# Agent-Space Transfer Learning for Sign-Based World Model

Maksim Rovbo
*Laboratory of Robotics*
*National Research Center «Kurchatov Institute»*
Moscow, 123182, Russia
rovboma@gmail.com

***Abstract*—Several algorithms were studied for transfer learning problem for a foraging task. These algorithms are suitable for a semiotic control system to facilitate experience transfer between two learning agents with different agent-space descriptions of the state in a predicate form. They require that the target agent's description of the task is a subset of the source agent's description.**

**The algorithms are based on the Q-learning algorithm and uses a custom transfer function to initialize the target agent's state-action table, which matches the corresponding predicates. The test problem used to test the algorithms is a foraging task, where an agent must gather randomly generated food in a grid world.**

**The results show that they provide improvement in the learning curve for the target agent after transferring experience from an agent with more input predicates. The improvement is inconsistent and sometimes does not bring noticeable difference in performance, but the target agent performs at least as good as an agent with no prior experience.**

***Keywords*—transfer learning, reinforcement learning, semiotic control, robotics**

## I. Introduction

Robotic systems, both single- and multi-agent, need to have adaptive properties to robustly achieve various goals in real, uncontrolled environments, which is often the case for mobile robots. One of the prominent research areas that can achieve that is reinforcement learning, which views robots as agents with a predefined input space and discrete or continuous actions that act in an environment and receive rewards for useful actions. This allows robots to learn and optimize goal-directed behavior. On the other hand, semiotic control systems allow robots to have a structured and more interpretable world models based on rules and predicates. To use reinforcement learning methods requires some adaptation of the corresponding models and algorithms since semiotic control systems use signs and connections between them as a basis for description of the world, while reinforcement learning mostly uses a vectorized description.

Sign has four parts, including the name, the percept, the functional meaning and the personal meaning [6]. The percept is a set of predicates used to describe the concept encoded by the sign. It can be used to connect the sensors of the agent and its logical control part of the control system by implementing some of the predicates as algorithmic functions of the sensor data. Different agents can have different predicates available. To use reinforcement learning in a multi-agent system for such agents requires a way of using different predicate descriptions when transferring experience between agents.

Transfer learning in reinforcement learning deals with the application of the knowledge about solving a problem by an agent to another, somewhat similar problem, which can also be used to transfer experience between different agents. A distinction can be made between action-space and problem-space descriptions of the task, which is important to consider, since it can lead to more efficient learning algorithms [3]. Problem-space descriptions are usually full descriptions of the state of the environment and are thus impractical for mobile robots, so the focus in this work is on agent-space descriptions. Transfer learning can be done in different ways, which also depend on the problem specifics such as whether the input representation is the same space, are there common useful sequences of actions between tasks, etc. Transfer learning methods can be based on modifying action selection, for example, by adding a heuristic function that contains the relevant knowledge [1], deep learning [7], transfer of samples from one task to another [5]. Manual mapping between tasks and transferring data between approximators [10] is the most similar to the problem considered here, since it can be used for agents with different input spaces and also uses averaging.

The idea of the current paper is to use descriptions of the world inherent in a semiotic control system to facilitate experience transfer between two learning agents with different agent-space descriptions of the state. In terms of the taxonomy given in [4], the problem considered here is the transfer across tasks with different domains (from the agent-space point of view) or transfer across tasks with fixed domain (from the problem-space point of view), while the algorithm uses a special case of parameter transfer that accounts for the difference in state representations of the agents.

Partially supported by RFBR 18-37-00498 mol_a

A foraging problem is considered as a test environment for the algorithms since it is common for multi-agent robotic research, was used for reinforcement learning evaluation and can serve as a model for some application such as resource gathering or energy collection [2], [11].

## II. Methods and algorithms

The objective of the method is to improve the performance of the learning agent by transferring experience to it from a source agent with a possibly different input space. To connect them, a description is proposed based on the notion of a sign [6] as used in some semiotic control systems [8].

Formally, a sign is defined as

$$S =< n, p, m, a >, \tag{1}$$

where n is the name of the sign, p is the percept that describes it (in this case, a corresponding sensor description), m is the knowledge about the sign encoded in rules (which is not important for task in this paper) and a is the encoded personal experience of the agent, such as actions associated with the sign. The name can be used to match different agents' descriptions of the input even when the sensors used for detection of the objects are different and the personal experience can be used to store a combination of an agent's state sign and the corresponding action and results for reinforcement learning.

A grid foraging problem is used as a test environment in the following sections, but it is useful to introduce some of the elements here. An agent moves on a grid and receives the local state of the environment as input. Two types of agents' inputs are considered. One of them will be called 'type A', which has sensors that allow it to determine, what is located in the 3x3 square around it. The other one, which will be called 'type B', does not have diagonal sensors, so it only has 5 squares as the input (a cross with itself in the center).

The developed transfer algorithm is based on the idea of grouping state-action values using the common parts of predicate descriptions or names of the corresponding semiotic state descriptions. In the relatively simple case considered in this work, they allow matching between different state descriptions of agents by determining which subset of states of one agent corresponds to a single state description of the other agent. Let's consider a world description for an agent of type A shown in "Fig. 1". For simplicity, statements about sensor data such as $"agent1"$ has "$S1$"; "$S1$" has $("0")$ will be shortened to $S1("0")$. The agent sign connects an agent with the name $"agent1"$ to its sensor objects in *p*, expresses rules that determine when a certain action can be performed in *m* and can store its experience as transition rules in *a*.

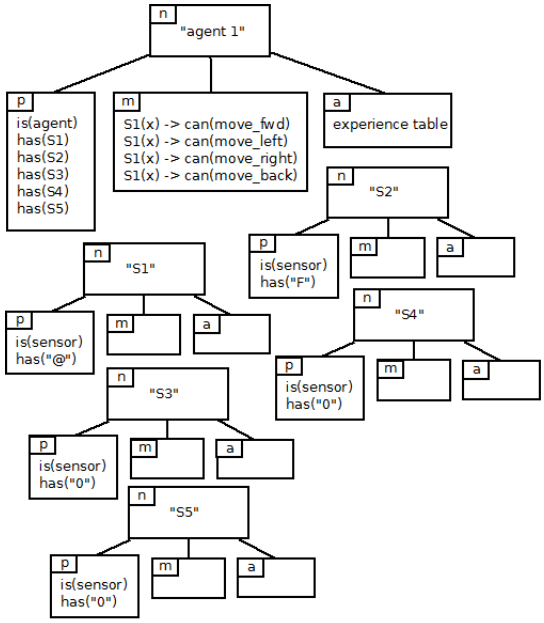


Figure 1: Sign-based description of a world model for an agent with 5 sensors that can each show an object denoted by "@" (name of a food item), "0" (name of an empty cell), etc.

If another agent ("agent 2" of type A) had a similar description, but with more sensors (with names "S1", "S2", ..., "S9"), so that "agent 1" has a subset of sensors of "agent 2", then any state of "agent 2" of type A is a part of the subset of a state of "agent 1", which can be matched by names of the sensors. More complex descriptions of states in an agent based on semiotic descriptions can involve multiple signs with intersecting predicate descriptions, but those cases are not considered in this work and are a subject of further research.

The $\epsilon$-greedy Q-learning algorithm [9] is used for individual learning of the agents, while the transfer function is custom as described further, so the individual learning step updates the state-action value function according to the formula

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + Q(s',a') - Q(s,a)), \tag{2}$$

Action selection for the Q-learning algorithm in this work is done in two different ways, continuous using the softmax function

$$P(a') = \frac{e^{Q(s',a')}}{\sum_{i=0}^{num\_actions} e^{Q(s',a_i)}}, \tag{3}$$

and discontinuous using the argmax function

$$a' = argmax_{a_i} Q(s', a_i), \tag{4}$$

After a mapping between different agent state spaces has been established, a way to transfer the state-action value function needs to be defined. In this case a simple averaging function is used for the corresponding subset. So, the state-action value function of the source agent $Q_s(s_s, a)$ is transferred to the initial state-action value function of the target agent by the formula:

$$Q_t(s_t, a) = \frac{1}{n_t} \sum_{s_s \subseteq s_t} Q_s(s_s, a), \tag{5}$$

where $n_t$ is the number of nonzero $Q_s(s_s, a)$ such that $s_s \subseteq s_t$.

However, it does not take into account the probability of the agent to end up in the corresponding state. A useful estimate may be to track the number of visits $c(s, a)$ to the corresponding states by the source agent and use a weighted average for the target agent with the rate of visits as the weight. In this case, the initialization is performed as:

$$Q_t(s_t, a) = \frac{\sum_{s_s \subseteq s_t} c(s_s, a) Q_s(s_s, a)}{\sum_{s_s \subseteq s_t} c(s_s, a)}. \tag{6}$$

## III. Experimental setup

Foraging problem was chosen to evaluate the algorithms. The environment is a grid in which an agent can move left, right, up and down (or remain in the same place). The steps and actions are discrete. Each cell of the grid can contain a number of objects or be empty. The objects are: the agent, a food item, an obstacle. A reward (1 point) is received by the agent when it picks up a food item. Food items are automatically picked up when the agent is in the same cell as a food item and it only gathers one food item at a time. Food items are initially placed randomly in cells without obstacles and are instantly created in a random place after being picked up. The environment is reset and generated randomly each experiment. The size of the grid world is 15 by 15 cells with 100 units of food ("Fig. 2").

The agent can either have a problem-space input or an agent-space input for the problem. The problem-space input is the coordinates of the agent in the grid. This description of the state is a full observation of the state of the environment. The agent-space descriptions, on the other hand, are partial observations and consist of the descriptions of the local surroundings of the agent. As mentioned in algorithm descriptions, two agent-space descriptions were used. Type A agent receives a description the nearby 9 cells (its own included) with a list of objects in each of them ("Fig. 3a"). Type B agent only receives predicates about the up, down, left, right and its own cell ("Fig. 3b"). The example in (a) corresponds to a tuple of length 9 $[@, F, 0, 0, 0, \#, F, 0, 0]$, while the (b) highlights

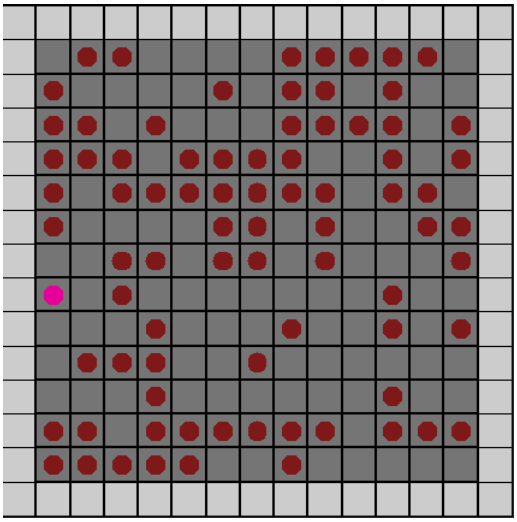

Figure 2: Simulation of the foraging problem environment. The light squares around the field are the obstacles, the bright circle is the agent, other circles are food items.

type B agent's cells, which are the center, left, right, up and down cells of a state $[@, F, 0, 0, 0]$.

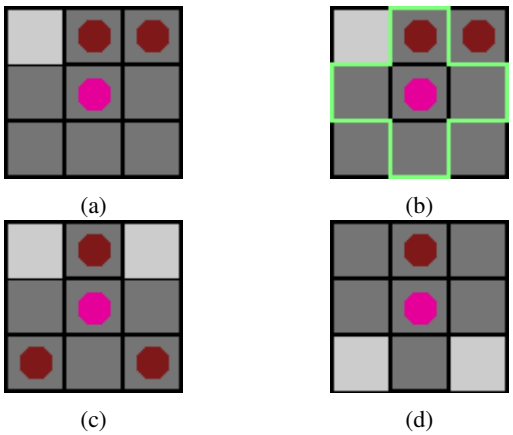


Figure 3: Examples of agent-space states, where dark red circles are food items, bright pink circle in the center is the agent, light grey squares are cells with obstacles and dark gray are empty cells. (a) shows an example of an agent-space state for an agent of type A. (b) has the 5 elements of type B agent's state highlighted. (a), (c) and (d) are examples of agent-space states of and agent of type A that are considered elements of the subset of states generated by the agent-space state of (b).

The transfer task is defined as a source agent of type A and the target agent of type B acting in this environment. First, the source agent learns for a fixed amount of steps. Then, its experience is transferred to the target agent of type B by transforming the state-action value table according to the function defined in the algorithms section and initializing the table of the

agent of type B with it. That agent then acts and learns for the same amount of steps. The results were averaged over a number of experiments (10 or more) and grouped into 'episodes' (the task itself is continuous, not episodic) consisting of several steps (1000 unless noted otherwise) for a more readable plot.

To define the problem formally, the corresponding spaces and variables must be written out. The state of the agent in agent-space is a tuple of objects seen by the corresponding sensors, for example, $[@, F, 0, 0, 0]$ for type B agent means that there is only the agent itself on the central cell, a food item in front of it and nothing on all other sides. Since only the 'topmost' object is recorded by the sensor and there are a limited amount of objects, this state *s* is an element of a finite set of possible states $S_B$. Similarly, for the agent-space of the agent if type A, the tuple looks like $[@, F, 0, 0, 0, \#, F, 0, 0]$, where @ denotes the agent, $F$ is a food item, $\#$ is an obstacle, 0 is an empty cell, and defines a finite set $S_A$. It can be noted that a state $s_b \in S_B$ forms a template for a set of states $s_a \in S_A$, where the corresponding elements of the tuple are the same. When a state $s_a \in S_A$ is from the corresponding set generated by $s_b$, it is denoted in this work as $s_b \subseteq s_a$ for simplicity. For example, the state of an agent of type B in "Fig. 3b" generates a set of states for an agent of type A, where the corner elements can have any object in them, like in "Fig. 3c" or "Fig. 3d".

The possible actions are always the same for all types of agents, available in any state (moving into an obstacle just does not change the current state) and simply correspond to a number in the set $A = 0, 1, 2, 3, 4$. Thus, the state-action pairs sets $S_A \times A$ and $S_B \times A$ are also finite and their value estimates can be expressed as a state-action table $\{Q(s, a) | s \in S, a \in A\}$ for the Q-learning algorithm.

The reward function $R : S \times A \times S \mapsto \mathbb{R}$ is deterministic both in agent-space and in problem-space and does not need to include a probability distribution. The reward $r(s, a, s')$, where $s$ is the previous state, $a$ is the chosen action, $s'$ is the state after the transition, equals 1 when $s'$ includes a food item in the agent's cell (first element of the state tuple) and 0 otherwise.

According to the goal of maximizing the agent's food-gathering efficiency, the goal function to maximize is the discounted return from the current time *t*:

$$G_t = \sum_{k=0}^{\inf} \gamma^k R_{t+k+1} \tag{7}$$

The learning algorithm used for the agent's individual learning is $\epsilon$-greedy Q-learning with the following parameters:

- random action chance $\epsilon = 0.05$;
- learning rate $\alpha = 0.1$;
- reward discounting factor $\gamma = 0.9$;
- initial state-action table is generated as a uniform distribution U(A) for each state (in target algorithm a transfer function is used to set it).

The corresponding policies that govern the agent's choice of actions are the $\epsilon$-greedy and softmax Q-learning alternatives respectively:

$$\pi(s, a) = \frac{\epsilon}{num\_actions} + (1 - \epsilon)\mathbb{1}_{argmax_{a_i} Q(s', a_i)} \tag{8}$$

$$\pi(s, a) = \frac{e^{Q(s', a')}}{\sum_{i=0}^{num\_actions} e^{Q(s', a_i)}} \tag{9}$$

For comparison, during the evaluation of performance of the algorithms, the performance of the agent of type B without transfer is also plotted so that the effects of experience transfer can be compared directly to the same type of agent.

Computational experiments were carried out in a custom Python simulation system.

## IV. Results and Discussion

The transfer algorithm based on averaging of state-action values grouped by common parts of predicate descriptions for the source agent of type A and the target agent of type B had the learning curve in "Fig. 4" for argmax action selection and "Fig. 5" for softmax action selection.


Figure 4: The asymptotically best performing algorithm of these three is the source algorithm which is in full agent-space (type A), its experience is transferred to the target algorithm with limited agent-space (type B) which shows a better starting efficiency and the worst performing algorithm is a type B agent without transferred experience that is a baseline for comparison. The source and target algorithms use argmax for action selection.

The weighted average alternative did not show any improvement over the base algorithm "Fig. 6"

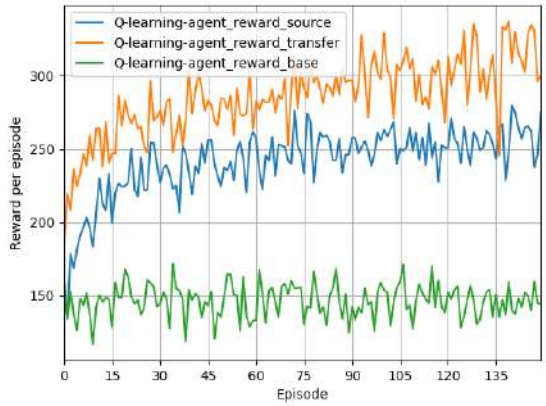


Figure 5: The full agent-space agent (type A) acts as a source and has its experience is transferred to the target algorithm with limited agent-space (type B) which shows a better starting efficiency and the worst performing algorithm is the baseline agent. The source and target algorithms use softmax for action selection.

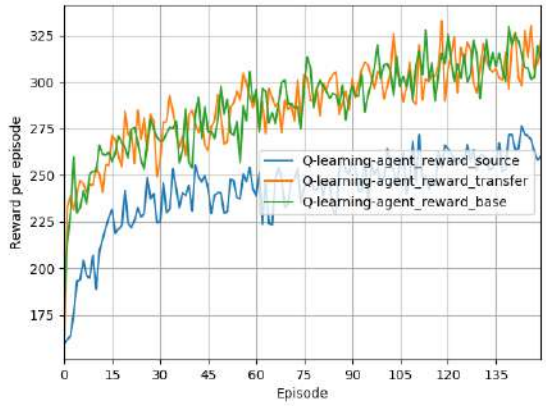


Figure 6: The weighted averaging algorithm alternative. The source agent with full agent-space (type A) here did not learn its optimal performance and the subsequent transfer to the agent of type B shows no improvement over the base non-transferred agent. The source and target algorithms use softmax for action selection.

The transferred experience immediately improves the efficiency of the agent in some cases (in the argmax experiment) while the learning speed is mostly improved in the softmax experiment. These results may indicate that continuous action selection benefits more from useful hints for better learning and exploration, while non-continuous argmax selection can get stuck on the initial strategy for some time before adjusting the estimations enough and thus needs a good initial strategy. For the foraging environment with agent-space input the full state is only partially observable, which favors probabilistic (softmax) strategies over deterministic (argmax). Both the target agent and the untrained type B agent eventually learn policies with similar efficiency. The alternative weighted algorithm may have overestimated common but not beneficial situations and thus not given any improvement over an untrained agent. It is also important to note that the whole batch of transfer experiments were done from a single pretrained agent and thus may have a strong dependency on the input pretrained model. Instead, initialization from an aggregated model (for example, averaged action-value tables over several such agents) may be able to provide more stable results, but this was not explored here.

## V. Conclusion

Several algorithms were studied for transfer learning problem for a foraging task. These algorithms can be used for special cases of agents with a semiotic control description of world models. The results show that they provide improvement in the learning curve for the target agent after transferring experience from an agent with more sensors (in the form of input predicates), but this improvement is inconsistent and sometimes does not bring noticeable difference in performance.

The most noticeable limitation of the method is the requirement of the target agent's description of the task to be a subset of the source agent's description. This constraint may be relaxed by making better use of the connections between signs within the agents' world models, and applying logical inference, which is a subject of further research.

## Acknowledgment

This work was supported in part by grant RFBR 18-37-00498 mol_a.

## References

[1] Bianchi, R. A. C., Celiberto, L. A., Santos, P. E., Matsuura, J. P., & Lopez De Mantaras, R. "Transferring knowledge as heuristics in reinforcement learning: A case-based approach," Artificial Intelligence, vol. 226, pp. 102—121. 2015. DOI: 10.1016/j.artint.2015.05.008.

[2] Kernbach, S., Nepomnyashchikh, V. a., Kancheva, T. "Specialization and generalization of robot behaviour in swarm energy foraging," Mathematical and Computer Modelling of Dynamical Systems, vol. 18, no. 1, pp. 131—152. 2012. DOI: 10.1080/13873954.2011.601421.

[3] Konidaris, G., & Barto, A. "Building portable options: Skill transfer in reinforcement learning," IJCAI International Joint Conference on Artificial Intelligence, pp. 895—900. 2007.

[4] Lazaric, A. "Transfer in Reinforcement Learning: A Framework and a Survey," Reinforcement Learning. Adaptation, Learning, and Optimization, vol. 12, pp. 143—173. 2012. DOI: 10.1007/978-3-642-27645-3_5.

[5] Lazaric, A., Restelli, M., & Bonarini, A. "ransfer of samples in batch reinforcement learning," Proceedings of the 25th International Conference on Machine Learning, pp. 544—551. 2008. DOI: 10.1145/1390156.1390225.

[6] Osipov, G. S., Panov, A. I., Chudova, N. V., & Kuznecova, J. M. “Znakovaja kartina mira sub’ekta povedenija (Semiotic view of world for behavior subject),” Artificial Intelligence, vol. 226, p. 261. 2017.
[7] Parisotto, E., Ba, J., & Salakhutdinov, R. “Actor-mimic deep multitask and transfer reinforcement learning,” 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, pp. 1—16. 2016.
[8] Rovbo, M. A., & Sorokoumov, P. S. “Control system architecture of an intelligent agent based on a semiotic network,” Open Education, vol. 22, no. 5, pp. 84—93. 2018. DOI: 10.21686/1818-4243-2018-5-84-93.
[9] Sutton, R. S., & Barto, A. G. “Reinforcement learning: an introduction,” UCL,Computer Science Department, Reinforcement Learning Lectures. Cambridge, MA: The MIT Press. p. 552. 2018. DOI: 10.1109/TNN.1998.712192.
[10] Taylor, M. E., Stone, P., & Liu, Y. “Transfer learning via inter-task mappings for temporal difference learning,” Journal of Machine Learning Research, vol. 8, pp. 2125—2167. 2007.
[11] Yogeswaran, M., & Ponnambalam, S. G. “Reinforcement learning: Exploration-exploitation dilemma in multi-agent foraging task,” Opsearch, vol. 49, no. 3, pp. 223—236. 2012. DOI: 10.1007/s12597-012-0077-2.

# ПЕРЕДАЧА ОПЫТА В ЗАДАЧЕ ОБУЧЕНИЯ В ПРОСТРАНСТВЕ АГЕНТА ДЛЯ ЗНАКОВЫХ МОДЕЛЕЙ МИРА

Ровбо М.А.

Исследовано несколько алгоритмов для задачи передачи опыта при обучении на примере задачи фуражировки. Эти алгоритмы подходят для семиотической системы управления, чтобы облегчить передачу опыта между двумя обучающимися агентами с различными описаниями состояния в пространстве агента в форме предикатов. Они требуют, чтобы описание задачи целевого агента было подмножеством описания исходного агента.

Алгоритмы основаны на алгоритме Q-обучения и используют специальную функцию передачи опыта, инициализирующую таблицу состояний-действий целевого агента, которая сопоставляет соответствующие предикаты. Тестовая задача, используемая для оценки алгоритмов, является задачей фуражировки, когда агент должен собирать случайно сгенерированную пищу в мире-сетке.

Результаты показывают, что они обеспечивают улучшение кривой обучения для целевого агента после передачи опыта от агента с большим количеством входных предикатов. Улучшение непостоянно и иногда не приносит заметной разницы в производительности, но целевой агент работает по крайней мере так же хорошо, как агент без предшествующего опыта.

Received 30.01.2020