Scalable Computing: Practice and Experience
Volume 12, Number 3, pp. 283–291. http://www.scpe.org
ISSN 1895-1767

VEHICLE ROUTING PROBLEMS WITH THE USE OF MULTI-AGENT SYSTEM

LUKASZ CHOMATEK* AND ANETA PONISZEWSKA-MARANDA†

Abstract. Increasing number of vehicles on the roads caused the increase of popularity of GPS devices that the drivers can install in their cars. Efficient vehicle routing is very significant task nowadays, as the number of vehicles on the roads is growing rapidly. As many drivers have an ability use a computer while planning their itinerary, they need to have an application to find the best route for them.

The paper describes a new approach for path finding problem. The proposition of solving the path finding problem with the use of multi-agent system is proposed. The idea of multi-agent system includes cooperation between autonomous software agents to complete a certain task.

Key words: vehicle routing, path finding problem, single source shortest path problem, multi-agent systems

AMS subject classifications. 68N02, 68T02

1. Introduction. Increasing number of vehicles on the roads caused the increase of popularity of GPS devices that the drivers can install in their cars. Internet sites, where it is possible to compute efficient route from one point to another are also popular, because they are not only maps but also can fulfill some needs of the user. The user's needs can be divided into some groups:

- saving of time - user of the system only needs to know the destination address and in some cases he needs to enter the start point as well because the path is computed automatically by the system,
- finding some information - system can show the way to the nearest restaurant, gas station or shop,
- informing about the situation on the road, e.g. traffic jams, speed cameras, road works, accidents.

Both mentioned websites and GPS devices have to execute large numbers of queries about the route between points on the map. The website is a service dedicated for the large number of users and GPS device has to reflect dynamically changing road conditions (i.e. driver was supposed to turn left on the crossroads but went straight and now the system must compute a detour). Large number of queries can only be handled when either users' requests can be processed in a longer time or by use of very efficient path finding algorithms.

The most efficient algorithms for solving Single Source Shortest Path (SSSP) are hierarchical approaches [1]. They are usually based on the fact that some road segments can be marked as having higher importance than others. What is more, road network can be preprocessed by removing some nodes and introduce some shortcuts instead (i.e. there is only one connection from one node to another, so all nodes between them can be substituted by a direct link for this nodes).

The paper describes the proposition of solving the path finding problem with the use of multi-agent system. The idea of multi-agent system includes cooperation between autonomous software agents to complete a certain task [9, 10]. For solution of SSSP problem based on road network hierarchy,the agents can be divided into some groups: graph constructing agents, agents interacting with the system user and miscellaneous agents. The second significant term in the domain of multi-agent systems is the environment, in which the agents are located [11]. In the case of road traffic it is very well defined and contains hardly any subjective factors. It includes vehicles, roads, road signs and signals and some important places which are usually named points of interest (POI). Road environment is obviously dynamic, due to the fact that hardly any part of it remains unchanged for a long time.

The paper is structured as follows: section 2 presents the approaches used to solve the problem of hierarchical single source shortest path. Section 3 describes the proposition of algorithm for road network division while section 4 deals with multi-agents system for road network hierarchization problem. Section 5 describes the results obtained during the use of system application.

2. Approaches of hierarchical single source shortest path. Typical algorithms designed for solving SSSP problems do not use any preprocessing of the graph. Preprocessing phase can take a long time, so that such algorithms can be easily applied, when there is a little number of queries about the shortest path. Most popular SSSP solving algorithms are Dijkstras algorithm, Bellman-Ford algorithm and A* algorithm.

*Institute of Information Technology, Technical University of Lodz, Poland (lukasz.chomatek@p.lodz.pl)
†Institute of Information Technology, Technical University of Lodz, Poland (anetap@ics.p.lodz.p)

Table 2.1: N_3^0 for all vertices of the sample graph

v	$N_3^0(v)$
0	$\{0, 2, 3\}$
1,2	$\{0, 1, 2\}$
3	$\{0, 3, 4\}$
4,5	$\{2, 4, 5\}$

Hierarchical algorithms include some kind of preprocessing of the graph in order to shorten the time required to process a single query. It is notably important when number of queries is very high and sometime can be expended before deployment of the system.

The algorithm of Hierarchical Path Views proposed in the literature [4, 5] was based on the following ideas:

- base road network was split into some fragments - places of split were chosen by its geographical coordinates,
- connections which are outside generated fragments belong to higher hierarchy level,
- division and level transfer is an iterative process.

The result of such a division are the matrices containing the shortest path lengths for each segment and each level. After the division phase, to perform a query, A* algorithm was used.

Base for other branch of hierarchical algorithms for solving SSSP problem was Highway Hierarchies algorithm proposed in [1, 2]. Dijkstras algorithm is used in the preprocessing phase to calculate the neighborhood for each vertex. Next, the vertices that fulfill some criteria are moved to the higher hierarchy level. When this phase is done, the higher hierarchy level is preprocessed that allows to generate shortcuts between certain vertices. Number of hierarchy levels and size of the neighborhood are parameters of the algorithm. Proper choose of them influences on the amount of time needed to process a single query.

2.1. Highway Hierarchies Algorithm. Highway Hierarchies algorithm requires two parameters: H that identifies the degree to which the requests for the shortest way are met without coming to a higher level in the hierarchy, and L, which represents the maximum permissible hierarchy level. The method used to iteratively generate a higher level with number $l + 1$ for a graph G^l is as follows:

1. For each vertex $v \in V$, build the neighborhood N_H^l for all vertices reached from v by using Dijkstras algorithm in graph G^l, respecting the H constraint. Set the state of the vertex V to "active".
2. For each vertex:
   - Build the partial tree $B(v)$ and assign to each vertex its state. The state of the vertex is inherited from the parent vertex every time when the vertex is reached or settled. Vertex becomes "passive" if on the shortest path $< v, u, ..., w >$, where $v \neq u \neq w$:

     $|N_H^l(u) \wedge N_H^l(w)| \leq 1$

     Partial tree is completed, when reached but unsettled vertices don't exist.
   - For each vertex t, which is a leaf node in the tree $B(v)$ move each edge (u, w), where $u(N_{\downarrow}H^{\uparrow}l(t), w(N_{\downarrow}H^{\uparrow}l(v)$ to the higher hierarchy level.

During the first stage, a highway hierarchy is constructed, where each hierarchy level G^l, for $l < L$, is a modified subgraph of the previous level graph G_{l-1}. Therefore, no canonical shortest path in G_{l-1} lies entirely outside the current level for all sufficiently distant path endpoints. This ensures that all queries between far endpoints on level $l - 1$ are mostly carried out on level l, which is smaller, thus speeding up the search.

2.2. Example of Highway Hierarchies Algorithm use. Let consider how the algorithm works for a simple graph. Let $L = 1$ and $H = 3$. First, N_3^0 has to be calculated for each vertex $v \in V$ using Dijkstras algorithm. The results are shown in table 2.1.

The construction of $B(v)$ for the example of vertex v_0 is shown above. This process is similar for other vertices:

1. Initial state of obtained v_0 is "active".

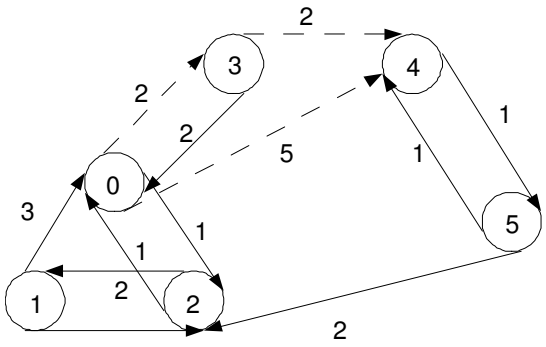

Fig. 2.1: Example of road network graph

2. Dijkstra's algorithm:
   (a) Vertex v_0 is settled, the path is empty.
   (b) Vertices v_2 and v_3 are reached from v_0 with, respectively 1 and 2. Their state is set to "active" (inherited from v_0).
   (c) Vertex v_2 with cost 1 is settled on the path $< v_0, v_2 >$. Passivity condition is not satisfied, because there are too few nodes on the path.
   (d) Vertex v_1 is reached from v_2 (cost 3) and its state is set to "active".
   (e) Vertex v_3 is settled with cost 2 on the path $< v_0, v_2 >$.
   (f) Vertex v_4 is reached from v_3 (cost 4).
   (g) Vertex v_1 is settled on the path $< v_0, v_2, v_1 >$ with cost 3. As $N_3^0(v_2) \wedge N_3^0(v_1) = \{v_0, v_2\}$, vertex v_1 stays "active".
   (h) Vertex v_4 becomes settled with cost 4 on the path $< v_0, v_3, v_4 >$. As $N_3^0(v_3) \wedge N_3^0(v_4) = \{v_3\}$, its state is set to "passive".
   (i) Vertex v_5 is reached from v_4 with cost 5 and its state is set to "passive" (inherited from v_4).
   (j) While there are no reached and active vertices, the algorithm terminates.
3. Leaf vertices are v_1 and v_5.
   (a) For vertex v_1 we iterate back on the path $< v_0, v_2, v_1 >$. For pair (v_1, v_2): $v_1 \in N_3^0(v_2)$ and $v_2 \in N_3^0(v_1)$. Therefore, that edge stays on level 0. The edge (v_2, v_0) also stays on level 0.
   (b) We perform the backward iteration process on the path $< v_0, v_3, v_4, v_5 >$. For example $v_\downarrow 3(N_\downarrow H^\uparrow l(v_\downarrow 5)$ and $v_\downarrow 4(N_\downarrow H^\uparrow l(v_\downarrow 0)$, so the $< v_3, v_4 >$ is moved to level 1.

The result of Highway Hierarchies algorithm is shown on figure 2.1. Dashed lines represent the edges on the level 1 and continuous lines represent edges on level 0.

3. Proposed road network division algorithm. Some parts of the construction phase of Highway Hierarchies algorithm can be performed concurrently:

- weight assignment for each road segment, in general using different rules,
- construction of N_h^l neighborhoods for each vertex in graph,
- construction of $B(v)$ trees.

We decided to try performing division of road network graph, so that Highway Hierarchies algorithm can be performed on a single part of this graph. After completion of the algorithm on each part, all subgraphs should be merged to obtain a final Highway Hierarchies graph.

To perform the devision of a graph, Breadth First Search (BFS) algorithm was applied for certain vertices as follows:

1. Get a list BFS_{start} of vertices mentioned to be start points for BFS.
2. For each vertex $v \in BFS_{start}$ create empty lists E_v and V_v to store the information about edges and vertices that belong to the subgraph.
3. For each vertex $v \in BFS_{start}$:
   (a) For the vertices from BFS queue, check if their children are allocated in any subgraph. If not, add them to BFS queue for current vertex and to V_v. Add corresponding edges to E_v.

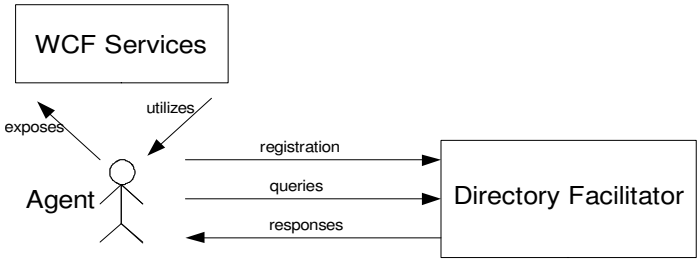

Fig. 4.1: System dependencies from the agent point of view

4. Check if all vertices of the base graph are not allocated in on of the subgraphs. If not, go to step 3.
5. Perform representation dependent postprocessing for each set V_v (i.e. reorder vertices if needed).

Such an algorithm can be applied to connected graph, if there are more than one connected components in the road network graph. Described division can be performed for each connected component treated as a base graph.

In the presented algorithm some edges can be not included in any graph. They can be denoted as E'. After performing construction phase of Highway Hierarchies algorithm on each subgraph, these edges must be included in the result graph. It is a two-step process:

1. For each vertex $v \in BFS_{start}$ add all of the vertices from V_v and all edges from E_v to the final graph.
2. For each edge $e \in E'$ that connects vertices V_s and V_d get the highest hierarchy level from all incoming edges of vertex V_s.

4. Multi-agent systems and its application for road network hierarchization problem. The standards of architecture for multi-agent systems were described by FIPA organization [7]. Due to this specification, multi-agent system consists of some number of Agent Platforms that were as a parts of the system and they can be used to host the agents. Each Agent Platform consists of three parts to handle the management of agents:

- Message Transport System (MTS) that is supposed to be used by the agents for communication,
- Agent Management System (AMS) that represents a catalog of existing agents,
- Directory Facilitator (DF) that stores the information about the services provided by the agents.

The analysis of modern programming techniques shows that some practices can be applied in newly designed multi-agents systems:

- use Service Oriented Architecture (SOA) to simplify and improve the possibilities of agent communication,
- make Directory Facilitator the mandatory part of multi-agent system,
- try to apply the enterprise design patterns such as dependency injection to coordinate the communication of agents on a single machine,
- simplify the architecture using the Windows Communication Foundation (WCF) [6, 7].

The introduction of web services allowed the developers to connect the applications based on different software and hardware platforms, for example Java and .NET Framework. The Web Services use a specific protocol to expose a schema of transferred data and allow the clients to make the synchronous calls of exposed methods [3].

Some generalization of such a system is described in [3]. In this approach the extended Directory Facilitator component plays the major role in the system because it keep all the information about services offered by the agents. All the services offered by agents are Web Services or some of their extensions such as WCF services (Fig. 4.1) [6].

4.1. Application of agents from building the road network hierarchies graph. Application of multi-agent system for building Highway Hierarchies graph was proposed in [8]. Two main assumptions were made for proposed application of multi-agent system for building Highway Hierarchies graph:

- system must be able to take into account the user's preferences (i.e. route should be the shortest, traveling time should be lowest) and environmental conditions (i.e. weather, time of a day),

- computations should be done concurrently, where it is possible to be done.

To complete the first of these assumptions, weights of the road segments must be assigned using different criteria, such as length, average traveling time, speed limits, etc. It was decided to introduce some number of reactive agents that collect the data from different road segments. This type of agents can work in two different ways, depending on the data structure which is used to store the road network technology. The first way is associated with the nodes as it is easy to get information about edges connected to the node. Second way is related to edges. If list of edges in the graph is directly provided, it can be divided into some parts and each part can be analyzed by a single agent.

However graph are usually represented in a hierarchical way, where nodes are on the top level and data for edges is usually kept as a list for each node. The complete list of edges is helpful for weight assignment criteria based only on some properties of a single edge (i.e. length, speed limit). On the other hand, some important information can be kept in nodes one can consider criterion of avoiding bigger crossroads so that all of the road segments connected to such node should have its weight properly adjusted.

In our system both graph nodes and edges are kept in the separate lists. However references are duplicated and it simplifies the way of access to the needed data and allows the simple and complex weight assignment rules.

Regardless of the chosen solution, this process can be performed in parallel, what means sharing work for several agents. Depending on the selection criteria by which individual weights are calculated, work on each road section may perform one or more agents (each can calculate the weight using different method). If the weight of the segment is calculated on the basis of several criteria, use of a coordinating agent for the weights assignment process can be considered. The coordinating agent can calculate weight in accordance with certain rules (e.g. use the weighted average of the values calculated by the agents). Coordinating agent may have some adaptive abilities, depending on the application of the system [8].

Concurrent computation can be also applied in the other parts of Highway Hierarchies graphs creation process. Obviously, calculation of neighborhood N_H^l for each vertex is independent of each other. The only nuisance is that for each vertex, different queue of vertices intended to be visited must be kept. Any number of agents can be used to calculate such a neighborhood. Depending on the developer choice, these agents cooperate directly with agents responsible for assigning weights to graph edges or with the coordinating agent.

The creation of trees $B(v)$ is another process that can be done in parallel by agents for the individual vertices of the graph. This process should to be implemented through the cooperation with agents that build the neighborhoods.

The responses to user's queries for the system should take into account his preferences regarding the itinerary and the current conditions on the road. It might be necessary to create several Highway Hierarchies graphs, which will be used to obtain a system response depending on certain factors. Different graphs can be prepared for example for the city center during peak hours and at night. To implement this assumption, the introduction of a special type of agent can be considered. Such an agent will redirect the user query to the appropriate Highway Hierarchies graph. Relay agent may assist in work of coordinating agent by suggesting the criteria by which the weight of the edge should be calculated [8].

Proposed architecture of multi-agent system described above is shown on figure 4.2. The tests revealed that for diverse criteria the calculated hierarchies differ very much. The results obtained for three proposed criteria: speed limits, traveling time and road length, shown that these hierarchies graphs have at each level only a few common edges with other hierarchies graphs. Moreover, expected convergence between dominating user's preference and number of common edges with the hierarchies graph for this criterion was observed.

4.2. Fastening Highway Hierarchies graph construction process. Application of multi-agent system described above shown that such architecture can be successfully used both for handling user's preferences and speeding up construction process of Highway Hierarchies graph. In order to apply improvements described in section 3, an architecture of multi-agent system must be significantly changed.

There are two new types of agents required to perform such process:

- *Graph splitting agents*, which are supposed to prepare proper split of the graph and pass the information to corresponding neighborhood calculators. Graph splitting agents can be considered as social agents as they have to cooperate with other graph splitting agents while doing their work, because some edges can be allocated in different subgraphs.
- *Graph merging agent*, which task is to merge subgraphs prepared by splitting agents according to certain

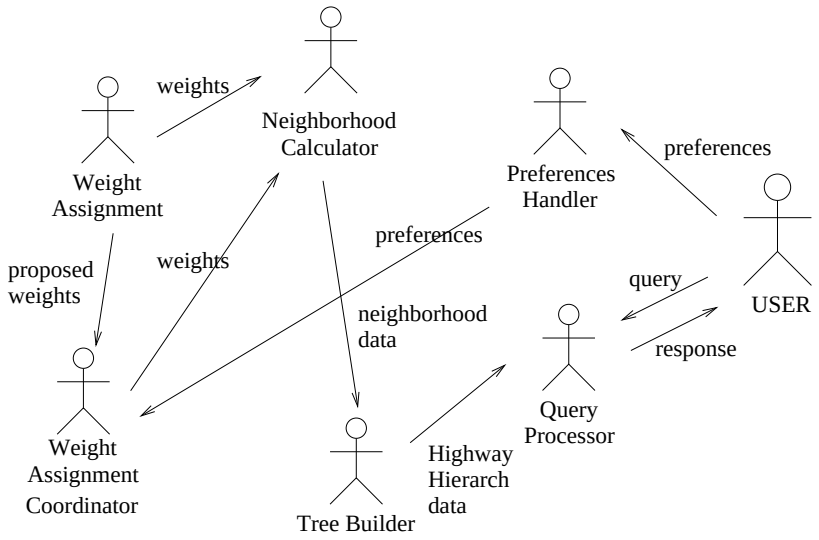

Fig. 4.2: Types of agents in Highway Hierarchies algorithm

rules. Although the work of this agent looks complicated, it is reactive agent it has to wait for the graph splitting agents to complete their work, so that he can perform the merging process.

Introduction of new types of agents to the system implies changes in the data flows in the system. *Preferences Handler agent* still cooperates with *Weight Assignment Coordinator agent* in order to pass the information about user's needs. *Graph Splitting agents* can divide graph into subgraphs and then cooperate with *Neighborhood Calculator agents* and *Tree Builder agents* to prepare partial Highway Hierarchies graphs. Next, *Graph Merging agent* can prepare final division of the road network into hierarchy levels. *Query processor agent* cooperates directly only with *Graph Merging agent*. Dependencies between other types of agents remain unchanged.

5. Results of system application. Algorithm described above was implemented using C# 4.0 language in Windows environment. Tests were run on different maps both for single and split road network graph. Exemplary result of full graph division is shown on figure 5.1. Figure 5.2 presents the results of algorithm performed on a subgraph.

In general, road segments on the top hierarchy levels in the full graph are on high hierarchy level in a part of the graph too, overall number on the highest hierarchy level is smaller for the subgraphs. It is caused by the fact that when the number of edges is smaller, promotion to the higher hierarchy level is harder.

The tests performed for both maps shown that the time needed to execute the calculations depends in slight degree on the number of edges in the subgraph. In case of real road map, both subgraphs contain after a division the exact number of vertices. However, the first part contains significantly more edges. The time needed to compute the hierarchy levels for both parts were almost identical. The second graph was an artificial road mesh, where after a division the number of edges was the same in the both parts. Performed tests shown that the time of computing for both subgraphs was also the same in this case.

Highway Hierarchies algorithm was run for the whole artificial and real road graph using such parameters:

- maximum hierarchy level: 3,
- Dijkstras neighborhood size: 5.

In the next step, Highway Hierarchies was run for the subgraphs with identical parameters. Obtained results for subgraphs show that when a division is made, number of edges on each level differs from number of these edges when the whole graph is taken into account in Highway Hierarchies. For real road network, such a difference was up to 70%. It was caused by the fact that large number of edges was included in the first subgraph. This difference for the artificial road network was smaller then in the real network.

It is harder to reach higher hierarchy level in smaller graphs (larger graphs usually contain longer paths). Therefore, we decided to decreases the size of Dijkstra neighborhood used in the division into the hierarchies. This was supposed to facilitate the promotion to higher hierarchy levels (step 2*b* of Highway Hierarchies al-

Fig. 5.1: Example of Hierarchical Division for neighborhood of Technical University of Lodz HH(3,5)

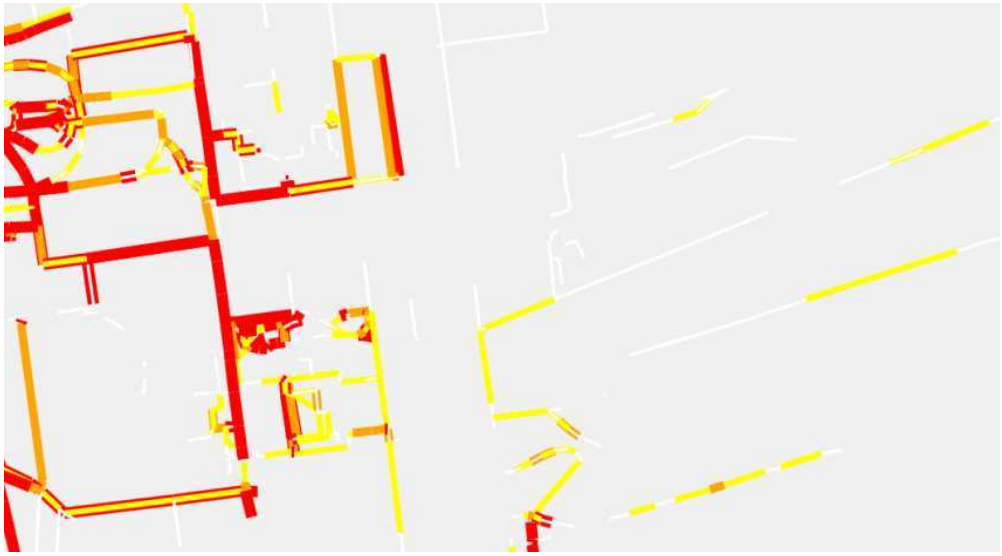

Fig. 5.2: HH algorithm performed for a subgraph HH(3,5)

gorithm). In the case of real road network, number of edges on each hierarchy level was closer to one in the reference division (about 37% better result). In the case of artificial road graph, the difference in number of edges on each level between result obtained for Highway Hierarchies computed for whole graph and for two subgraphs was about 7%. Generally, decreasing size of Dijkstra neighborhood resulted in lower differences between reference hierarchical division and a division made for subgraphs.

The second test was performed to check, whether HH algorithm for spitted graph is faster than the algorithm ran for the whole graph. Results are gathered in the table 5.2. Values are given in percents that represent the amount of time needed to perform each phase of the algorithm in addition to time needed to construct Highway Hierarchies for the whole graph. When the algorithm is supposed to build the hierarchies with greater number of levels, the gain is the highest. When number of maximum level is set to 1, the gain is not so high.

Table 5.1: Number of edges on each level for different HH parameters and maps

Road Network	Level	HH(3,5), Whole graph	HH (3,3), 2 subgraphs	% diff	HH (3,5), 2 subgraphs	% diff
Technical Univ. of Lodz	0	707	750	6,08203678	922	30,41018
	1	451	343	23,9467849	274	39,24612
	2	136	131	3,67647059	93	31,61765
	3	613	234	61,8270799	177	71,12561
Artificial mesh	0	242	215	11,1570248	251	3,719008
	1	264	239	9,46969697	261	1,136364
	2	184	170	7,60869565	184	0
	3	440	452	2,72727273	380	13,63636

Table 5.2: Time amount needed to complete each phase of the algorithm for split road graph in addition to time needed for HH construction for whole graph

HH parameters	First subgraph	Second subgraph	Merge	Split
Technical University of Lodz				
(3,5)	10,7%	11,4%	3,5%	2,1%
(3,3)	7,3%	7,5%	5,0%	3,0%
(1,3)	11,8%	11,9%	25,4%	15,2%
Artificial mesh				
(3,3)	18,0%	17,6%	18,5%	18,9%
(3,5)	19,6%	19,4%	9,5%	9,7%

6. Conclusions. The presented paper focuses on the problems of efficient vehicle routing. Nowadays these problems are very important because of increasing number of vehicles on the roads. More and more drivers have the abilities of use the devices to support the planning of their itinerary and it is important to find the new solutions, new algorithms to improve the applications for finding the best routs taking into consideration different criteria, static and dynamic. The use of agent concept and use of multi-agent approach seems to be the interesting solution for solving the problems with vehicle routing and finding the optimal itinerary.

Performing the hierarchical division of road network on the split of the road graph can improve the construction phase processing time due to the lower computational complexity. Number of edges on certain levels in subgraphs can differ very much from these from division of whole graph. Adjusting hierarchical algorithm parameters can improve results of divisions of subgraphs.

Multi-agent system can be utilized to solve this problem, what allows to compute most parts of the algorithm in parallel. Architecture of presented multi-agent system is extensible. There is a possibility to implement new types of agents for different graph split methods.

REFERENCES

[1] P. Sanders and D. Schulters, *Engineering highway hierarchies*, LNCS 4168, pages 804-816, 2006.
[2] P. Sanders and D. Schulters, *Highway hierarchies hasten exact shortest path queries*, LNCS 3669, pages 568-579, 2005.
[3] L. Chomatek and A. Poniszewska-Maranda, *Modern Approach for building of Multi-Agent Systems*, LNCS 5722, pages 351-360, 2009.
[4] G. Nannicini, P. Baptiste, G. Barbier, D. Krob and L. Liberti, *Fast paths in large-scale dynamic road networks*, Computational Optimization and Applications , 45 (1), pages 143-158, 2008.
[5] D. Eppstein, M. Goodrich and L. Trott, *Going off-road: transversal complexity in road networks*, Proceedings of 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pages 23-32, 2009.
[6] C. Vasters, *Introduction to Building WCF Services*, MSDN Library, 2005.
[7] FIPA, *Abstract Architecture Specification*, 2002.
[8] L. Chomatek, *Multi-agent approach for building highway hierarchies graph*, Proceedings of 31th International Conference Information Systems, Architecture and Technology, Szklarska Poreba, Poland, September 2010.
[9] M. Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons, 2002.
[10] G. Weiss, *Multi-Agent Systems*, The MIT Press, 1999.
[11] M. Singh and M. Huhns, *Readings in Agents*, Morgan-Kaufmann Pub., 1997.

Edited by: Dana Petcu and Jose Luis Vazquez-Poletti
Received: August 1, 2011
Accepted: August 31, 2011