, it will use travel times to update normal part and virtual part of the routing table.

Figure 34: Example path for abstract level agent

Now, let's consider what probabilities a backward agent updates in the routing table of the node on its way back to the source node S. Table-16 shows the probabilities that are updated and the times which are used for the update.

Table 16: Probabilities that are updated by the backward agent

Agent located on	Probability to update	Used travel time
N_{k+4}	$P_{k+n,k+5}$	$T_{k+4}+T_{k+5}+T_{k+6}+\cdots+T_{k+n}$
N_{k+3}	$P_{k+n,k+4}$	$T_{k+3}+T_{k+4}+T_{k+5}+T_{k+6}+\cdots+T_{k+n}$
……	……	……
N_{k+1}	$P_{k+n,k+2}$	$T_{k+2}+T_{k+3}+T_{k+4}+T_{k+5}+T_{k+6}+\cdots+T_{k+n}$

When backward agent reach source node S, it also update the virtual part of the routing table of node S. Table-17 shows this kind of update.

Table 17: Virtual probabilities that are updated by the backward agent

Agent located on	Probability to update	Used travel time
N_s	$P_{sector(n),k+1}$	$T_{k+1}+T_{k+2}$
	$P_{sector(n+1),k+1}$	$T_{k+1}+T_{k+2}+T_{k+3}+T_{k+4}+T_{k+5}$

Chapter 4 Implementation

4.1 Introduction

In this chapter, we will explain the implementation of our software prototype in detail. As we described before, this hierarchical routing system is developed from Kroon's simulation version. And as a consequence we made use of the main parts of the Kroon's program and made some necessary adjustments for hierarchical routing requirements. Furthermore, the hierarchical traffic network is quite different from conventional traffic network. The traffic networks defined by Kroon are not appropriate for our hierarchical routing system. So we must develop one program for hierarchical traffic network generating. Using this traffic network generating program, we can make hierarchical traffic network according to our requirements.

We chose Borland Delphi 7.0 as our developing tools (Kroon's program is developed using Borland Delphi 5.0 Professional). Let's remind the Kroon's simulation for dynamic routing system. For convenience, we call Kroon's simulation K-Simulation in the following sections. There are two main function modules in K-Simulation, one is the city program, which is responsible for simulating the real traffic and controlling the running of the whole simulation; the other one is the dynamic routing system, which is responsible for computing optimal route according to the motorist's request using ABC-algorithm. Detailed information could be found in section ***3.1.2***. The figure-35 shows system design. We must state that there exists little difference in this system design. The hierarchical city network generating program will help us to create one example city network based on our requirements.

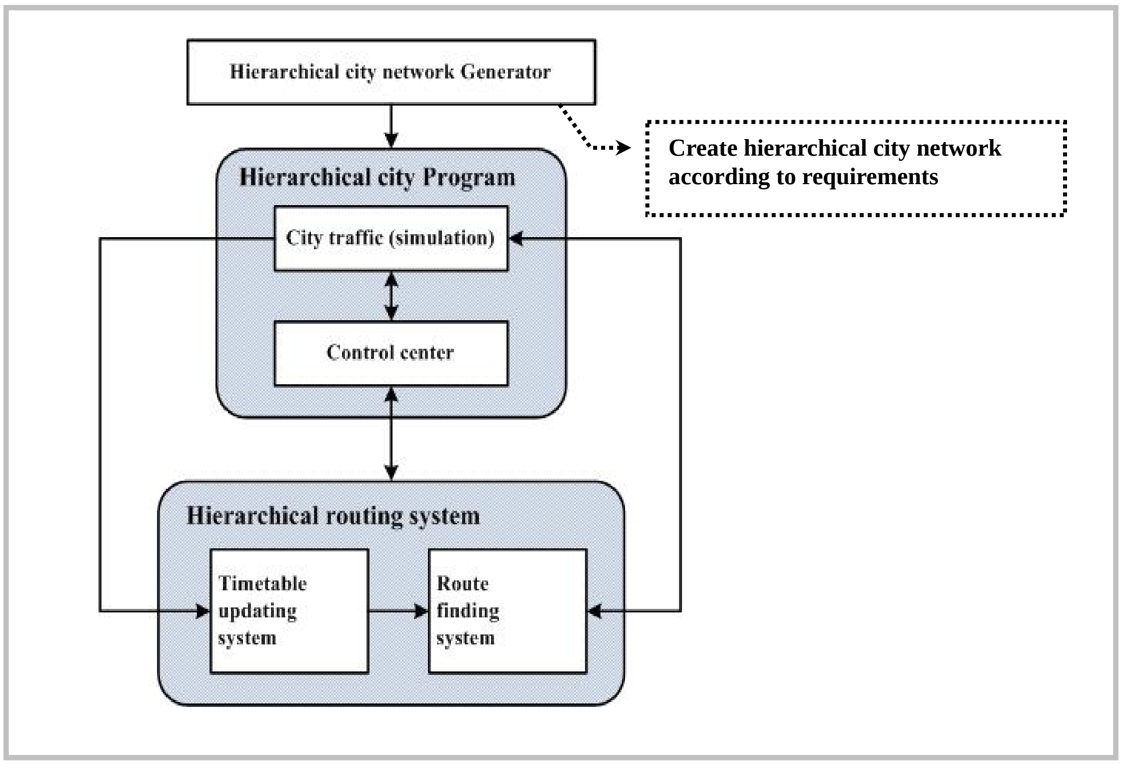

Hierarchical city network Generator
Create hierarchical city network according to requirements
Hierarchical city Program
City traffic (simulation)
Control center
Hierarchical routing system
Timetable updating system
Route finding system

Figure 35: Adjusted system design

In this chapter, the implementation will be discussed in the following order:

1. Hierarchical city network generating program

2. Hierarchical city program

- Loading and changing hierarchical traffic network
- Showing hierarchical information
- Collecting and sending route information
- Activate routing system

3. Hierarchical routing system
    - Routing systems activation
    - Virtual node generating
    - Timetable updating
    - Ants generating and movement
    - Routing table generating
    - Optimal route computation

4. UML diagram of the hierarchical routing system

4.2 Hierarchical city network generation

We developed one program to produce the real traffic networks. Through this program, we can create city networks according to our requirements. As we said before, in our simulation city networks are made up of several nodes and links. Through them, we can draw city networks randomly. Following figure shows screenshot of the network generator. In this screenshot, we can see one traffic network with three independent sectors (sector 2 ~ sector 4).

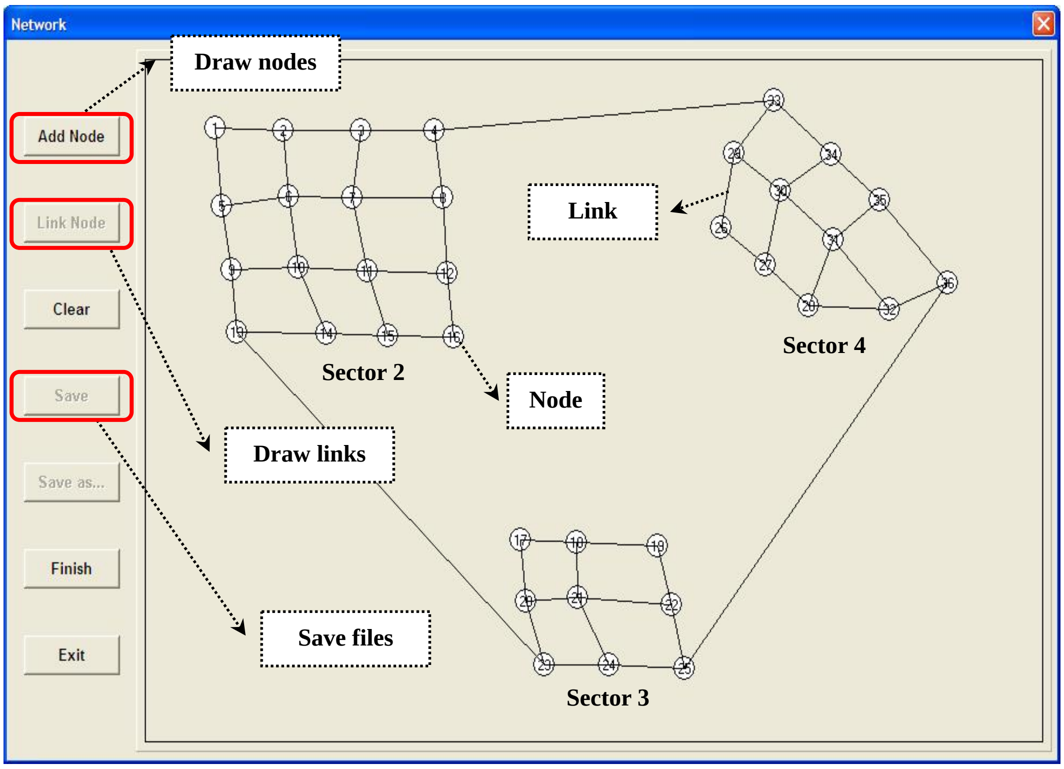


Figure 36: Screenshot of city network generating program

After studying on Kroon’s program, we know that if we want to describe traffic network clearly and precisely, several types of files are necessary to store information. Ronald Kroon has given the detailed description in his master thesis. As we said before, our hierarchical routing system is developed from Kroon’s version, so we will use these files in our system. Besides, according to the requirements of hierarchical routing system, we need some new types of files to keep the hierarchical information of the traffic network. And then we use ‘.sec’ and ‘.hier’ files to store such kind of data in our routing system. We will give a rough explanation on files types, which Kroon defined in his program, using table-18.

Table 18: File types in Kroon's program

File extension	File function description
.city	This file is the main file; it tells the system which files are used together. Loading this file will automatically load the appropriate files of a city environment. In our hierarchical routing system we add two new file types ('.sec' and '.hier' files).
.map	The intersections and roads are called 'nodes' and 'links' in traffic simulation. Every node has position attribute, which consists of nonnegative x- and y- coordinate. For every link, the nodes it connects (nodes starts from and ends) and direction must be known.
.int	Every node can be a normal intersection, an intersection with traffic light or a round about.
.road	For every link, we must know the number of lanes, the length, the average speed and the fact whether or not it is a main road. Only city program needs the number of lanes and the priority of the road.
.dis	The node must belong to one routing system. The routing system will deal with the updates that vehicles send about their position and with the request for routes from the vehicles. In our routing system, the routing system could be divided into global routing system (responsible for abstract level network) and sector routing system (responsible for detailed level network). Routing system number is equal to the sector number of nodes. The first routing system (routing system 1) is reserved for global routing system.
.light	A traffic light table describes for every road that ends at the intersection at what time the light is green. Using this information, Kroon's program simulates the real traffic light in the street.
.route	In some situations, vehicles will not request route information from routing system, and move according to the default information stored in this file. Table-19 will show the how vehicles are divided in Kroon's program.

In Kroon's simulation, vehicles are divided into four types. The table-19 shows such kind of definition. We need state that '1' means vehicle do some action, and '0' means do not execute action. For example, request route: 1 and send update: 0 means vehicles only request route from routing system and do not send route information it collected.

Table 19: Vehicle types in Kroon's program

Vehicle type		Vehicle actions
Request route	Send update	
0	0	Move without request route from routing system and send updating information
Type 1		
0	1	Only send updating information to routing system
Type 2		
1	0	Only request route from routing system
Type 3		
1	1	Not only request route from routing system, but also send update information to routing system
Type 4		

If vehicle belongs to type 1 and type 2, it will not request route information from routing system, and it will move according to some default route. Default routing tables are introduced for this requirement. It will be stored in '.route' files. Additionally, hierarchical routing system owns some unique attributes, such as sector assignment, nodes hierarchical types. To store hierarchical information, we need some other kind of files besides files mentioned above. We will give detailed explanation in the following sections.

- ***Sector information***

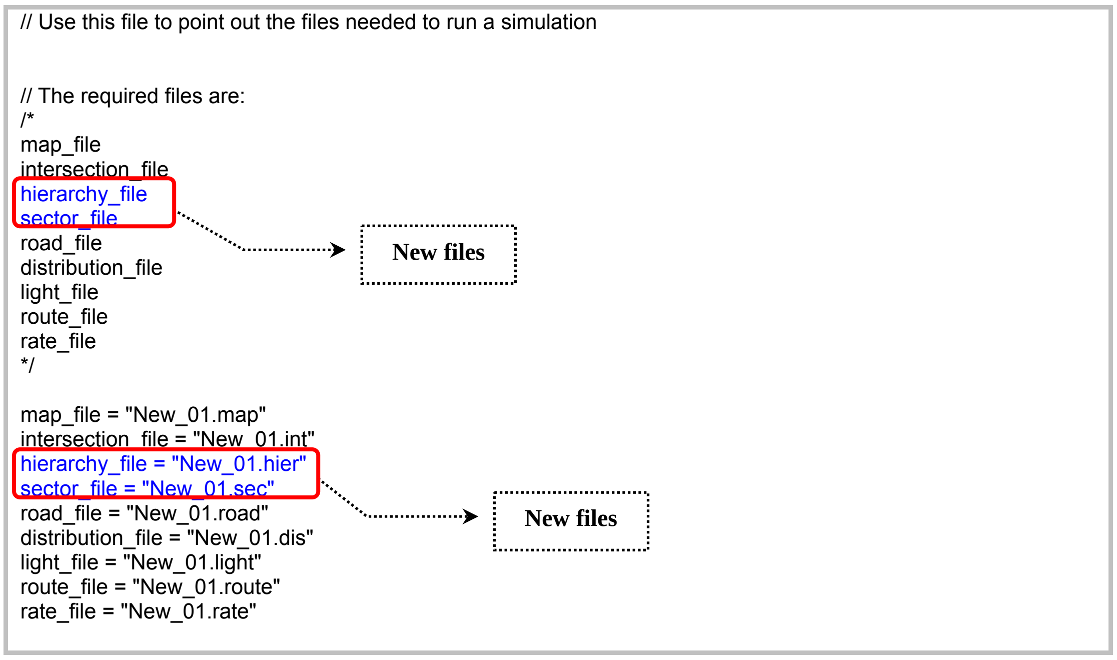
Based on our hierarchical view, real traffic network is made up of several independent city networks. We call this kind of network 'sector'. Every node and link in the network must belong to some sector. As we explained before, one obvious property of our hierarchical routing system is that it could be divided into two levels, abstract level and detailed level. Global routing system will be responsible for global city network (abstract level network); sector routing systems will be responsible for corresponding city sector. In our system, the first sector (sector 1) is reserved for global routing system (abstract level routing system), and then the sector routing system begins from the second sector (sector 2). The sector information is stored in '.sec' files.

- ***Nodes and links hierarchical types***

Besides the normal attributes defined in Kroon's program, every node and link must have hierarchical attributes. For nodes these hierarchical attributes are *intersection, slip road, departure and motorway intersection*; for links are *street (links located within one city sector), motorway, global way (link between two different sectors).* Using hierarchical information about nodes and links, routing system could know how compute optimal route based on hierarchical way. That is, compute optimal route within one sector is different from the one among several sectors. The nodes hierarchical types will be stored in '.hier' files. The links hierarchical types will be assigned when routing system running.

After adding these two hierarchical files, the '.city' file will look like following:

Table 20: '.city' files format in hierarchical routing system



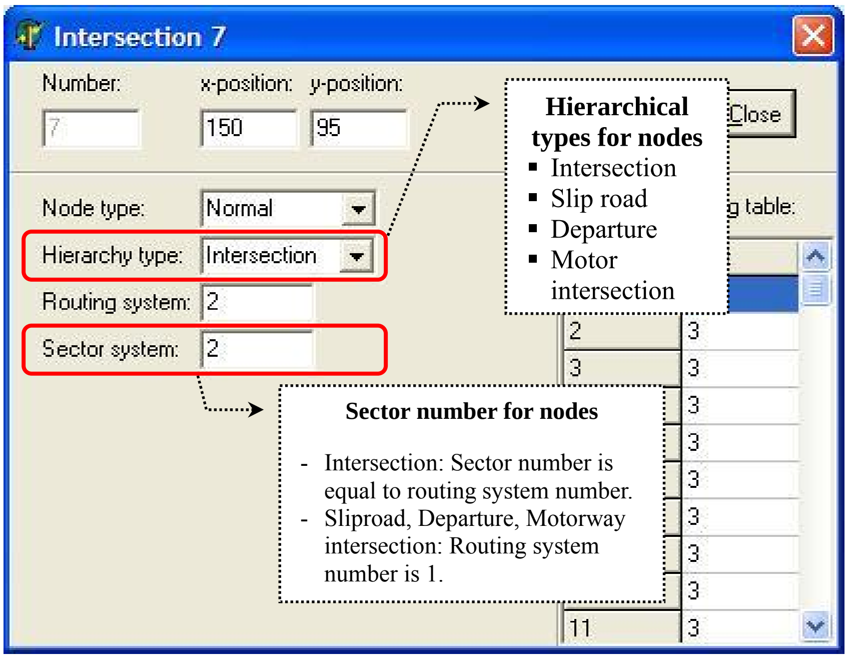
```
// Use this file to point out the files needed to run a simulation


// The required files are:
/*
map_file
intersection_file
hierarchy_file
sector_file
road_file
distribution_file
light_file
route_file
rate_file
*/

map_file = "New_01.map"
intersection_file = "New_01.int"
hierarchy_file = "New_01.hier"
sector_file = "New_01.sec"
road_file = "New_01.road"
distribution_file = "New_01.dis"
light_file = "New_01.light"
route_file = "New_01.route"
rate_file = "New_01.rate"
```

4.3 Hierarchical city program

City program is responsible for traffic simulation and controls the running of the routing system. Because of the introduction of a hierarchical definition, the conventional city program needs some modifications.

4.3.1 Loading and changing the hierarchical traffic network

Using a hierarchical network generating program, we could create random network according to our requirements. After loaded '.city' file, which is made by network generating program, we could set hierarchical attributes. The following several screenshots show how the system set these properties for nodes and links.



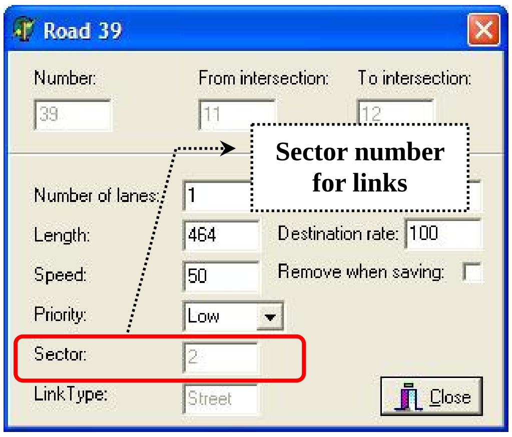
Intersection 7

Number: 7
x-position: 150
y-position: 95
Node type: Normal
Hierarchy type: Intersection
Routing system: 2
Sector system: 2

Hierarchical types for nodes
- Intersection
- Slip road
- Departure
- Motor intersection

Sector number for nodes
- Intersection: Sector number is equal to routing system number.
- Sliproad, Departure, Motorway intersection: Routing system number is 1.

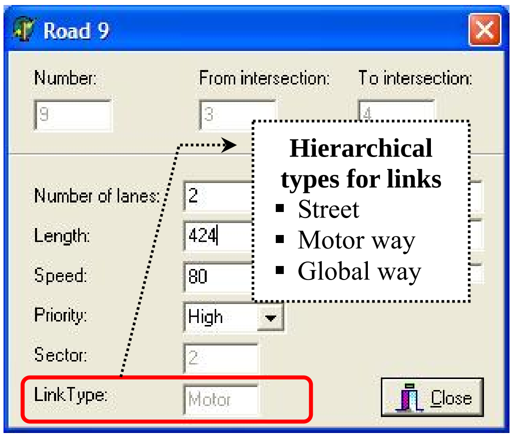
Figure 37: Setting hierarchical attributes for nodes



Road 39

Number: 39
From intersection: 11
To intersection: 12
Number of lanes: 1
Length: 464
Destination rate: 100
Speed: 50
Remove when saving: ☐
Priority: Low
Sector: 2
LinkType: Street

Sector number for links



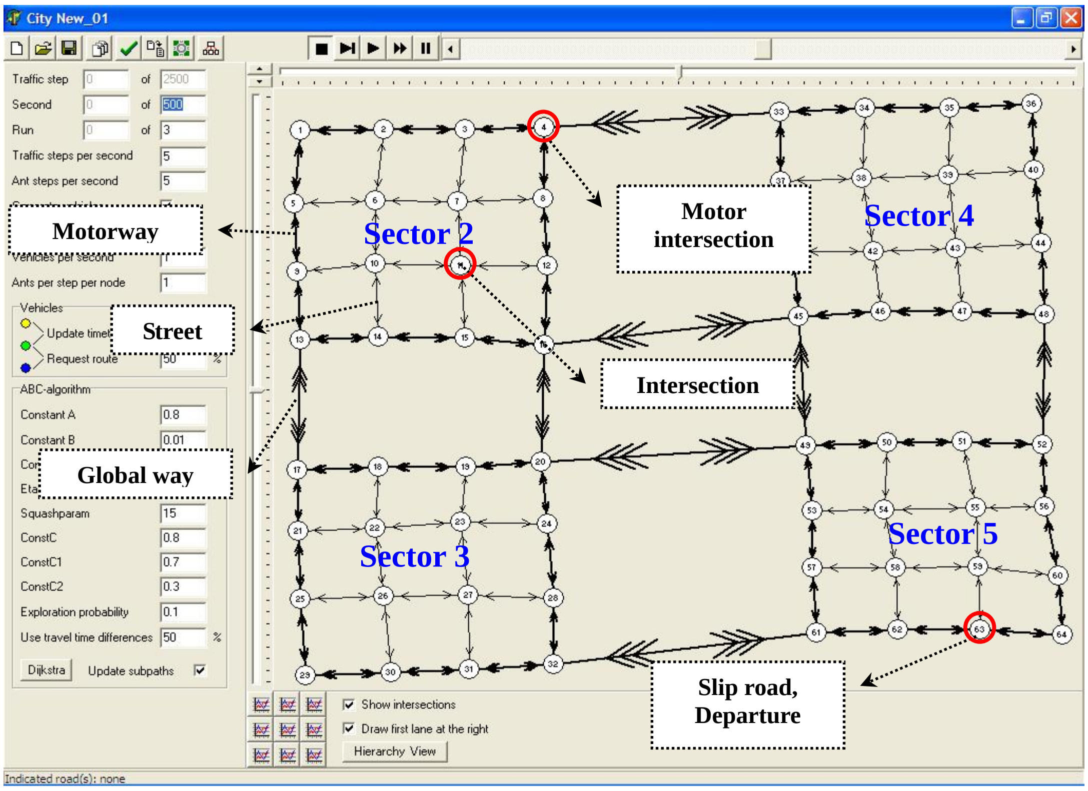
Road 9

Number: 9
From intersection: 3
To intersection: 4
Number of lanes: 2
Length: 424
Speed: 80
Priority: High
Sector: 2
LinkType: Motor

Hierarchical types for links
- Street
- Motor way
- Global way

Figure 38: Setting hierarchical attributes for links

As we described in the former section, hierarchical types for links are dynamic and will be set after '.city' file load automatically by city program. Let's see the entire hierarchical network in the city program.


City New_01

Traffic step 0 of 2500
Second 0 of 500
Run 0 of 3
Traffic steps per second 5
Ant steps per second 5
Vehicles per second 1
Ants per step per node 1
Vehicles
Update timet
Request route 50 %
ABC-algorithm
Constant A 0.8
Constant B 0.01
Squashparam 15
ConstC 0.8
ConstC1 0.7
ConstC2 0.3
Exploration probability 0.1
Use travel time differences 50 %
Update subpaths ☑
☑ Show intersections
☑ Draw first lane at the right
Indicated road(s): none

Figure 39: Entire hierarchical traffic network in city program

4.3.2 Showing hierarchical information

We also developed a function to let the city program to tell users detailed information about the hierarchical traffic network.

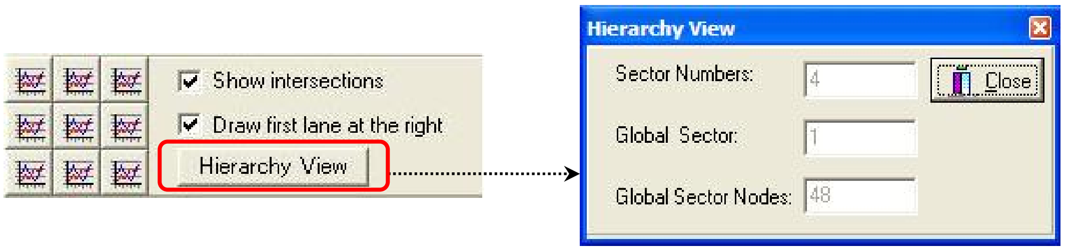

☑ Show intersections
☑ Draw first lane at the right
Hierarchy View
Sector Numbers: 4
Global Sector: 1
Global Sector Nodes: 48

Figure 40: Hierarchical information about network

From the figure-40 above, we can see that there exist four city sectors (2~5) in our network and only one global sector (sector 1, abstract level network). Additionally, we also knew that there are 48 nodes located on the motorway. This information will be computed when '.city' file is loaded.

4.3.3 Collecting and sending route information

The city program is responsible for simulating the traffic in the real world. The vehicles will collect the information about a route, which is covered by the vehicle, and send to the corresponding routing information to update timetable. The dynamic information is different from Kroon's program. The vehicles must provide the times when motorway was entered, the motorway was left respectively and a new city network was entered. To satisfy these requirements, we must make some changes for vehicle move. We will present the changes in following flow chart.

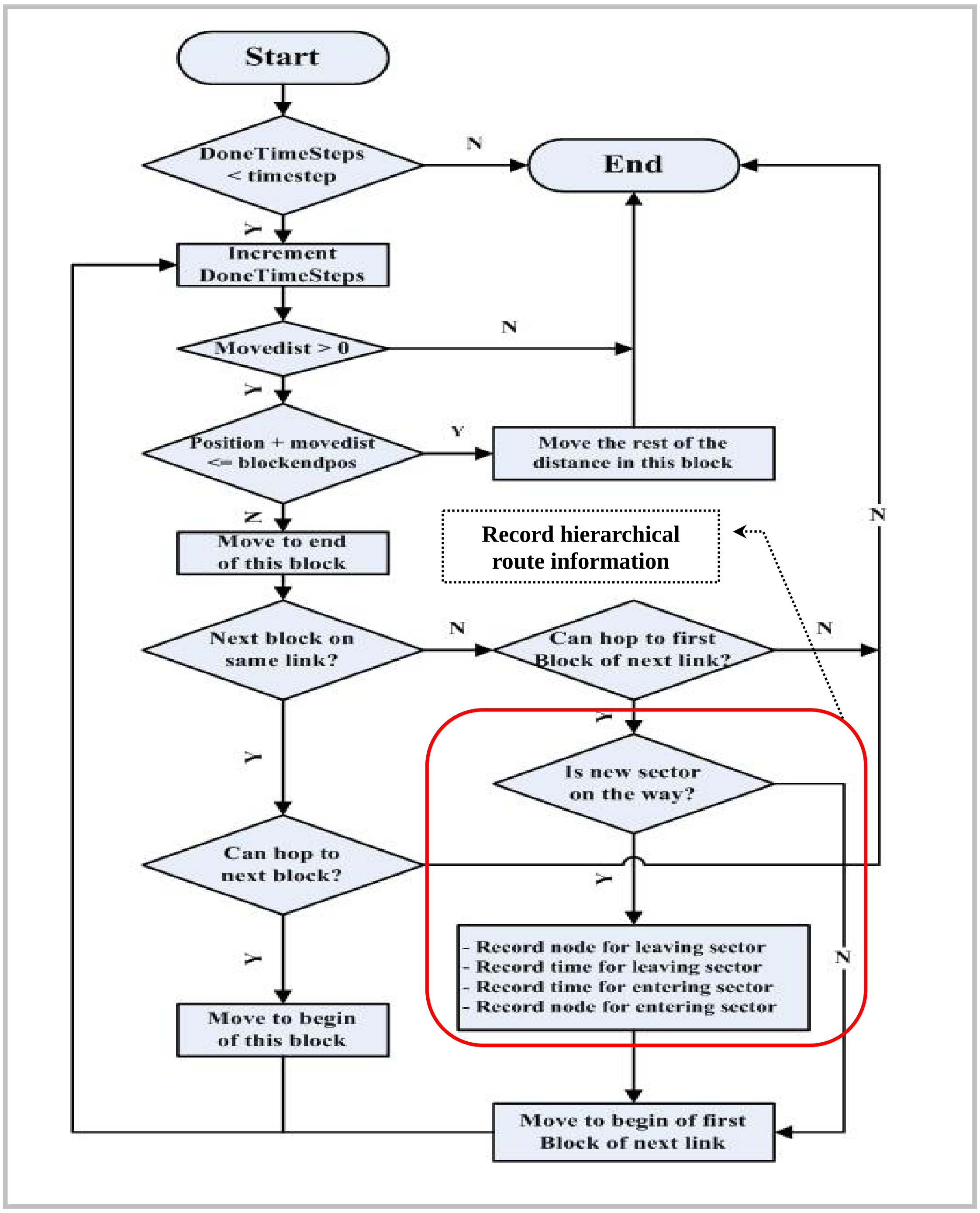

Start
DoneTimeSteps < timestep
N
End
Y
Increment DoneTimeSteps
Movedist > 0
N
Y
Position + movedist <= blockendpos
Y
Move the rest of the distance in this block
N
Move to end of this block
Record hierarchical route information
N
Next block on same link?
N
Can hop to first Block of next link?
N
Y
Is new sector on the way?
Y
Y
Can hop to next block?
Y
- Record node for leaving sector
- Record time for leaving sector
- Record time for entering sector
- Record node for entering sector
N
Move to begin of this block
Move to begin of first Block of next link

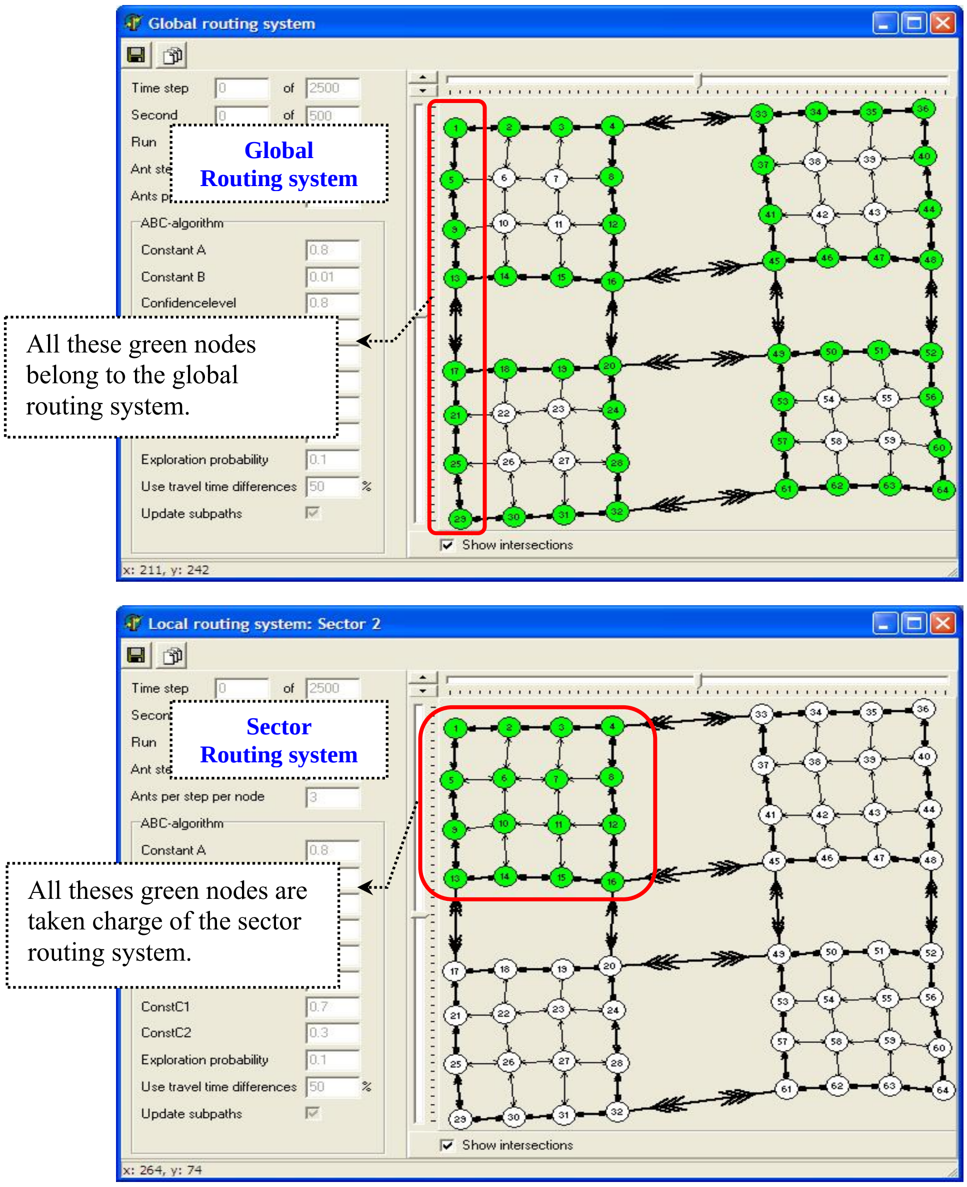
Figure 41: Flow chart of the vehicle movement in hierarchical environment

4.4 Hierarchical routing system

Based on basic knowledge of hierarchical routing system, we know that the procedure of optimal route computing is quite different from the conventional routing system (Kroon's program). In this section, we will show how we implement the hierarchical routing system in our software prototype.

4.4.1 Routing systems activation

When hierarchical attributes setting in the city program was finished, the routing systems will be activated. Based on the hierarchical view, each city network will activate one sector routing system (detailed level routing system) and the whole city networks will activate only one global routing system (abstract level routing system). The figure-42 shows the situation after all routing systems have been activated.


Global Routing system
All these green nodes belong to the global routing system.
Show intersections
Sector Routing system
All theses green nodes are taken charge of the sector routing system.
Show intersections

Global routing system

Time step	0	of	2500
Second	0	of	500

ABC-algorithm

Constant A	0.8
Constant B	0.01
Confidencelevel	0.8
Exploration probability	0.1
Use travel time differences	50 %
Update subpaths	✓

x: 211, y: 242

Local routing system: Sector 2

Time step	0	of	2500
Ants per step per node	3		

ABC-algorithm

Constant A	0.8
ConstC1	0.7
ConstC2	0.3
Exploration probability	0.1
Use travel time differences	50 %
Update subpaths	✓

x: 264, y: 74

Figure 42: Global and sector routing system

Nodes, which are responsible by the corresponding routing system, have a green color. Nodes on the motorway are taken care of global and sector routing systems. Based on the example network, displayed above, there are one global routing system and four sector routing systems.

4.4.2 Virtual node generating

In a hierarchical routing system, we must create a virtual node for every city network. Based on the example shown in figure-39, we need to create four virtual nodes for the city networks (sector 2 ~ sector4). Each virtual node must know how many real nodes it includes. In our program, virtual nodes are created when the city network is loaded. The following flow chart shows the procedure of virtual nodes creation.

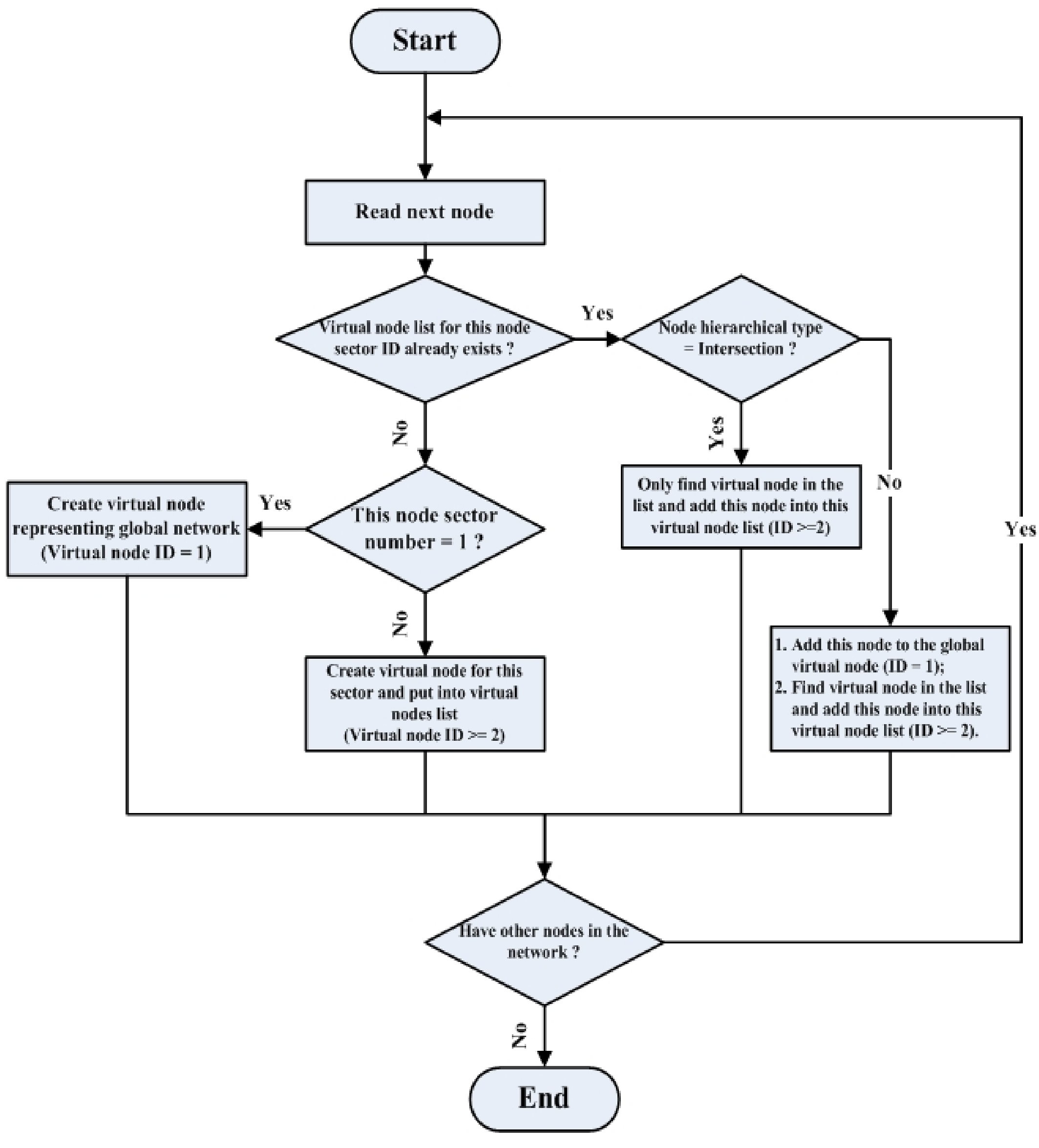
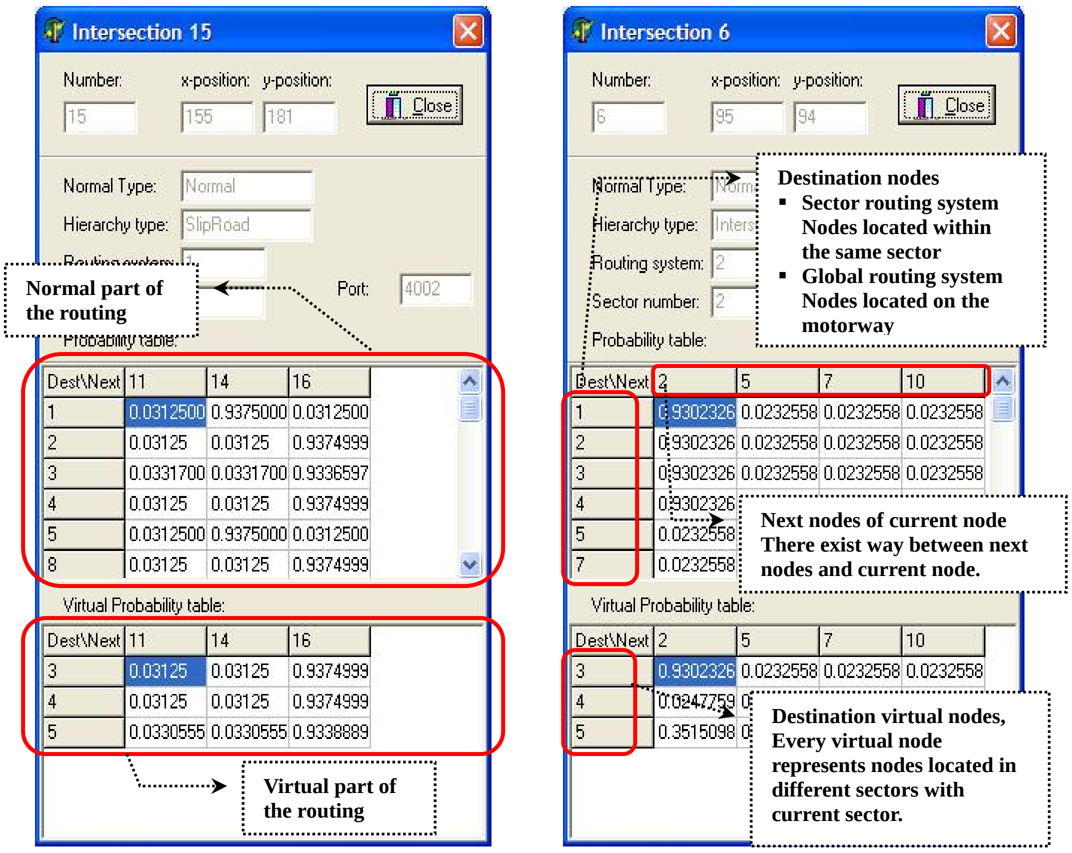

Start
Read next node
Virtual node list for this node sector ID already exists ?
Yes
Node hierarchical type = Intersection ?
No
Yes
Only find virtual node in the list and add this node into this virtual node list (ID >=2)
No
Create virtual node representing global network (Virtual node ID = 1)
Yes
This node sector number = 1 ?
Yes
No
Create virtual node for this sector and put into virtual nodes list (Virtual node ID >= 2)
1. Add this node to the global virtual node (ID = 1);
2. Find virtual node in the list and add this node into this virtual node list (ID >= 2).
Have other nodes in the network ?
No
End

Figure 43: Generating virtual nodes

4.4.3 Routing table

- ***Creation of the routing table***

The most important data structure in Ant Based Control algorithm is the routing table. Every node must own one routing table (global routing table or sector routing table) in a routing system. We already gave a clear definition about routing table in former Chapter 3. In our routing system routing table is made up of two parts, a normal part and a virtual part. The virtual parts represent nodes in different distant sectors. Nodes on the motorway can have a global routing system in the global routing system and a sector routing table in the corresponding sector routing system. Destination nodes in one routing table must belong to the same hierarchical level as the routing table owner. That is, in the global routing system, destination nodes must be located on the motorway. In the sector routing system, destination nodes must belong to the same sector as the routing table owner. We create a routing table for every node when the routing system is activated. Routing table structure can be displayed using the following figure-44.


Intersection 15
Close
Normal part of the routing
Virtual part of the routing
Intersection 6
Close
Destination nodes
Sector routing system Nodes located within the same sector
Global routing system Nodes located on the motorway
Next nodes of current node There exist way between next nodes and current node.
Destination virtual nodes, Every virtual node represents nodes located in different sectors with current sector.

Intersection 15

Number:	15
x-position:	155
y-position:	181
Normal Type:	Normal
Hierarchy type:	SlipRoad
Routing system:	1
Port:	4002

Probability table:

Dest\Next	11	14	16
1	0.0312500	0.9375000	0.0312500
2	0.03125	0.03125	0.9374999
3	0.0331700	0.0331700	0.9336597
4	0.03125	0.03125	0.9374999
5	0.0312500	0.9375000	0.0312500
8	0.03125	0.03125	0.9374999

Virtual Probability table:

Dest\Next	11	14	16
3	0.03125	0.03125	0.9374999
4	0.03125	0.03125	0.9374999
5	0.0330555	0.0330555	0.9338889

Intersection 6

Number:	6
x-position:	95
y-position:	94
Normal Type:	Norm
Hierarchy type:	Inters
Routing system:	2
Sector number:	2

Probability table:

Dest\Next	2	5	7	10
1	0.9302326	0.0232558	0.0232558	0.0232558
2	0.9302326	0.0232558	0.0232558	0.0232558
3	0.9302326	0.0232558	0.0232558	0.0232558
4	0.9302326	[illegible]	[illegible]	[illegible]
5	0.0232558	[illegible]	[illegible]	[illegible]
7	0.0232558	[illegible]	[illegible]	[illegible]

Virtual Probability table:

Dest\Next	2	5	7	10
3	0.9302326	0.0232558	0.0232558	0.0232558
4	0.0247759	[illegible]	[illegible]	[illegible]
5	0.3515098	[illegible]	[illegible]	[illegible]

Figure 44: Global (left) and Sector (right) routing tables

In our program, according to the routing system type and node hierarchical type, the system could produce the corresponding routing table for the node. If current routing system is a global routing system, the system will create global routing tables for nodes except for inner intersections in the traffic network; and if the current routing system is a sector routing system, the system will create sector routing tables for nodes in this sector. Nodes located on the motorway of city network will have one sector routing table in sector routing system and one global routing table in the global routing system. We give a flow chart of the creation of routing tables.

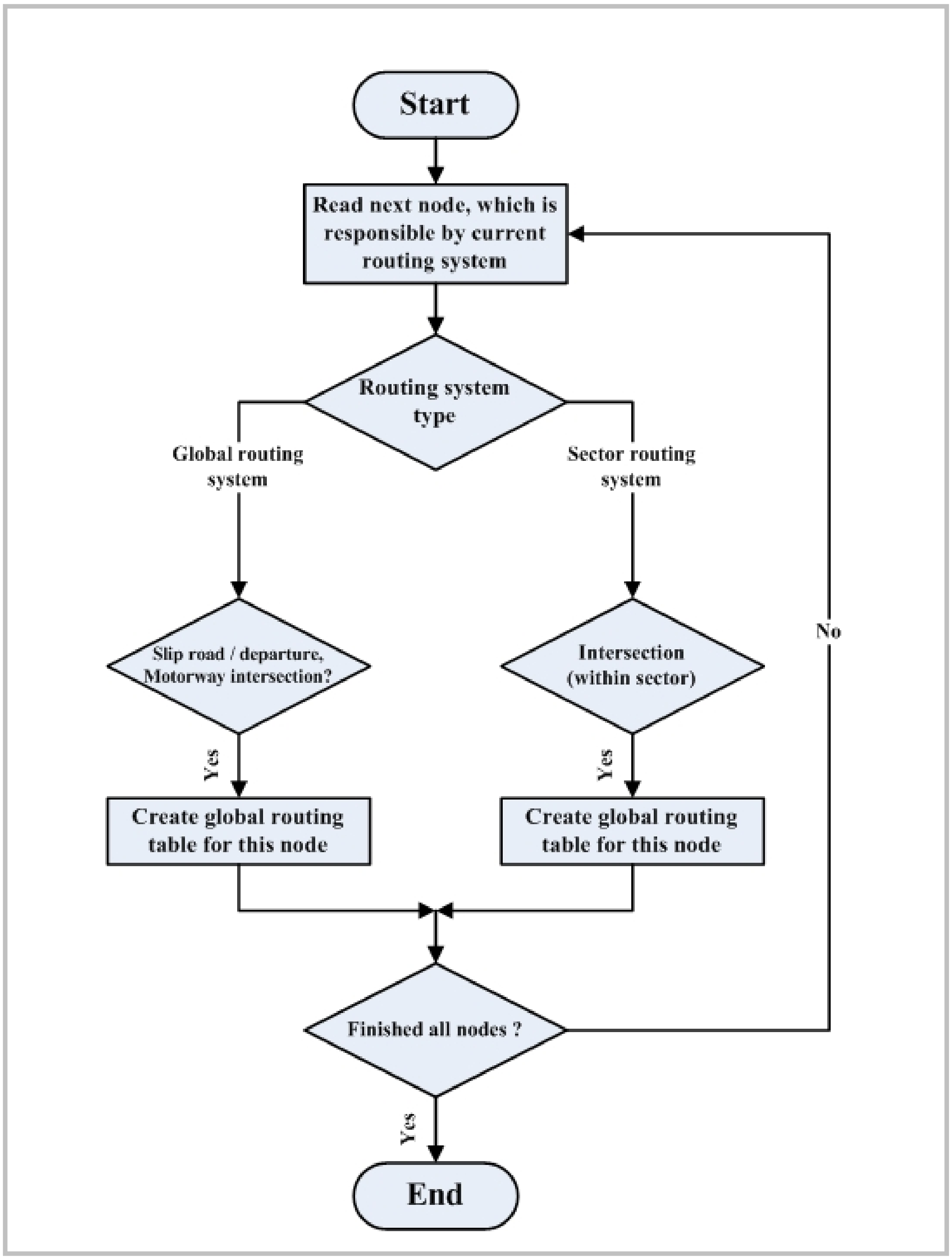
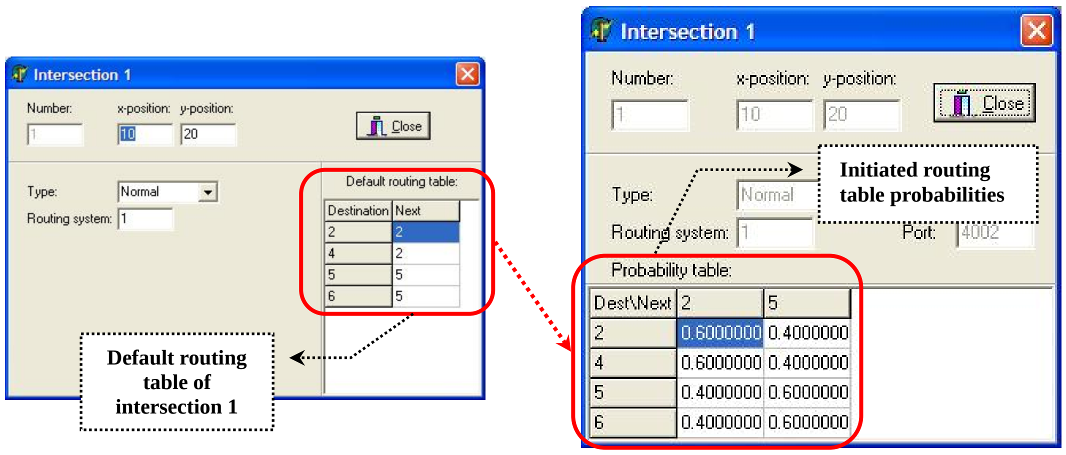

Start
Read next node, which is responsible by current routing system
Routing system type
Global routing system
Sector routing system
Slip road / departure, Motorway intersection?
Intersection (within sector)
No
Yes
Yes
Create global routing table for this node
Create global routing table for this node
Finished all nodes ?
Yes
End

Figure 45: Creating routing table for node

- ***Routing table initiation***

In Kroon's program, the probabilities in a routing table are initiated using a default routing table created in the city program corresponding to Dijkstra's algorithm. The next node, which is in the default routing table, will have a higher probability. In the hierarchical routing system, the probability is initiated according to the type of routing system, where the current node is located. That is, in a global routing system, next node on the motorway will get the higher probability; in the sector routing system, next node on the street will get the higher probability. In our program, we will assign the probability value based on network situations. The following screenshot will show such difference.


Intersection 1

Number: 1 x-position: 10 y-position: 20

Type: Normal

Routing system: 1

Default routing table:

Destination	Next
2	2
4	2
5	5
6	5

Default routing table of intersection 1

Intersection 1

Number: 1 x-position: 10 y-position: 20

Type: Normal

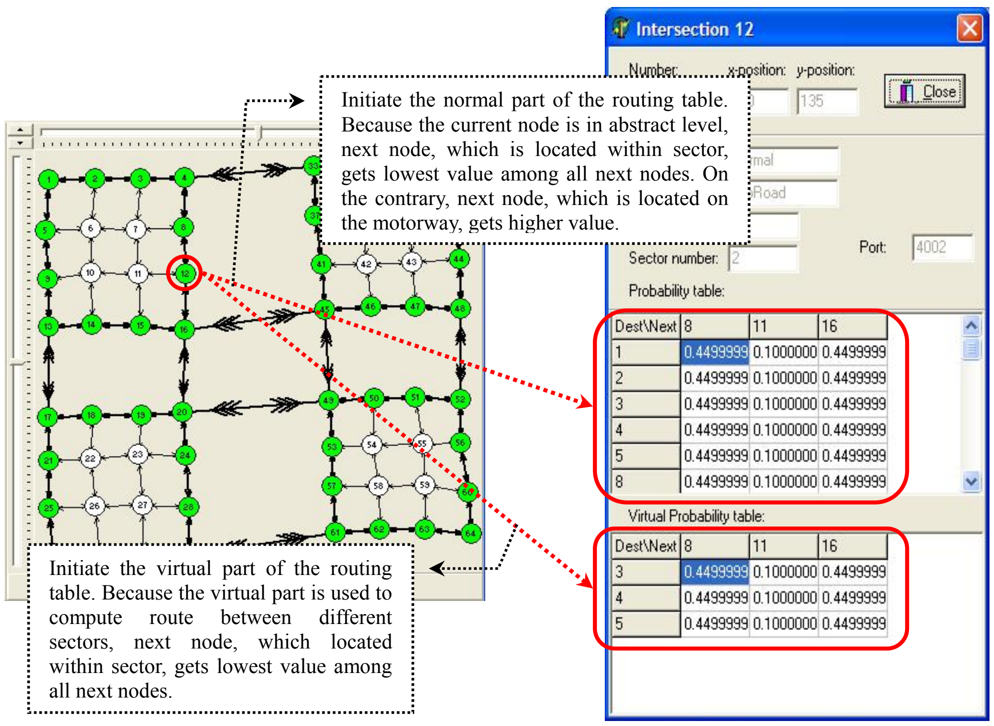
Routing system: 1 Port: 4002

Initiated routing table probabilities

Probability table:

Dest\Next	2	5
2	0.6000000	0.4000000
4	0.6000000	0.4000000
5	0.4000000	0.6000000
6	0.4000000	0.6000000

Figure 46: Routing table initiating in Kroon's program


Initiate the normal part of the routing table. Because the current node is in abstract level, next node, which is located within sector, gets lowest value among all next nodes. On the contrary, next node, which is located on the motorway, gets higher value.

Initiate the virtual part of the routing table. Because the virtual part is used to compute route between different sectors, next node, which located within sector, gets lowest value among all next nodes.

Intersection 12

Number: x-position: y-position: 135

Sector number: 2 Port: 4002

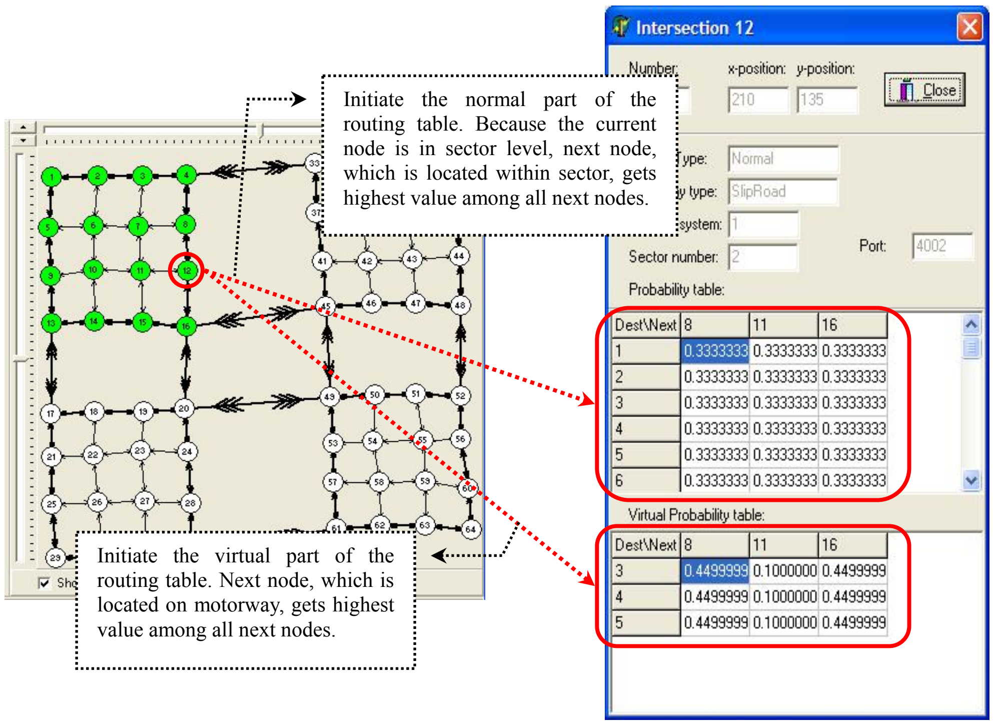
Probability table:

Dest\Next	8	11	16
1	0.4499999	0.1000000	0.4499999
2	0.4499999	0.1000000	0.4499999
3	0.4499999	0.1000000	0.4499999
4	0.4499999	0.1000000	0.4499999
5	0.4499999	0.1000000	0.4499999
8	0.4499999	0.1000000	0.4499999

Virtual Probability table:

Dest\Next	8	11	16
3	0.4499999	0.1000000	0.4499999
4	0.4499999	0.1000000	0.4499999
5	0.4499999	0.1000000	0.4499999

Figure 47: Initiating global routing table



Initiate the normal part of the routing table. Because the current node is in sector level, next node, which is located within sector, gets highest value among all next nodes.

Initiate the virtual part of the routing table. Next node, which is located on motorway, gets highest value among all next nodes.

Intersection 12

Number: x-position: 210 y-position: 135

Type: Normal

y type: SlipRoad

system: 1

Sector number: 2 Port: 4002

Probability table:

Dest\Next	8	11	16
1	0.3333333	0.3333333	0.3333333
2	0.3333333	0.3333333	0.3333333
3	0.3333333	0.3333333	0.3333333
4	0.3333333	0.3333333	0.3333333
5	0.3333333	0.3333333	0.3333333
6	0.3333333	0.3333333	0.3333333

Virtual Probability table:

Dest\Next	8	11	16
3	0.4499999	0.1000000	0.4499999
4	0.4499999	0.1000000	0.4499999
5	0.4499999	0.1000000	0.4499999

Figure 48: Initiating sector routing table

From screenshots above, we could see that because intersection 12 located on motorway, it has two routing tables (global routing table and sector routing table).

4.4.4 Intelligent agents generating and movement

In the hierarchical routing system, the routing systems are divided into two categories. One is the global routing system, which takes charge of the abstract level network; the other one is the sector routing system, which is responsible for detailed level network. In our simulation, we divided intelligent agents into two types according to different routing systems. They are global agents and sector agents. Global agents only move in the abstract level network, and sector agents move with a sector. In Kroon’s program, agents are generated at nodes with random destination (any nodes in the traffic network). We must make some adjustments on agents’ generation and movement.

- ***Global routing system (abstract level network)***
  a) Only generate agents (global ants) at nodes (source nodes), which are located on abstracted level network. These nodes could be *sliproad, departure, motorway intersection*.
  b) Destination nodes of agents must locate on the abstracted level network.
  c) Agents move according to the global routing tables.
- ***Sector routing system (detailed level network)***
  a) Only generate agents (sector ants) at nodes, which are located on a detailed level network. These nodes should belong to this detailed level sector.
  b) Destination nodes of agents must be located within the same sector as the source nodes.
  c) Agents move according to the sector routing tables.

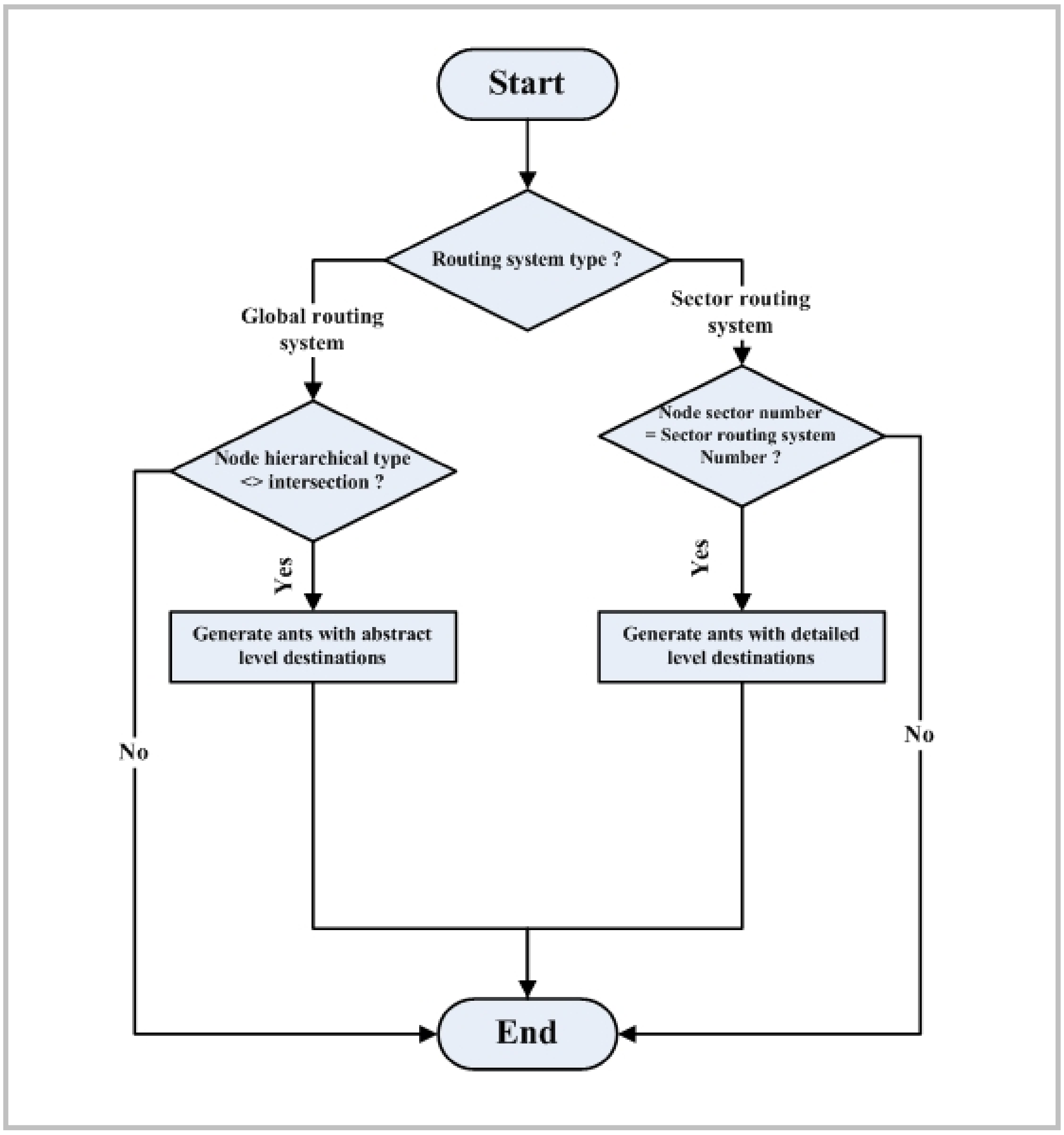

Start
Routing system type ?
Global routing system
Sector routing system
Node hierarchical type <> intersection ?
Node sector number = Sector routing system Number ?
Yes
Yes
Generate ants with abstract level destinations
Generate ants with detailed level destinations
No
No
End

Figure 49: Agents generating

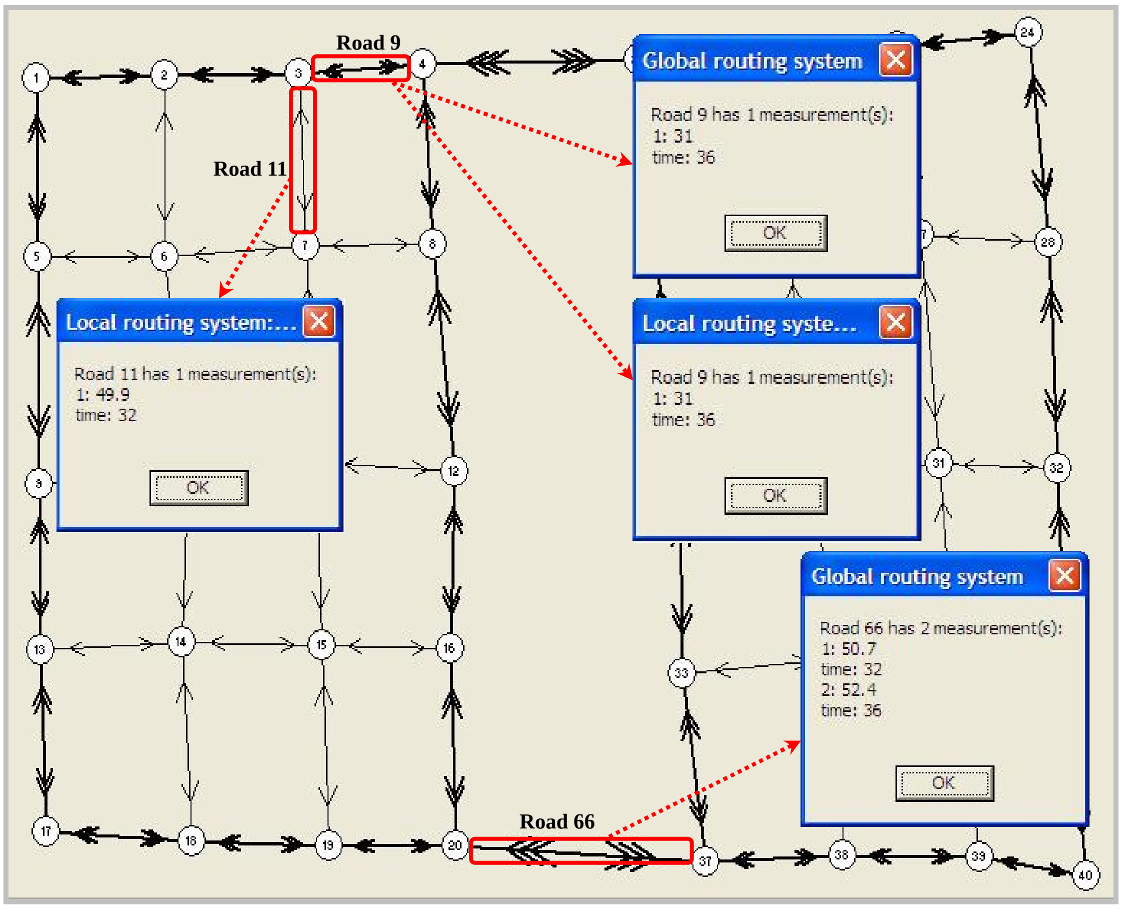
4.4.5 Hierarchical timetable updating

When the routing system received the update information, which is sent by vehicles, it will update the corresponding links based on the hierarchical view. How to update timetable depends on the hierarchical type of links.

Table 21: Timetable updating

Hierarchical type of link	Routing system action
Global	Charged by global routing system
Motorway	Charged by corresponding sector routing system and global routing system
Street	Charged by corresponding sector routing system

The following screenshot shows the procedure of the timetable updating.



Road 9 has 1 measurement(s):
1: 31
time: 36

Road 11 has 1 measurement(s):
1: 49.9
time: 32

Road 9 has 1 measurement(s):
1: 31
time: 36

Road 66 has 2 measurement(s):
1: 50.7
time: 32
2: 52.4
time: 36

Figure 50: Time table updating

From the figure above, we can see that road 9 is motorway and located within the left sector. When a vehicle send the travel time information of road 9 to the routing systems, both the global timetable updating system and sector timetable updating system will compute the travel time measurement for the road 9. The road 11 is the street and located within the left sector. Then only sector timetable updating system will take charge of this road. The road 66 is the global way and located between these two sectors, as a result, only the global timetable updating system will be responsible for it.

4.4.6 Hierarchical routing table updating

Hierarchical routing table updating is different from the conventional routing system (ABC routing). Global routing system takes care of nodes located on motorway; the sector routing system is responsible for nodes located within one sector. As we explained before, global agents only move in abstract level network and sector agents only move in detailed level network.

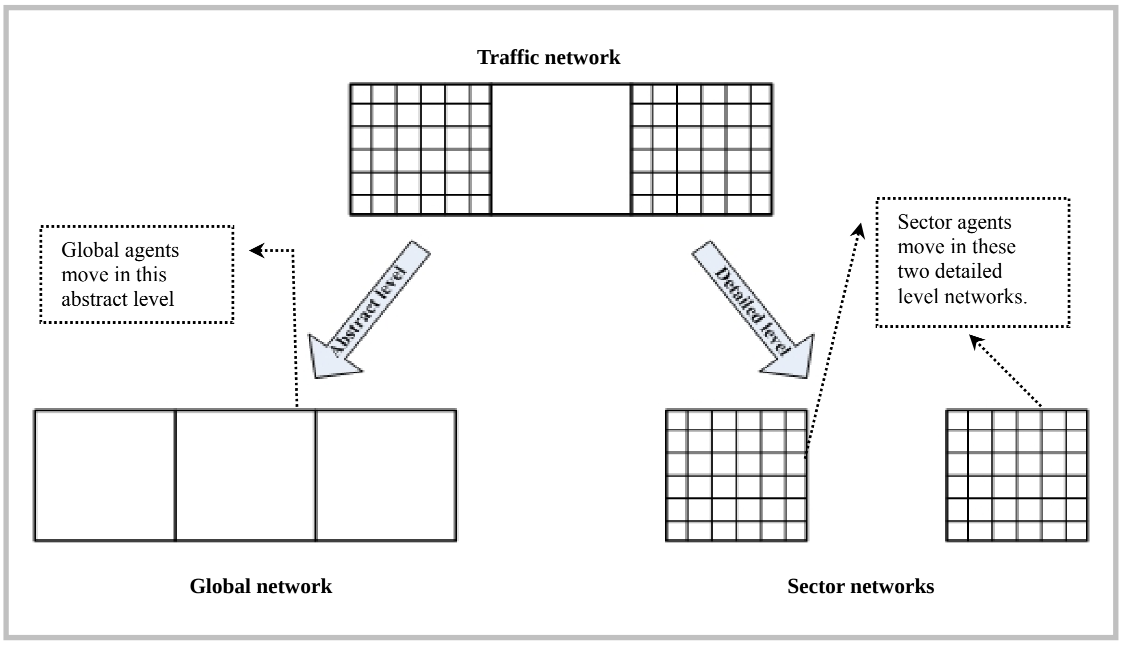

Traffic network
Global agents move in this abstract level
Abstract level
Detailed level
Sector agents move in these two detailed level networks.
Global network
Sector networks

Figure 51: Agents movement in hierarchical network

We need to explain sector agents' movement in detail. Because they only move within a sector, how they get necessary information to update virtual part of the routing table? In our system, when sector agents' destination is a node located on the motorway of the city network, they will request virtual travel time from this node (virtual travel time means travel time from this node to other remote sectors). By virtual travel time, sector ants could update the virtual part of the routing table.

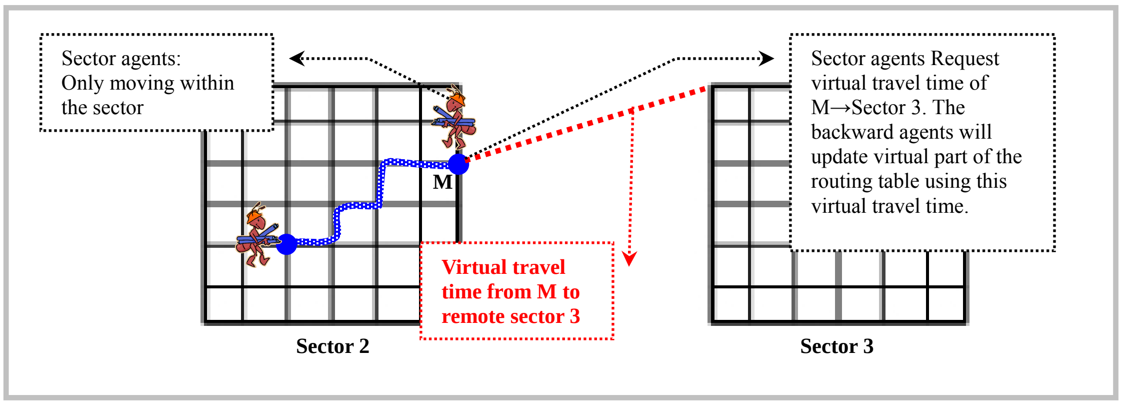
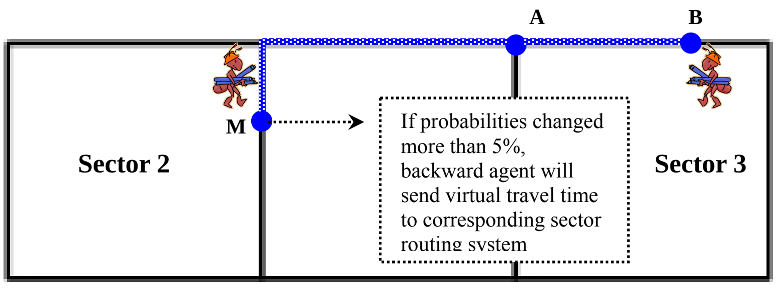

Sector agents: Only moving within the sector
Sector agents Request virtual travel time of M→Sector 3. The backward agents will update virtual part of the routing table using this virtual travel time.
M
Virtual travel time from M to remote sector 3
Sector 2
Sector 3

Figure 52: Sector agents request virtual travel time

As we know, sector agents only move within a sector. So the virtual travel time will be provided by the global routing system through message passing. If the global routing system found that probabilities in routing table have been changed more than 5% than before, it would record such difference and send it to the corresponding sector routing system. Based on the example traffic network above, virtual time of M is provided by global routing system.


A
B
M
Sector 2
Sector 3
If probabilities changed more than 5%, backward agent will send virtual travel time to corresponding sector routing system

- Global agent will record the travel from M to A (this is virtual travel time from M to sector 3).
- When global agent reached destination B, it will change to backward agent and go back to M.
- When backward agent arrived M (source node), it will update the virtual part of the routing table of M. If backward agent found probabilities in routing table of M changed more than 5%, it would send this virtual travel time to the corresponding node in the sector routing system.

Figure 53: Global agents send virtual travel time to sector routing system

4.4.7 Hierarchical optimal route computation

In our program, the routing system could be divided into two types according to the hierarchical view. One is a global routing system (responsible for abstract level network); the other is a sector routing system (responsible for detailed level network). How these two kinds of routing systems cooperate is explained in the former chapter. We make some necessary adjustments based on hierarchical routing algorithm. We know that in Kroon’s program distributed routing systems have the same functionality and they are responsible for several nodes in the network. When a vehicle, in city program, requests route from the routing system, the control center will send this request to the corresponding routing system. The following figure shows optimal route information from source node A to destination node F. The corresponding routing system number is displayed on top of the node. For example, routing system 2 is responsible for node B and C.

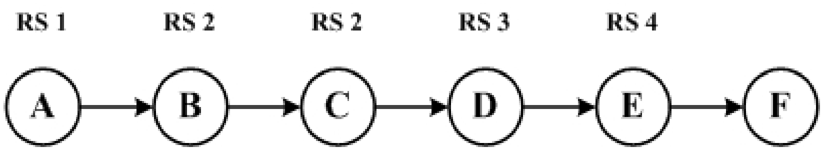


Figure 54: Example optimal route computed by routing systems

Let’s see the detailed computation procedure based on the route above. In the city program, a vehicle requests the route information from source node A to destination node F.

1) Request is sent to routing system 1 and node B is returned (current node is B);
2) Request is sent to routing system 2 and node C is returned (current node is C);
3) Request is sent to routing system 2 and node D is returned (current node is D);
4) Request is sent to routing system 3 and node E is returned (current node is E);
5) Request is sent to routing system 4 and node F is returned (current node is F);
6) Destination node F is reached and computation finish.

We must state that, node B and C both are charged by the routing system 2. Request is sent twice to the routing system 2 and this makes message passing too frequently. In a hierarchical routing system, sector routing system is responsible for the whole nodes located within this sector and global routing system is responsible for the whole nodes located on the motorway. We must reduce request frequency in the same routing system.

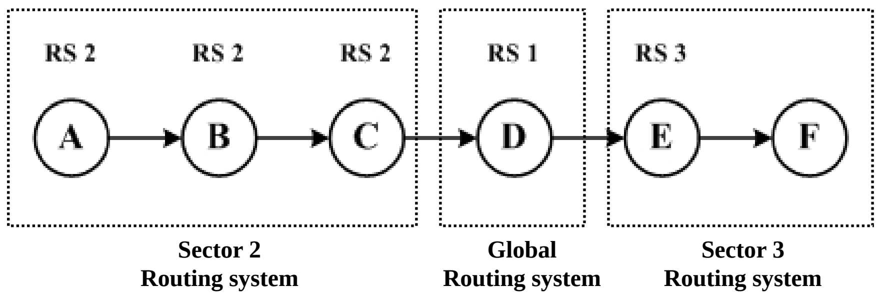


Figure 55: Example optimal route computed by hierarchical routing systems

Figure 52 shows route computation in hierarchical routing system. We could see that A located within sector 2 and destination node F located within the sector 3 (different sector with sector 2). The vehicle in the city program requests route from A to F.

1) Request is sent to sector routing system 2, and next nodes B, C and D are returned (current node is D);
2) Request is sent to the global routing system, and next node E is returned (current node is E);
3) Request is sent to sector routing system 3, and next node F is returned (current node is F);
4) Destination node F is reached and computation finish.

Comparing with computation procedure of Kroon's program, the message passing in our system is reduced drastically and then system could run more stably. When the routing systems receive request of route computation, they will give responses based on their hierarchical routing system type. Global routing system is only responsible for navigating between different sectors; sector routing system is responsible for navigating within a sector. The following figures show route computation by different routing systems.

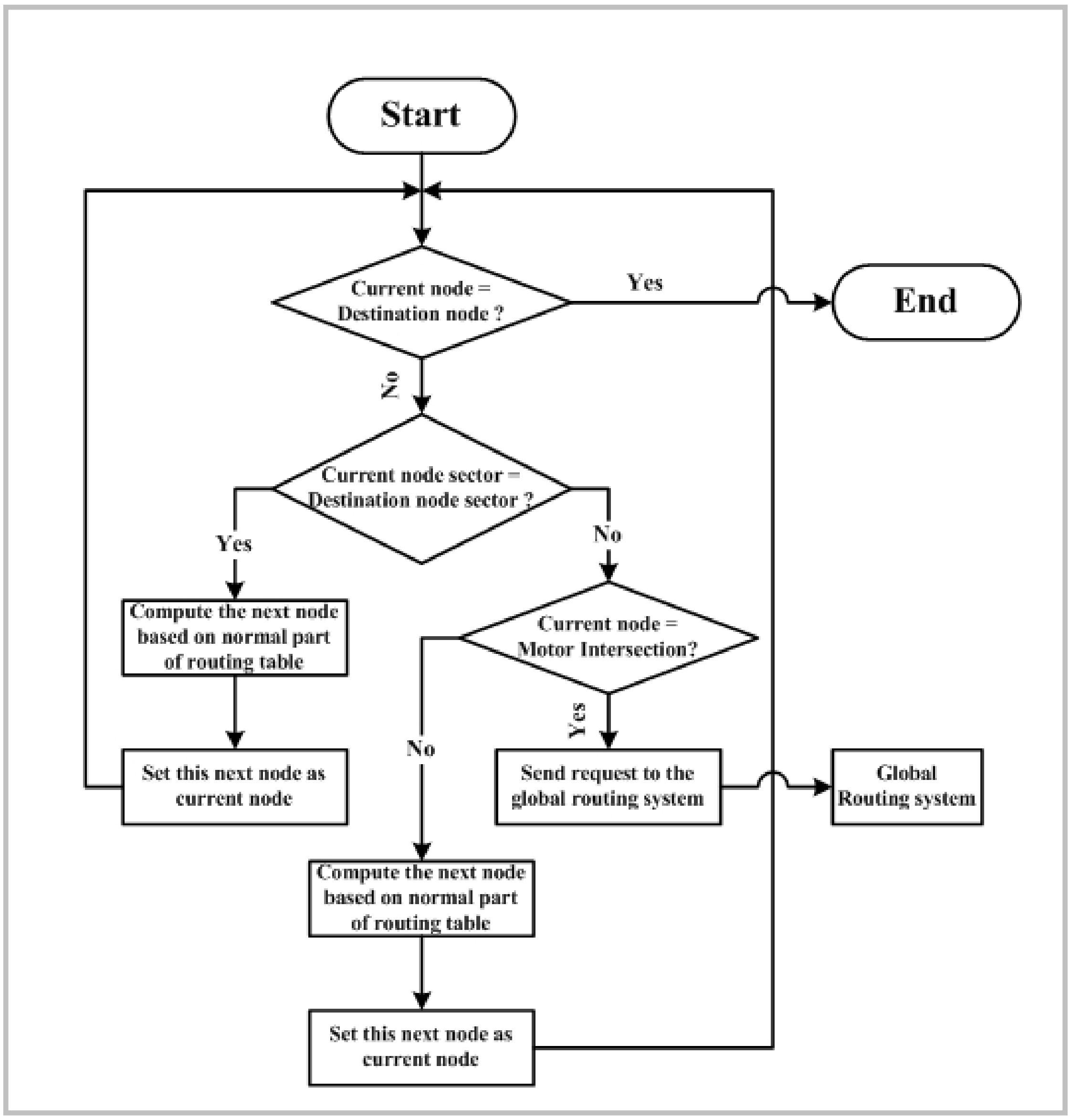

Start
Current node = Destination node ?
Yes
End
No
Current node sector = Destination node sector ?
Yes
No
Compute the next node based on normal part of routing table
Current node = Motor Intersection?
Yes
No
Set this next node as current node
Send request to the global routing system
Global Routing system
Compute the next node based on normal part of routing table
Set this next node as current node

Figure 56: Route computation of sector routing system

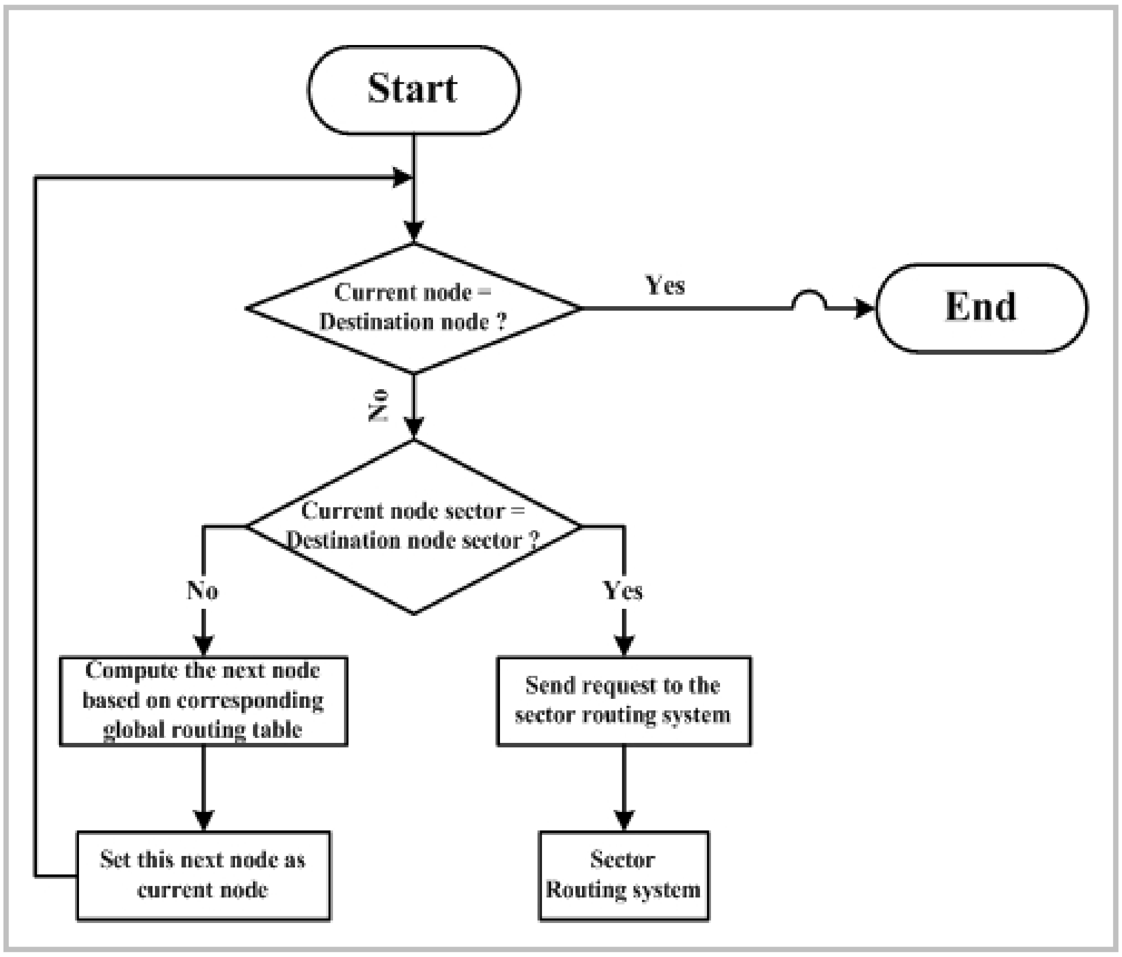
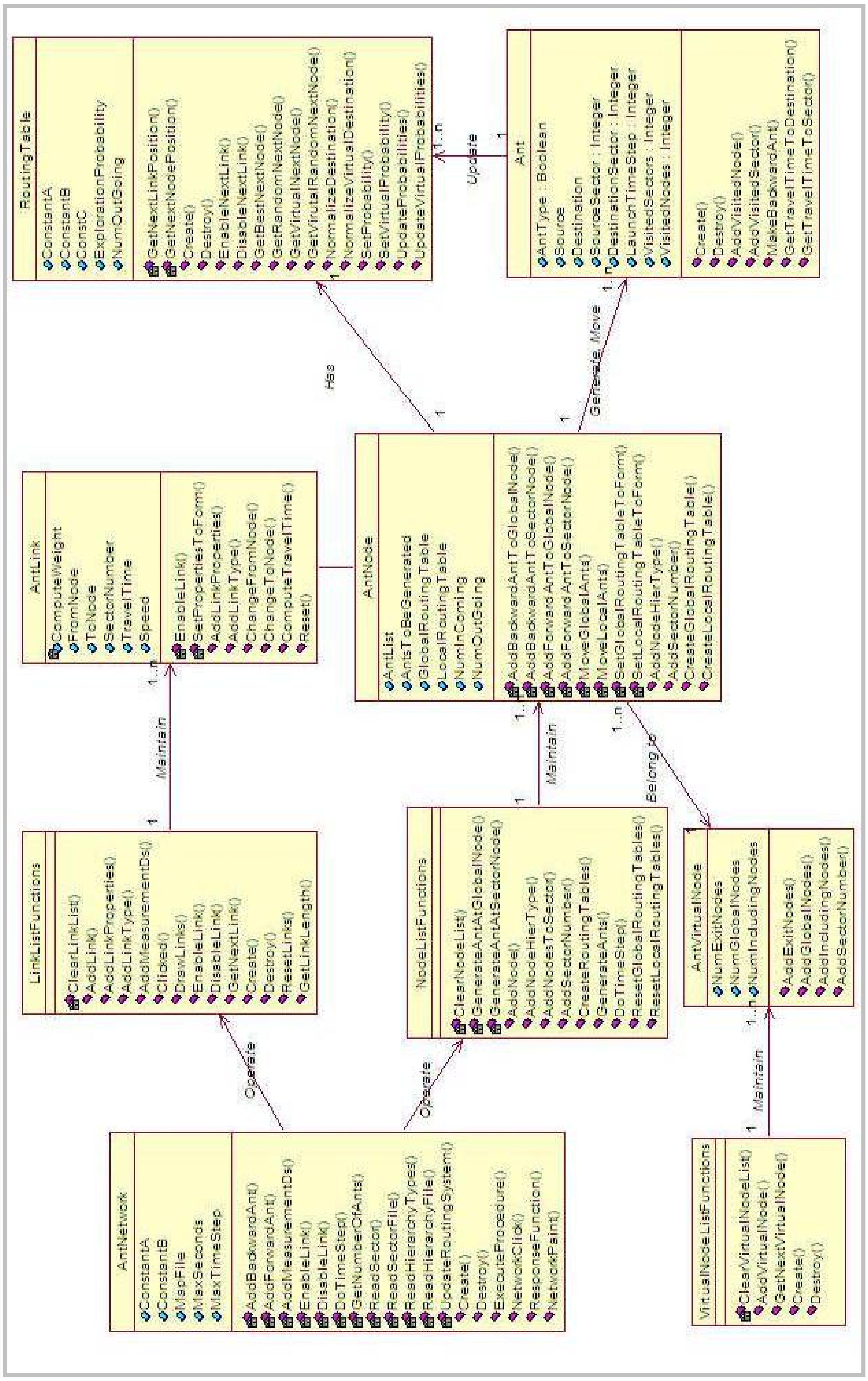

Start
Current node = Destination node ?
Yes
End
No
Current node sector = Destination node sector ?
No
Yes
Compute the next node based on corresponding global routing table
Send request to the sector routing system
Set this next node as current node
Sector Routing system

Figure 57: Route computation of global routing system

4.5 UML diagram

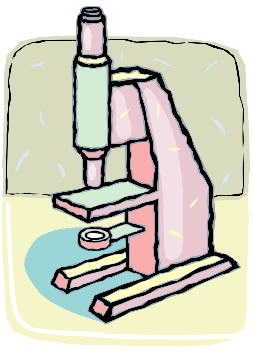
As we stated in the former chapter, our hierarchical routing system is developed based on Kroon's program. We make some necessary adjustments on the original program according to the hierarchical requirements. Then there will be many classes in our hierarchical routing system. Before we implement our hierarchical routing system, we must study these classes in very detail. We should know the relationship existing in these classes; we also need to master the functionalities of these classes.

In this master project, hierarchical routing is the central part of our work, and classes related to this part need to be improved for the hierarchical requirements. We will show the class diagram of these classes as follows.



The class diagram for the hierarchical routing system

Chapter 5 Experiments and Results



Based on the basic idea about hierarchical routing system, we developed a software prototype. To present characteristics of this hierarchical routing system, we designed some special experiments. In this chapter, we will discuss the results of these experiments. The experiments were executed in different environments and with different parameter settings. The quality and effectiveness of the hierarchical routing system could be shown through the results of experiments. The designed environments are different from traffic situations in the real city networks. This is mainly because the large realistic environments need a lot of computer system resource. In a hierarchical routing system, each independent sector is charged by a corresponding sector routing system, besides one global routing system takes care of the abstract level city network. In practice, city networks could have several sectors, including many intersections and streets. To simulate such complex traffic situation using our hierarchical routing system, we need a stronger computer system. For all experiments we will describe the goal, the environment with the settings, the expectations and the final results.

As we described before, our hierarchical routing system is developed from a conventional routing system (Kroon's program). We used the main structures of Kroon's program and make many necessary adjustments according to hierarchical routing requirements. Comparing to the conventional routing system, the hierarchical topology (abstract level and detailed level) could save more system resource when running simulations. Routing systems, built using this kind of hierarchical topology, could compute optimal route more quickly and more effective than conventional routing system when traffic network become more and more complicated.

The designed experiments focus on the following aspects:

- Comparing with conventional routing (ABC routing)

In this experiment, we will let conventional and hierarchical routing system execute one example traffic network. Through graphics and statistic data, we show the differences between these two types routing systems.

- Effectiveness and stability of a hierarchical routing system

The most obvious advantage of hierarchical routing system is that it divides one complex problem into several simple problems. In this experiment, we will use a little complex traffic network to test the effectiveness and stability of hierarchical routing system.

We will explain these experiments in more detail in the following sections.

5.1 Experiment 1: comparing with conventional routing

5.1.1 Goal

With this experiment we want to compare the hierarchical routing system with the conventional routing system. Through comparing, we could learn more about characteristics of hierarchical routing system. Firstly, let’s recall the differences between hierarchical and conventional routing system.

Hierarchical routing system

a) City program

In a hierarchical view, one real complex traffic network could be divided into several smaller city networks. Besides, one city network could be divided into several smaller sections. Thus, in a city program, every node and street must belong to certain sector.

b) Routing system

- There are two types of routing systems in a hierarchical routing system. One is a global routing system (abstract level routing system), which is responsible for nodes located on the motorway; the other a is sector routing system (detailed level routing system), which takes care of nodes located within certain sector (including nodes located on the ring of this sector).
- Global and sector routing systems will cooperate to compute optimal route for vehicle according to combination of source and destination node.

Conventional routing system

a) City program

In a conventional routing system, the traffic network is treated as a whole. There is no sector definition for city network, that is nodes and links have no sector attributes.

b) Routing system

Conventional routing system computes optimal routes in a distributed way and these distributed routing systems are equal in functionality.

5.1.2 Environment and settings

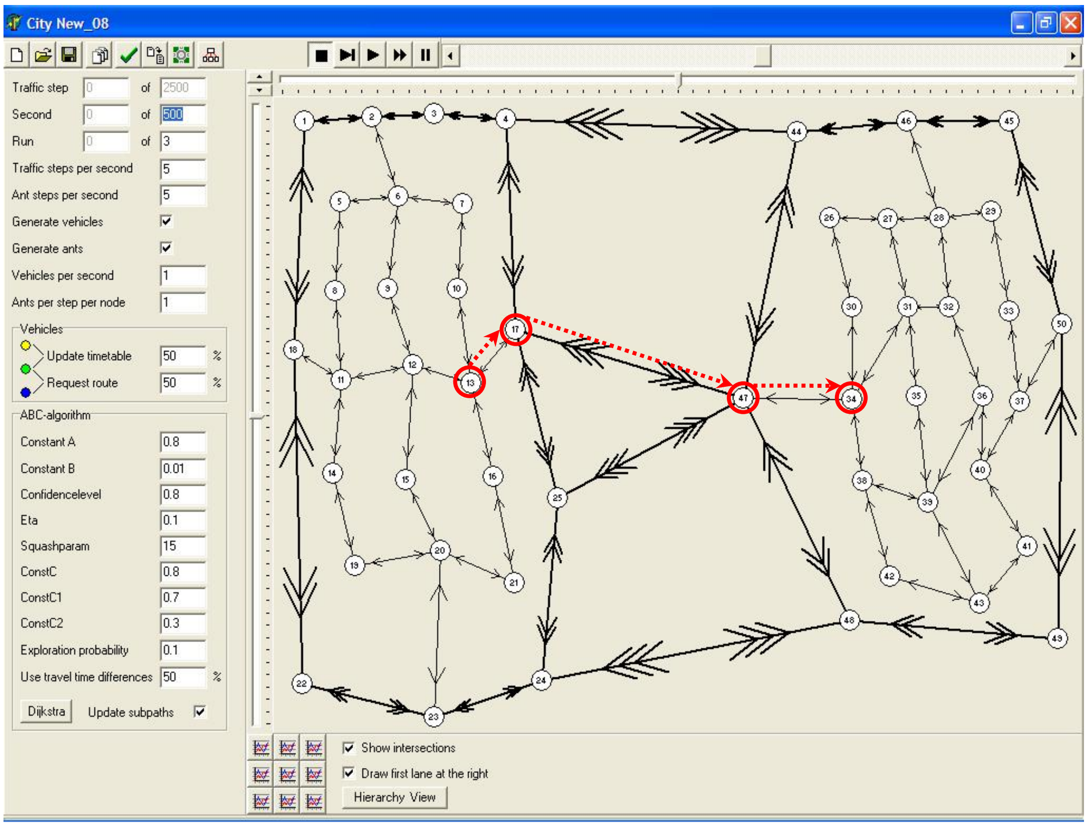
For a hierarchical routing system, the example traffic network is made up of two different sectors. And there exists some motorway connecting these two sectors. Correspondingly, it will have three routing systems, two sector routing systems and one global routing system.

For the conventional routing system, the example traffic network is only one complex city network. It will have three similar distributed routing systems.

We will initially use the default parameters for this simulation. These defaults are:

5.1.2.1 City program

- Traffic steps per second = 5
- Ant steps per second = 5
- Generate vehicles = True
- Generate ants = True
- Vehicles per second = 1.0
- Ants per step per node = 1.0
- Update timetable = 50%
- Request route = 50%
- Constant A = 0.8
- Constant B = 0.01
- Confidence level = 0.8
- Eta = 0.1
- Squashparam = 15
- ConstC = 0.8
- ConstC1 = 0.7
- ConstC2 = 0.3
- Exploration probability = 0.1
- Use travel time differences = 50%
- Update subpaths = True


City New_08

Traffic step	0	of	2500
Second	0	of	500
Run	0	of	3
Traffic steps per second	5		
Ant steps per second	5		
Generate vehicles	✓		
Generate ants	✓		
Vehicles per second	1		
Ants per step per node	1		

Vehicles

Update timetable	50	%
Request route	50	%

ABC-algorithm

Constant A	0.8	
Constant B	0.01	
Confidencelevel	0.8	
Eta	0.1	
Squashparam	15	
ConstC	0.8	
ConstC1	0.7	
ConstC2	0.3	
Exploration probability	0.1	
Use travel time differences	50	%

Dijkstra Update subpaths ✓

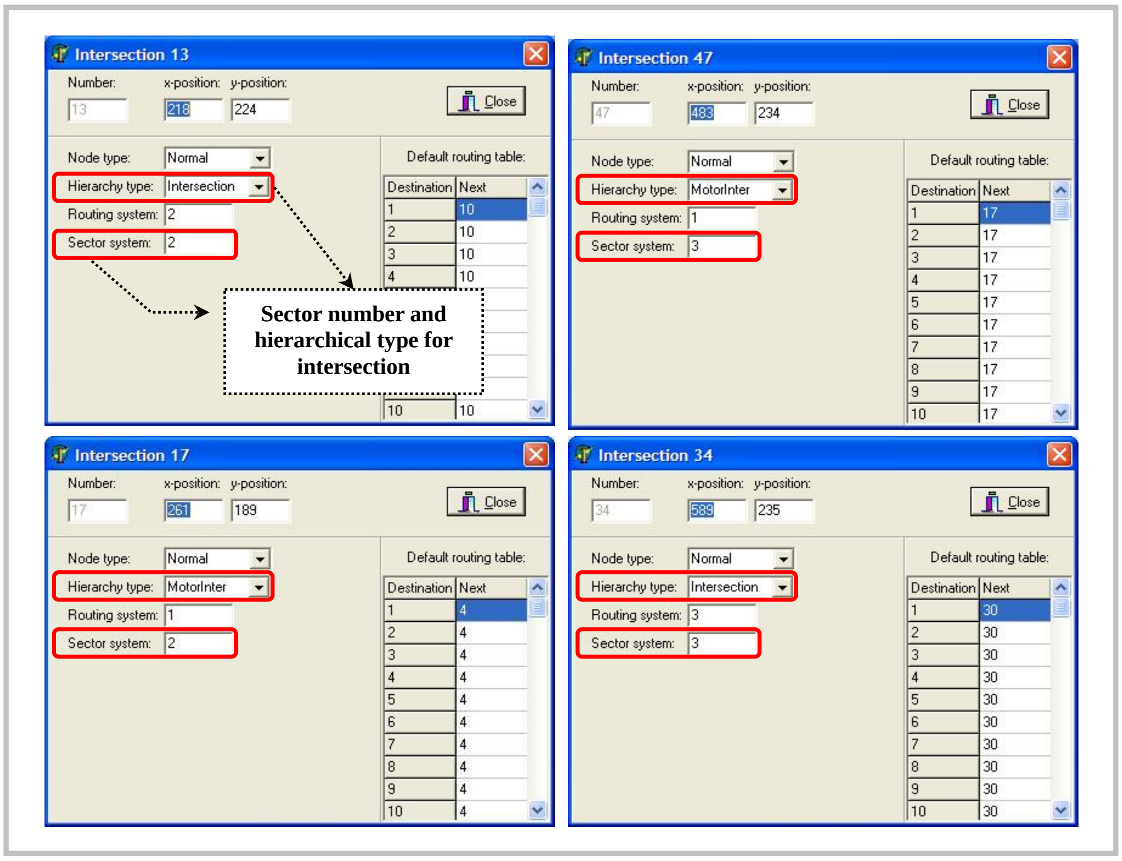
✓ Show intersections
✓ Draw first lane at the right
Hierarchy View

Figure 58: Traffic environment for experiment 1

We choose some intersections and roads from traffic environment to show the difference between hierarchical city program and conventional city program. These nodes and roads are:

$\boldsymbol{intersection13} \xrightarrow{road\,43} \boldsymbol{intersection17} \xrightarrow{road\,53} \boldsymbol{intersection47} \xrightarrow{road\,104} \boldsymbol{intersection34}$

Following pictures will display this difference.



Intersection 13

Number: 13 x-position: 218 y-position: 224

Node type: Normal
Hierarchy type: Intersection
Routing system: 2
Sector system: 2

Default routing table:

Destination	Next
1	10
2	10
3	10
4	10
10	10

Intersection 47

Number: 47 x-position: 483 y-position: 234

Node type: Normal
Hierarchy type: MotorInter
Routing system: 1
Sector system: 3

Default routing table:

Destination	Next
1	17
2	17
3	17
4	17
5	17
6	17
7	17
8	17
9	17
10	17

Intersection 17

Number: 17 x-position: 261 y-position: 189

Node type: Normal
Hierarchy type: MotorInter
Routing system: 1
Sector system: 2

Default routing table:

Destination	Next
1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4
9	4
10	4

Intersection 34

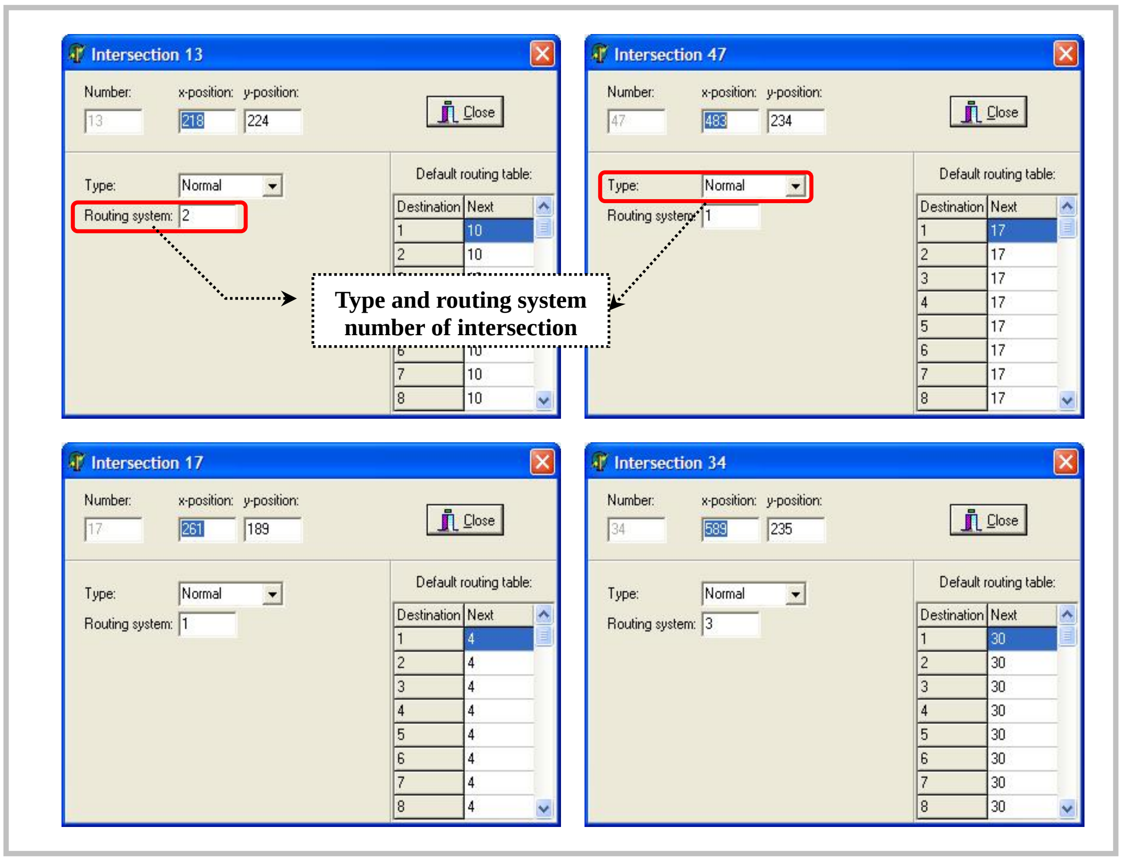
Number: 34 x-position: 589 y-position: 235

Node type: Normal
Hierarchy type: Intersection
Routing system: 3
Sector system: 3

Default routing table:

Destination	Next
1	30
2	30
3	30
4	30
5	30
6	30
7	30
8	30
9	30
10	30

Figure 59: Nodes attributes in hierarchical city program



Intersection 13

Number: 13 x-position: 218 y-position: 224

Type: Normal
Routing system: 2

Default routing table:

Destination	Next
1	10
2	10
6	10
7	10
8	10

Intersection 47

Number: 47 x-position: 483 y-position: 234

Type: Normal
Routing system: 1

Default routing table:

Destination	Next
1	17
2	17
3	17
4	17
5	17
6	17
7	17
8	17

Intersection 17

Number: 17 x-position: 261 y-position: 189

Type: Normal
Routing system: 1

Default routing table:

Destination	Next
1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4

Intersection 34

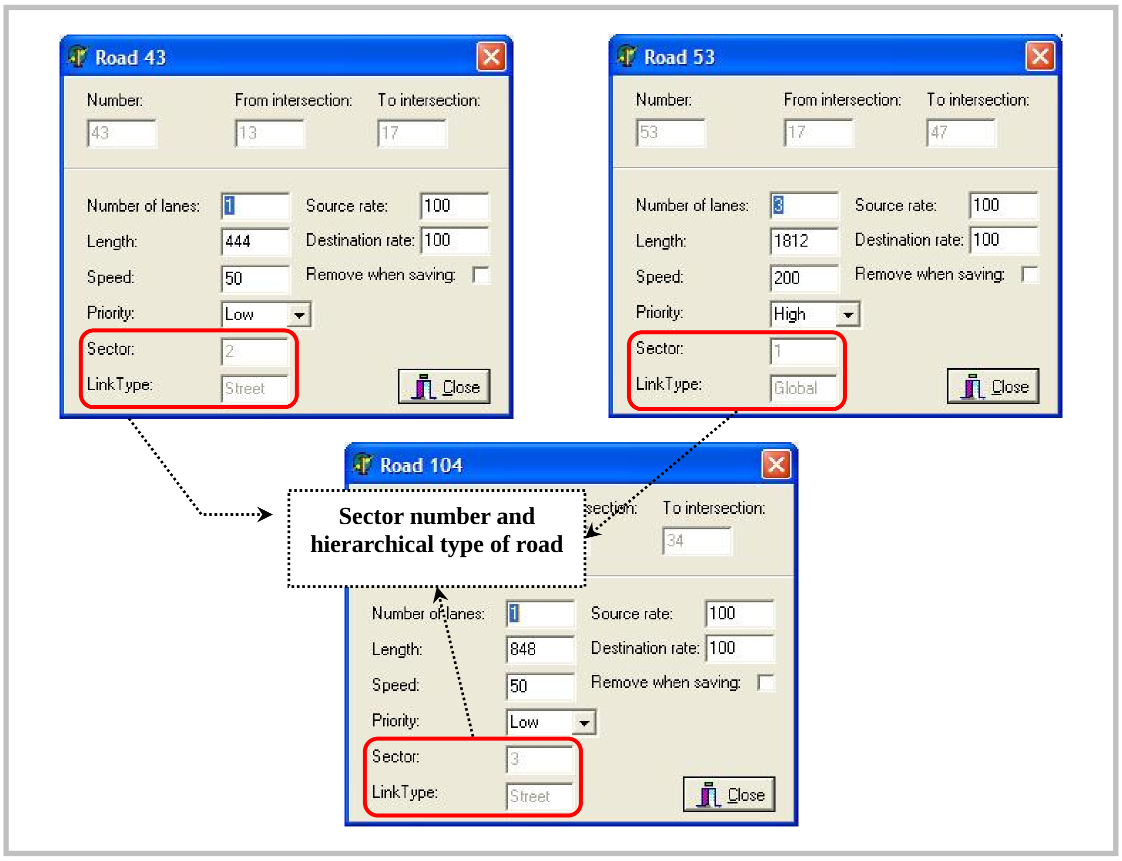
Number: 34 x-position: 589 y-position: 235

Type: Normal
Routing system: 3

Default routing table:

Destination	Next
1	30
2	30
3	30
4	30
5	30
6	30
7	30
8	30

Figure 60: Nodes attributes in conventional city program



Road 43

Number: 43
From intersection: 13
To intersection: 17
Number of lanes: 1
Length: 444
Speed: 50
Priority: Low
Sector: 2
LinkType: Street
Source rate: 100
Destination rate: 100
Remove when saving: ☐

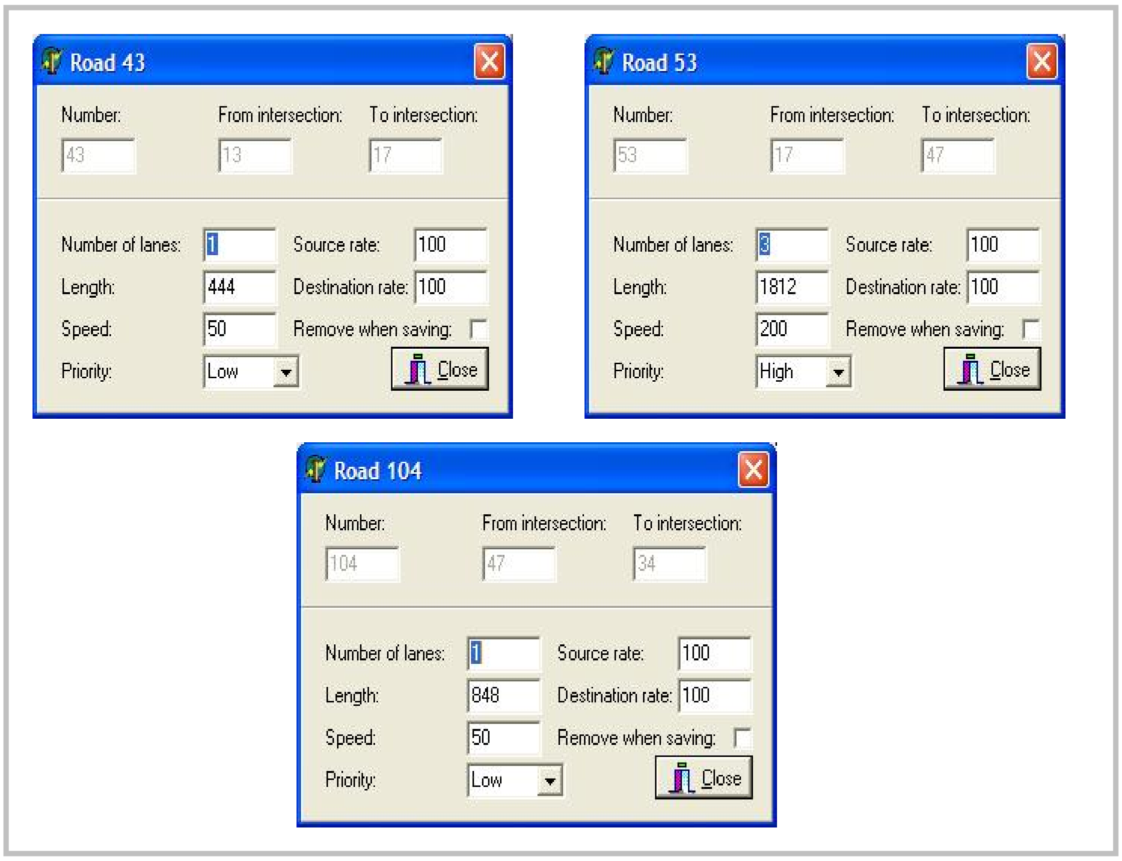
Road 53

Number: 53
From intersection: 17
To intersection: 47
Number of lanes: 3
Length: 1812
Speed: 200
Priority: High
Sector: 1
LinkType: Global
Source rate: 100
Destination rate: 100
Remove when saving: ☐

Road 104

section:
To intersection: 34
Number of lanes: 1
Length: 848
Speed: 50
Priority: Low
Sector: 3
LinkType: Street
Source rate: 100
Destination rate: 100
Remove when saving: ☐

Figure 61: Roads attributes in hierarchical city program



Road 43

Number: 43
From intersection: 13
To intersection: 17
Number of lanes: 1
Length: 444
Speed: 50
Priority: Low
Source rate: 100
Destination rate: 100
Remove when saving: ☐

Road 53

Number: 53
From intersection: 17
To intersection: 47
Number of lanes: 3
Length: 1812
Speed: 200
Priority: High
Source rate: 100
Destination rate: 100
Remove when saving: ☐

Road 104

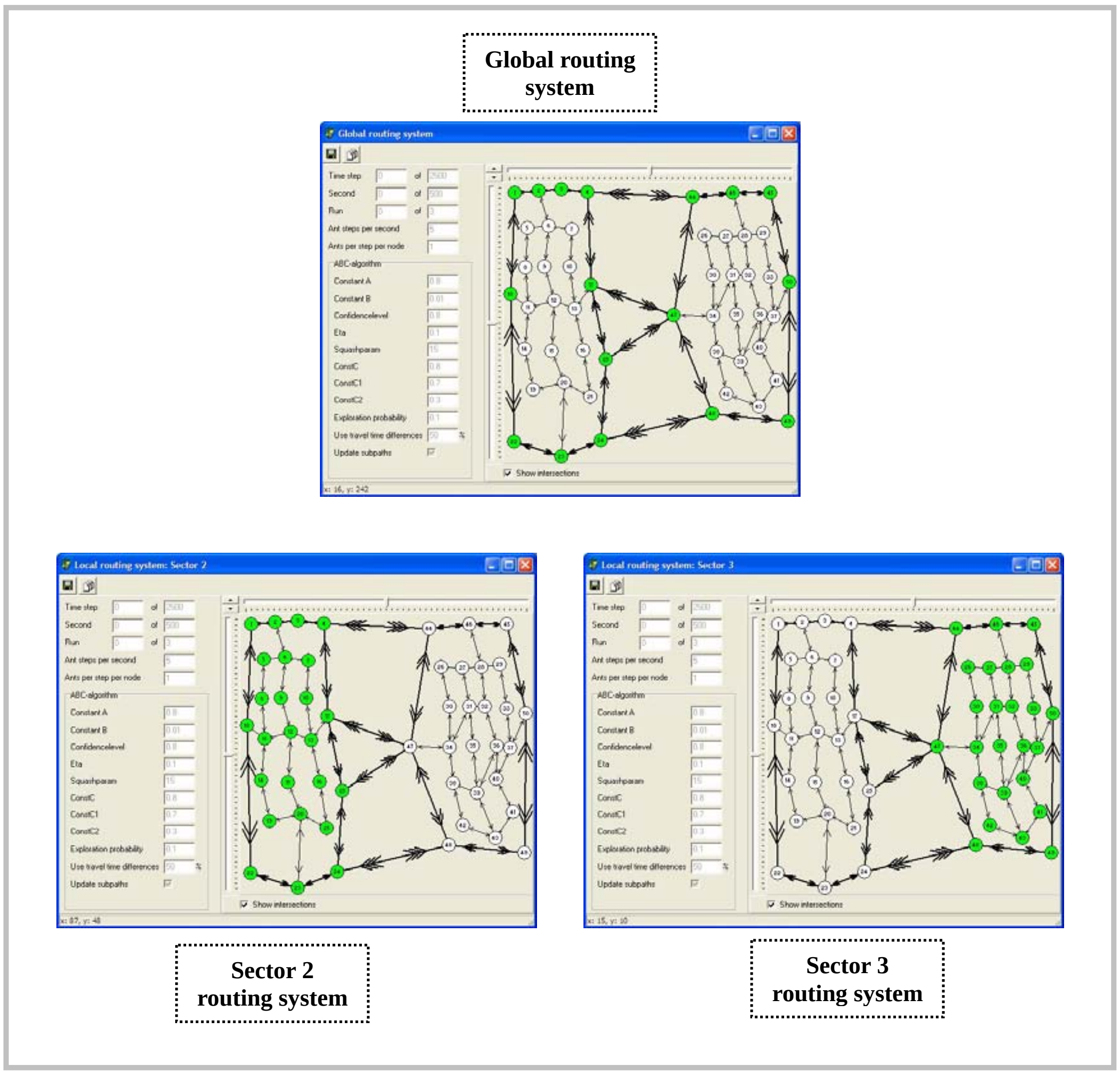
Number: 104
From intersection: 47
To intersection: 34
Number of lanes: 1
Length: 848
Speed: 50
Priority: Low
Source rate: 100
Destination rate: 100
Remove when saving: ☐

Figure 62: Roads attributes in conventional city program

5.1.2.2 Routing system

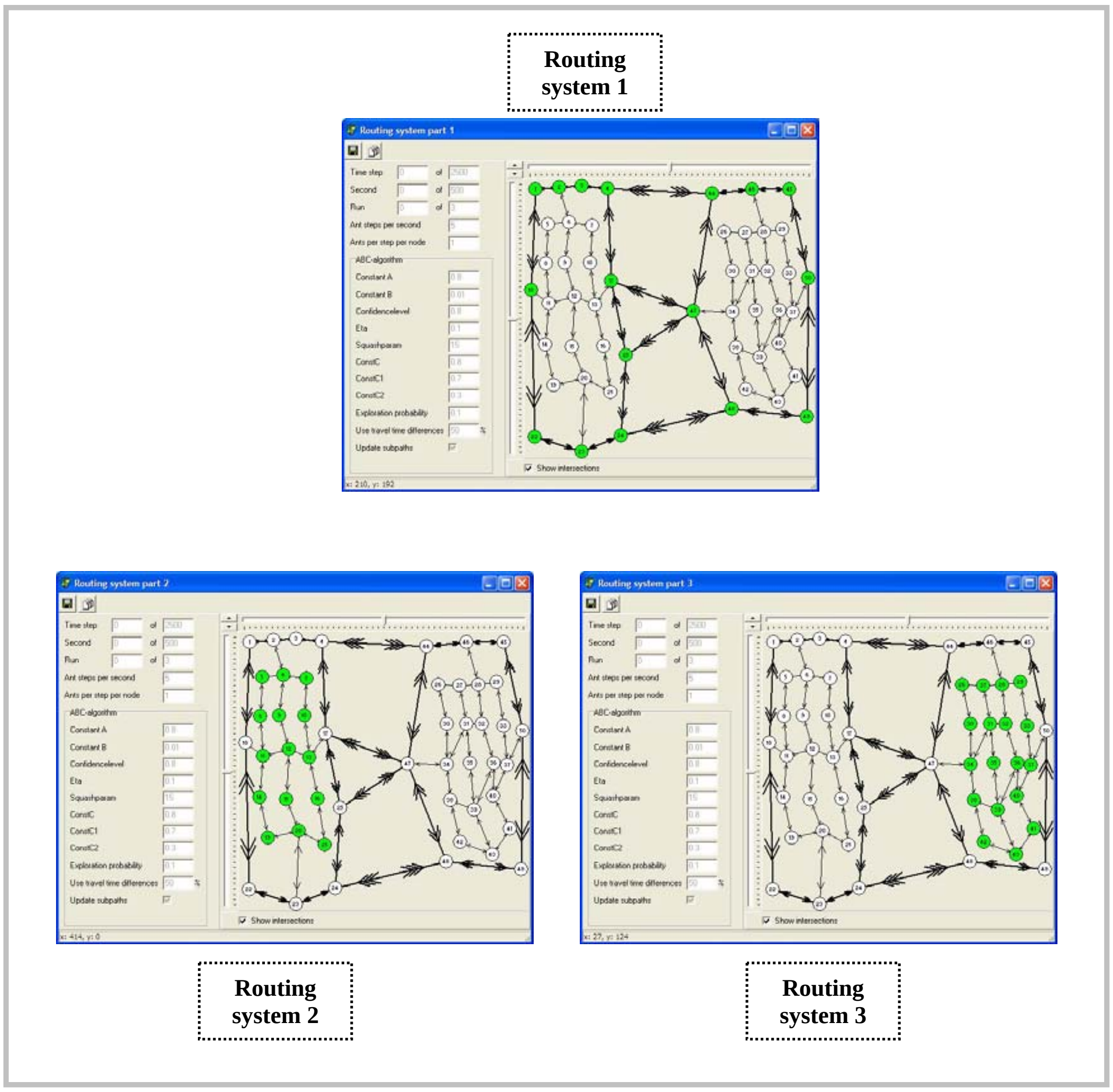
In a hierarchical routing system, example traffic network will be divided into one abstract level network and two detailed level networks. Global routing system is responsible for the abstract level network and sector routing system takes care of the corresponding sector network. We must state that each kind of routing system only take charge of nodes located within this level network. For the sector routing system, source and destination nodes must belong to this sector; for the global routing system, source and destination nodes must belong to global traffic network. The cooperation between sector and global routing system obeys some rules. We can find detailed information from the former chapter.

In conventional routing system, example traffic network will be charged of three distributed routing system, routing system 1, 2, 3. These routing systems' functionality are equal. In every routing system, source nodes must belong to scope of this routing system and destination nodes could be any nodes in the traffic network.



```
Time step                    0   of 2500
Second                       0   of 500
Run                          0   of 3
Ant steps per second             5
Ants per step per node           1
ABC-algorithm
  Constant A                     0.8
  Constant B                     0.01
  Confidencelevel                0.8
  Eta                            0.1
  Squashparam                    15
  ConstC                         0.8
  ConstC1                        0.7
  ConstC2                        0.3
  Exploration probability        0.1
  Use travel time differences    50 %
  Update subpaths                ☑
```

Figure 63: Hierarchical routing systems



Routing system part 1

```
Time step                   0   of 2500
Second                      0   of 500
Run                         0   of 3
Ant steps per second            5
Ants per step per node          1
ABC-algorithm
  Constant A                    0.8
  Constant B                    0.01
  Confidencelevel               0.8
  Eta                           0.1
  Squashparam                   15
  ConstC                        0.8
  ConstC1                       0.7
  ConstC2                       0.3
  Exploration probability       0.1
  Use travel time differences   50 %
  Update subpaths               ☑
x: 210, y: 192
```

Routing system part 2

```
Time step                   0   of 2500
Second                      0   of 500
Run                         0   of 3
Ant steps per second            5
Ants per step per node          1
ABC-algorithm
  Constant A                    0.8
  Constant B                    0.01
  Confidencelevel               0.8
  Eta                           0.1
  Squashparam                   15
  ConstC                        0.8
  ConstC1                       0.7
  ConstC2                       0.3
  Exploration probability       0.1
  Use travel time differences   50 %
  Update subpaths               ☑
x: 414, y: 0
```

Routing system part 3

```
Time step                   0   of 2500
Second                      0   of 500
Run                         0   of 3
Ant steps per second            5
Ants per step per node          1
ABC-algorithm
  Constant A                    0.8
  Constant B                    0.01
  Confidencelevel               0.8
  Eta                           0.1
  Squashparam                   15
  ConstC                        0.8
  ConstC1                       0.7
  ConstC2                       0.3
  Exploration probability       0.1
  Use travel time differences   50 %
  Update subpaths               ☑
x: 27, y: 124
```

Figure 64: Conventional routing systems

5.1.2.3 Optimal route computation

When computing optimal route according to the motorist's requirements, these two types routing system will obey different rules. To illustrate computation procedure, we choose one pair of source and destination nodes based on example traffic network above (figure 55). The source node is intersection 19 (located on the left section of the traffic network) and destination node is intersection 28 (located on the right section of the traffic network). Figure 62 shows the source and destination in more detail. We will let two types routing systems compute one optimal route from 19 to 28.

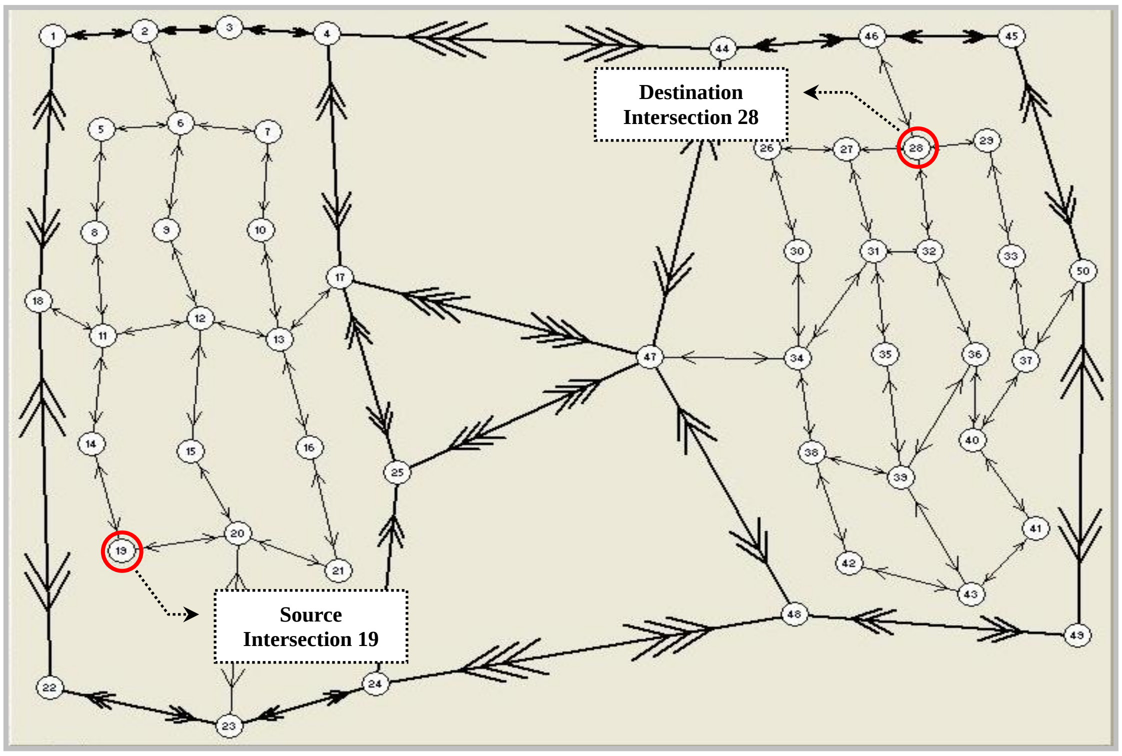


Figure 65: Source and destination node on the traffic network

5.1.3 Expectations

a) *Hierarchical routing system*
Intersection 19 and intersection 28 are located on different sectors. The detailed information about these two intersections is shown by following table.

Table 22: Hierarchical attributes of intersections

Intersection number	Hierarchical type	Sector number
19	Inner intersection	2
28	Inner intersection	3

According cooperation rules of hierarchical routing system, the computational procedure must be as follows:

I. Sector 2 routing system navigates a vehicle to the nearest motorway intersection using virtual part of the routing table. This is because the destination node is in the different sector with source node. From motorway intersection, vehicle could reach destination sector as soon as possible.
II. When reached motorway intersection, global routing system navigate vehicle to the remote sector (sector 3) along the motorway.
III. When reached destination sector, sector 3 routing system navigate vehicle to the destination node (intersection 28).

b) *Conventional routing system*
Distributed routing systems will provide one optimal route from source to destination without considering sector definition. This route should be the shortest route based on current traffic situation.

c) *Running time*

Because of hierarchical topology, hierarchical routing system will run faster than conventional routing system using same computer system. Two types simulation will run 200 steps (40 seconds). We will use computer system with following specification.

- CPU: Intel Pentium III mobile CPU, 1133MHz
- Memory: 640MB

For the accuracy of the experiment, we execute the experiment for 20 times and compute the average value of some necessary parameters (Average travel time, system running time, and so on).

5.1.4 Result

a) *Hierarchical routing system*

The optimal route from 19 to 28 is following:

$$19 \xrightarrow{49} 20 \xrightarrow{64} 23 \xrightarrow{26} 24 \xrightarrow{44} 48 \xrightarrow{43} 47 \xrightarrow{44} 44 \xrightarrow{26} 46 \xrightarrow{41} 28$$

Travel time between two nodes is displayed above the arrow.

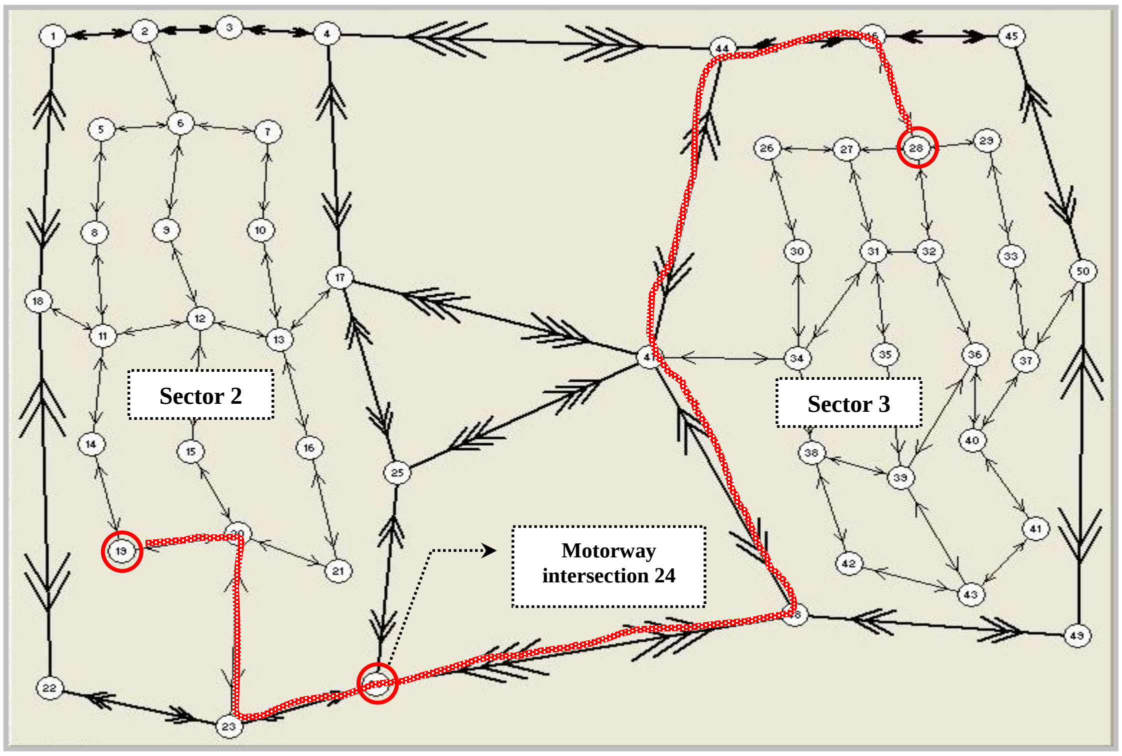


Figure 66: Optimal route of hierarchical routing system

We display some statistic as following.

- Average system running time for 200 traffic steps: ***144 seconds***.
- Average travel time from source to destination: ***361 seconds***.
- Average travel time from source to motorway intersection: ***139 seconds***.

b) *Conventional routing system*

The optimal route from 19 to 28 is following:

$$19 \xrightarrow{49} 20 \xrightarrow{64} 23 \xrightarrow{26} 24 \xrightarrow{37} 14 \xrightarrow{36} 11 \xrightarrow{29} 18 \xrightarrow{36} 1 \xrightarrow{16} 2 \xrightarrow{14} 3$$

$$3 \xrightarrow{17} 4 \xrightarrow{41} 44 \xrightarrow{26} 46 \xrightarrow{41} 28$$

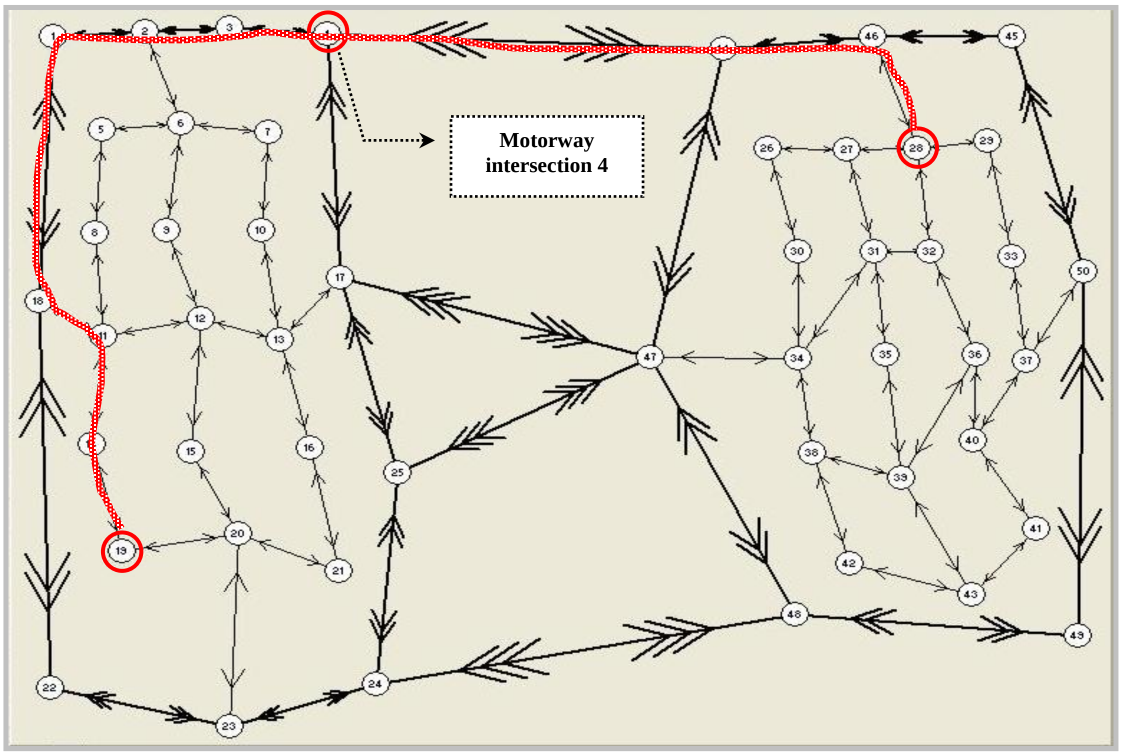


Figure 67: Optimal route of conventional routing system

Some statistics are following.

- Average system running time for 200 traffic steps: ***437 seconds***.
- Average travel time from source to destination: ***293 seconds***.
- Average travel time from source to motorway intersection: ***185 seconds***.

We use following table to illustrate the differences between hierarchical routing system and conventional routing system.

Table 23: Statistics of two types' routing system

Statistic item	Hierarchical routing system	Conventional routing system
Average System running time	144 seconds	437 seconds
Average travel time (19→28)	361 seconds	293 seconds
Average travel time (19→motor intersection)	139 seconds	185 seconds
Distance (19→motor intersection)	2421 meters	4198 meters

From statistics on table-23, we could see that hierarchical routing system always navigate vehicle from intersection 19 to the nearest motorway intersection. In this example traffic network, the nearest motorway intersection of 19 is 24. Travel time from 19 to 24 is 139 seconds, and distance from 19 to 24 is 2421 meters. When traffic network become more and more complex, hierarchical routing system will illustrate advantages comparing to the conventional routing system. The disadvantage of the hierarchical routing is shown from this experiment. The hierarchical routing may navigate vehicle to a sub optimal route.

5.2 Experiment 2: effectiveness of hierarchical routing

5.2.1 Goal

The first Experiment uses simple (just two sectors) traffic network to show the difference between hierarchical routing system and conventional routing system. We could learn more about the characteristics of the hierarchical routing system through such kind of comparing. In this experiment, we will design a more complex traffic network to test the effectiveness of the hierarchical routing system. This traffic networks will contain more sectors than in the first experiment. We know that in practice, one traffic network could be very complex, including many sectors. We use this experiment to simulate the real city network, and let hierarchical routing system compute optimal route based on this complex environment.

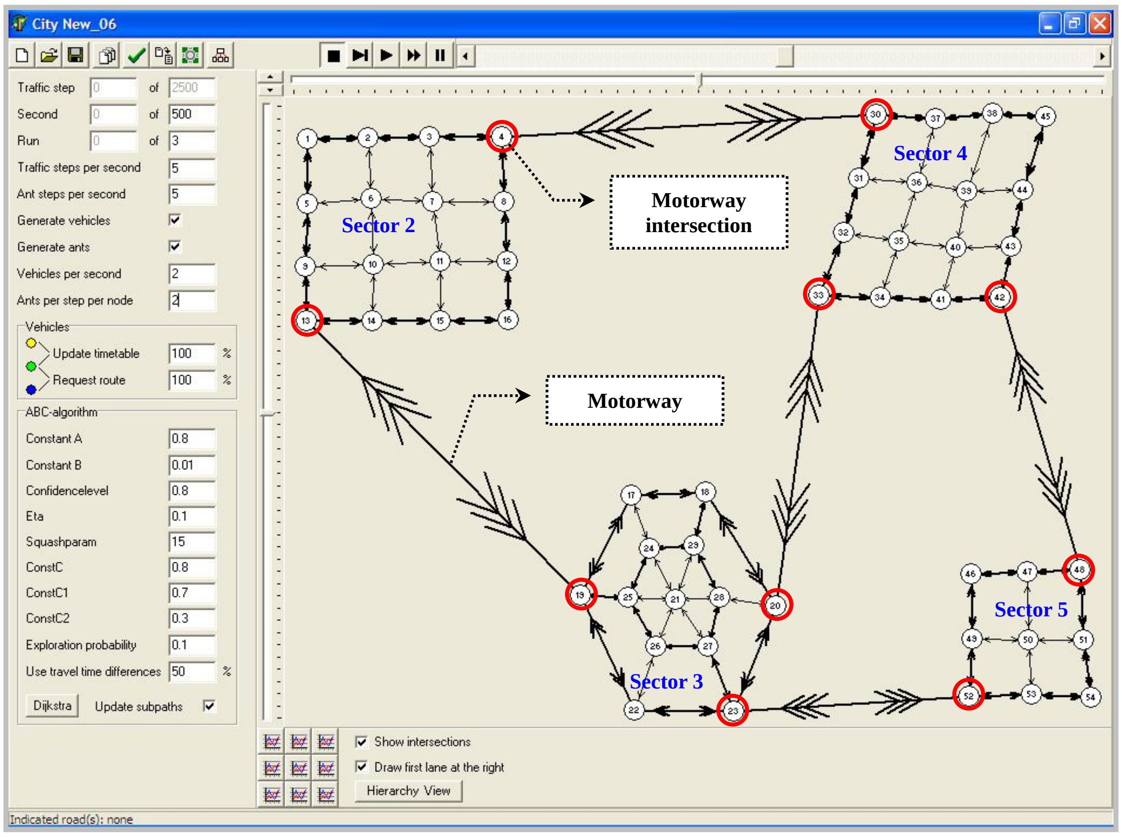
5.2.2 Environment and setting

5.2.2.1 City program

The parameters about city program are set as following:

- Traffic steps per second = 5
- Ant steps per second = 5
- Generate vehicles = True
- Generate ants = True
- Vehicles per second = 2.0
- Ants per step per node = 2.0
- Update timetable = 100%
- Request route = 100%
- Constant A = 0.8
- Constant B = 0.01
- Confidence level = 0.8
- Eta = 0.1
- Squashparam = 15
- ConstC = 0.8
- ConstC1 = 0.7
- ConstC2 = 0.3
- Exploration probability = 0.1
- Use travel time differences = 50%
- Update subpaths = True

This traffic network is made up of four sectors. There exists high way between two sectors.



Traffic step 0 of 2500
Second 0 of 500
Run 0 of 3
Traffic steps per second 5
Ant steps per second 5
Generate vehicles ✓
Generate ants ✓
Vehicles per second 2
Ants per step per node 2

Vehicles
Update timetable 100 %
Request route 100 %

ABC-algorithm
Constant A 0.8
Constant B 0.01
Confidencelevel 0.8
Eta 0.1
Squashparam 15
ConstC 0.8
ConstC1 0.7
ConstC2 0.3
Exploration probability 0.1
Use travel time differences 50 %
Dijkstra Update subpaths ✓

✓ Show intersections
✓ Draw first lane at the right

Indicated road(s): none

Figure 68: City program setting

5.2.3 Expectations

Because traffic network in our experiment have four sectors, hierarchical routing system will activate one global routing system and four sector routing systems (sector 2~ sector 4). The routing table of each node will be created based on the type of the node, like hierarchical type and sector number. When the hierarchical routing systems receive requests from vehicles, they will cooperate to compute optimal routes according to hierarchical routing rules. We display these hierarchical routing rules as follows to help us test the effectiveness of hierarchical routing systems.

Table 24: Hierarchical routing rules

Sectors of source and destination nodes	**Hierarchical routing rules (including routing systems)**
Different sector between source and destination	▪ Sector routing system of source node (from source node to the nearest motorway intersection), using virtual part of the routing table ▪ Global routing system (from source sector to the destination sector), using virtual part of the routing table ▪ Sector routing system of destination node (to the destination node),using real part of the routing table
Same sector between source and destination	▪ Sector routing system of source node (from source node to the destination node), using virtual part of the routing table

5.2.4 Result

5.2.4.1 Routing systems

As we expect, the hierarchical routing system activates following routing systems.

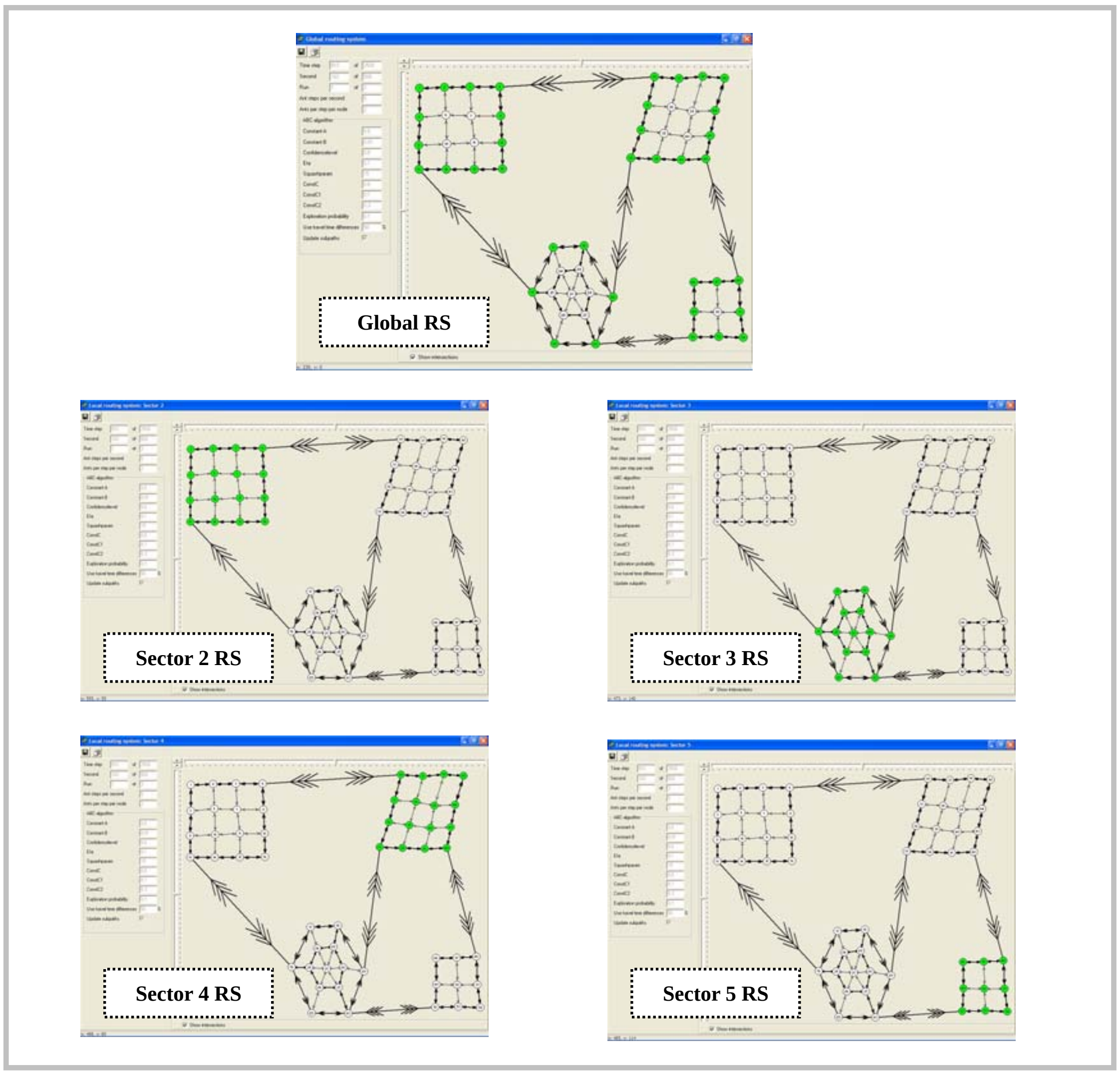


Figure 69: Global routing system and sector routing systems

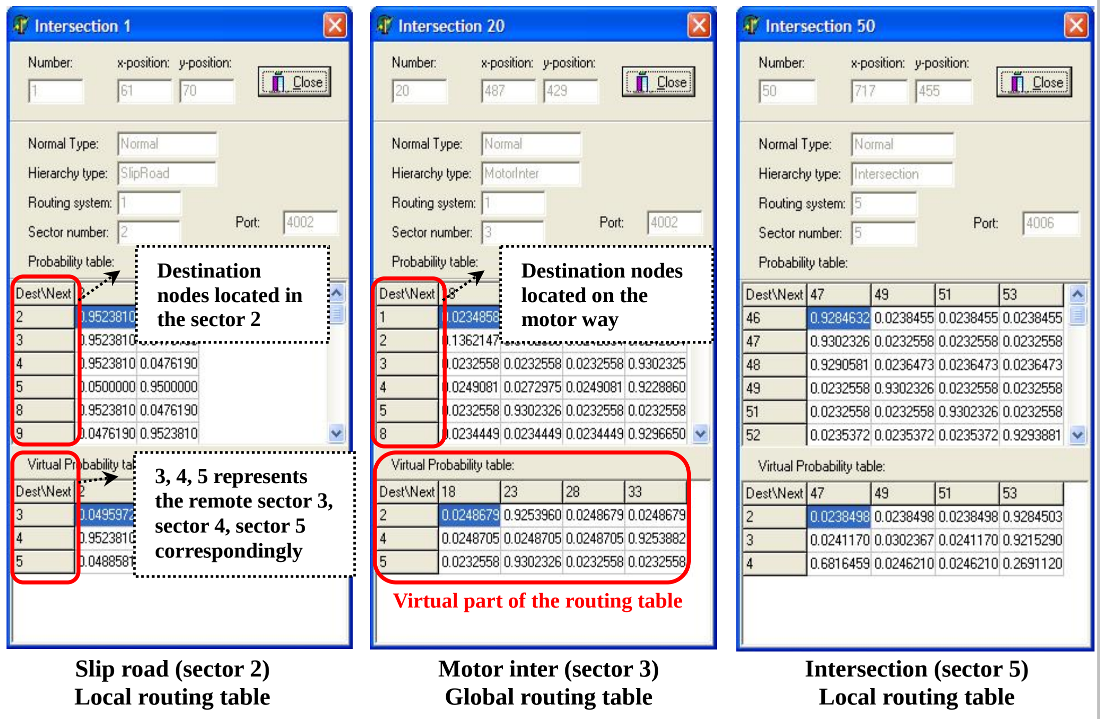
5.2.4.2 Routing table

In the hierarchical routing system, we have two types of routing tables. One is a local routing table; the other is a global routing table.

Table 25: Routing table types

Hierarchical type of node	Routing table type	Characteristics
Intersection	▪ Local routing table	▪ Destination nodes located within the same sector
Slip road, departure Motorway intersection	▪ Local routing table ▪ Global routing table	▪ Local: destination nodes located within the same sector ▪ Global : destination nodes located on the motorway

The most important characteristic of the hierarchical routing table is a virtual node. The virtual part represents the sector, which is different from current sector. In this example environment there are four different sectors, and then every routing table contains three entries for the virtual node.


Intersection 1
Close
Destination nodes located in the sector 2
3, 4, 5 represents the remote sector 3, sector 4, sector 5 correspondingly
Intersection 20
Close
Destination nodes located on the motor way
Virtual part of the routing table
Intersection 50
Close

Number: 1 x-position: 61 y-position: 70

Normal Type: Normal
Hierarchy type: SlipRoad
Routing system: 1
Sector number: 2
Port: 4002

Probability table:

Dest\Next	2	
2	.9523810	
3	.9523810	
4	.9523810	0.0476190
5	.0500000	0.9500000
8	.9523810	0.0476190
9	.0476190	0.9523810

Virtual Probability table:

Dest\Next	2
3	.0495972
4	.952381
5	.0488581

Slip road (sector 2)
Local routing table

Number: 20 x-position: 487 y-position: 429

Normal Type: Normal
Hierarchy type: MotorInter
Routing system: 1
Sector number: 3
Port: 4002

Probability table:

Dest\Next	8			
1	.0234858			
2	.1362147			
3	.0232558	0.0232558	0.0232558	0.9302325
4	.0249081	0.0272975	0.0249081	0.9228860
5	.0232558	0.9302326	0.0232558	0.0232558
8	.0234449	0.0234449	0.0234449	0.9296650

Virtual Probability table:

Dest\Next	18	23	28	33
2	0.0248679	0.9253960	0.0248679	0.0248679
4	0.0248705	0.0248705	0.0248705	0.9253882
5	0.0232558	0.9302326	0.0232558	0.0232558

Motor inter (sector 3)
Global routing table

Number: 50 x-position: 717 y-position: 455

Normal Type: Normal
Hierarchy type: Intersection
Routing system: 5
Sector number: 5
Port: 4006

Probability table:

Dest\Next	47	49	51	53
46	0.9284632	0.0238455	0.0238455	0.0238455
47	0.9302326	0.0232558	0.0232558	0.0232558
48	0.9290581	0.0236473	0.0236473	0.0236473
49	0.0232558	0.9302326	0.0232558	0.0232558
51	0.0232558	0.0232558	0.9302326	0.0232558
52	0.0235372	0.0235372	0.0235372	0.9293881

Virtual Probability table:

Dest\Next	47	49	51	53
2	0.0238498	0.0238498	0.0238498	0.9284503
3	0.0241170	0.0302367	0.0241170	0.9215290
4	0.6816459	0.0246210	0.0246210	0.2691120

Intersection (sector 5)
Local routing table

Figure 70: Hierarchical routing tables

5.2.4.3 Optimal route computation

To test the cooperation between global and sector routing systems, we let the hierarchical routing system compute an optimal route based on our requirements.

The source node and destination node are as follows:

- Source node 50 → destination node 6
- Source node 50 → destination node 36
- Source node 50 → destination node 24

With figures, we will show the optimal route according to the routing table of the hierarchical routing system.

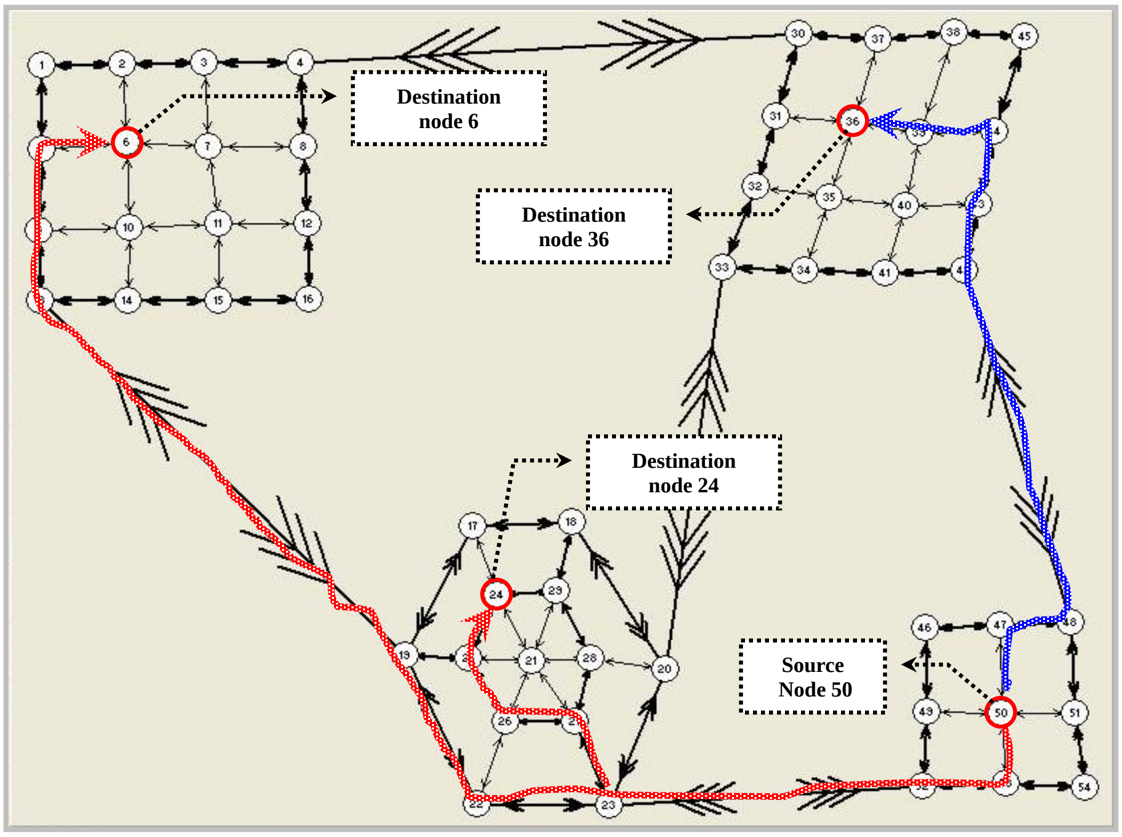


Figure 71: Optimal routes

Optimal routes are:

- Source node 50 ⟶ destination node 6

Route: $50 \rightarrow 53 \rightarrow 52 \rightarrow 23 \rightarrow 22 \rightarrow 19 \rightarrow 13 \rightarrow 9 \rightarrow 5 \rightarrow 6$

a) Sector 5 routing system navigates the vehicle from source to the 52. Node 52 is the nearest node to the sector 2 (because destination node 6 located on sector 2).

b) Global routing system navigates the vehicle from 52 to the destination sector 2 along the motor way. As we see that path between 52 and 13 is a motor way indeed.

c) Sector 2 routing system navigates the vehicle from 13 to the destination node 6

- Source node 50 ⟶ destination node 36

Route: $50 \rightarrow 47 \rightarrow 48 \rightarrow 42 \rightarrow 43 \rightarrow 44 \rightarrow 39 \rightarrow 36$

  - Sector 5 routing system navigates the vehicle from source to the 48. We could see that because destination node 36 located on sector 4, 48 is the nearest node to the sector 4.
  - Sector 4 routing system navigates the vehicle from 42 to 36

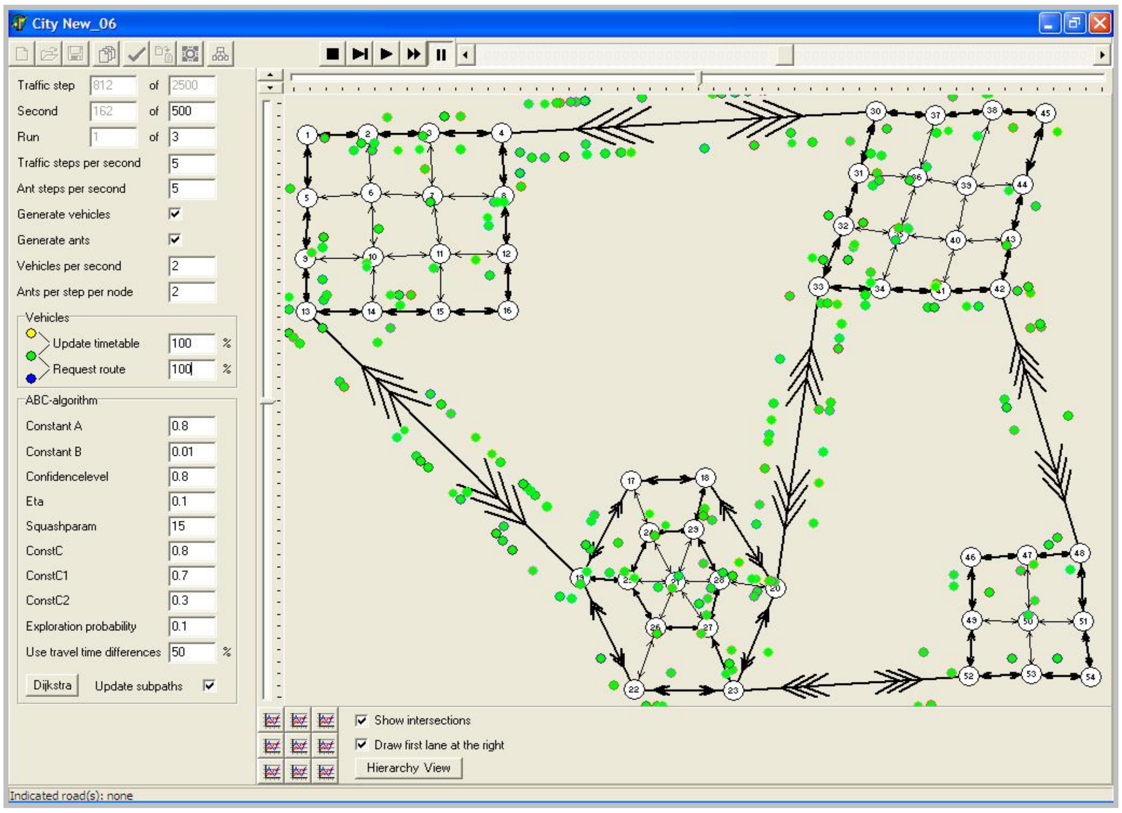
- Source node 50 ⟶ destination node 24

Route: $50 \rightarrow 53 \rightarrow 52 \rightarrow 23 \rightarrow 27 \rightarrow 26 \rightarrow 25 \rightarrow 24$

a) We could see that the vehicle will move from 23 to 24 along the best route.

5.2.4.4 Computer system resource consumption

In this experiment, we let the simulation run with the fastest model. The computer system we used is the same as in the first experiment. We use the stopwatch to compute the exact time for executing simulation. The result is that simulation uses 1187 seconds for running 812 traffic steps in the fastest model. The following figure shows the situation about simulation after 812 steps.



City New_06

Traffic step	812	of	2500
Second	162	of	500
Run	1	of	3
Traffic steps per second			5
Ant steps per second			5
Generate vehicles			✓
Generate ants			✓
Vehicles per second			2
Ants per step per node			2

Vehicles

Update timetable	100	%
Request route	100	%

ABC-algorithm

Constant A	0.8	
Constant B	0.01	
Confidencelevel	0.8	
Eta	0.1	
Squashparam	15	
ConstC	0.8	
ConstC1	0.7	
ConstC2	0.3	
Exploration probability	0.1	
Use travel time differences	50	%

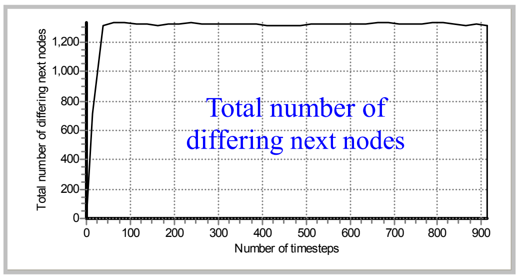
Dijkstra Update subpaths ✓

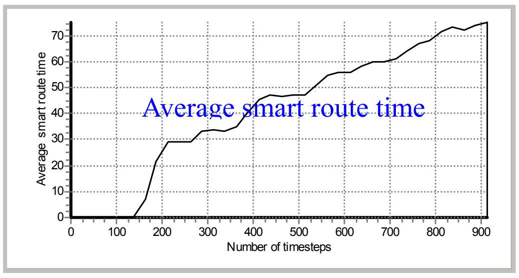
✓ Show intersections
✓ Draw first lane at the right
Hierarchy View

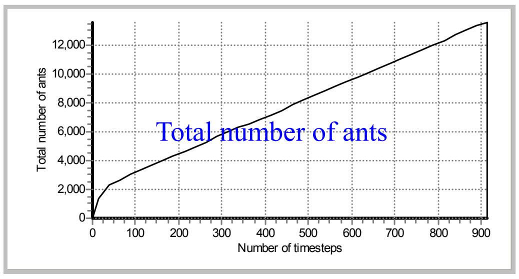
Indicated road(s): none

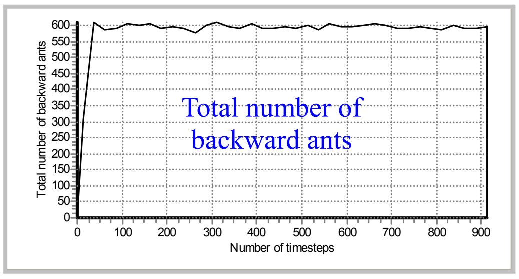
Figure 72: Executing situation

We also show some statistic figures:









Chapter 6 Conclusions and Recommendations

After nine months' hard working on the project, I mastered many characteristics of the hierarchical routing system. In this chapter, I will discuss the hierarchical routing system in the aspects of limitations and recommendations. And then I will present a conclusion on the project.

6.1 Limitations and recommendations

The tests and experimentations shown in the previous chapter could tell us that the hierarchical routing system application is not perfect yet. For some reasons, we could not make improvements on these aspects. We will elaborate on them in following sections.

As we described before, our hierarchical routing system is developed from Kroon's program. And as the result, our simulation inherits many characteristics of conventional routing system, including some deficiencies. Detailed information about these inherited disadvantages could be found in Kroon's master thesis. Now, we will talk about some limitations in our hierarchical routing system simulation.

6.1.1 City program

6.1.1.1 Adjacent sectors

A city network sector could be adjacent to another sector in one real traffic network, if both sectors contain a city network and there is more than one node that belongs to both sectors. For an example have a look at the traffic network shown in figure-73. The only adjacent sector of sector 2 is sector 3. The only adjacent sector of sector 4 is sector 5. Sector 6 has no adjacent sector.

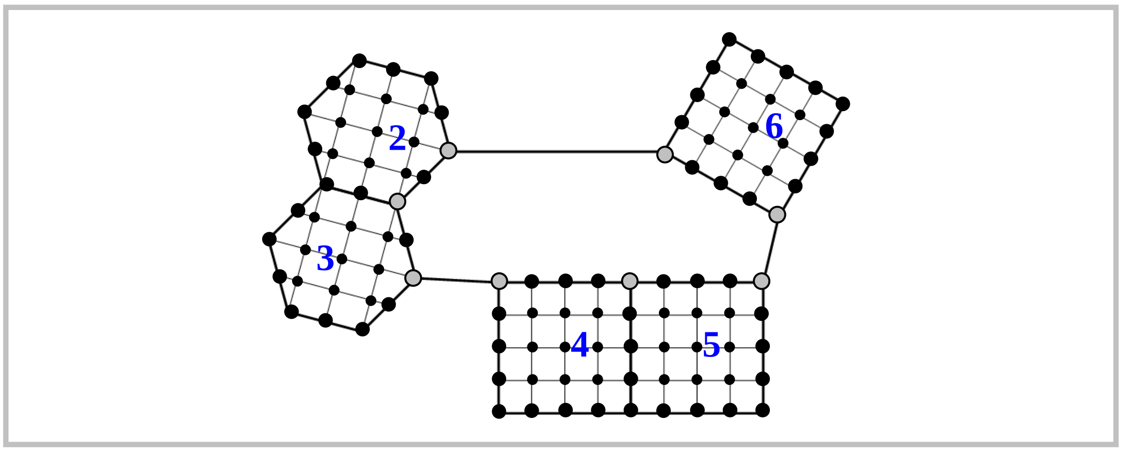


Figure 73: Traffic network with adjacent sectors

From the picture above, we could see that some nodes maybe belong to several sectors. The city program must remember this information. Recording sectors information, we need to use a special data structure, like array. Correspondingly, introducing array will bring some new additional problems. Currently, nodes in our hierarchical routing system just can belong to only one sector.

6.1.1.2 Global nodes

We explained that nodes in hierarchical routing system have hierarchical types. They are *intersection, slip road, departure and motorway intersection.* But in practice, there exist some nodes located between two distant city networks. These nodes could be gas station, small town, toll station or other things. Based on hierarchical view, these nodes don't belong to any real city network and can only belong to the abstract level network. Figure-74 shows such kind of nodes.

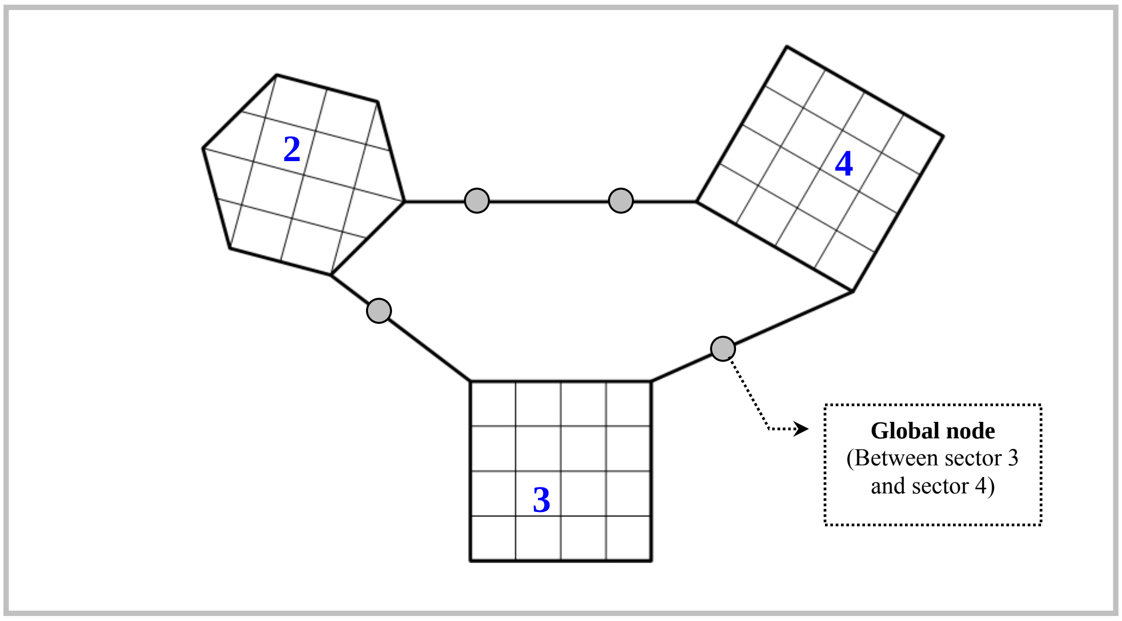


Figure 74: Global nodes on the hierarchical network

For the sake of convenience, global nodes don't exist in our hierarchical routing system.

6.1.1.3 Traffic simulation details

Since the most important part of the study is a hierarchical routing system, in hierarchical city program, we ignore some city program details used in the Kroon's program, like *Roundabout* type of the node and disability of the road. This is because introduction of this detailed information will make the routing system more complex and weaken the running of the routing system. Of course, we should make traffic simulation including more detail information of real traffic as possible as it can.

6.1.2 Hierarchical routing system

6.1.2.1 Routing table initialization

In a conventional routing system, the routing table is initialized according to the default routing table. In a hierarchical routing system, the routing table is initialized based on the hierarchical type of the node. The detailed information could be checked on the former chapter. We know that optimal routing table initialization could make the routing system more effective.

Now we will give one idea about the better way to initialize the routing table. As we know, in Kroon's program routing table is initialized according to the default routing table. But there are no hierarchical characteristics in this default routing table. For hierarchical routing requirements, we could create the default routing table based on the hierarchical traffic network structure. As a result, the default routing tables will be divided into two types, one is a global default routing table, and the other one is a local default routing table. Before we activate distributed routing system, we use the Dijkstra's algorithm to compute the default routing table for every node. There are some rules for the default routing table computation.

1) Create default routing table according to the hierarchical type of node
    - If node is *Sliproad, Departure or Motorway Intersection*, the city program will create two default routing tables for it, one is local and the other one is global.
    - If node is *Intersection*, the city program will create one local default routing table for it.
2) Use The Dijkstra's algorithm based on the hierarchy network division
    - If node is *Sliproad, Departure or Motorway Intersection*, the city program will use the Dijkstra's algorithm to compute the optimal next node based on the abstracted level network. That is, the destinations of this node are located on the abstracted level network.
    - If node is *Intersection*, the city program will use the Dijkstra's algorithm to compute the optimal next node based on the detailed level network, which this node belongs to. That is, the destinations are located on the same

sector as this node.

3) Store the information about default routing table to the file
   - After computation, the city program will store the default routing table as the file. For example, '.Groute' is used to record the global default routing table; '.Lroute' is use to record the local default routing table.
4) Initialize the routing table according to corresponding the default routing table
   - Each routing system will read these files based on the routing system type. The global routing system will read global default routing table files and initialize the global routing table of a node. The local routing system will read local default routing table files and initialize the local routing table of a node.

Through this way, the initialization of the routing table will be more reasonable and the hierarchical routing system will be more efficient than before.

6.1.2.2 Routing table updating

Because of difference between abstract level network and detailed level network, change of the global routing table must be reflected to the corresponding local routing table. And then a sector routing system could use this information to update the virtual part of the routing table. We must define fixed change threshold to reduce the message passing frequency. Since more frequently message passing could make instability problem to the routing system.

In our system, such kind of threshold is defined as 5%. That is when probabilities in global routing table change exceed 5%, global routing system will send a message to the relevant sector routing system. We are not sure about that whether this value is the best one. If the change threshold is too big, local routing table can not be update in time. And then the optimal route would be outdated and useless. On the contrary, if threshold is too small, messages passing become more and more frequent and routing system is more and more instable.

- Probability updating formulas

The formulas we used to update the probabilities are very simple. They only take the old probability and the experienced trip time to compute the new probability. The adaptation to suddenly occurring new traffic situations is very slow. There exist much better but more complicated approaches that update the probabilities faster when the travel times dramatically change. Such approaches use heuristics and statistical approaches that consider for example also the last n trip times that have been experienced. If we want to achieve a better performance of the ABC-algorithm, these formulas are the key for an improvement.

6.1.2.3 Distributed routing system

The routing task of the routing system could be distributed over any number of applications in the range from one (only one sector routing system) to the number of sectors in the traffic network. This means that one task does not have to be executed on one computer. Being restricted with some reasons, we did not execute hierarchical routing system on several distributed computers. Maybe through experiment, we could find some special communication problems.

6.1.2.4 Performance issues

Although hierarchical topology speeds up the routing system execution, hierarchical routing system will use system resource as much as it could. When the traffic network is complex, such as including three or more sectors, the simulation will consume about 80% ~ 100% of the CPU-Time (1133MHz CPU, 640MB Memory). With improvement of the ABC algorithm, the occupation of CPU-Time will become less and less. In our experiments, the numbers of sectors are not more than 5.

6.2 Conclusions

In this report, we have examined the functionality of simulation for a hierarchical routing system, which uses Ant Based Control as main dynamic routing algorithm. This hierarchical routing system simulation is developed from a conventional routing system, and gets some important improvements based on hierarchical routing topology.

We can conclude that we have succeeded in building a routing system simulation that meets the hierarchical routing requirements. After several special experiments test, the simulation is proved to be realistic, sufficiently fast, adaptable and expandable. The global routing system and sector routing system could cooperate effectively to provide good routes, although no optimal routes can be guaranteed, based on motorist’s requirement. The execution speed and stability of simulation are improved obviously comparing with conventional routing system. Hierarchical routing mechanism let simulation deal with complex traffic network more efficient and easier than Kroon’s routing simulation.

As Kroon said in his thesis, “in science most answers arouse even more questions”. This is also applies to this master project. Although our hierarchical routing system simulation has some problems and need to be refined further, I hope what I did about hierarchical routing system simulation could provide a little help to those people, who also are interested in this science field.

Appendix A: Bibliography

[1] ANWB, http://anwb.nl

[2] Ant Colony Optimization, http://iridia.ulb.ac.be/~mdorigo/ACO/index.html

[3] Artificial Intelligence Depot, http://ai-depot.com

[4] C. Marco, *Mastering Delphi 7*, San Francisco: Sybex, 2003.

[5] Delphi Basics, http://delphibasics.co.uk

[6] D. Subramanian, P. Druschel, and J. Chen, *Ants and reinforcement learning: A case study in dynamic networks*, In Proceedings of IJCAI-97, International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1997.

[7] G. Eggenkamp, *Dynamic multi modal route planning: An artificial intelligence approach*, Master thesis, Delft University of Technology, 2001.

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

[9] G. Di Caro, M. Dorigo, *An adaptive multi-agent routing algorithm inspired by ants behavior*, In Proceedings of PART98-5th Annual Australasian Conference on Parallel and Real-Time Systems, 1998.

[10] G. Di Caro, M. Dorigo, *AntNet: Distributed stigmergetic control for communications networks*, Journal of Artificial Intelligence Research (JAIR), 9:317 – 365, 1998.

[11] H. Dibowski, *Hierarchical routing system using Ant Based Control*, Master thesis, Delft University of Technology, 2003.

[12] K. Bittner, I. Spence, *Use Case Modeling*, Addison-Wesley, 2003.

[13] L.M. Gambardella, M. Dorigo, *An ant colony system hybridized with a new local search for the sequential ordering problem*, 2000.

[14] M. Heissenbuttel, T. Braun, *Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks*, Kommunikation in verteilten Systemen (KiVS03), Leipzig, Germany, February 25-28, 2003.

[15] M. Dorigo, Artificial Life: *The Swarm Intelligence Approach*, Tutorial TD1, Congress on Evolutionary computing, Washington, DC, 1999.

[16] M. Dorigo, L.M. Gambardella, *Ant colony system: A cooperative learning approach to traveling salesman problem*. IEEE Transactions on Evolutionary Computation, 1(1): 53-66, 1997.

[17] M. Dorigo, G. Di Caro, *Ant Algorithms for Discrete Optimization*, Technical Report 98-10, IRIDIA, University Libre de Bruxelles, 1998.

[18] R. Kroon, *Dynamic vehicle routing using Ant Based Control*, Master thesis, Delft University of Technology, 2002.

[19] R. Schoonderwoerd, O. Holland, J. Bruten, and L.J.M. Rothkrantz, *Load Balancing in Telecommunication Networks*, Adaptive Behavior, Vol. 5 No. 2, 1997.

[20] R. van der Put, L.J.M. Rothkrantz, *Routing in packet switched networks using agents*, Simulation Practice and Theory, 1999.

[21] S. Russell, P. Norvig, *Artificial Intelligence, A modern approach*, Prentice Hall, 1995.

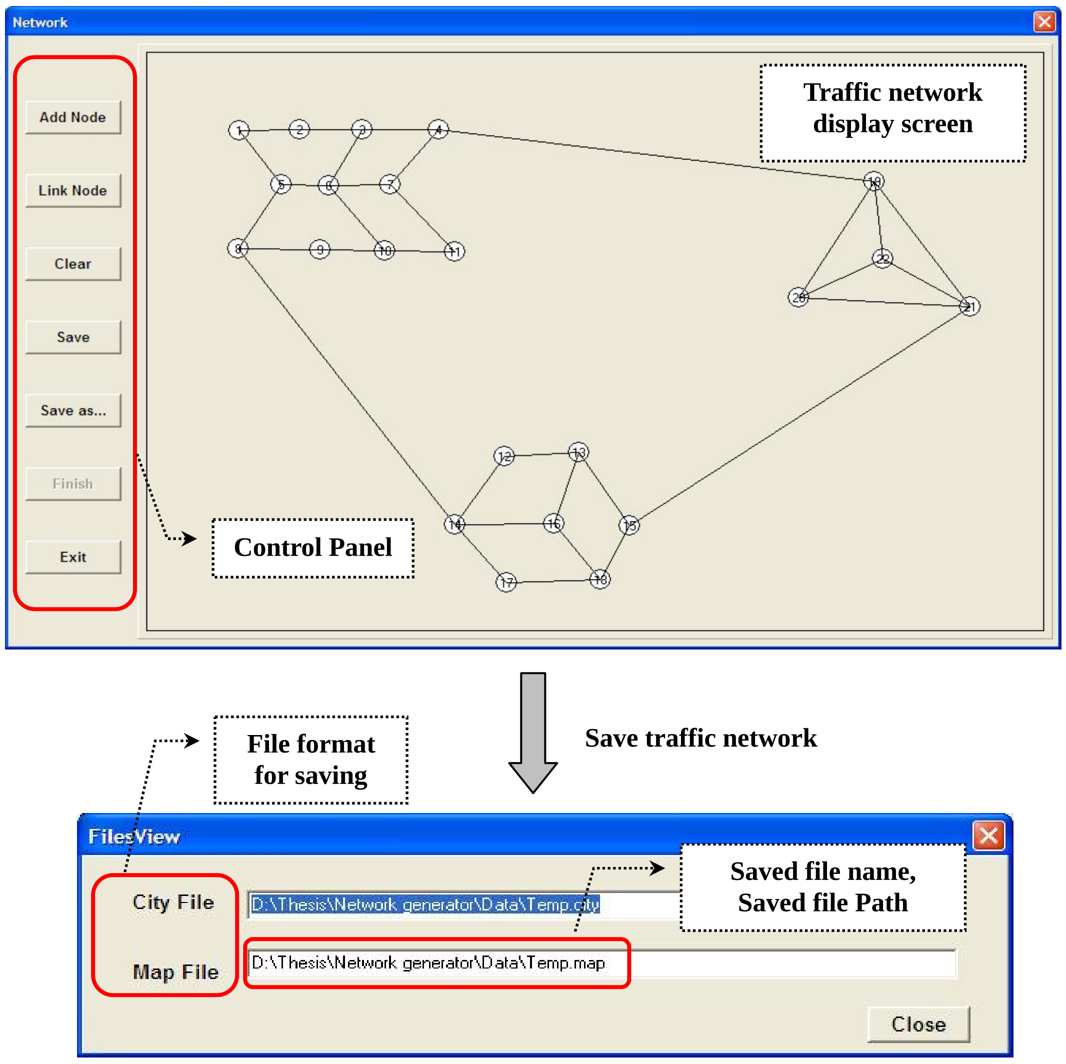
[22] W. Savitch, *Pascal (fourth edition)*, The Benjamin / Cummings Publishing Company, Inc, 1995

Appendix B: User manual

In this appendix, we will explain our simulation based on user's point. We will skip what are already interpreted in Kroon's Master thesis and show special characteristics of this hierarchical routing simulation.

B.1 Create the hierarchical traffic network

We use traffic network generator to create example traffic network according our requirements. The generator interface is following:



D:\Thesis\Network generator\Data\Temp.city

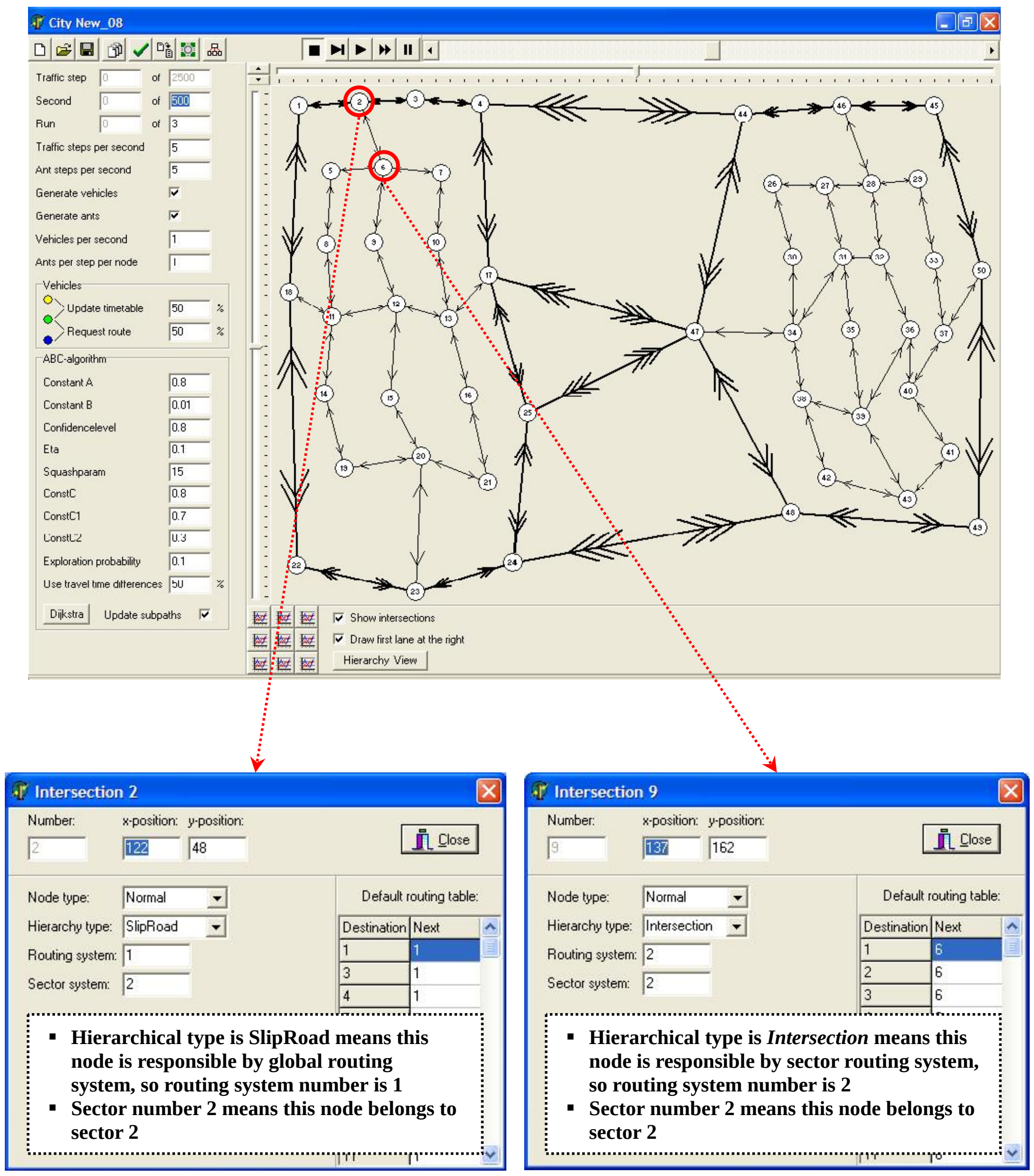
D:\Thesis\Network generator\Data\Temp.map

Figure 75: Traffic network generator

This network generator can help user to construct random traffic network. The network will be saved as two type's files, which are accepted by simulation program.

B.2 Hierarchical features of the city program

Before the whole simulation running, we must set several hierarchical features for the traffic network. Such as sector number, hierarchical type and so on.



Traffic step	0	of	2500
Second	0	of	500
Run	0	of	3
Traffic steps per second			5
Ant steps per second			5
Generate vehicles			✓
Generate ants			✓
Vehicles per second			1
Ants per step per node			1

Vehicles

Update timetable	50 %
Request route	50 %

ABC-algorithm

Constant A	0.8
Constant B	0.01
Confidencelevel	0.8
Eta	0.1
Squashparam	15
ConstC	0.8
ConstC1	0.7
ConstC2	0.3
Exploration probability	0.1
Use travel time differences	50 %
Update subpaths	✓

Intersection 2

Number:	2
x-position:	122
y-position:	48
Node type:	Normal
Hierarchy type:	SlipRoad
Routing system:	1
Sector system:	2

Destination	Next
1	1
3	1
4	1

- **Hierarchical type is SlipRoad means this node is responsible by global routing system, so routing system number is 1**
- **Sector number 2 means this node belongs to sector 2**

Intersection 9

Number:	9
x-position:	137
y-position:	162
Node type:	Normal
Hierarchy type:	Intersection
Routing system:	2
Sector system:	2

Destination	Next
1	6
2	6
3	6

- **Hierarchical type is *Intersection* means this node is responsible by sector routing system, so routing system number is 2**
- **Sector number 2 means this node belongs to sector 2**

Figure 76: Hierarchical features setting

About other functionality of this hierarchical routing system simulation, users can read corresponding part in the Kroon's master thesis.