Bee Inspired Online Vehicle Routing in Large Traffic Systems

Horst Wedde, Sebastian Senge, Sebastian Lehnhoff, Fabian Knobloch, Tim Lohmann, Robert Niehage, Manuel Sträßer
School of Computer Science
Technical University of Dortmund
Dortmund, Germany
{horst.wedde, sebastian.senge, sebastian.lehnhoff, fabian.knobloch, tim.lohmann, robert.niehage, manuel.straesser}@tu-dortmund.de

***Abstract*—Traffic congestions have been a major problem in metropolitan areas worldwide, causing enormous economical as well as ecological damage. We argue that, due to the highly dynamic character of congestion forming and dissolving, any adequate solution for individual online vehicle routing in large traffic systems will require distributed, adaptive coordination of local navigators in order to transmit directions in due time before any road intersection which would still be valid when carried out. In this paper a completely distributed and adaptive swarm intelligence based multi-layered approach termed BeeJamA is presented. It features dynamic deadlines. There is no need of global or centralized information. We report on extensive simulation experiments with the MATSim simulator verifying BeeJamA's superior performance compared to existing models.**

***Index Terms*—Traffic management; traffic information system; distributed systems; swarm intelligence**

I. Motivation

Traffic congestions in densely populated areas induce high economical and ecological costs worldwide. The total economic loss in Europe alone, during the year 2000 was estimated to exceed 270 billion euro [1].

For an adequate individual online vehicle routing to be practically successful in large traffic systems it is necessary that driver directions should be transmitted and processed in due time before each intersection, and that they should still be valid when carried out. This calls for dynamic and very tight deadlines. (Otherwise the measures may even be counterproductive.) Also successful solutions should be seamlessly scalable to very large systems, and at the same time, they should be robust considering that when introduced the drivers will only gradually accept the routing system. None of the so far existing approaches satisfies all three constraints. Due to space limitations we have to mostly skip previous and related work, however, it is not difficult to see that any satisfactory model has to be based on distributed agent control and a dynamic management of deadlines in the range of very few seconds. Global information handling would make the system too slow. For coping with the highly dynamic complexity of individual vehicle traffic we developed a novel self-organizing online routing system termed BeeJamA (for *Bee Jam A*voidance). Here autonomous agents (navigators) coordinate their area information through a multi-layer communication structure, the latter relying on our own novel routing protocol BeeHive/BeeAdhoc [9] for large computer networks which had been inspired by the honey bee foraging communication [2], and was stepwise adapted to the road guidance situation. The objectives are congestion avoidance and minimization of individual travel times. After achieving a proof of concept through a fairly simple traffic model [3] we deal with the general case here. A notable related work is the H-ABC [4] , an adaption of AntNet [5] (and thus decentralized approach) to traffic networks, however, for a detailed comparison between BeeHive and AntNet as underlying agent model we refer to [9].

The next sections are structured as follows. Section II briefly outlines the behavior of honey bees in nature. Section III describes our communication and system architecture and Section IV introduces our novel Generic Routing Framework, as the central modeling contribution and at the same time the key for implementing BeeJamA (and other algorithms) into a simulator, and to compare it to traditional algorithms. Section V sketches the main ideas and concepts of the BeeJamA algorithm. In Section VI we report on our simulation experiments with the MATSim [6] simulator and summarize the results in Section VII.

II. Honey Bees in Nature

A honey bee colony reacts flexibly and adaptively to countless changes in the forage pattern outside the hive, and to changes inside the hive, through a decentralized and sophisticated communication and control system (by watching so-called waggle dances. In this way each honey bee follows simple stochastic rules relying on local information only. A reinforcing and self-regulating behavior emerges in a decentralized fashion where the amount of collected food is optimized in a decentralized fashion, and food sources are not overworked.

The exploitation of such principles by means of a *multi agent system* (MAS) is frequently termed Swarm Intelligence [7]. For a complete description of the biological background, which we used as the foundation of our MAS, see [2].

To sum up, the global optimization problem (of collecting enough food in the natural scenario or of jam reduction in the

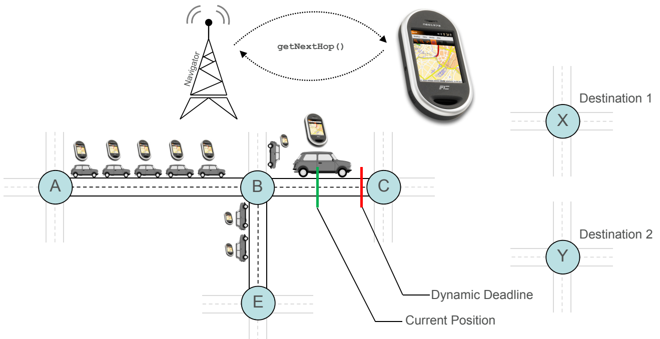

Fig. 1. Dynamic Deadline

road traffic scenario) is tackled by agents solving local and dynamic assignment problems under distributed control. Our algorithm borrows from this decentralized and self-organizing foraging behavior.

III. V2I Communication

The BeeJamA system uses an approach similar to Floating Car Data (FCD) [8] to get up-to-date traffic information. The BeeJamA algorithm disseminates this information and calculates individual routing instructions based on it.

In this section, we propose a vehicle-to-infrastructure (V2I) based architecture for the necessary communication with the vehicles. The cornerstone of the architecture is a decentralized network of so-called *navigators*. A navigator has a spatially limited area of responsibility, its *navigation area*, where it handles the communication with each vehicle and returns routing instructions on request.

The interaction between a vehicle and a navigator is depicted in Figure 1. After the driver specified a destination, a (GPS-enabled) personal navigation assistant (*PNA*), e.g. a smart cell phone, continuously submits the vehicle's position to its responsible navigator. In addition, a routing instruction request for the next intersection (hop) is sent from the PNA to the navigator each time a vehicle enters a new road. The navigator calculates an instruction (see Section V-D) based on up-to-date traffic information (which, in turn, is based on the continuously transmitted position information of the vehicles and disseminated by a flooding procedure). Ultimately, a hop-to-hop routing emerges, where each vehicle in a navigation area receives an individual next-hop instruction before each intersection in due time. This is a tough real-time problem as these individual deadlines depend on the speed of the vehicle moving within the area of its responsible navigator. Therefore, the size of a navigation area must be small enough to allow for timely detour calculation in the presence of congestions (and it is further limited by the actual wireless communication range).

To cope with these real-time constraints a completely distributed routing algorithm (based on the described decentralized V2I architecture) is necessary to achieve these objectives.

IV. Generic Routing Framework

In order to conduct simulations easily, we developed a middleware (termed *Generic Routing Framework* (GRF)) to hide the complexity of traffic simulation from the routing algorithms and to offer a common interface to routing algorithms from the simulation tools. In such a way we could easily switch to another simulator, or use a different set of routing algorithms. The GRF manages a graph G (representing the road network) and a set of agents called tokens (representing the vehicles) on the so-called *physical layer*. The physical layer maintains domain specific physical attributes like the length of a road and the position of each token at each time step. For that purpose, the simulator computes the vehicle positions and updates the GRF data structures accordingly (using the appropriate interfaces).

The weighted directed graph is given by $G = (V, E, w)$, where V is the set of vertices, E the set of edges and $w : E \times \mathbb{N} \rightarrow \mathbb{R}^+$ a time variant weight function (the second parameter $t \in \mathbb{N}$ represents a discrete time step). Edges represent roads and vertices represent intersections, thus the graph correlates directly with the actual road network. The weight function on the physical layer simply indicates the number of vehicles at time t on edge e, or their current travel times, as given by the road traffic simulation tools. The graph is assumed to be strongly connected.

The implemented routing algorithms may add abstract layers to maintain algorithm specific information for the computation of the routing decision, e.g., the BeeJamA algorithm adds two layers as described in the next section.

V. The BeeJamA Algorithm

The basic idea of our BeeJamA algorithm presented in this section is for the sake of scalability to follow a three layer approach (Figure 2) to allow real-time hop-to-hop routing in road networks of a realistic size. The first layer is given by the aforementioned physical layer. The BeeJamA algorithm has two more layers each consisting of a graph, a set of routing tables, and rules for the generation of bee agents used to disseminate local weight changes.

On the first additional layer, called *navigation area layer*, or *area layer* for short, the graph layout is nearly congruent the physical layer's graph structure but is partitioned into the areas. Each area belongs to a navigator which is responsible for maintaining the routing tables associated to the area, sending and receiving bee agents for the routing table updates and for communicating with the vehicles within the area. A vehicle which travels over larger distances thus traverses more than one area to reach its destination. On the second additional layer, called *net layer*, the information needed for routing between areas (and 'in the direction of the destination area") is managed. The actual techniques are described in the following sections.

On both layers two kinds of routing tables are used: *Next-Hop* (NH) and *Node-to-Hierarchy* (NTH) tables. Table I and II depict generic instances of these tables.

TABLE I
STRUCTURE OF A NEXT HOP TABLE

$NH(v)$	W_1	$\cdots$	$W_{\|W\|}$
$S(v)_1$	$c_{1,1}$	$\cdots$	$c_{\|W\|,1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$S(v)_{\|S(v)\|}$	$c_{1,\|S(v)\|}$	$\cdots$	$c_{\|W\|,\|S(v)\|}$

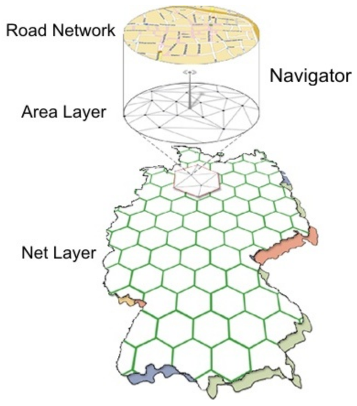

Fig. 2. Layered Routing Model

Let $S(v)$ denote the set of successors of node $v \in V$ and $W \subseteq V$ a set of destinations. Then, the cost to travel from node v to W_i over successor $S(v)_j$ is $c_{i,j}$ in a next hop table. The actual set W differs on both layers.

TABLE II
STRUCTURE OF A NODE-TO-HIERARCHY TABLE

NTH	W_1	$\cdots$	$W_{\|W\|}$
	H_1	$\ldots$	$H_{\|W\|}$

The node-to-hierarchy table is a simple associative memory, mapping nodes bijectively from a set W to sets $H_1, \ldots, H_{|W|}$. The elements of H_i are structural components like the aforementioned navigation areas. Such node-to-hierarchy tables are static and do not change throughout the simulation. In contrast, the table entries of next hop tables are time variant and are updated by incoming bee agents (see Section V-C).

A. Area Layer

The basic notion of our distributed routing algorithm is to divide the physical layer graph (i.e., the actual road network) into smaller parts called *navigation areas* (done on the area layer) and to connect them to allow inter-area travels (done on the net layer). So the navigator associated to each area could refrain from maintaining global information and must thus maintain only small routing tables with local (and therefore up-to-date) information for intra-area routing and information about a limited neighborhood for inter-area routing on the net layer. Technically, the area layer's set of vertices and edges are equal to their counterparts on the physical layer, $G_A = (V_A, E_A) = (V, E)$. The weight function w_A, however,

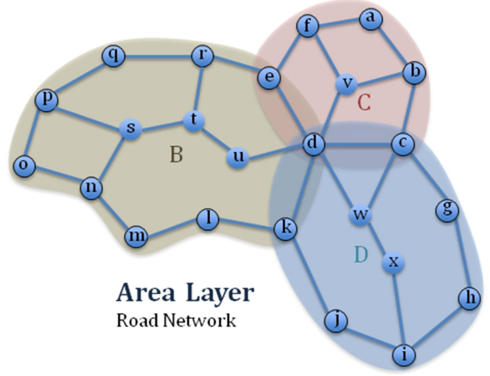

Fig. 3. Area Layer

differs, since it transforms the information available on the physical layer into (estimated and thus abstract) travel times for each edge. E.g., we use a simple moving average of the last n travel times over an edge (given by the simulator) to determine an edge's weight. An area is defined using an edge partition of the area layer's graph. Thus, given an edge partition $E_{A,1}, \ldots, E_{A,n}$, an area is a subgraph $A_i = (V_{A,i}, E_{A,i})$ with

$$V_{A,i} = \bigcup_{(e_k,e_l) \in E_{A,i}} \{e_k, e_l\}.$$

The set of border nodes $B(A)$ of an area A is given by $B(A) = \{b \in V_A | \exists i \in V \setminus V_A : (b,i) \in E\}$ and the set of inner nodes by $I(A) = V_A \setminus B(A)$. Please note, that border nodes belong to at least two areas. At the same time, the set of edges is partioned such that each edge is assigned to exactly one navigation area (e.g., edge (d,e) in Figure 3 belongs either to area B or C). We use a simple grid algorithm to obtain an edge partition. Each navigator maintains two tables: for each node $v \in V_{A,i}$ the next hop IFZ_{area} (intra foraging zone) table and a copy of the static and global node-to-hierarchy FRM_{area} (foraging region membership) table. The destination set $W = V_{A,i}$ is utilized for intra-area next hop routing (see section V-D for details on routing decisions). To locate a destination node's area, the navigator checks its FRM_{area} table which maps each node to its associated areas.

B. Net Layer

Typically vehicles may drive across several areas to reach their individual destinations. To satisfy those routing requests, for each area on the area layer a so-called *net area* on the second layer, the *net layer*, is created. These net areas are mapped onto fixed *foraging regions*, modeling the vicinity of a destination node (see Figure 4 and Figure 5 for examples).

Each foraging region is represented by a *representative node*. In addition, each node v on the net layer maintains a specific *foraging zone* $FZ^r_{net}(v)$ that consists of all neighboring nodes within a certain hop range r. It models the direct vicinity of a source node for which accurate routing information is available. They are constructed by a flooding procedure originally developed for the BeeHive algorithm [9].

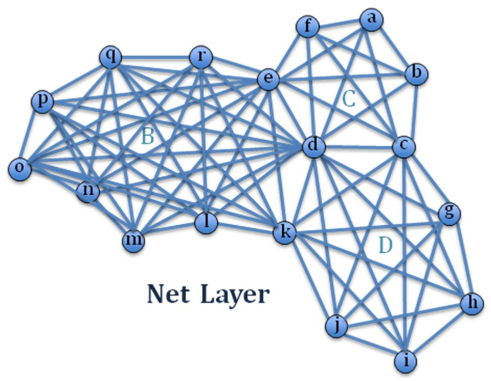

Fig. 4. Extended Net Layer with 3 Areas

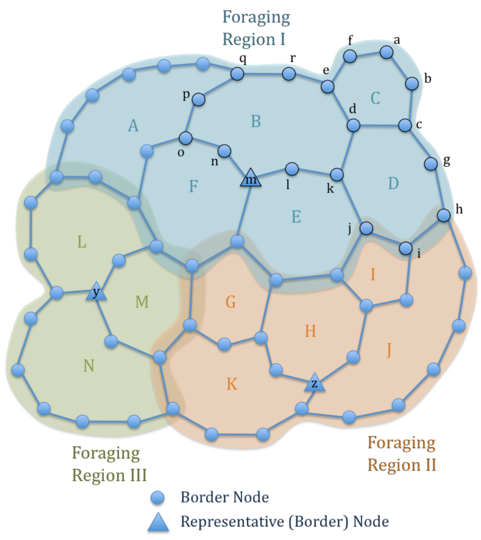

Fig. 5. Net Layer with Foraging Regions

Let $A_1, \ldots, A_n$ be the areas on the area layer, then the net area N_i is given by the fully connected graph $N_i = (B(A_i), E_i)$ with

$$E_N = \bigcup_{i=1}^{n} \{(v_1, v_2) \in E | v_1 \in B(A_i) \wedge v_2 \in B(A_i)\}\,.$$

The net area's navigator maintains three tables: for each node $v \in B(A_i)$ the next hop tables IFZ_{net} (inter foraging zone) and IFR_{net} (intra foraging region) and a copy of the static and global node-to-hierarchy FRM_{net} table . The IFR_{net} table (in analogy to the IFZ_{area} table) stores costs from each neighbors to each nodes in the same foraging zone on the net layer ($W = FZ^r_{net}(v)$). In addition, the IFR_{net} table stores costs from each neighbors to each representative nodes and is used to forward vehicles if the destination is far away (see Section V-D for more details) and finally the FRM_{net} table maps each net layer node to its foraging region (and thus to its associated representative node).

C. Table Updates

In order to continuously update the routing tables, we follow a multi agent system (MAS) approach. We use two types of agents, inspired by the honey bee behavior: the majority of the foragers exploit food sources in the direct vicinity of the hive, while a minority visit food sources which are further away. We adapted this concept into *Short Distance Bee Agents* and *Long Distance Bee Agents*. Both types of agents are responsible for disseminating routing cost information. They only differ in the distance (hops) that they are allowed to travel starting from their launching node. Due to page limitations we refer the reader to a comprehensive description of the MAS based update process in [9].

Here, we present a shortend informal version of this process:

1) Each non-representative node periodically sends a short distance bee agent, by broadcasting replicas of it to each neighbor site.
2) When a replica of a particular bee agent arrives at a site it updates routing information there, and the replica will be flooded again (it will not be sent to the neighbor from where it arrived). This process continues until the life time (number of hops) of the agent has expired, or if a replica of this bee agent had been received already at a site. In the latter case the new replica will be killed there.
3) Representative nodes launch long distance bee agents that would be received by the neighbors and propagated as in 2. However, their life time (number of hops) has a higher limit, the long distance limit.

As a result of this flooding based approach, each node always has up-to-date cost information (depending on the flooding period and the graph layout) of the travel times to the nodes in its own foraging zone and to the representative nodes.

D. Routing decisions

In the area and net layer model, there are three possible cases for routing a vehicle from node x within area X to a destination node y within a destination area Y:

1) The destination node y is listed in the current area X's IFZ_{area} table. Thus, the destination node y lies within the current area X (areas X and Y are the same):

$$\textbf{RC I}: y \in X$$

2) Case 1 is not true but the destination area Y is listed (relative to every border node of Y) in the IFZ_{net} tables of the current area X's border nodes. Thus the destination node y lies outside of area X but within an area Y that lies within the net layer foraging zone[1] of area X:

$$\textbf{RC II}: \neg\text{RC I} \wedge B(Y) \subseteq FZ^r_{net}(X)$$

3) Case 1 and 2 are not true (i.e., the destination node y lies outside of area X and the destination area Y lies outside of the net layer foraging zone of area X):

$$\textbf{RC III}: \neg\text{RC II}$$

[1]$\forall X \subseteq V : FZ^r_{net}(X) = \bigcap_{x \in X} FZ^r(x)$

The cases above reflect a sequence of scenarios in which the destination node y lies further and further away from the current area X.

The navigator in X selects the vehicle's next hop by creating a set of *cost paths* $C(x, y)$, evaluating the costs $c(p)$ of each cost path $p \in C(x, y)$ as appropriate in the particular routing case and drawing a cost path p (including a next hop) accordingly. Thus, the decision process in each routing case is described by stating the set of cost paths $C(x, y)$ and the associated cost function $c(p), p \in C(x, y)$. The cost paths[2] $C(x, y) = \left\{ \left(x \dot{\rightarrow} S(x) \rightarrow \ldots \rightarrow y \right) \right\}$ are no complete paths from x to y but a sequence of nodes used to approximate the costs of choosing each $s \in S(x)$ as a next hop.

Routing in case 1 is fairly simple. The vehicle's next hop is chosen probabilistically according to the (normalized) costs of the next-hop entries in the IFZ_{area} table of area X. The cost path expresses that the destination node y is reachable within the current area by each neighbor $s \in S(x)$:

$$C(x, y) = \left\{ \left(x \dot{\rightarrow} S(x) \rightarrow y \right) \right\}$$

Thus, the number of cost paths to evaluate is

$$|C(x, y)| = |S(x)|$$

Each cost path $p = (x \dot{\rightarrow} s \rightarrow y) \in C(x, y)$ can be easily calculated by a single IFZ_{area} table lookup:

$$\begin{aligned} c(p) &= c(x \dot{\rightarrow} s' \rightarrow y) \\ &= IFZ^{X}_{area}(x, s', y) \end{aligned}$$

This table is managed by the current navigator, so that no additional intra-navigator network traffic is required for routing a vehicle within the destination area. For an adequate routing in case 2 travel times have to be accumulated across the different layers in BeeJamA.

Therefore, travel costs are composed of three parts:

- The travel times from the vehicle's current position at node x to a border node of area X (over neighbor $s \in S(x)$).
- The minimum travel times from a "suitable" border node of the current area X to a border node of the destination area Y on the net layer.
- The minimum travel times from a border node of area Y to the destination node y.

The following cost paths are used to approximate these travel costs:

$$C(x, y) = \left\{ \left(x \dot{\rightarrow} S(x) \rightarrow B(X)_i \overset{\min}{\rightarrow} B(Y)_j \overset{\min}{\rightarrow} y \right) \right\}$$

and

$$|C(x, y)| = |S(x)| \cdot |B(X)| \cdot |B(Y)_i|$$

The cost function is given by:

$$\begin{aligned} c(p) &= c(x \dot{\rightarrow} s \rightarrow B(X)_i) \\ &\quad + \check{c}(B(X)_i \rightarrow B(Y)_j) + \check{c}(B(Y)_j, y) \\ &= c(IFZ^{X}_{area}(x, n', B(X)_i)) \\ &\quad + \min(IFZ^{X,Y}_{net}(B(X)_i, B(Y)_j))) \\ &\quad + \min(IFZ^{Y}_{area}(B(Y)_j, y))) \end{aligned}$$

(Here the min notation is used to denote the minimum table entry from the first argument as current node and the second argument as destination). Only the last addend is unknown to the current navigator and the appropriate minimum value must be requested from the navigator of area Y.

For an adequate routing in case 3 travel times are composed of only two parts:

- The travel times from the vehicle's current position at node x to a border node in its current area X.
- The minimum travel times from a "suitable" border node of the current area X to the representative node R_Y of area Y's foraging region.

The key idea (adopted from the foraging behavior of bees in nature) is to route a vehicle in the rough direction towards the foraging region of its target area Y until it gets close enough to its destination for it to be listed in the IFZ_{net} table of a visited node.

The cost paths to approximate the travel costs are

$$C(x, y) = \left\{ \left(x \dot{\rightarrow} S(x) \rightarrow B(X)_i \overset{\min}{\rightarrow} R_Y \right) \right\}$$

with

$$|C(x, y)| = |S(x)| \cdot |B(X)|$$

$$\begin{aligned} c(p) &= c(x \dot{\rightarrow} n' \rightarrow B(X)_i) + \check{c}(B(X)_i \rightarrow R_Y) \\ &= c(IFZ^{X}_{area}(x, n', B(X)_i) \\ &\quad + min(IFR^{X}_{net}(B(X)_i, R_Y)) \end{aligned}$$

Finally, to draw a next hop $s \in S(x)$ the aforementioned costs must be transformed into probabilities, so that higher costs (higher travel times) result in lower next hop probabilities. For a cost path $p_i \in \{p_1, \ldots, p_n\} \in C(x, y)$ the probability is given by

$$\begin{aligned} P(p_i) : &= \frac{\left(\frac{c(p_i)}{\sum_{j=0}^{n} c(p_j)} \right)^{-1}}{\sum_{k=0}^{n} \left(\frac{c(p_k)}{\sum_{j=0}^{n} c(p_j)} \right)^{-1}} \\ &= \frac{1}{c(p_i) \sum\limits_{k=0}^{n} \frac{1}{c(p_k)}}, \end{aligned}$$

[2]Please note, that $\{(x \rightarrow \ldots \rightarrow M \rightarrow \ldots \rightarrow y)\} = \{(x \rightarrow \ldots \rightarrow m_1 \rightarrow \ldots \rightarrow y), \ldots, (x \rightarrow \ldots \rightarrow m_n \rightarrow \ldots \rightarrow y)\}$, if $M = \{m_1, \ldots, m_n\}$.

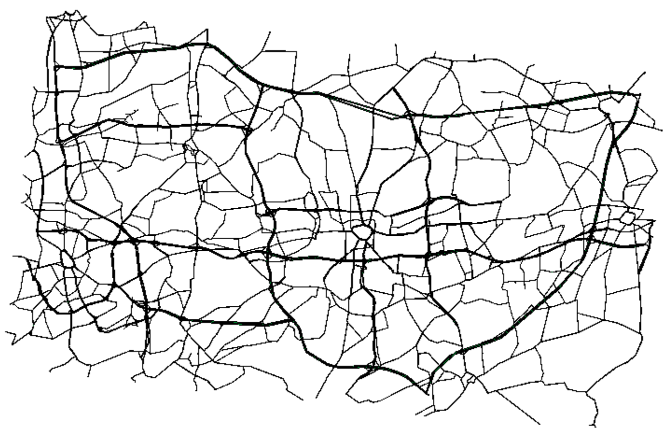

Fig. 6. Eastern Ruhrgebiet District

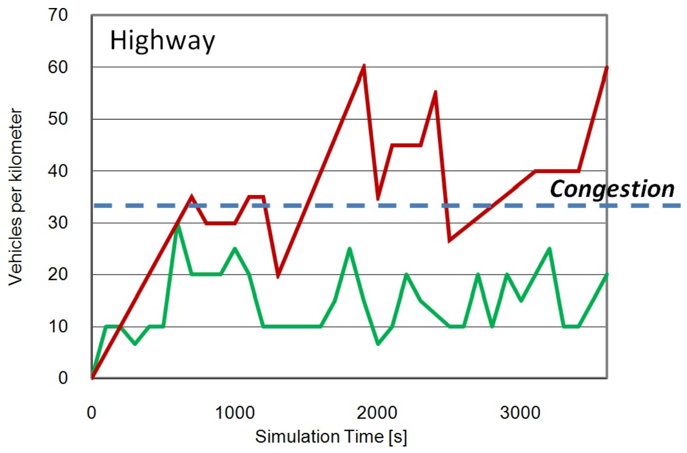

Fig. 7. Density

which is simply the normalization of the inverted normalized costs.

If the cost path $p_i = (x \dot{\rightarrow} s \rightarrow \ldots \rightarrow y)$ is chosen, then $s \in S(x)$ is the next hop sent back to the vehicle.

Bees in nature perform the waggle dance only if the quality of a discovered food source exceeds a certain threshold. Otherwise, the food source is discarded and no new foragers will be recruited. In analogy to this behavior, a route may only be selected if its travel cost does not exceed a dynamic threshold (and thus would avoid unacceptable detours).

VI. Simulation Studies

For our simulation experiments on MATSim [6] we selected the major part of the Eastern Ruhr District (see Figure 6, bold lines indicate highways) in Germany, a densely populated, formerly heavy industry area, featuring a good of approx. 1850 sections and approx. 3800 intersections. We also selected a variety of characteristic real-world initial configurations and utilized our own graphical tools to follow the events as they occurred under BeeJamA and one of the commercially available shortest-path based services. We picked quite a few road sections that turned into bottlenecks under the shortest-path approach. This history for one section is to be observed in Figure 7. Here the car density is plotted over the simulation time. A threshold (see [10] for more details) is marked which indicates that any value above means congestion on this section. As it can be observed, the shortest-path approach (dark color) results in quite a far serious congestion while on the other hand BeeJamA (light color) leaves this section congestion-free. In another course of study - which we cannot display in detail, due to page limitations - we selected, starting again from real-world scenarios, a substantial subset of cars with fixed source and destination, and we kept track of their travel times in subsequent runs. Here again, it was clear that in the average over the runs, the travel times were kept considerably lower compared to shortest-path solutions.

VII. Conclusions

The BeeJamA traffic guidance system is a highly flexible distributed multi-agent system. While self-organizing under the challenge of dynamic and very tight deadlines it makes sure that any directions are still accurate when carried out. No global information is needed for decision making. So altogether BeeJamA has the potential from the beginning to be robust under gradually growing degrees of penetration (among the drivers). Our results so far support this claim clearly. Also scalability is not a serious issue since the new structure over the navigation areas could be layered further in order to cover systems of the size of Germany. The idea behind is that the more distant a disturbance is the less influential it will be in a particular city, a principle that is borrowed from the swarm intelligence of honey bees.

The upcoming research will focus on more and more realistic scenarios and a much more detailed experimental insights. This will be subject of forthcoming publications.

References

[1] W. Rothengatter. *External costs of transport, 2004,* http://www.uic.org/html/environnement/cd_external/pages/introduction.html*, last access at: 08/27/2010*

[2] M. Farooq. *Bee-Inspired Protocol Engineering - From Nature to Networks*. Springer, Berlin, 2009.

[3] T.D. Seeley. *The Wisdom of the Hive*. Harvard University Press, Cambridge, 1995.

[4] H. F. Wedde, S. Lehnhoff, S. Senge, and A. M. Lazarescu et. al. *A Novel Class of Multi-Agent Algorithms for Highly Dynamic Transport Planning Inspired by Honey Bee Behavior*. In: Proc. of the 12th IEEE Conf. on Emerging Technologies and Factory Automation, Patras, 2007.

[5] B. Tatomir, L.J.M. Rothkrantz, *H-ABC A scalable dynamic routing algorithm*, In: Recent Advances in Artificial Life, World Scientific Publishing Co. Pte. Ltd. 5 Toh Tuck Link, Singapore 596224

[6] G. Di Caro, M. Dorigo, *AntNet: Distributed Stigmergetic Control for Communications Networks*, In: Journal of Artificial Intelligence Research, vol. 9, pag. 317-365, 1998

[7] MATSim - Multi-Agent Transport Simulation Toolkit, official homepage at: http://www.matsim.org, last access at: 08/27/2010

[8] E. Bonabeau, M. Dorigo and G. Theraulaz. *Swarm Intelligence - From Natural to Artificial Systems*. Oxford University Press, Ney York, 1999.

[9] B.S. Kerner et al. *Traffic State Detection with Floating Car Data in Road Networks*. IEEE Intelligent Transportation Systems, 2005.

[10] L. Neubert et. al. *Statistical Analysis of Freeway Traffic*. In: Traffic and Granular Flow '99, Springer, 2000.