ant colony system, car navigation, ambulance

Wojciech BURA[1], Mariusz BORYCZKA[2]

ANT COLONY SYSTEM IN AMBULANCE NAVIGATION

This paper describes the ant-based vehicle navigation algorithm (known from literature) and then proposes its modification. Because the performance of these two versions of the algorithm is unsatisfactory, the new, really revised algorithm is introduced. Experiments performed on real-world data show, that this new algorithm is much more efficient and gives better results than its predecessors. The proposed algorithm may be used in the rescue, where ambulances need the useful and efficient method to plan a route to the patient.

1. INTRODUCTION

In recent years we witness the rapid growth in popularity and development of car navigation systems. This is primarily due to the increasing availability of portable electronic devices equipped with a GPS module, which may be used for this purpose. The object of this work is to analyze the possibility of using ant algorithms in the search process of the optimal route between two selected points on the map. The proposed algorithm may be used in the rescue, where ambulances need the useful and efficient method to plan a route to the patient. It is important that we search for an algorithm which is able to propose not only the shortest route (as e.g. the Dijkstra's algorithm), but a set of good solutions. Thus, it is possible to chose the alternative route when the unforeseeable situation (traffic jam, temporarily closed road etc.) occurs. The deterministic algorithms solving this problem give only one solution, optimal, but not always acceptable in the specific situation. Also, some other methods were used (e.g. [1, 3, 4]), but here we propose to use the ant algorithm for this purpose.

The ant algorithm realizing this functionality was proposed in [6-8]. This algorithm is based on the basic version of the ant algorithm described for example in [2, 5]. It finds the optimal (or nearly optimal) route between two points on the map, using user preferences for distance, traffic load, road width, risk of collision, quality and number of intersections. Parameterization of the algorithm is performed by setting the corresponding coefficients for the different criteria of optimality. During calculations, the time of departure is also taken into account due to the fact that the weight of the individual segment of the route may have assigned different values at different times of day or night. Unfortunately, the mentioned algorithm not always provides the optimal solution (especially for real data). Therefore this paper proposes firstly its modification and secondly, the new, really revised version of that algorithm.

This work is organized as follows. Section 2 shortly describes ant systems, section 3 presents an original algorithm (Ant-based Vehicle Navigation algorithm — AVN) and its slightly modified version (CAVN). In section 4 a new ant algorithm (NAVN), much more efficient and capable of use for the actual data, is proposed. Section 5 presents the results of computational experiments carried out on several data sets: on which the AVN algorithm was tested in [6] and [8] (project "Kerman"), on data similar to the project "London" [7] and on new data obtained from the Open Street Map system. Section 6 summarizes the work.

2. ANT ALGORITHMS

Ant algorithms take inspiration from the behaviour of real ant colonies to solve combinatorial optimization problems. They are based on a colony of artificial ants, that is, simple computational agents that work cooperatively and communicate through artificial pheromone trails [2].

The artificial ant in turn is a simple, computational agent that tries to build feasible solutions to the problem tackled exploiting the available pheromone trails and heuristic information. It has some

[1] Institute of Computer Science, University of Silesia, Będzińska 39, 41-200 Sosnowiec, Poland. wojciech.bura@asseco.pl,
[2] Institute of Computer Science, University of Silesia, Będzińska 39, 41-200 Sosnowiec, Poland. mariusz.boryczka@us.edu.pl.

characteristic properties. It searches minimum cost feasible solutions for the problem being solved. It has a memory, storing information about the path followed until that moment. It starts in the initial state and moves towards feasible states, building its associated solution incrementally. The movement is made by applying a transition rule, which is a function of the locally available pheromone trails and heuristic values, the ant's private memory, and the problem constraints. When, during the construction procedure, an ant moves, it can update the pheromone trail associated with the edge. The construction procedure ends when any termination condition is satisfied, usually when an objective state is reached. Once the solution has been built, the ant can retrace the travelled path and update the pheromone trails on the visited edges/components.

3. AVN ALGORITHM AND ITS MODIFICATION

The AVN algorithm is designed for computing the optimal route between two points on the map. It is based on the ant system introduced by Dorigo, Maniezzo and Colorni.

3.1. ORIGINAL AVN ALGORITHM

The Ant-based Vehicle Navigation algorithm (AVN) finds optimal route, which best fits preferences desired by the user. The set of user parameters consists of coefficients controlling the importance level of distance, width, number of intersections, traffic, risk and quality of the proposed route. Steps of the algorithm are as shown on the AVN procedure. Below we shortly describe the elements of the mentioned algorithm.

```
Procedure AVN
1  begin
2      Initialize;
3      foreach loop do
4          Locate Ants;
5          foreach iteration do
6              foreach ant do
7                  if ant is active then
8                      Construct Probability;
9                      Select Route;
10                     Update Tabu List;
11                     Kill blocked ant;
12         Value Ants;
13         Award Winner Ants;
14         Punish Loser Ants;
15         Evaporate Pheromone;
16     Select Best Optimized Direction;
```

Fig. 1. AVN procedure.

First, initial setup of algorithm parameters is performed. Then ants are located at the starting point. Each ant is active until it reaches the destination point and is not blocked at the intersection. An ant is blocked when there is no way to choose to continue the travel. In the next step, the probability of each possible next direct route is calculated based on the cost function for each active ant. The probability of the move from node *i* to node *j* for ant *k* is calculated as:

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\prod\limits_{l\in parameters}\xi_{ij_l}^{-2\alpha_l}}{\sum\limits_{h\notin tabu_k}\tau_{ih}^{\alpha}\prod\limits_{l\in parameters}\xi_{ih_l}^{-2\alpha_l}} & \text{if} \quad j\notin \text{tabu}_k \\ 0 & \text{otherwise}\end{cases} \tag{1}$$

where:

τ_{ij} — value of pheromone trail on edge ij,

α — coefficient that controls the importance level of τ_{ij},

tabu_k — a set of unavailable nodes, already visited by the ant k,

ξ_{ijl} — value of cost functions for parameter l for edge ij,

α_l — coefficient controlling the importance level of parameter l,

and cost functions for all parameters for edge ij are [6]: $\xi_{ij_{distance}}$, $\xi_{ij_{width}}$, $\xi_{ij_{traffic}}$, $\xi_{ij_{risk}}$ and $\xi_{ij_{quality}}$.

Based on the calculated probability ant k selects a route to go. To do this, a random value q in range $\langle 0,1\rangle$ is compared with parameter $Q\in\langle 0,1\rangle$, to choose an exploitation or an exploration:

$$j=\begin{cases}\arg\max\limits_{h\in J_i^k}\{p_{ih}^{k}\} & \text{if } q\le Q \text{ (exploitation)} \\ S & \text{otherwise (exploration),}\end{cases} \tag{2}$$

In next steps the node selected by the ant is added to its tabu list and if ant k arrives the destination or is blocked at the certain node, it is deleted from the active ant list. For each ant that reached the destination the complete cost ψ of the whole route is calculated and for all calculated costs ψ the average cost χ is worked out. If $\psi_k<\chi$ then ant k is added to AWA (Award Winner Ants) list, otherwise it is added to PLA (Punish Loser Ants) list. In addition, for an ant (if any) with cost ψ lower than the cost of the global best solution then the new best solution is remembered.

At the end of every loop updating of the pheromone trail takes place. The pheromone trail on the edge ij for ant k is updated as follows:

$$\tau_{ij}(t)=\begin{cases}\tau_{ij}(t-1)+\dfrac{av}{\psi_{ij}} & \text{if } k\in \text{AWA} \\ \tau_{ij}(t-1)\cdot pv & \text{if } k\in \text{PLA}\end{cases} \tag{3}$$

where av is the awarding coefficient, $av>1$, and pv is the punishment coefficient, $0<pv<1$. Then the global evaporation of the pheromone trail is performed as $\tau_{ij}(t)=\rho\cdot\tau_{ij}(t-1)$, where ρ is the evaporation coefficient, $0<\rho<1$. After the assumed number of repetitions of the algorithm the global best solution is returned as the result.

3.2. CORRECTED AVN ALGORITHM

We have implemented the original AVN algorithm but experiments performed showed that this algorithm does not always give optimal results. Therefore we prepared its slightly improved version (CAVN). The main change was a normalization procedure of each element of the cost function and the way of parameterization of user preferences. In the original algorithm particular α coefficients work the same way as user preferences and they play a role of the normalization. Thus, proper set-up of these parameters is not an easy task, and what is even more important, it is not comfortable for the user.

In the CAVN algorithm, the normalization coefficients are initially calculated at the beginning and they are then used during calculations of the cost function and probabilities. These values depend on the map only, so that they may be prepared before algorithm starts.

```
Procedure CAVN
1  begin
2      Prepare Normalization;
3      Initialize;
4      foreach loop do
5          Locate Ants;
6          foreach iteration do
7              foreach ant do
8                  if ant is active then
9                      Construct Probability;
10                     Select Route;
11                     Update Tabu List;
12         Value Ants;
13         Award Winner Ants;
14         Punish Loser Ants;
15         Evaporate Pheromone;
16         Update Q;
17     Select Best Optimized Direction;
```

Fig. 2. CAVN procedure.

Procedure CAVN presents steps of this algorithm. Below we describe new and modified procedures of the CAVN algorithm (underlined in pseudo-code).

Prepare Normalization. In this step for each element of the cost function (distance, width, traffic, risk, quality and intersections) normalization coefficient η_l is calculated. Therefore user preference parameters have values in range $\langle 0,1 \rangle$ and there is no need to adjust them to particular data on the map.

For calculations of the cost function maximum and minimum values of the user preferences are assigned to variables: $\max_{distance}$, $\max_{width}$, $\max_{traffic}$, $\max_{risk}$, $\max_{quality}$, and $\max_{all} = \max\{\max_{distance}, \max_{width}, \max_{traffic}, \max_{risk}, \max_{quality}\}$ is calculated. Then particular coefficients are calculated as follows:

$$\eta_{distance} = \frac{\max_{all}}{\max_{distance}} \tag{4}$$

$$\eta_{width} = \frac{\max_{all}}{\max_{width} / \min_{width}} \tag{5}$$

$$\eta_{traffic} = \frac{\max_{all}}{\max_{traffic}} \tag{6}$$

$$\eta_{risk} = \frac{\max_{all}}{\max_{risk}} \tag{7}$$

$$\eta_{quality} = \frac{\max_{all}}{\max_{quality} / \min_{quality}} \tag{8}$$

$$\eta_{intersection} = \max_{all} \tag{9}$$

Construct Probability. Here, normalization coefficients ηl are calculated (they are exploited in calculations of probabilities):

$$\xi_{ij_{distance}} = d(i, j) \cdot \eta_{distance} \tag{10}$$

$$\xi_{ij_{width}} = \frac{\max_{width}}{w(i, j)} \cdot \eta_{width} \tag{11}$$

$$\xi_{ij_{traffic}} = tf(i, j, t) \cdot \eta_{traffic} \tag{12}$$

$$\xi_{ij_{risk}} = r(i, j, t) \cdot \eta_{risk} \tag{13}$$

$$\xi_{ij_{quality}} = \frac{\max_{quality}}{q(i, j)} \cdot \eta_{quality} \tag{14}$$

Value Ants. Normalization coefficients are finally used in the complete cost calculation: $\xi^k_{intersection} = \text{sizeof}(tabu_k) \cdot \eta_{intersection}$.

Award Winner Ants. In this step usage of parameter *av* was slightly modified:

$$\tau_{ij}(t) = \begin{cases} \tau_{ij}(t-1) \cdot av & \text{if } k \in \text{AWA} \\ \tau_{ij}(t-1) \cdot pv & \text{if } k \in \text{PLA} \end{cases} \tag{15}$$

where *av* is the awarding coefficient, $av > 1$, and *pv* is the punishment coefficient, $0 < pv < 1$.

Update Q. In this step parameter *Q* is decreased by coefficient φ ($\varphi \in \langle 0,1 \rangle$) which is an algorithm's parameter $Q(new) = Q(old) \cdot \varphi$. This way in each loop we gradually reduce exploration on behalf of exploitation.

4. NEW VERSION OF AVN — THE BACKTRACKING ALGORITHM (NAVN)

Running original and improved AVN algorithms on bigger datasets sometimes resulted in no solution. Removal of blocked ants has proven to be too aggressive, and none of ants has reached the destination. As a result of our research a new ant algorithm (NAVN) has been constructed. This algorithm is more similar to classical form of ACO algorithms.

```
Procedure NAVN
1  begin
2      Prepare Normalization;
3      Initialize;
4      foreach loop do
5          Locate Ants;
6          foreach iteration do
7              foreach ant do
8                  if ant is active then
9                      Construct Probability;
10                     if ant has no move then Move Back;
11                     else
12                         Select Route;
13                         Update Tabu List;
14         Value Ants;
15         Award Best Solution;
16         Punish Loser Ants;
17         Modify Q;
18     Select Best Optimized Direction;
```

Fig. 3. NAVN procedure.

In NAVN algorithm, killing the blocked ants is replaced by their returns from dead ends. Pheromone trail is updated both locally (when an ant is traveling on the map) and globally (after every loop) on the paths from the best solution. Additionally, while moving back a blocked ant highly reduces the pheromone trial on the edge that led it to a node without a way out. This considerably reduces the probability of selecting this edge by other ants. Procedure NAVN presents the algorithm and below we describe its main steps:

Initialize. Setting the parameters of the algorithm. The initial amount of pheromone on edges is set to value τ_0.

Construct Probability. Calculation of the components of the probabilities is more similar to the classical ant algorithm and is expressed by the formula:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij} \cdot \left(\dfrac{1}{\psi_{ij}}\right)^{\beta}}{\sum\limits_{h \notin \text{tabu}_k} \tau_{ih} \cdot \left(\dfrac{1}{\psi_{ih}}\right)^{\beta}} & \text{if } j \notin \text{tabu}_k \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where τ_{ij} — value of pheromone trail on the edge from i to j, β — coefficient controlling importance of the cost, ψ_{ij} — the cost of the edge from i to j calculated as follows:

$$\psi_{ij} = \sum_{l \in \text{parameters}} \xi_{ij_l} \cdot \alpha_l \tag{17}$$

where ξ_{ij_l} is the value of the cost function for parameter l and edge (i, j), calculated as in the CAVN algorithm, and α_l is the coefficient controlling importance of parameter l assigned by the user $(0 < \alpha_l < 1)$.

Move Back. If ant k is locked (there is no edge it can travel), it makes one step back. The edge used to go back is added to the list called $blindEdges_k$, which contains list of forbidden edges which will be no more chosen by ant k in current loop. The pheromone trail on the edge is updated as follows: $\tau_{ij}(new) = \tau_{ij}(old) \cdot bv$, where bv is the blind edge pheromone change parameter.

Select Route. Choosing the edge is performed the same way as in previous algorithms but after selecting the edge an ant updates a pheromone trail according to the formula:

$$\tau_{ij}(new) = (1-\rho) \cdot \tau_{ij}(old) + \rho \cdot \tau_0 \tag{18}$$

where ρ is the trail evaporation coefficient ($0 \le \rho \le 1$).

Value Ants. Solutions found by the ants are evaluated. If there is a solution better than the best one already found, it is saved as the new best solution. As in the previous algorithms, total cost includes the cost connected with the number of intersections:

$$\xi^k_{intersection} = \text{sizeof}(\text{tabu}_k) \cdot \eta_{intersection} \tag{19}$$

Award Best Solution. The best solution is rewarded through a global pheromone trail updating rule according to the principle expressed by the formula:

$$\tau_{ij}(new) = (1-\rho) \cdot \tau_{ij}(old) + \gamma \cdot \rho \frac{\theta_{best}}{\psi_{best}} \tag{20}$$

where θ_{best} — number of edges belonging to the best solution's path, ψ_{best} — the cost of the best solution and γ — the parameter reinforcing the award for the best solution.

Punish Non Active Ants. In this step, ants which failed to find a solution within a assumed number of iterations are punished — the pheromone trail on the edges belonging to their routes is decreased with the punishment coefficient $pv \in (0,1)$:

$$\tau_{ij}(new) = pv \cdot \tau_{ij}(old). \tag{21}$$

5. COMPUTATIONAL EXPERIMENTS

Experiments were performed on different data sets: (a) original data used in the experiment "Kerman "[6], (b) crafted data corresponding to the project "London"[7] and (c) large real data from the system Open Street Map (http://www.openstreetmap.org/).

Table 1. Preferences of parameters.

	α_d	α_w	α_t	α_r	α_q	α_i	Preference
Kerman1	1.0	0.45	0.45	0.25	0.25	0.3	from [6]
Kerman2	1.0	0.75	0.60	0.75	0.50	0.50	from [6]
Distance	1.0	0.1	0.1	0.1	0.1	0.1	distance
Width	0.5	1.0	0.1	0.1	0.1	0.1	width
Traffic	0.1	0.1	1.0	0.1	0.1	0.1	traffic
Risk	0.1	0.1	0.1	1.0	0.1	0.1	risk
Quality	0.1	0.1	0.1	0.1	1.0	0.1	quality
Inter	0.1	0.1	0.1	0.1	0.1	1.0	intersection

Experiments were carried out on the following computer system: processor: AMD Athlon 64 1.8 GHz 3000+, RAM: 1 GB, OS: Windows XP SP2, JDK: 1.6. During the experiments the following algorithms have been used for comparison: AVN, CAVN, NAVN and Dijkstra's classic algorithm (as an example of an algorithm giving the optimal but only one solution). The algorithms were run with different settings of preferences (Table 1) and with the parameters, including those for the ant algorithm, defined in Table 2 (m is the number of ants and $N_{\max}$ — the loop count). The values of parameters of the ant algorithm were, as usual, set as a result of preliminary experiments.

Below we present only the main results of experiments limited by the permissible volume of this work.

Table 2. Parameters of the algorithms.

Parameter	α	β	av	pv	bv	ρ	Q	φ	τ_0	m	$N_{\max}$
AVN	2	–	950	0.9	–	0.9	0.9	–	–	10	50
CAVN	2	–	1.05	0.9	–	0.9	0.9	0.99	–	10	50
NAVN	–	2.5	–	0.9	0.1	0.2	0.9	0.99	1/6000	10	50

6. EXPERIMENTS ON DATA FROM PROJECTS "KERMAN" AND "LONDON"

The data for these experiments were obtained from the authors of the AVN algorithm. Graph consists of 27 nodes and 67 edges, and represent partial map of the city of Kerman (Fig. 4). The average number of edges (incoming and outgoing) per node is 2.48. The route was sought from point 8 to 22. Time of departure was 17:30 and average speed — 40 km/h.

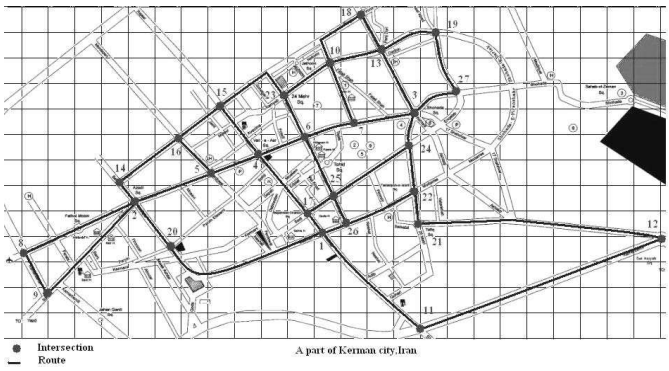

Fig. 4. Fragment of Kerman city map used in the experiments [6].

Algorithms were searching for a solution with the lowest possible cost. All algorithms, except AVN, found optimal solutions. Execution time of ant algorithms was greater than execution time of the deterministic algorithm.

The data for experiment "London" were prepared so that the number of nodes was comparable with the data described in [7] (the original data were not available). Map consisted of 54 nodes and 184 edges. The average number of edges (incoming and outgoing) per node was 3.4. The route was sought from point 1 to 54. Time of departure was 17:30 and average speed — 40 km/h.

In this experiment the NAVN algorithm found the optimal solution, the same one as the deterministic algorithm. Both AVN and CAVN algorithms did not find this solution, but CAVN produced better results. Dijkstra's algorithm was still working at the non-measurable time. The overall results of experiments are presented in Table 3 (time is expressed in milliseconds).

Table 3. Results of experiments with data of “Kerman” and “London”.

	Kerman			London		
Algorithm	Time	Cost	Route	Time	Cost	Route
AVN	156	6290	8,2,5,4,6,7,3, 24,22	266	1185	1,3,6,7,8,9,11,28, 30,31,29,32,34,33,40, 44,45,46,51,54
CAVN	79	5425	8,2,20,1,26,22	188	880	1,4,7,8,10,14,15, 19,39,43,47,52,53,54
Dijkstra	≈0	5425	8,2,20,1,26,22	≈0	860	1,3,6,13,14,15,19,39, 43,47,52,53,54

7. EXPERIMENTS ON THE REAL DATA

The data for this experiment were collected from the system Open Street Map (OSM) for the area of the city of Katowice (Fig. 5). Map size used in this experiment exceeds several times the size of the maps used in the projects “Kerman” and “London”. This map consists of 31044 nodes and 68934 edges. The average number of edges (incoming and outgoing) per node is 2.2. In this experiment only the preferences of the distance and width were used, other values were constant. This is due to the limited scope of information available in the OSM system. The start node was OSM id.: 383783583 (suburbs of Katowice) and end node id. was 384912139 (downtown of Katowice). The time of departure was 17:30 and travel speed: 40 km/h.

Fig. 5. Map used in experiments on real data.

During the preliminary experiments we found that the original algorithm AVN and its improved version (CAVN) were unable to find any solution, so they are omitted from presented results, and the NAVN algorithm was compared only with the classical Dijkstra’s algorithm. Results of experiments for the project “Katowice” presents Table 4 (as previously, time is expressed in milliseconds).

The results show, that the NAVN algorithm can find good (although not optimal) solutions for large real map. This is important information considering the fact that the original algorithms AVN and CAVN do not give any solutions for such large data. Another important observation is that the execution time of the NAVN algorithm starts to be comparable to Dijkstra’s algorithm. During some experiments this time was even shorter. Observing the trend of the execution time of both algorithms, we hope that with further increasing the complexity of the maps, the NAVN algorithm may be as fast

as Dijkstra's, of course, assuming a certain acceptable inaccuracy of solutions found. Especially interesting seems to be the result of the NAVN algorithm for preferences of width. The solution is regarded by many people (driving between the mentioned points of Katowice) as the best route, based mainly on the expressways (one of the authors uses the same one). The Dijkstra's algorithm, although it calculates the cost in exactly the same way, suggests a different route.

Table 4. Results of experiments with data of "Katowice".

	Distance			Width		
	Time	Cost	Distance	Time	Cost	Distance
NAVN	9531	38705	15466	11984	298984	19180
Dijkstra	2515	27144	13495	2531	212459	14317

8. CONCLUSIONS

In the work we have presented three versions of AVN algorithm finding the best (or nearly best) direction between desired origin and destination points on the map, based on ant algorithms. We introduced a really new version of this algorithm (NAVN). We showed its usefulness during the tests on real data, where the AVN algorithm did not give good results. We observed that the execution time of the NAVN algorithm for bigger problems appears to be comparable to that of Dijkstra's algorithm. We plan parallelizing the NAVN algorithm to obtain both better results and shorter execution time.

The proposed algorithm produces the set of solutions, thus the best one, taking into consideration the dynamic situation in the traffic, may be chosen at the moment. It is especially important in the planning of the ambulances' route. To date the deterministic algorithms (e.g. Dijkstra's) were used to solve this problem, but they give only one solution, optimal, but not always acceptable in the specific situation.

BIBLIOGRAPHY

[1] CHAO–LIN L., Best–Path Planning for Public Transportation Systems. Proceedings of the Fifth International IEEE Conference on Intelligent Transportation Systems, Singapore, 2002, pp. 834–839.

[2] CORDON O., HERRERA F., STŰTZLE T., A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends. Mathware & Soft Computing, Vol. 9, 2002, pp. 1–36.

[3] DELAVAR M.R., SAMADZADEGAN F., PAHLAVANI P., A GIS–Assisted Optimal Urban Route Finding Approach Based on Genetic Algorithms, International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences, Vol. 35, Part 2, 2004, pp. 305–308.

[4] DEREKENARIS G., GAROFALAKIS J., MAKRIS C., PRENTZAS J., SIOUTAS S. and TSAKALIDIS A., Integrating GIS, GPS and GSM technologies for the effective management of ambulances, Computers, Environment and Urban Systems, Vol. 25, 2001, pp. 267–278.

[5] DORIGO M., GAMBARDELLA L.M., Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation, Vol. 1, 1997, pp. 53–66.

[6] SALEHINEJAD H., FARRAHI–MOGHADDAM F., An Ant Based Algorithm Approach to Vehicle Navigation, First Joint Congress on Fuzzy and Intelligent Systems, Ferdowsi University of Mashhad, Iran, 2007.

[7] SALEHINEJAD H., POULADI F., TALEBI S., A New Route Selection System: Multiparameter Ant Algorithm Based Vehicle Navigation Approach, CIMCA 2008, IAWTIC 2008, and ISE 2008.

[8] SALEHINEJAD H., TALEBI S., A New Ant Algorithm Based Vehicle Navigation System: A Wireless Networking Approach, Int. Symp. on Telecomm, 2008.