Agent-Based System with Learning Capabilities for Transport Problems

Bartłomiej Śnieżyński, Jarosław Kożlak

Department of Computer Science
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland

Abstract. In this paper we propose an agent architecture with learning capabilities and its application to a transportation problem. The agent consists of the several modules (control, execution, communication, task evaluation, planning and social) and four knowledge bases (global one shared by all modules and local ones to store information and learned knowledge specific to single modules). The proposed solution is tested on the PDPTW problem. Agents using supervised and reinforcement learning algorithms generate knowledge to evaluate arriving requests. Experimental results show that learning increases agent performance.

Keywords: agent-based system, machine learning, transport problem

1 Introduction

The multi-agent and machine learning approaches are useful methods for finding efficient solutions for transportation problems. Transportation problems are based on serving a set of transportation requests using vehicles with the lowest possible costs. The theoretical problem (for instance, VRPTW or PDPTW) assumes constant velocities and travel times between locations, but, in practice, transportation problems are characterized by a high degree of dynamism and uncertainty - new requests may come while vehicles are on the move and the delays may occur because of the traffic on the roads.

As a result of the distributed character of the problem, applying the multi-agent approach, where individual agents have a high degree of autonomy in decision making but cooperate in the realization of the requests, may be very useful and should be analyzed. Events may take place according to some regularities, not known to the designer of the decision system a priori. Therefore, the application of machine learning could be useful here.

In this work, we are focusing on solving Pickup and Delivery Problem with Time Windows (PDPTW) [10]. In this problem, the requests are described by the location of pickups and deliveries, time windows (time periods when the loading and unloading of the cargo may be performed) and freight capacity. The

* The work has been financed by a grant from the Polish Ministry of Science and Higher Education no. NN516 366236
vehicles are described by their maximum capacities. Every vehicle has the same velocity and travel time between the locations is constant. We can distinguish a static version of the problem, where all requests are known in advance and only the solving of the optimization problem is necessary; and dynamic, where the requests are incoming while the system is functioning and alongside the optimization problem, also the modeling of vehicle movements and operations is necessary.

The goal of the presented work is to design and test a hybrid learning agent, applied to the domain of transportation planning and scheduling and an environment where such agents may function. The agent consists of several modules which use different forms of knowledge representation and apply different decision and machine learning algorithms. The current possibilities of the agent will be verified with the use of selected experimental scenarios concerning decisions about accepting requests.

2 Related research

Because of the high computational complexity of the PDPTW problem (which is an extension of the travelling salesman problem) heuristic approaches are the most widely used methods. Different methods are used: tabu search, evolutionary algorithms, simulated annealing or ant colony approaches.

Apart from the various heuristic approaches, especially important from the point of view of research described in this paper is the application of the multi-agent approaches. The examples of such systems are MARS [5] and Teletruck, which use algorithms and methods such as Contract Net Protocol [14] and Simulated Trading [2]. In some cases the machine learning algorithms are applied, some of these solutions are presented below.

The most popular learning technique in multi-agent systems is reinforcement learning, which allows agent to learn its strategy: what action should be executed in a given situation. Other techniques can also be applied: neural networks, models coming from game theory as well as optimization techniques (like the evolutionary approach, tabu search etc.). Good survey of learning in multi-agent systems working in various domains can be found in [12]. Only several examples are described here.

In [17], agents learn coordination rules, which are used in coordination planning. If there is not enough information during learning, agents can communicate additional data during learning. Airiau [1] adds learning capabilities into the BDI model. Decision tree learning is used to support plan applicability testing. Nowaczyk and Malec are also using learning for evaluating plans. Inductive Logic Programming is used to generate knowledge for choosing the best partial plan for an agent [11].

There are also some works carried out in the transport domain. For example, in [7] machine learning was applied to evaluate travel times between locations for solving dynamic transportation problems with uncertain travel times.
In [9] the agents solving the problem VRP used two techniques: a reinforcement learning for a choice of the best sequences of optimization operators and mimetic learning based on exchange of learned solutions between agents.

In [6] rule induction is used in a multi-agent solution for the vehicle routing problem. However; in this work learning is done off-line. First, rules are generated by AQ algorithm (the same as used in this work) from traffic data. Next, agents use these rules to predict traffic. Continuation of this work is [16], where on-line learning is applied to generate a routing strategy. Agents use the hybrid learning approach: Q-learning is combined with rule induction. Rules are used to predict traffic on the roads and allow the decrease of space for Q-learning.

3 Model of the system

The model of the system consists of an environment which represents a road network, an agent Dispatcher, responsible for providing transportation requests and a set of agents which represents individual vehicles which manage the travel routes and models the movement of vehicles.

This paper focuses on the presentation of the model of the agent which represents the vehicle. It is presented in Fig. 1. The agent consists of modules and knowledge bases. Control module is responsible for managing all the other modules. Communication module exchanges information with other agents. Task evaluation module is used to accept or reject a given transport request. Planning module generates a plan to serve accepted requests. Social module is responsible for modeling the other agents. All modules, except the Control module, have access to the Global knowledge base, where they can store information. The last three modules have their own knowledge bases, where they store knowledge specific to their tasks. The description below concentrates on the following elements: knowledge, performed actions, behavior's and learning processes.
Knowledge. The knowledge of the agent is stored in knowledge sources: global $(K_M)$ and locals. Local knowledge is accessible for processing by individual modules, but global knowledge may be accessible from every module.

The global knowledge base, $K_M$, contains information from the following domains: State, Map, Requests and Plans.

The State contains the following knowledge: current location of the vehicle $(P)$, current travel goal $(NT)$ – next target node), current size of the freight $(CL)$ – current load), maximal allowed capacity $(MC)$ – maximum capacity), information about other vehicles $(OV)$ for example their locations and current travel paths.

Map contains current information about the state of the road network, which is described by a directed graph with weights: $N$ – set of nodes (locations and intersections $N_i$), $E$ – set of edges representing roads $E_i$ (described by the distance $d_i$ and current travel time $t_i$).

Requests $(Rec)$ is a set of accepted requests $Rec_j$ described by locations of pickup and delivery $(Rec_j^{Pickup}$ and $Rec_j^{Deliv})$, time windows of pickups $(t_j^{Pickup}$, $l_j^{Pickup})$ and delivery $(t_j^{Deliv}$, $l_j^{Deliv})$ and needed capacity $c_j$;

Plan (called Route) is a sequence of operations of movements between nodes $Mov(N_{start}, N_{end})$, loadings $(Pickup(req))$ and unloadings $(Delivery(req))$.

One can distinguish the following local knowledge bases: Task Evaluation KB, Planning KB, Social KB.

Task Evaluation KB $(K_{TE})$ contains a knowledge concerning request acceptance: $(K_M, K_{TE}) \rightarrow \{accept, reject\}$.

Planning KB $(K_P)$ contains knowledge about the constructed routes of travel and loading/unloading operations $TR$ and their evaluations (travel cost $TC$, travel distance $TD$, travel time $TT$, waiting time $WT$), also specific knowledge about planning (how to check if a plan is good) can be stored here. $KB$ also contains particular models of road network $Map$ with information which may be taken into consideration during the planning process: graphs describing the states of the map in subsequent time periods $(t,t+\tau)$: $Map(t) = (N(t), E(t))$, where for given nodes, information is associated about the distribution of transportation request location in given time intervals $r_i^{(t)}$, and with edges – an average travel times $t_i^{(t)}$ in that interval. Using such models of a network, it is possible to define criteria $crit_j$ which describe the travel which may be assumed in the planning with expected confidence levels: $Eval(Map^{(j)} : j = \tau \ldots t_{max}, crit_j)$. Another part of planning KB are sets of identified patterns $pat$, which describe dependencies between the states of the traffic on different roads/edges and the changes that occur.

Social KB $(K_S)$ stores information about strategies of communication (frequency of updating of information concerning given subjects $j IU(j)$ and relations between agents (conviction/trust $TR(V_j, Op_k)$ that a given agent $V_j$ will be able to execute a given operation $Op_k$.
Actions. Below the actions performed by each of the distinguished modules of the agent are described. Actions are described by knowledge which is used and agent state parameters which are changed as a result of their execution.

Task Evaluation Module has only one operation - task evaluation: \( \text{EvalRequest} : (K_{TE}, \text{Route}, \text{Req}, \text{StrongC}, \text{WeakC}) \rightarrow \{ \text{accept}(\text{Req}), \text{reject}(\text{Req}) \} \), where \( \text{StrongC} \) represents strong constraints, in our case time window constraints, and \( \text{WeakC} \) are weak constraints based on the following information:

- change of the route length if \( \text{Req}_i \) is accepted,
- sum of accepted requests capacities \( \sum_{j \in \text{Rec}} c_j \),
- current distance to the pick up point \( \text{Req}_i^{\text{PickUp}} \),
- smallest distance of all vehicles to the pick up point \( \text{Req}_i^{\text{PickUp}} \),
- smallest change of the route length if \( \text{Req}_i \) is accepted by some vehicle,
- average sum of accepted requests capacities.

Last three are known because of communication with other agents.

Planning Module's operations are:

- plan creation: \( \text{CP} : (K_M, K_P) \rightarrow \text{Route} \);
- request adding: \( \text{AR}(R_j) : (\text{Route}, \text{Reg}_j) \rightarrow (\text{Route}) \);
- request removal: \( \text{RR}(R_j) : (\text{Route}, \text{Reg}_j) \rightarrow (\text{Route}) \);
- update plan \( \text{UP} : (\text{Route}, \text{conditions}) \rightarrow (\text{Route}) \).

Social Module's actions are:

- \( \text{SetVehConf} \) – describe the confidence to agent vehicle \( k \) considering execution of action \( \text{Op}_j \): \( \text{SetVehConf}(\text{value}, V_k, \text{Op}_j) : \text{value} \rightarrow \text{TR}(V_k, \text{Op}_j) \);
- \( \text{SetComFreq} \) – determine a frequency of communication with subject \( \text{sub} \) for a given state of the network: \( \text{SetComFreq}(\text{sub}, \text{value}) : \text{value} \rightarrow \text{IU}(\text{sub}) \).

Communication Module is a module responsible for an exchange of messages between agents, the messages concern:

- Conditional offer of request: \( \text{OR}(\text{Request}, \text{condition}) \). The condition identifies other action(s) which the agent is willing to has assigned instead;
- Conditionally accept request if specified conditions are fulfilled: \( \text{AR}(\text{Request}, \text{Condition}) \). Condition is specified by an agent which declares a conditional willingness of realization of the request. It may concern an acceptance of some requests previously assigned to this agent by other agents or different reasons (for example neither of other agent is able to serve a considered request). In both mentioned actions \( \text{OR}, \text{AR} \), the condition may be empty, which means the unconditional offer/acceptance of request;
- Send Info \( \text{SI}(\text{Info}) \) – sending information about state of the world.

Execution Module is responsible for the information concerning physical operations of the vehicle:

- freight loading: \( \text{Pickup}(\text{Req}_i) : (\text{CL}) \rightarrow (\text{CL} + c_i) \);
- freight unloading: \( \text{Delivery}(\text{Req}_i) : (\text{CL}) \rightarrow (\text{CL} - c_i) \);
- vehicle movement from old to new location: \( \text{Move}(N_i^{\text{old}}, N_j^{\text{new}}) : (P_i^{\text{old}}, NT_i^{\text{old}}) \rightarrow (P_j^{\text{new}}, NT_j^{\text{new}}) \).
Behaviors. The actions are executed in the contexts of several parallel behaviors:

- evaluation of request (assigning request or conditional acceptance of the request) - a negotiation with the use of Contract Net Protocol [14] is performed, the request is assigned to the vehicle which is able to realize it with the lowest costs;
- exchange of the requests (the vehicle is trying to get rid of a request assigned to it previously – negotiations may be performed with the use of simulated trading algorithm [2];
- modification of plan (in response to the changing conditions – a change of the state of the agent or change of its model of the world, the agent may change the previous plan. It may be done by activities Conditionally Offer Request (OR) or Conditionally Accept Request (AR);
- learning – the learning may concern several representation elements or decision schemes of the agent, the versions of learning will be described in the subsequent section.

Learning. The learning process concerns the modification of the agent model of the world, performed usually as a consequence of new information incoming. The model of the agent considers the following kinds of learning: evaluation of how profitable an acceptance of the given requests is ($L_1$), qualification of the points on the map which are worth visiting ($L_2$), learning of the characteristics of the road network ($L_3$), learning the best frequencies of the propagation of the information about the environment ($L_4$) and learning of the behavior of other vehicles ($L_5$).

The main goal of this research is to test $L_1$ learning. Two learning strategies are tested here: supervised learning and reinforcement learning. In the former, the agent learns to classify decisions as good or bad. To learn the classifier, it needs labeled examples. Every example consists of the attributes representing WeakC and the decision (accept, or reject). Example related to Req is stored after delivery of the request or if it is too late for delivery. It is labeled as good if the request was accepted, delivered and if other requests appear between accepting the Req and delivery of Req were not rejected, or if Req was rejected and accepted by some other agent. Example is labeled as bad if the request was delivered but at least two other requests were rejected or Req was not delivered by any agent and the agent could accept it (StrongC were fulfilled). In the case of reinforcement learning, the state is defined by variables representing WeakC. If the agent accepts the request and delivers it on time, it gets a reward equal to 1. In all other cases the reward is equal to 0.

$L_2$. On the basis of the gathered information about the spacial and temporal distribution of request locations $Map^{(t)}$, some conditions conditions necessary to be fulfilled by the routes are identified (set by action $UP(route, condition)$, for example points are identified where the road should preferably pass. These are the points characterized by a high density of locations of request points. Some works concerning the modification of vehicle routes to achieve this goal were presented in [8].
A L3 takes into consideration the history of its travels and travels of other agents \((\text{Map}^{(t)})\). The agents may create patterns \(\text{pat}\) describing changes of the network characteristic, for example, dependencies between the changes of travel times between different roads. A model of learning of the current travel times was described in [7].

L4. Getting information about the changes of the road network characteristics may be executed at higher or lower frequencies. As a result of learning, the agent gets the preferred frequency of the model of the world update for the given network configuration. This is based on the evaluation of the results obtained for different state of the traffic \(\text{Map}^{(t)}\) and different \(\text{IU}(j)\) and how the increase of IU changes the average values from Task Evaluation KB. Some preliminary tests concerning this kind of learning were described in [7].

L5 relates to learning of the preferences of other agents/vehicles concerning acceptance of different kinds of requests (for example, depending on the given operation locations or size of the freight), on the basis of it, the agent may determine chances of delegation of a request to other given vehicles.

4 Testing environment and performed experiments

To test the performance of the learning agent, we built a multi-agent system and performed several experiments. The environment is implemented in Java language and makes it possible to solve the PDPTW problem in static and dynamic versions. We started with Acceptance Module learning. Other modules are fixed. The system and results are described below.

Software used in the experiments. For the purpose of this research we used a multi-agent system Environment for Learning Agents (ELA), build for testing learning agents in various environments. This system uses Weka [19] for supervised learning, and PIQLE library [4] for reinforcement learning. It provides a general framework, communication facilities and basic classes (such as agent, environment etc.) which can be adapted to create a system for a chosen domain. So far it was used to perform experiments in the Farmer-Pest problem [15], Predator-Prey domain, and PDPTW, described here.

Results of the experiments. As it was mentioned, the first set of experiments was prepared to test Task evaluation module learning \((L1)\). The following supervised learning algorithms were applied: Naïve Bayes classifier, C4.5 [13] (its Weka implementation is called J48), RIPPER [3] (JRip implementation). Also Watkins's \(Q(\lambda)\) [18] reinforcement learning algorithm was used (shorter name "Watkins" will be used). It is a modification of well known Q-Learning algorithm [18]. As a reference, greedy algorithm was used. In this setting the first agent for which \(\text{StrongC}\) are satisfied accepts the request.

Every experiment consists of 100 sequences of 10 (for the first experiment) or 20 simulations. Agents' knowledge was kept between simulations, but it was cleared between sequences. All vehicle agents used the same algorithm. There
were as many repetitions as many algorithms tested. Figures present the averages from sequences. Supervised learning algorithms were executed between simulations. Acceptance actions were chosen according to the Boltzmann strategy [18] with $\tau = 0.2$ to provide appropriate exploration of possibilities. Default parameters of the algorithms were used. Watkins algorithm was also executed between simulations with $\lambda = 0.4$, $\gamma = 0.9$, and $\alpha = 0.8$. Exploration strategy was $\epsilon$-greedy with constant $\epsilon = 0.1$ (which gave better results than Boltzmann strategy in initial experiments).

We tested the mentioned learning algorithms in several situations, with various numbers of vehicles, various capacities and speeds, various numbers, frequencies and space distributions of requests.

The first experiment is simple. There are 2 vehicles with capacity 1, and speed 10 which should distribute 90 packages. There were two clusters of places on the map. Let us call them $A$ and $B$. Source and destination places were in the same folder. There were 80 requests in a cluster $A$, appearing every 5 time steps and 10 requests in cluster $B$ appearing every 15 steps. Time windows in $B$ were 5 times larger than in $A$. Results representing the number of requests served are presented in Fig. 2-(1). In the second experiment there were 4 vehicles with speeds 12 and capacity 1. Requests were appearing in two clusters every 5 turns, there were 100 steps for pickup and 500 for delivery. The learning process is slower, therefore there are 20 simulations in every sequence. Results are presented in Fig. 2-(2). Configuration of the third experiment was almost the same as above, the only difference was that the vehicle speed limit decreased to 10. Results are presented in Fig. 2-(3). The last, fourth experiment, there were 9 vehicles with a capacity of 2 and speed 15. There were 90 requests of size 1 appearing in every one of the three clusters, every step. Time for the pickup was 50 and time for delivery was 200. Results are presented in Fig. 2-(4).

Discussion of results. For all experiments, we can observe that the learning algorithm has better results than greedy. $t$-Student at the final simulation confirms that they are significantly better at $p<0.05$.

All agents using supervised learning algorithms improved throughout time. In experiments 3 and 4 we can observe initial fall of J48 and JRip performances, because not enough experience is initially collected. Next, the performance increases. At the beginning, the increase is rapid because a lot of significantly new examples are stored, which results in the generation of new knowledge. After several simulations, performance becomes stable because examples do not contain any new information.

Watkins algorithm behaves more stable, it does not improve. Probably a better parameter setting should be used here. Also in the first experiment it produces very good results at the beginning. The reason may be in another exploration strategy. However, this needs further investigation.

In the first experiment the best results are archived by agents using JRip. However, the difference is not statistically significant. In experiments 3 and 4 the best is Naive Bayes and the difference is significant at $p<0.05$. These envi-
environments are difficult and their stochastic character seems to be responsible for good performance of Naïve Bayes.

5 Conclusion

As a result of the work the model of learning agent for solving transport problems was designed and applied. The environment for modeling transportation problems was implemented together with the modules necessary for applying different machine learning techniques. Experimental results show that machine learning algorithms allow the agents to update and correct their knowledge which results in the development of better plans in the partially unknown environment.

Future works will mainly concern the integration and evaluation of all learning agent modules. Also performance of the system in environments with changing characteristics should be evaluated.

In this paper, the solving of classical PDPTW problem is presented; however, the presented infrastructure is designed for solving in the future more complex
problems, related to: solving soft time windows and variable travel times between locations.

Acknowledgments. The authors thank students of Computer Science from AGH-UST, especially M. Mlostek and M. Pulchuny, for the participation in the development of simulation environment and realization of the experiments.

References