

# Using Meta-agents for Multi-agents in Networks

**Anne Håkansson**

Department of Information Science, Computer  
Science  
Uppsala University  
Uppsala, Sweden

**Ronald L Hartung**

Department of Computer Science  
Franklin University  
Columbus, Ohio, USA

**Abstract** *In this paper, we propose an approach using meta-agents for monitoring multi-agents and controlling their behaviour while moving between states in a network. The meta-agents are built on the agents in the system and then used to inspect the agents' behaviour when reaching a result as well as perceiving the reason for that result. Each meta-agent can comprise everything from one agent to dozen or more agents depending on the task assigned to the system. The benefit of using meta-agents for a multi-agent system is the ability to provide the fastest way between nodes under given circumstances and to handle a vast number of nodes in graphs and networks such as travelling salesman problem (TSP).*

**Keywords:** Intelligent Agents, Multi-agent Systems, Meta-agents, Graphs, Networks

## 1 Introduction

Multi-agent systems use intelligent agents to perform tasks. A multi-agent system is a program with independent problem solvers i.e., agents, situated in interactive environments [6]. An agent is defined in terms of its behaviour and a definition is social ability, where agents interact with other agents via a communication language for coordination [13; 3]. Additionally, agents are defined as situated, where agents receive inputs from the environment and can effect changes. In the environment, the agents are autonomous and flexible [6]. Autonomous is defined as an agent interacting with the environment without any intervention from other agents. Flexibility is defined as agents being responsive and adaptive to the current situation.

Intelligent agents are associated with the ability to adapt to the environment and learn from it [10; 4]. Adaptively is achieved through the choice of alternative problem-solving rules or algorithms. Learning proceeds from the capability of introspection and analysis of successful behaviour.

The multi-agent system performs tasks assigned in different kinds of environments. The agent performs in a task environment containing static, dynamic or semi-dynamic characteristics [10]. In a static environment, the agent does not continuously need to check the environment when deciding an action. However, when the environment continuously changes, as in dynamic environments, the agent reconfigures its behaviour, accordingly, and changes its moves according to the new environment. However, the environment in networks can be of semi-dynamic nature

and include characteristics of both. Some environments include constraints, static hindrances or properties of the environment, or obstacles, dynamic problems or conditions that have short-term characteristics [5]. The agents react to the environment by knowing the static properties and by continuously checking the dynamic conditions.

The cooperation between agents is important in multi-agent systems. In these systems [14], one single agent does not have all the data required to achieve a task and must cooperate with other agents to jointly perform the given task. This cooperation needs both coordination to work properly, and cooperation strategies are used as means to interact with other agents. There are attempts where a single learner is applied to discover joint solutions to multi-agent problems, so called team learning, or using multiple simultaneous learners, often one per agent, so called concurrent learning [8].

Teamwork coordinates cooperative agents in dynamic environments and supports a group of agents that can achieve goals, flexibly and robustly, in a distributed and dynamic environment [15]. Thus, the interaction makes the agents collectively achieve their goals [1]. To achieve goals, each individual needs to reason with its environment and about the behaviour of other agents. The coordination mechanism handles the interaction between agents and can be used by meta-agents as well. A coordinating mechanism is the use of meta-level reasoning [1], which we implement in the meta-agents.

Meta-reasoning is a technique that supports the system's ability to reason about its own operation. In a meta-agent system, meta-reasoning implies implementing the agents' capability to reason about other agents. This reasoning can be used in multi-agent systems to reconstruct the agents' behaviour [9] but also to help in the interaction among agents and in the implementation strategies or plans for responding to requests [2].

Meta-agents can support the intelligent agents by observing their environment and prescribe scheduling actions. The meta-agents are used to yield predictions about other agents' decisions where meta-agents can use active learning techniques to improve their models [12]. Meta-agents can perform reasoning, plan actions, maintain individual agents' state information and attempt to control future behaviour. The meta-agents can also classify conflicts and resolve them [1]. From the intelligent agents' observation of the environment, the meta-agents assist the meta-agents evaluating the alternatives and scheduling actions.

Commonly, the meta-agents have been used for machine learning and robotics, but we apply them on networks and

use them to solve the problems with a large amount of data. The meta-agents are built on the agents in the multi-agent systems. The goal is to use the meta-agents to provide the fastest way between nodes in a network considering the circumstance given at the time, where the circumstances include static and dynamic characteristics in the environment. This requires a plan or an algorithm to find the goal and in a network it is generally a route-finding problem. This is closely related to touring problems, such as travelling salesperson problem [10], and we apply our approach using meta-agents for monitoring multi-agents for TSP.

The meta-agents can reason about the multi-agents in the system and find an optimal route, i.e., the best or safest, way for a route, which is less of a distraction for the task environment of the intelligent agents. The system, presented in this paper, works with multi-agents in a network to solve the problems with a large number of nodes and calculate the fastest way between two nodes using alternative paths. The meta-agents are applied to the multi-agents to speed up the computing but also strengthen the reasoning with the agents. The meta-agents can provide all the static and dynamic information found for the intelligent agents.

## 2 The Intelligent agents in the system

In the multi-agent system, the intelligent agents work in an uncomplicated task environment, in which the environment is fully observable, deterministic, episodic, discrete and semi-dynamic. The agent has access to the complete state of the environment and can detect all the aspects that are relevant to the action. In the system, the intelligent agents have full access to the environment and do not need to change internal state to keep track of the external world.

The agents do not need to handle uncertainty, which the agents in the current system ignore. In a deterministic and fully observable environment, the next step is completely determined by current state. The agents' action is deterministic in that sense that the agents do not react to unexpected events. Instead, the agents have constant actions.

The intelligent agents work in an episodic task environment where the task is divided into atomic episodes. In the episode, each agent performs a single action. The choice of action only depends on the episode itself but the result of the action performed by the agent is used as the total cost for travelling.

Discrete describes the state of the environment, i.e., how time is handled, as well as, perception and actions of the agents. Each agent has a finite number of states and actions it can take. Moreover, the agent is aware of the travel time it takes to fulfil the task of moving along the arc between two nodes. The time differs depending on the circumstances in the environment at the given time, which can change often since it depends on the dynamic characteristics in the environment.

The agents' task environment is semi-dynamic in the sense that the agents' environment, the graph structure itself, does not change but the agents' performance score changes as some properties in the environment are changing

over time. The static environment contains the constraints or hindrances, which are the properties inherent in the network. They are static in the sense that they seldom change because of the nature of the environment and because of the impact it requires to change these. The dynamic environment contains the obstacles, which are temporary conditions that change frequently [5]. The intelligent agents in the multi-agents system have knowledge about the static and the dynamic characteristics in the system, with following structure:

$$\forall Ag(x) \exists x(Task(x) \wedge Static(x) \wedge Dynamic(x))$$

Intelligent Agents (Ag) denote all the agents in the system where each agent has a specific task to perform (Task), while consider the data about the environment, i.e., (Static) and (Dynamic).

Each agent has one task assigned. The task is to move between two nodes and is the only action available to the agent. The Task(x) is structured as:

$$agent(Node A, Road number, Arc Id, Node B)$$

The agent goes from one node to another (Node A) and (Node B) depending on the task assigned to the system and relation  $\mathfrak{R}$  is symmetric. The  $\mathfrak{R}(\text{State A}, \text{State B})$  is equal to  $\mathfrak{R}(\text{State B}, \text{State A})$ . Thus, the agent will have the same result of the task regardless of the travelling direction. The agent works with the same information irrespectively of the differences in the environment. This is a simplifying assumption used in the current system; it can be removed, representing the situation where travel in opposite directions can vary. Hence, the agents have data about the states they work between, where the nodes are either the initial node or the goal node. Moreover, a number, corresponding to the road number, is also given as an information piece. Using road number informs that the agents use the same road while moving between several different states. Since the users decide the ultimate use of result of the agents, the information can be interesting for them. These agents also have an identification number to distinguish them, which connect them to static and dynamic information.

The basic data, i.e., the nodes, number and identification for the agents can never change as long as this agent is working. Static information, on the other hand, can change but does so infrequently and does not need much attention. For the dynamic information, however, changes occur frequently about every hour to one day. Each agent constantly has access to environment according to following:

$$\forall Ag(x) \exists x(Static(x)[D * S * T * L] \wedge Dynamic(x)[W * Q * R * C])$$

For the agents, the static environment include Distance (D), Speed (S), Typography (T) and Lanes (L) and the dynamic environment consists of weather (W), road quality (Q), vision range (R) and road constructions (C).

The static environment defines the distance between the nodes and the average speed for the distance. It is also the

defines topography in the environment ranging from flat to very strong uphill and very strong down hill and using about eight different characteristics. The topography, except for flat and easy downhill, affects the speed of the agent. The static environment includes the number of lanes for the road where one lane affects the agent's speed.

The dynamic environment includes the weather, the road quality, the field of vision and whether there is any road construction. The weather, road quality and vision range spans from good to very bad with five different categories where good does not influence the speed and the very bad has the greatest impact. Road construction affects the speed regardless of number of lanes.

In the system, the each agent keeps track of the time it takes to perform its task, i.e., travelling from one node to another. The agents start by checking its static environment and calculate the path costs, i.e., the time it takes to move. Then, the agents check the dynamic characteristics and take those into account while moving. These characteristics slow the agents down and do not improve the travel time. The number and kind of problems affects the performance and can slow down the travel time considerably. Taken all together, the affects of the static and dynamic characteristics that negatively impact agents, represents a speed deduction up to 60 percent.

### 3 Multi-agents

The intelligent agents work in a multi-agent environment. Graphs and networks, like travelling salesman problem, usually are too complex for a single agent to efficiently compute and, hence, multi-agent system can be used to perform the task. Accomplishing a task among several agents is achieved by the interactions between the agents.

Multi-agent systems manifest self-organization and complex behaviours even though the individual strategies of the agents are simple. The agents have a limited viewpoint and do not have all the data available to achieve a task. Therefore, these agents cooperate with other agents to reach goals, collectively, by activating the intelligent agents concerned with performing a particular task [5]. When a task is assigned to the system, all the agents involved for the task prioritize that task.

The cooperation between the agents is a kind of teamwork. The agents do not learn from each other but become a starting point for the others agents. The cooperation occurs when an agent has reached its end node and, thereby, launches the agents that contain that node. There is no global control of the agents, but the system is dependent from the beginning since it needs an initial and end node from the user. Thus, the agents know the start and end points and compute until the connections between these are found. When the agents have accomplished their task, they reach their end node and check whether this node is the ending node for the task. As long as the task is not fulfilled, the agents become an assigning agent for the next agent, which continues to work with next node. Moreover, as long as the cost for executing the agents for a task is lower than the agents that have already accomplished the task, the agents will invoke the connected agents.

The environment will be probed and changes in the environment lead to behaviour changes of the agents. As mentioned above, these change frequently in the dynamic environment but, nonetheless, will change in the static as well. However, the changes in the static environment are slow enough to ignore. All the agents are obligated to take the dynamic changes into account but the static environment can be checked from time to time. In the current system, both static and dynamic environment are checked when an agent is moving. The example used is that of a road network. This example provides an interesting collection of static and dynamic elements. It is likewise interesting as an application area. Routing traffic over the real world highway system is both interesting and useful.

The multi-agents cooperate through communication. Communication can be achieved by holding the data open for others to use. There is not any message passing between the agents in the current system. Instead the data, used by several intelligent agents, is handled by meta-agents. The meta-agents can collect and make available all the data produced by the agents used for an assignment.

The meta-agents need to work on a higher level than the intelligent agents. Hierarchies of agents, e.g., scheduling agents, can be useful for teams of agents [11]. These agents include features, such as, event handling, communication and appending the specific mechanism that the agents use for cooperation. The scheduling agents are used to build teams of cooperating agents on several levels of the scheduling hierarchy [11]. In the multi-agent system the scheduling can support sending data between levels where the schedule consists of information about time, constraints and obstacles. The idea of communication in hierarchies of agents is used in the meta-agents.

### 4 Meta-reasoning and Meta-agents

Meta-reasoning is to use meta-knowledge about the object-level entities, where the object-level consists of agents that knows the goal to obtain [2]. The reasoning can be useful for exploiting properties of relations, such as symmetry, and also for suggesting interactions between the objects and the meta-level.

Meta-agents are used to improve the decision-making required to perform a task in an environment. These agents can observe the intelligent agents (i.e., at object-level) in the environment and prescribe scheduling actions. The meta-agents maintain information on other agents and can classify conflicts. The meta-agents are analysed before the intelligent agents and used to check whether the conditions are viable.

In the multi-agent system, the meta-agent is superior to the intelligent agent. The meta-agent is capable of reasoning about intelligent agents and can have a method to evaluate them. For each set of intelligent agents, that have performed the assigned task and reached the specified goal, a meta-agent is developed. The meta-agents have the information about the initial node and end node but also the total travelling cost. For each task, several meta-agents are developed and these can be used to evaluate the optimal way in the point of view of what is best and what is fastest. What is optimal is, of course in the eye of the beholder,

and in our system it is passable roads with good quality and good weather.

We use meta-agents similar to the use of meta-rules. The meta-rules are meta-knowledge about how problem-specific rules are used (see, e.g., Negnevitsky, [7]) and reflect the body of knowledge gained from these problem-specific rules.

As the meta-rules should be given the highest priority in a knowledge base, we give meta-agents the highest priority in our system. The system looks at the meta-agent and checks whether these have the solution for the given task. If not, the multi-agents will perform the task and the result becomes a meta-agent.

The meta-agents select the fastest way for given circumstances. For each task assignment, commonly, several meta-agents are constructed. The result of the system is to show all the agents together with the path costs. An example of meta-agents for intelligent agents is illustrated in Figure 1.

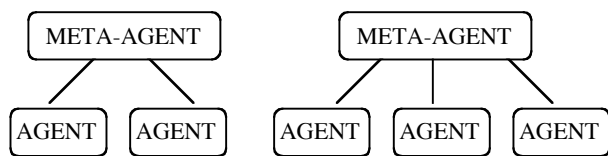


Figure 1. Meta-agent for one or several intelligent agents.

In the system, the meta-agents have the structure:

*meta\_agent(Initial\_state, End\_state, WayList, Costs).*

Each meta-agent is a four-tuple where the route (WayList) includes all the intelligent agents involved in the task of calculating the travelling time between a departure node (Initial state) and a destination node (End state). Moreover, the total cost for the agents in the WayList is calculated and presented (Costs).

As mentioned above, meta-agents are used to get information about and from the agents. Each agent's conditions can be provided and the users can pick the path that seems the most suitable for a given moment. Sometimes the fastest path is not preferable. Instead, the safest path may be more appealing. However, the system can show the weather conditions at the same time as the cost is provided.

The meta-agents are also used to allow the system to apply machine learning. The meta-agents are more effective as a learning structure than the agents. This is because the meta-agents have access to the results produced by the agents and, thus, have more data to react to. In Panait and Luke [8] several learning strategies for agents are discussed, such as, evolutionary approaches, and stochastic methods and reinforcement learning.

For our system, reinforcement is chosen. The meta-agent can compare the results collected from the agents. By collecting data about the agents, the meta-agents can suggest to the agents they should only perform again if the change in the environment is over some threshold. This is a variable threshold, as the changes in other areas of the graph may allow a previously high cost route to become

the preferred route. This approach is especially effective in controlling the re-computation of the solution without re-running the entire calculation from the start. This adapting quality is a major advantage of the meta-agent based solution.

## 5 The algorithm for the agents

The algorithm for the multi-agent system is a mix of depth-first and breadth-first. Each individual meta-agent follows path in a depth-first strategy. However at a branch point, meta-agents are created, providing an overall breadth-first strategy.

The system starts to explore the intelligent agents associated to the start state with breadth-first, see left figure in the Figure 2 (Ag A and Ag B). These agents (A and B) become the first agents for two different meta-agents. Each of these meta-agents explore with a depth-first tactic, see the right part of the Figure 2. If there is one agent invoking several other agents, it is duplicated for each meta-agent, i.e. it will be one meta-agent spawned for Ag B and Ag C and one meta-agent spawned for Ag B and Ag D.

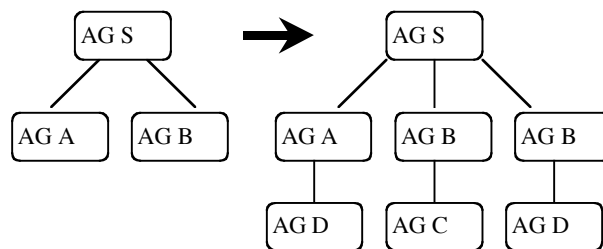


Figure 2. An illustration of the algorithm of the system.

If the end state is not reached by any of the intelligent agents, the system continues with the first explored intelligent agent (left branch in the figure to the right), which is also expanding the meta-agent for the current intelligent agent. Thereafter, the order of the execution of the intelligent agents is determined by the computation system.

The duplication supports building up a unique set of agents while executing the system and, thereby, being able to apply parallel computing for the agents. Thus, order of the execution of the agents is determined by the scheduler of the parallel computing system.

The system can support parallel computing either by assigning agents to a collection of processors in a static assignment or by using a central dispatch queue of ready agents, which are then assigned to processor as available. The trade off is that the first approach may underutilize the processors if the computation is very localized in the graph. On the other hand, the scheduling queue approach is rather inefficient since the individual computations of the agents are rather small. Therefore, the dispatch time may be larger than the computation time. In the current multi-agent system, we simulate the central queue scheduler.

The meta-agents are an interesting twist since they interact with multiple agents and also as they maintain

more state information as the computation progresses. These meta-agents may need to migrate from processor to processor and having them follow the graph makes sense. So the meta-agents in the first case, i.e., assigning agents to a collection of processors, can move from processor to processor by their own choice. In the centre-scheduling model, they are scheduled just like the agents.

The system avoids repeated states. Already explored nodes, or visited states, are marked in the system and the agents moving against an explored node will be removed from further evaluation. For some routing problems, exploring the node and not revisiting it, is a more efficient solution. Nevertheless, the intelligent agent that reaches the node first is also the fastest considering the environment.

Conclusively, a node is able to perform under two conditions. The node has not been visited and is, therefore, unmarked. The other condition is when the dynamic conditions on an arc changes. When this happens the nodes, attached to that arc, are marked executable. Under these rules, the system will recomputed best paths when the conditions change.

The meta-agents collect the results from the agents. Agents have a purely local view whereas meta-agents can capture a global view of the graph. When a node becomes executable, the meta-agents need to become aware of the changes in the results.

## 6 Meta-agents for travelling salesman problem

Meta-agents for intelligent agents in networks can be a feasible solution strategy for the travelling salesman problem. The meta-agents monitor the shortest path between nodes and are applied as a meta-level for solving the TSP. These agents can support checking that each node has been visited, according to the environment, and also get the shortest path between the different nodes. Since the system always checks alternatives, it can be assured that the meta-agents find the fastest way between the nodes.

To solve the TSP, the work is performed in two steps. The first step is generating the meta-agents for the intelligent agents in the system together with the costs and the second is comparing the meta-agents against each other. The latter requires an effective search algorithm, because the number of created meta-agents can increase rapidly. We use the costs determined by intelligent agents to calculate the total costs of the meta-agent.

For each intelligent agent in the system, either a new meta-agent is produced or the existing meta-agent is expanded with the intelligent agent's results. This result is combined with the costs determined by prior agents. When the intelligent agent has reached a goal or stops for another reason, the total cost is calculated.

An example producing meta-agents for intelligent agents in solving the TSP is illustrated in Figure 3 a). In the figure, nodes and arcs are defined, where each arc comprises an intelligent agent.

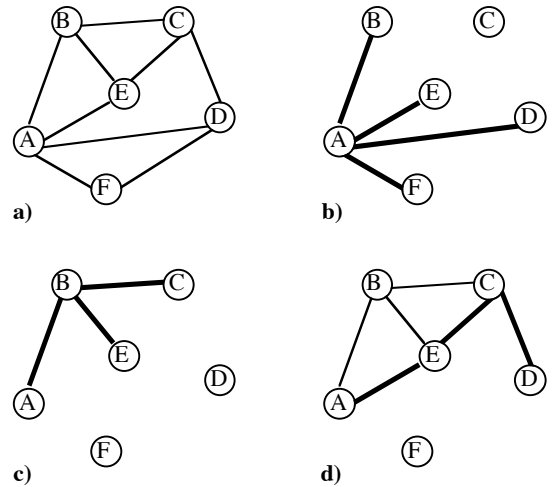


Figure 3. Meta-agents and intelligent agents for TSP.

Lets suppose the following scenario. The user wants to find the optimal solution starting from and ending at A. From the node A, four intelligent agents are operating, one for each arc from A, see b) in Figure 3. Each agent calculates the cost for the task and initiates a meta-agent, one for each intelligent agent with the structure:

- meta-agent [Ag A-B, Cost]* (1)
- meta-agent [Ag A-E, Cost]* (2)
- meta-agent [Ag A-D, Cost]* (3)
- meta-agent [Ag A-F, Cost]* (4)

The meta-agent (1) reaches B node and continues by activating two intelligent agents, B-E and B-C, see c) in the figure. The meta-agent (1) is extended with the intelligent agents' results, but since our system operates with meta-agents containing a single unique path, the meta-agent (1) needs to be distinctive. Accordingly, the meta-agent (1) is duplicated, and each meta-agent is extended with one of the intelligent agent's calculated costs. Hence, from meta-agent (1) following meta-agents are produced:

- meta-agent [Ag A-B, Cost, Ag B-C, Cost]* (1 extended)
- meta-agent [Ag A-B, Cost, Ag B-E, Cost]* (5 a new copy of 1)

The process above continues through the graph and, eventually, twelve characteristic meta-agents are created. Some meta-agents reach the goal without visiting all nodes in the network. These become superfluous and should not be taken into account in the comparison of the meta-agents. However, in the example the meta-agent (1) results in only one meta-agent, which has visited all nodes:

- meta-agent [Ag A-B, Cost, Ag B-E, Cost, Ag E-C, Cost, Ag C-D, Cost, Ag D-F, Cost, Ag F-A, Cost, Totalcost]* (12)

Even though the meta-agent (1) has a solution for TSP, the meta-agents (2), (3) and (4) perform to find alternative paths. For the meta-agent (2), the result is:

*Meta-agent [Ag A-E, Ag E-B, Ag B-C, Ag C-D, Ag D-F, Ag F-A]*

and for the meta-agent (4):

*Meta-agent [Ag A-F, Ag F-D, Ag D-C, Ag C-E, Ag E-B, Ag B-A].*

The meta-agent (3), on the other hand, must at least visit a node twice and does not give a satisfactory result for TSP.

Conclusively, several meta-agents contain a Hamiltonian path for the network and the total cost. As the agents traverse the network, they maintain a list of the nodes visited. The list is used to control the result against the content of the meta-agents to assure that a Hamiltonian path is being generated.

In the example, we have three different possible routes, i.e., three meta-agents with all the nodes visited. The next step is to compare the results of these meta-agents. Each meta-agent has a total cost calculated and, even though the relation is symmetric, the costs can be asymmetric. As mentioned above, there are no guarantees that going from A-B costs the same as B-A, as in road construction, which might only affect travelling in one direction. Hence, also these routes must be taken into consideration. When a meta-agent is generated for the path, it can reverse the order and cause the intelligent agents recalculate the costs and reassemble the total cost.

In a large network, a considerable number of meta-agents are generated. To find the optimal route among those meta-agents, an effective search algorithm is required. A master meta-agent is available to determine the meta-agent that has completed the path at the lowest costs. In the current system, an effective results comparison for the meta-agents is a linear time-algorithm. This search algorithm compares each solution as it is created and retains the lowest cost solution.

Since the system uses parallel computing, the linear algorithm needs to be notified when the meta-agents have completed their goals. Hence, the master meta-agent collects information about the meta-agents performance, which is sent as messages. Because the meta-agents finish at different rates of time, the linear algorithm is overlapped with the operation of the meta-agents.

In the case of TSP, the meta-agents speed up the computation by collecting the results from the agents. The results are used to form the final solution, but the more interesting use for the meta-agents is to control the computation of the agents. The information collected by the meta-agents can detect the best paths found to date and cause agents suspend the execution of lower ranked paths. The suspended paths can be allowed to perform again later, provided the promising path becomes a poor choice. This will be used in a parallel implementation to optimize the use of processor resources.

Similarly, the meta-agents can apply heuristics to speed up the computation time for a solution. The classic heuristics and approximation algorithms can be constructed by the meta-agents.

It is also possible to apply a dynamic algorithm in the meta-agents. Then the meta-agents are used to adjust the route by changing the order of the intelligent agents, as mentioned above. For each change, the system needs to re-evaluate the route and the total cost for the meta-agent must be recalculated.

Moreover, the meta-agents can respond to environment requirements specified by the user. The user might want to avoid some characteristics in the environment and, by applying these to the meta-agents, they can look for those among the intelligent agents' environment. The meta-agents that consist of these intelligent agents can be removed from consideration.

A future addition is dynamic route reconfiguration. When a route has been partially traversed and conditions change, the system could be triggered to update the route. This would be accomplished by running the route from the current endpoint of arc the vehicle has reached. A new route will be recomputed. This will be a major use of the learning and meta-agents controlling the computation.

## 7 Conclusions and further work

We have applied meta-agents on agents in a multi-agent system. The system works in networks with multi-agents. The system calculates the fastest way from one node to another, by taking several different approaches into account. The meta-agents are built on top of successful and fastest agents in the network but can be also used to show the status of the static and dynamic environment.

The system is still a small network and needs to be expanded with more nodes and arcs. As an example, we have applied the meta-agents and multi-agents on maps for part of Sweden and United States of America. However, it will be expanded with more cities but also countries.

In the current system, the meta-agents are only at one level. In the future, there will be a hierarchy of levels of meta-agents. At the top level is the abstract meta-agent and at the lowest level are the concrete meta-agents. The architecture with several levels of agents can perform a task faster.

For the moment, the static environment does not take all characteristics of an environment into account. For example, sharp curves which makes the agents slow down. Hence, the static environment needs to be expanded with number of curves but also the curves severity.

As a user interaction feature, the weather should be given in their meteorological names and not a range from good to very bad with a five scale. The system can supply translation from a user-given weather type to a number that is used in the calculation as possible delays in the agents' performance.

The system is implemented in logic programming with a text-based interface. Next step is to combine the system with geographic information system (GIS) for the static characteristics to utilise the topography in the GIS. Moreover, the dynamic characteristics need constantly to be updated in the system and therefore, the system needs to work real-time.

## 8 References

- [1] David Chelberg, Lonnie Welch, Arvind Lakshmikummar, Matthew Gillen, and Qiang Zhou. *Meta-Reasoning For a Distributed Agent Architecture*. 2000 <http://zen.ece.ohiou.edu/~robocup/papers/HTML/SSST/SSST.html>.
- [2] Stefania Costantini. *Meta-reasoning: a survey*. In A. Kakas and F. Sadri, editors, *Computational Logic: From Logic Programming into the Future: Special volume in honour of Bob Kowalski* (in print). Springer-Verlag, Berlin. 2002.
- [3] Michael R. Genesereth and Steven P. Ketchpel. *Software Agents*. Communication of the ACM, Vol. 37, No. 7 July, 1994.
- [4] Carl Hewitt, and Jeff Inman. DAI Betwixt and Between: From Intelligent Agents to Open Systems Science. *IEEE Transactions on Systems, Man, and Cybernetics*. Nov./Dec, 1991.
- [5] Anne Håkansson, and Ronald L. Hartung. Calculating optimal decision using Meta-level agents for Multi-Agents in Networks. (KES2007) *11th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*. Vietri sul Mare, Italy 12<sup>th</sup> – 14<sup>th</sup> September 2007.
- [6] George F. Luger. *Artificial Intelligence – Structures and strategies for Complex Problem Solving*. Fourth Edition Pearson Education. 2002
- [7] Michael Negnevitsky. *Artificial Intelligence – A guide to Intelligent Systems*. Pearson Education. ISBN 0201-71159-1. 2002.
- [8] Liviu Panait, and Sean Luke. Cooperative Multi-Agent Learning: The State of the Art. *In Autonomous Agents and Multi-Agent Systems*, Springer Netherlands, Computer Science, Volume 11, Number 3 / November. 2005.
- [9] Michal Pechoucek, Olga Stepánková, Vladimír Marík, and Jaroslav Bárta. Abstract Architecture for Meta-reasoning in Multi-agent Systems. (*CEEMAS 2003*) *3rd International Central and Eastern European Conference on Multi-Agent Systems*, Prague, Czech Republic, June 16-18. 2003.
- [10] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc. 1995.
- [11] Jürgen Sauer and Hans-Jürgen Appelrath. Scheduling the Supply Chain by Teams of Agents. (*HICSS'03*), *36th Annual Hawaii International Conference on System Sciences*. p. 81, 2003.
- [12] Jan Tozicka, Filip Zelenzny, and Michal Pechoucek. Modelling of agents' behavior with semi-collaborative meta-agents. In Pechoucek M.; Petta P., and Varga L. Z. (Eds.): *CEEMAS 2005, LNAI 3690*, Springer-Verlag Berlin Heidelberg. pp. 572–575, 2005.
- [13] Michael Wooldridge and Nicholas R. Jennings. *Intelligent agents: Theory and practice*. Knowledge Engineering Review 10(2), 1995.
- [14] Michael Wooldridge. *An Introduction to MultiAgent Systems*, John Wiley & Sons Ltd, ISBN 0-471-49691-X. 2002.
- [15] Yang Xu, Elizabeth Liao, Paul Scerri, Bin Yu, Katia Sycara, and Mike Lewis. Towards Flexible Coordination of Large Scale Multi-Agent Teams, in *Challenges of Large Scale Coordination*, Springer, 2005.