Analysis of Semantic Probabilistic Inference Control Method in Multiagent Foraging Task

Vitaly Vorobiev, Maksim Rovbo
National Research Center «Kurchatov Institute»
Moscow, 123182, Russia
Vorobev_VV@rrcki.ru, rovboma@gmail.com

***Abstract*—Adaptation in robotics systems is often implemented as some form of learning. While much research is dedicated to studying policy and value approximation in reinforcement learning, some methods are based on rule inference and logical descriptions. One of these methods is based on a semantic probabilistic inference algorithm that has its roots in the theory of functional systems. In this article, the method is applied to a distributed multiagent foraging problem that has an important property of providing an environment that allows to study a decentralized system of individually learning agents. We compare the performance of this method to other methods: Q-learning and a random choice algorithm as a baseline. We also propose a modification of the algorithm that includes an exploration behavior. Experiments are carried out in a computer simulation system. The results show the performance of the algorithms with different parameters, as well as the effect of exploration on the performance.**

***Keywords*—adaptive control, robotics, semantic probabilistic inference, foraging, local interaction**

I. Introduction

Adaptive control system for robotics are of practical interest since they promise to increase robustness of existing systems, make the behavior closer to optimal as well as introduce the possibility to impart new behaviors to the robot by a system of rewards or examples. This may be especially important for multiagent systems as controlling them in a direct way to achieve a given goal is harder than single robots.

A lot of current research is dedicated to learning methods for virtual and robotic agents that is based on reinforcement learning methods using value and policy approximations, especially based on parametric descriptions of the functions and neural networks. Multiagent aspect introduces event more problems, like operating in a dynamic, non-markovian environment that makes even a static environment more challenging due to the activity of the agents themselves in relation to other agents. Robots often work in an environment, where only some information about the state is accessible, which means making decisions in a partially observable environment. Thus, seeking efficient ways to search the policy space for acceptable (and, preferably, optimal) observation-action mappings is important.

One of the ways to address this problem it to seek biologically inspired models of decision making or using different representations of the problem and policy space, such as logical. There are various approaches and methods that use logical descriptions that could be used for decision making, for example, semiotic networks [10], JSM method [2], semantic probabilistic inference [7], [9].

Semantic probabilistic inference (SPI) is a learning method for an agent that uses logical (rule-based) descriptions of the actions of the agent that was introduced in [9]. It is based on a mathematically formalized concept of a functional system from the theory of functional systems [7].

One of the main goals of this work was to study and compare capabilities of the SPI method and some reinforcement learning methods in a multiagent setting with physically distributed agents and also the effects of exploration and some other, problem-specific parameters, on the agents' performance. While SPI was used as a basis of a network composed of logic neurons and studied in a multiagent context in [1], in those works agents controlled tightly coupled (physically connected) elements of a robot, used a common reward from a centralized source and inferred the rules in a single system that could create rules specific to each agent, as well as general rules. In this work we emphasize that the studied system does not provide agents with a common reward (each reward is specific to the agent), the agents have only an indirect effect on each other's performance and they do not have to use a centralized rule-based learning system, but can learn separately from each other.

The chosen problem environment is a foraging problem, where agents must gather food units in a grid world since it can be seen as a reasonably representative problem for some simple group robotics tasks and it satisfies the environment requirements described above. We also propose a modification of the SPI algorithm that introduces exploration behavior into the system so that the agent is less susceptible to local minima of performance, especially in a stochastic environment.

On the other hand, an important issue is the question of the application of the logical system of adaptive control, which is based on the algorithm of semantic probabilistic inference, to a group of mobile robots that allows local interaction. In this regard, it is proposed to consider the

possibility of using such a control system for a group of robots that solve some common task. In this case, the main emphasis is placed not on the solution of the common task but on resolving the problem of organizing communication and capabilities of separated and moving robots.

The work is structured as follows. Firstly, methods and algorithms used in the paper are presented, as well as the proposed modification. Then the model of the problem is described. After that, the experiment parameters, simulation results and the analysis follow. Then, in the section "Organization of a group of robots for collective application of the logical model of an adaptive control system" further research is described, detailing the problems that need to be solved and a proposed approach to adapt the semantic probabilistic inference for a physically distributed group of mobile robots. Finally, a conclusion sums up some key points about the article.

II. Methods and algorithms

The main algorithm that is studied in this work is the semantic probabilistic inference that is described in [9] but without the mechanism of new functional system formation as it was observed in the original work that for a foraging problem forming new functional systems is not required. The algorithm also uses the concept of a goal predicate, but in the later works [1] a reward was used as a prediction, which is what we use here, but without creating logic neurons described in the latter work. The reward predicted by the rules always equals to one, so it can be written as a goal predicate that states that the agent gets a reward of one.

We also propose a modification of the algorithm by introducing an exploration behavior into the system. The original algorithm chooses a random action only in cases where the situation was never encountered before and / or there were no suitable rules inferred from the experience. Instead, we also add a random possibility of choosing an action randomly with uniform probability that is governed by an exploration rate ε. This is similar to exploration done by the Q-learning algorithm and should help the agent gather information about alternative choices of actions in situations that already have a suitable rule. There are also cases where such exploration strategies were successfully applied to foraging problems [6], so it seems reasonable to try it for the SPI algorithm in a multiagent setting.

The following is a short version of the SPI algorithm as it is implemented in this work. The proposed modification is marked by an asterisk and is basically everything that uses ϵ.

1) Parameters of the algorithm basic_rule_depth *brd*, max_plan_length *mpl* are set, the environment and agents are initialized.
2) Agent receives an observation *obs* from the environment, which includes the reward *r* from the previous action.
3) Agent updates its experience table (called here *spi_table*) by adding 1 to the record describing a combination of the last state s_{last}, sequence of last actions taken $aseq_{last}$ and the resulting reward r (0 or 1):
$$spi\_table[s_{last}, aseq_{last}, r] \mathrel{+}= 1$$
4) Rules for regularities detection are created by exploring a graph that has rule of the following form as its nodes:
$$P_1 \wedge P_2 \wedge ... \wedge A_1 \wedge ... \wedge A_{mpl} \rightarrow r$$
which contain state predicates (P_1 can be, for example, a fact "the left cell has food") and a sequence of actions A_i, in the precondition and a predicted reward *r* in the postcondition that always equals 1 (otherwise the rule would never be applied). Nodes are explored in two steps. Firstly, all possible rules with no more than *brd* predicates in the precondition including action predicates and no less than one action are built by expanding a node with a single new predicated added to the preconditions. During the second stage, only the rules that pass a positive rule regularity check are expanded. The first node is a $rule \rightarrow r$.
5) Positive regularity check for a rule means that its estimated probability to yield a reward *r* is higher than that of any subrule that can be formed with a subset of its preconditions. Only rules that pass a positive regularity check are added to the list of *regularities* for decision making. The probability check is the following inequality:
$$\frac{n(P_{rule} \wedge A_{rule} \wedge r)}{n(P_{rule} \wedge A_{rule})} > \frac{n(P_{subrule} \wedge A_{subrule} \wedge r)}{n(P_{subrule} \wedge A_{subrule})}$$
where P_{rule} — state preconditions of the rule, A_{rule} — action preconditions (planned actions) of the rule, $P_{subrule}$ — state preconditions of the subset rule, $A_{subrule}$ — action preconditions (planned actions) of the subset rule, n(predicates) — number of times the predicates were applicable to agent's situation stored in its experience table *spi_table*.
6) (*) Exploration action is carried out with ε probability, which is a randomly chosen action with a uniform probability and step 9 is performed. Otherwise the usual SPI action selection is applied.
7) If exploration action was not chosen, all discovered rules' applicability to current state from the regularities list are checked. If a rule's precondition is satisfied, its performance rule_performance (probability to get a reward following the precondition actions from the current state) is checked according to the formula:
$$rule\_performance(rule, state) =$$
$$= \frac{p(P_{state} \wedge A_{rule} \wedge r)}{p(P_{state} \wedge A_{rule})}$$
where P_{state} is all predicated describing the current observation (not just the predicated from the rule's precondition). A list of such rules is formed with their performances.
8) The applicable rule with the highest performance is chosen and its first action (if there are several in the rule action plan) is chosen to be performed:
$$chosen\_rule =$$
$$= \underset{rule}{argmax}(rule\_performance(rule, state))$$

$$action = (A_{chosen\_rule})_1$$

9) The chosen *action* is stored as the last action performed, the sequence of length *mpl* of last actions is updated, the current state is remembered as the last state and the action is returned to the environment to be performed.

The SPI method was expected to show relatively high performance, surpassing some classical reinforcement learning algorithms at least on initial episodes, since it can aggregate states from observation space by only deciding on a few variables (predicates) from it. The original (with exploration rate $\epsilon = 0$) SPI also quickly adapts rewarding behavior, but it can be hypothesized that it can fall into a local minima because it does not seek new rules actively for a state that already has a rewarding regularity discovered. Hence the proposal to add an exploration coefficient to it, so that it can sometimes check other actions.

A random choice algorithm was used as a baseline for comparison. Another algorithm to compare the performance against was the classical Q-learning (in tabular form), described, for example, in [4]. It maps states to actions by using a value function *Q(s, a)* that serves as an estimate of how much reward an agent can get from a given state *s* by choosing an action *a*. The actions are chosen to maximize the reward:

$$a(s) = \underset{a}{argmax}(Q(s, a))$$

The state-action values are stored in a table and are updated using both the actual received reward *r* at the current step after receiving information about the state *s'* the agent was transferred to and the estimate of the next rewards:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \cdot \underset{a}{max} Q(s', a) - Q(s, a))$$

The estimate is updated with a learning rate α that can be interpreted as indication of how much the observations are trusted and the discount rate γ, which indicates how much the agent values (or trusts the estimates of) the future rewards.

It should be noted that in partially observable environments observations must be used instead of the whole states and that the method uses the whole state information to make decisions. It means that if only one small part of the observation changes, the agent treats the situation as a completely new one and will have to learn it from scratch.

III. Experimental Setup

Foraging problem is a relatively common testing environment for multiagent reinforcement learning problems, but can be formulated differently. Here it is defined in the following way. There are three types of objects – food units, obstacles and agents situated in square grid cells. Neither agents nor food can be located on a cell with an obstacle, but there can be unlimited amount of food units or agents on any of the free cells. The obstacles are located only at the edges of the field (Fig. 1):

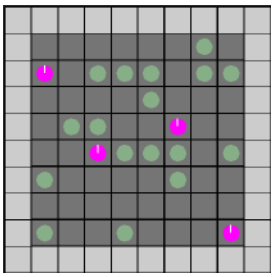

Figure 1. Foraging environment. Green circles are food locations, pink circle is an agent and the white line on it shows the direction it is currently facing, light squares on the fringes are obstacles.

Agent can move and when moving onto a cell with food objects it «eats» one of them and gets a reward of 1 as part of the observation on its next step. This is to have a unified and more realistic interface between the agent and simulation, since rewards are not a separate entity of the world, but rather an interpretation of the agent of the situation or an external signal that explicitly communicates a reward. Agent has a direction, where it is currently facing. It can choose one of three actions: moving forward one step in the direction it is facing, turning 90 degrees left or right.

An observation consists of a number of nearby cells and their simplified contents. The simplification is that only a type of the object is shown on the grid to the agent and none of its internal parameters, and when there are multiple objects in a cell, only one of them is detected. The exact cells to be observable depend on the observation radius and are calculated using a maximum norm. The direction of the agent is facing also determines the observation so that the agent also faces 'up'. For example, for radius of one the observation looks as shown in Fig. 2:

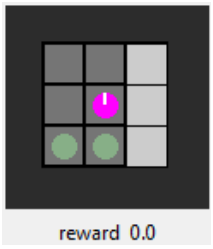

Figure 2. Observation with vision radius equal to 1. In this example, three obstacles are shown to the right, two unites of food are located behind the agent with one being diagonally to the left, the agent itself is in the center and it always looks forward ('up') in its local system of observation, and the reward received on the previous step is 0.

The goal of each agent is to maximize the cumulative reward (food it gathered) over an episode, where an episode is a fixed length sequence of steps. For con-

venience, instead of a rolling sum, all of the steps are separated into non-overlapping episodes.

IV. Results and discussion

The model of the world consisted of a 25 by 25 grid with 100 randomly placed food units. The amount of food on the field was constant which was provided by randomly placing another unit of food whenever a unit was gathered by an agent. There were simultaneously 10 agents on the grid with the same starting parameters, but that evolved independently of each other (without deliberate communication). At the start of each step, all agents decided on an action based on the observation of nearby surroundings with a vision radius of 1 (all contiguous to an agent cells, including diagonal, were seen by it). Then the simulation executed each of the actions sequentially. On the next step, agents were provided with a reward (if they gathered a unit of food on the previous step) of 1 by including that information in the observation.

Experiments were carried out 10 times for each set of parameters and the results were averaged over the experiments. They were further averaged over the agents to get a learning curve for an algorithm with a set of parameters. While a standard practice for single agent simulations in reinforcement learning, it also makes sense in this case for a multiagent problem, since agents of the same type (with the same algorithm and starting parameters) show similar average performance.

When a new experiment starts, the world and agents reset (learned parameters are not kept between experiments). Each experiment consisted of 2000 steps that are grouped into episodes — each episode is a 100 steps. The episodes are mostly a more convenient way to view the results of a simulation, since the task itself is continuous. Each episode's cumulative reward for each agent is recorded.

Semantic probabilistic inference had its rule depth limited to 1 (rules with more than one precondition were retained only after a successful regularity check was performed) and its plan length to 1 (so that no more than one action is in a precondition). It was tested both with the proposed exploration modification and several exploration rates ϵ., as well as without it as in the original article [9], which is equivalent to $\epsilon = 0$.

Q-learning parameters used in the experiments are the following: exploration rate $\epsilon = 0.05$, learning rate $\alpha = 0.1$, discount factor $\gamma = 0.1$.

Computational experiments were carried out in a custom Python simulation system. Experimental results are shown in Fig. 3.

The random learning agent serves as a baseline for comparison and also allows to check that the environment is not too easy for an agent — rewards must be relatively low between a random and a learning agent.

The foraging problem itself in this formulation has several interesting qualities — locally (within a single agent's viewfield) it is almost deterministic, that is, correct actions will always lead to the same reward, unless another agent interferes, which is relatively rare for such rate of agents to the field size (less than 2% of cells are occupied by agents, which observe at most about 14% of the world). Outside of the immediate observation area, however, the world is highly unpredictable to agents, since they only have a partial observation of the world state (and a considerably limited, at that), with food spawning randomly and agents having almost no information, which could allow to determine food location outside of the immediate observation.

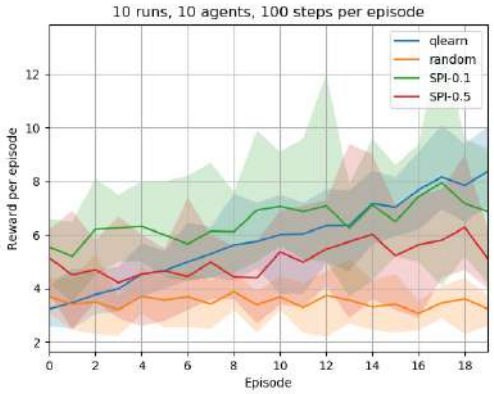

Figure 3. SPI – semantic probabilistic inference algorithm. The number after 'SPI' is the ϵ used. Random algorithm is a baseline for comparison. The bold line is the mean over all agents and experiments, the shaded area shows minimum and maximum performance between agents with the same algorithm.

This leads to a behavior that is similar to random choice when no food is observed (although, agents show a preference for forward movement after learning) and a set of rules that instruct to pick up observed food as soon as it appears in the agent's view. However, this environment did not demonstrate significant opportunities to trap an agent's policy in a local minimum because of the aforementioned properties. Most random significant drops in an agent's performance can be attributed to having collected all the food in a local area and randomly moving in a now-empty area that the agent has no way of determining without sufficient memory.

This might contribute to lower importance of exploration in this problem, despite it usually being important to get a high performance for some types of learning agents, as seen from, for example, comparison in the work [5]. We can also observe that high values (0.5 on the graph) lead to a decreased performance, which is expected since even optimal rules would be ignored half of the time.

Q-learning quickly becomes overwhelmed by the state space when increasing the radius of the agent vision and learns very slowly, while showing performance below

that of a random agent. It is not shown on the graph, but separate experiments have shown that with vision radius equal to 2 cells away, Q-learning surpasses performance of 50 by about 200th episode (which is 200 000 steps).

V. Organization of a group of robots for collective application of the logical model of an adaptive control system

Firstly, let's consider, instead of a shortened version studied above, a full logical model of an adaptive control system that is introduced in [9]. It can be formally written as $CS = \langle S, A, P, F \rangle$, where F is the description of the hierarchy of the basic elements of the system, which are functional systems, as $S = \{S_1, S_2, ..., S_n\}$ is the set of sensors of the robot (a robot with such structure will be further called animat), $A = \{A_1, A_2, ..., A_m\}$ is the set of its possible actions and $P = \{P_1, P_2, ..., P_l\}$ — the sensory information at a specific time in the form of predicates which can describe not only the current, but also the past states of the sensors.

Each functional system (FS) is a tuple $FS = \langle PG, G, PR \rangle$, where PG is a predicate-goal, describes a goal that is represented using the conjunction of sensory predicates P, i.e. $PG = P_1 \wedge P_2 \wedge ... \wedge P_l$. If this predicate is true, then the goal is achieved. $G = \{PG_1, PG_2, ..., PG_n\}$, where PG_i are goal predicates corresponding to the goals of the subordinate FS in the hierarchy. PR is a pattern, in the form of:

$$P_1 \wedge P_2 \wedge ... \wedge P_n \wedge PG_1 \wedge PG_2 \wedge ... \wedge PG_m \wedge A_1 \wedge A_2 \wedge ... \wedge A_k$$

which shows that:

- If the animat is in the state described by the sensory predicates $P_1, P_2, ..., P_n$;
- If in this situation it sequentially reaches the sub-goals $PG_1, PG_2, ..., PG_m$;
- If then it sequentially performs actions $A_1, A_2, ..., A_k$;
- Then it will reach the goal G with some probability.

With a given goal or sub-goal and known information about the world and the internal state of the FS, the task of this FS is to find the best way to achieve the goal by performing the actions chosen on the basis of the prediction.

The example of general scheme of this architecture is shown in Fig. 4

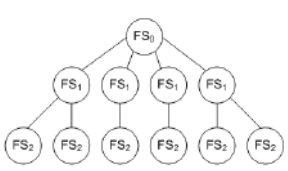

Figure 4. Example of adaptive control system architecture.

Since this case describes the model of the adaptive control system for a single animat, then considering a group of robots with local interaction, each robot in the group can be associated with a separate FS. The links between individual FSs are then identical with the local links between the robots of the group. Therefore, organizing the processes of transferring subgoals from higher-level FS-robots to lower-level FS-robots and transferring results or predictions of results from lower-level FS-robots, you can ensure that the algorithm for semantic probabilistic inference for a group of robots is similar to the same algorithm for a single robot. This allows to treat a group of robots as a single goal-directed entity driven by the SPI algorithm.

Thus, to adapt the logical model of the adaptive control system for a group of robots, it is necessary to build a hierarchy of relations within the group of robots, each of which is known for the adaptive control system architecture.

This first problem can be solved in two stages: first, a leader is selected in the group of robots that will be the top of the hierarchy. Restriction on the locality of interactions between agents makes leader selection non-trivial, but algorithms that can solve this problem are described in, for example, [3] or [8].

By applying one of these algorithms [8], which is based on the redistribution of weights between robots, a hierarchical communication structure can be created (Fig. 5):

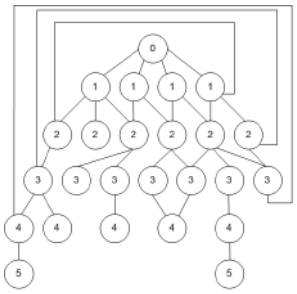

Figure 5. Hierarchical structure resulting from the use of a leader selection algorithm for a group of robots with local interaction.

It can be seen from the figure that it is a structure similar to the structure necessary for the functioning of the logical model of the adaptive control system, where the root agent '0' corresponds to the one of FS.

However, there are still links connecting several robots at the top level with the same robot at a lower level. In this regard, the second stage of preparation is necessary, i.e. carrying out the procedure for removing such connections, which consists in sending the leader a special message M with his number to his neighbors. As soon as the message M is received, the robot remembers the number of the "parent" robot from which it received it and will continue to receive messages only from it. Then he sends the message M with his number and the number

of his “parent” to his neighbors. Neighbors of the upper level, except for the “parent”, by accepting this message, will exclude him from their neighbors. Lower level neighbors will remember him as their “parent” unless they already have another. If messages from different “parents” came at the same time, a “parent” with a large number is selected.

This is a particular example of when the connections in the adaptive control system form a tree. In general, there is no need to remove additional links.

It remains to assign the remaining FS robots from $F^{(k)}$ in such a way that they satisfy the expressions describing the set $F^{(k)}$. This is achieved by assigning each neighbor of the robot a separate branch of a common tree, which describes the scheme of relations of individual FS with each other in the adaptive control system. Each neighbor thus gets its own tree for distributing its branches among its neighbors, etc. Obviously, with such a physical organization, a number of restrictions appear, for example, the number of lower-level FS that can be controlled by the upper-level FS is limited by the maximum number of local communication channels of the robot.

Consequently, in a given group of robots with local interaction, only some functional system hierarchies can be implemented. Another factor limiting the number of possible hierarchies is the number of robots in the group, i.e. it is impossible to implement systems with $|F^{(k)}| > R$, where R is the number of robots in a group. Moreover, even if $|F^{(k)}| > R$, it does not guarantee that robots will be able to organize such a control system. This is due to the fact that it is impossible to predict what hierarchical structure robots will build in the process of self-organization, knowing only their numbers and the number of their possible local neighbors. The probability of building a suitable structure increases if R is noticeably greater than $|F^{(k)}|$ and increasing L. In other words, the more robots in a group, and the more neighbors each robot can have, the more complex functional system hierarchy they can reproduce.

A similar approach can be used to create a group of robots with local interaction, which can be controlled as a whole using a logical model of an adaptive control system.

ACKNOWLEDGMENT

This work was supported in part by grant RFBR 18-37-00498 mol_a (parts concerned with the semantic probabilistic inference) and RFBR 17-29-07083 (multiagent Q-learning).

REFERENCES

[1] Demin A.V., Vityaev E.E. Adaptive Control of Modular Robots. Biologically Inspired Cognitive Architectures (BICA) for Young Scientists. BICA 2017. Advances in Intelligent Systems and Computing, 2018, vol. 636, pp. 204-212.

[2] Finn V.K. Plausible inferences and plausible reasoning. Journal of Soviet Mathematics, 1991, vol. 56, no 1, pp. 2201-2248.

[3] Karpov V., Karpova I. Leader election algorithms for static swarms. Biologically Inspired Cognitive Architectures, 2015, vol. 12, pp. 54-64.

[4] Sutton R.S., Barto A.G. Reinforcement Learning: An Introduction, Cambridge, MA: The MIT Press, 2018, 552 p.

[5] Tokic M. Adaptive ϵ-greedy exploration in reinforcement learning based on value differences. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2010, vol. 6359 LNAI, pp. 203-210.

[6] Yogeswaran M., Ponnambalam S.G. Reinforcement learning: Exploration-exploitation dilemma in multi-agent foraging task. Opsearch, 2012, vol. 49, no 3, pp. 223-236.

[7] Vityaev E.E. Printsipy raboty mozga, soderzhashchiesya v teorii funktsional'nykh sistem P.K. Anokhina i teorii emotsii P.V. Simonova [The principles of the brain from the Anokhin's theory of functional systems and P.V. Simov's theory of emotions]. Neiroinformatika [Neuroinformatics], 2008, vol. 3, no 1, pp. 25-78.

[8] Vorobiev V.V. Algoritmy vybora lidera i klasterizatsii v statich-eskom roe robotov [Leader choice and clustering algorithms in a static swarm of robots]. Mekhatronika, avtomatizatsiya, Upravlenie [Mechatronics, automation, control], 2017, vol. 18, no 3., pp. 166-172.

[9] Demin A.V., Vityaev E.E. Logicheskaya model' adaptivnoi sistemy upravleniya [Logical model of the adaptive control system]. Neiroinformatika [Neuroinformatics], 2008, vol. 3, no 1, pp. 79-107.

[10] Osipov G.S., et al. Znakovaya kartina mira sub"ekta povedeniya [Symbolic worldview of a subject of behavior]. Moscow, FIZMATLIT, 2017, 259 p.

АНАЛИЗ МЕТОДА УПРАВЛЕНИЯ НА ОСНОВЕ СЕМАНТИЧЕСКОГО ВЕРОЯТНОСТНОГО ВЫВОДА В МНОГОАГЕНТНОЙ ЗАДАЧЕ ФУРАЖИРОВКИ

Воробьев В. В., Ровбо М. А.

Адаптация в робототехнических системах часто представляет собой какую-либо форму обучения. Хотя многие исследования посвящены изучению приближения стратегии и функции полезности в обучении с подкреплением, некоторые методы основываются на выводе правил и логическом описании. Один из них основан на алгоритме семантического вероятностного вывода, который имеет корни в теории функциональных систем. В этой статье метод применяется к распределенной многоагентной проблеме фуражировки, которая имеет важное свойство в виде среды, позволяющей изучать децентрализованную систему индивидуально обучающихся агентов. Мы сравниваем эффективность этого метода с другими: Q-обучения и алгоритма случайного выбора в качестве основы сравнения. Мы также предлагаем модификацию алгоритма, включающую исследовательское поведение. Эксперименты проведвены в системе компьютерного моделирования. Результаты показывают эффективность работы алгоритмов для различных параметров, а также влияние исследовательского поведения.

Received 10.01.19