computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

INFLUENCE LEARNING FOR MULTI-AGENT SYSTEM BASED ON REINFORCEMENT LEARNING

Anton Kabysh [1], Vladimir Golovko [1], Arunas Lipnickas [2]

[1] Brest State Technical University, 224017, Moskovskaya str. 267, Brest, Republic of Belarus
anton.kabysh@gmail.com, gva@bstu.by
[2] Kaunas University of Technology, Student g. 48-111, LT-51367, Kaunas, Lithuania
arunas.lipnickas@ktu.lt

Abstract: *This paper describes a multi-agent influence learning approach and reinforcement learning adaptation to it. This learning technique is used for distributed, adaptive and self-organizing control in multi-agent system. This technique is quite simple and uses agent's influences to estimate learning error between them. The best influences are rewarded via reinforcement learning which is a well-proven learning technique. It is shown that this learning rule supports positive-reward interactions between agents and does not require any additional information than standard reinforcement learning algorithm. This technique produces optimal behavior of multi-agent system with fast convergence patterns.*

Keywords: *reinforcement learning, influence learning, multi-agent learning, multi-joined robot.*

1. INTRODUCTION

Multi-Agent approach – it is an entire paradigm in the development of complex systems consisting of interacting autonomous agents, which are operating with local knowledge and limited capacity, however, can be expected, in general, the behavior of the system. No agent lives in a vacuum, but typically must interact with other agents to achieve its goals. Distributed artificial intelligence is the subfield of artificial intelligence that focuses on complex systems that are inhabited by multiple agents. The main goal of research in this area is to provide principles for the construction and application of multi-agent systems as well as means for coordinating the behavior of multiple independent agents [1].

For example, many application domains are envisioned in which teams of software agents or robots learn to cooperate amongst each other and with human beings to achieve global objectives. Interacting agents may also be essential in many non-cooperative domains such as economics and finance. Teams of agents have the potential for accomplishing tasks that are beyond the capabilities of a single agent. An excellent and demanding example of multi-agent cooperation is in robot soccer. At the same time, Multi-Agent learning (MAL) poses significant theoretical challenges, particularly in understanding how agents can learn and adapt in the presence of other agents that are simultaneously learning and adapting [2,3].

In general multi-agent systems each agent possesses its own, arbitrary reward function which may be entirely unrelated to the rewards other agents receive. For this realm of problems a number of corresponding multi-agent reinforcement learning algorithms have been suggested [4, 5, 7, 8]. The *Nash-Q* learning algorithm shown multi-agent system as general-sum games where agents tend to find Nash equilibrium in common environment. This algorithm assumed an environment where each agent has full knowledge over the actions taken by other agents. Aiming at the reduction of necessary preconditions to be met for this algorithm to be applicable, Littman proposed the *Friend-or-Foe Q* learning algorithm, which, however, requires every other agent to be classified as cooperative or competitive a priori, and Greenwald and Hall developed *Correlated-Q-learning* which generalizes Nash-Q and FoF-Q.

Multi-agent systems with cooperative agents which collectively aim at achieving a common goal is to be found very frequently in practical applications and, thus, of special importance. Some practical usages include [4]:

- Mobile telephony (e.g. for channel allocation)

- Network technology (e.g. for data packet routing)
- Computing power management (e.g. for load balancing across servers)
- Autonomous robots (e.g. for robotic soccer) and distributed robotics.
- Job-shop scheduling [4].

This numeration of mostly real-life applications misses another important area the field of production planning and factory optimization, where machines may act cooperatively with the goal of achieving maximal joint productivity.

The universal multi-joined robot learning problem via multi-agent learning described in this paper. The section 2 describes reinforcement learning approach and section 3 influence learning framework to multi-agent learning. Section 4 describes and eplains particular methodology of adaptation reinforcement learning to multi-agent influence learning. Next sections, 5 and 6 describes model of universal multi-joined and learning experiments. Conclusion contains perspectives, prospects and limitation of presented learning framework.

2. REINFORCEMENT LEARNING

Reinforcement learning is an approach to artificial intelligence that emphasizes learning by the individual from its interaction with its environment that produces optimal behavior [4]. It is often used as one of the control techniques, especially for learning autonomous agents in unknown environment. It emerged at the intersection of dynamic programming, machine learning, biology, studies the reflexes and reactions of living organisms: reflex theory, animal cognition [5]. RL is highly popular for learning autonomous agents, for example autonomous robotics, negotiating agents and so on. The math foundation of RL is Markov Decision Process (MDP), so it widely used for learning in game theory, e.g. TD-Gammon [6].

The core of all most Reinforcement Learning methods is a Temporal Difference (TD) learning [4]. Temporal Difference technique measures the inconsistency between difference of quality for two *actions* done in some *state* and received reward, shows expectation of agent.

Agent execute action *a* in particular state *s*, goes to next state *s'* and receives reward *r* as a feedback of recent action. During learning agent try to select the best action in some state (best action usually more rewarded in future). Visually, iteration of RL-agent on MDP is shown at Fig. 1.

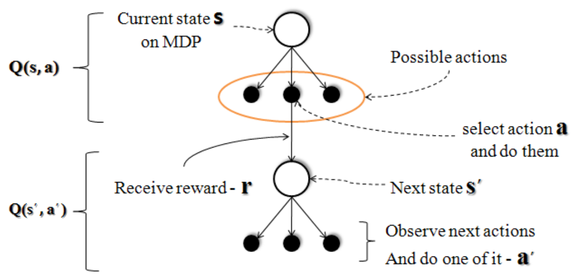

Fig. 1 – One iteration of Reinforcement learning

where α – learning rate, γ - discount factor.

Learning goal is to approximate *Q*-function, e.g. finding true *Q*-values of *Q*-function for each *action* in every *state*. Formulas 1,2 shows *SARSA* learning rule. Estimated value δ is *Temporal Difference error*.

$$\delta = r + Q(s',a') - Q(s,a) \qquad (1)$$
$$\Delta Q(s,a) = \alpha\delta \qquad (2)$$

The natural extension of standard RL algorithm is usage *eligibility traces* to remember previously visited states. Eligibility trace is a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action [4]. At every time step, when a TD error occurs, only the eligible states or actions are updated.

$$\Delta Q(s,a) = \alpha[r + Q(s',a') - Q(s,a)]e(s) \qquad (3)$$
$$e(s) = \begin{cases} \lambda\gamma e(s) & if\ s \neq s_t \\ \lambda\gamma e(s) + 1 & otherwise \end{cases} \qquad (4)$$

Formula (2) called for every previously visited state *s* if , where $e(s)$ – is a eligibility value, λ – is a eligibility discount factor, decay in time past eligibilities.

Reinforcement Learning works well in case of one agent and where external environment can be represented compactly via *Q*-function. In a multi-agent environment with multiple cooperative agents in general it is difficult to define appropriate state-action spaces for all interacting agents – the *course of dimensionality* problem. For example, in multi-agent system with *N* agents RL algorithm should search in *N*-dimensional state-space. Most often the tiling of the state space has to be rather fine to over all possibly relevant situations and there can also be a wide variety of actions to choose from. As a consequence there exists a combinatorial explosion problem when trying to explore all possible actions from all possible states. Solutions to this problem apply scale-spacing methods and/or function approximation methods to reduce and/or interpolate the searchable value-space.

To solve these problems in this work we use *influence learning framework* to define formal model of agent's interaction.

3. INFLUENCE LEARNING

Multi-Agent learning (MAL) poses significant theoretical challenges, particularly in understanding how agents can learn and adapt in the presence of other agents that are simultaneously learning and adapting [10]. In a distributed system, a number of individually acting agents coexist. If they strive to accomplish a common goal, i.e. if the multi-agent system is a cooperative one, then the establishment of coordinated cooperation between the agents is of utmost importance [2].

Influence – is active offer from one agent to another which can be accepted or not. If influence is accepted, than it affects the agent's behavior and (or) translates it into a new state. If influence is not accepted, than it can be returned to the sender with negative mark. Moreover if influence is accepted by receiver than it also can send a feedback showing how influence was good.

The main idea of influence learning is that good influences should be rewarded and negative should be excluded. Agents should cooperate to achieve the common goal finding in adaptive process best influences to each other. Only way to work together determines whether or not achieved the goal and only final goal determine what influences is valid from the others. Therefore, the adaptive learning process should be used to find best influences and their sequence that are correct for achieving the goal. In this paper we used reinforcement learning for these purposes, but this is not strict.

So, the influence learning – is simple and agile learning framework for cooperating agents in combination with an adaptive algorithm to find the best influences. In other words, influence learning is a *learning to coordinate behaviors*. The advantage of this approach is its universality; any multi-agent systems can be easily described in terms of influences. The limitation of this approach is that the agent should have a connection with each other. In any case, if the connection does not exist, then influence learning is simply reduced to the standard techniques of single agent learning.

4. REINFORCEMENT LEARNING ADAPTATION TO INFLUENCE LEARNING

Let's see how the reinforcement learning can be adapted to influence learning. Typically, in reinforcement learning, agent estimate error temporal difference error between two states and learn. In multi-agent system actions from one agent may be directed to another agents and change their states. The actions in this case will be the influences. If agent state is changed via influence the received reward in new state is transferred to influencing agent. It can lean via reinforcement learning and in these way positive influences is selected among others.

Let's see to interconnected agents *A* and *B* in states s_a and s_b respectively. Agent *A* in state s_a execute action *a* over agent *B*, and put them it into new state s_b. Agent in new state *B* select action *b* and execute it somewhere (on another agent, or on environment). Actions *a* and *b* has their Q-*values* $Q(s_a, a)$ and $Q(s_b, b)$ respectively. This situation is shown at Fig. 2.

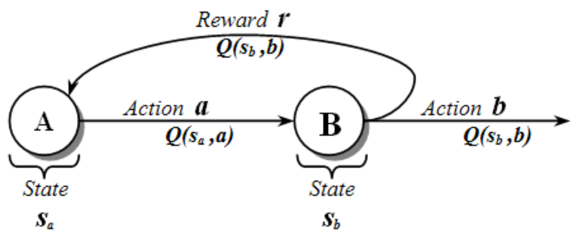

Fig. 2 – Influences between two agents. Reinforcement Learning view

Executing action and receiving reward agent *B* can send this reward as a learning feedback to *A*. This feedback include *Q-value* $Q(s_b, b)$ and reward value *r*. Receiving this feedback agent *A* can calculate *Temporal Difference* error for selected influence (action *a*) and learn. Learning process done using formulas (5, 6).

$$\delta_{AB} = r + \gamma Q(s_b, b) - Q(s_a, a) \qquad (5)$$

$$\Delta Q(s_a, a) = \alpha \delta_{AB} \qquad (6)$$

Formula (5) defines an *influence error* as a *temporal difference error* between agent *A* and *B*. Expression $r + \gamma Q(s_b, b)$ – is a feedback from agent *B*.

The most one important change in I-RL is that we suppose a $Q(s_b, b)$ - is a *"future"* Q-value of agent *A*.

In the end of learning, agent *A* can build the optimal influence selection policy for agent *B*. Feedback between agents included into update rule produces coherence of their behaviors. The advantage of this rule is simplicity and all single-agent RL algorithms can be used without serious changes. Easy to see that update rule is identical to standard reinforcement learning approach.

To support indirect interactions lets include eligibility traces into agent update rule. Introduce one more agent *C*. This situation is shown at Fig. 3.

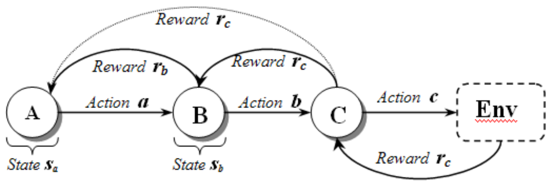

Fig. 3 – Influences between three agents. Reinforcemen learning view

As we can see at fig. 3, there are direct interaction between *A-B*, and *B-C*, and *indirect* interaction between *A* and *C*.

To support indirect interactions lets introduce *influence value* $i(d)$, where *d* is a *distance* between agents; This distance shows how *structurally* far away produced influence to this agent. For direct interactions *d* is equal to 0; for indirect interaction *d > 0*, depending on how many intermediate influences done between indirect agents. For direct interactions *A-B* and *B-C* the influence distance *d* equal to 1. For indirect interaction *A-C* influence distance is equal to 2, and so on.

The update rule changed in following way:

$$\delta_{AC} = r_c + \gamma Q(s_c, c) - Q(s_a, a) \qquad (7)$$

$$\Delta Q(s_a, a) = \alpha \delta_{AC} i(d) \qquad (8)$$

$$i(d) = \lambda^d \qquad (9)$$

where δ_{AC} - is a influence error between agent A and C, λ – is a coefficient of influence discount factor (0<λ<1).

Influence value $i(d)$ is depends from discount factor λ and reduced with increasing influence distance between agents. If $\lambda = 1$, then all influences will be updated with full power. If $\lambda = 0$ only direct interactions will be updated. If $0 < \lambda < 1$ then indirect interactions will be updated with decay.

5. MODEL OF MULTI-JOINED ROBOT

Multi-Joined Robot (MJR) learning task is a simple decentralized model, where simulate robot arm with N-degrees of freedom, where N – is a number agents in MAS. Every segment – is an intellectual agent learned via Reinforcement Learning with own *Q*-function. The goal of experiment is to learn MJR reach some target point. The goal can be achieved only if all actions were approval. Moreover, this model includes distributed and indirect interactions, which are also worth considering. This problem requires synchronization of local agent's.

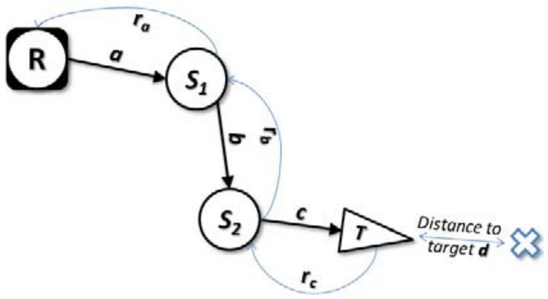

Fig. 4 – Multi-Joined Robot with 4 segments *R, S1, S2, T*. a,b,c - Agent actions. r_a, r_b, r_c – Feedback reward corresponds to actions

MJR contains one root segment ***R***, several intermediate segments ***S₁***, ***S₂***,..., ***Sₘ***, and one terminal segment ***T*** connected into chain from ***R*** to ***T*** (Fig. 6). Every segment, excluding terminal, can rotate at full circle (360^o) all next segments. At one time step each segment, excluding terminal, can rotate all next segments at 5^o to left or right, or do nothing.

First acts root segment ***R***, then first intermediate ***S₁***, then second ***S₂*** and so on, until ***Sₘ,***. Root segment can't move, can't be moved and don't change their position. Terminal segment verify reaching the target and receive actions from previous segments that change their own position.

Every segment – is an intellectual agent learned via reinforcement learning. Goal of multi-agent system is reaching a target grid. After learning MJR must reach by oneself any acceptable target cell of grid world.

Used next learning procedure (one training start):

1. MJR moved to initial position.
2. Every segment selects and executes action in order to structure of MJR. States of all next agents are changed.
3. Terminal segment calculate distance to target point.
4. If target is reached then MJR count grand-prix reward and learned. Go to 1.
5. Else, terminal segment produce feedback reward for previous agent to learn it. Feedbacks are propagated into MJR, so agents learn via RTD until root segment will be reached.
6. If simulation time is ended (1000 simulation steps) go to 1. If average RTD-Error (7) lover than limit value, then learning is over.
7. Next time step. Go to 2

6. EXPERIMENTAL RESULTS

RL parameters include: α (learning rate) = 0.05~0.1; γ (discount factor) = 0.7; λ (eligibility discount factor) = 0.7~0.99, d (influence discount factor) = 0.5~0.7.

In experiments we used MJR with 5 active segments. It is good compromise between model complexity and decentralization of agent actions.

The normal convergence process using *Q-Learning* update rule illustrated at fig 5.

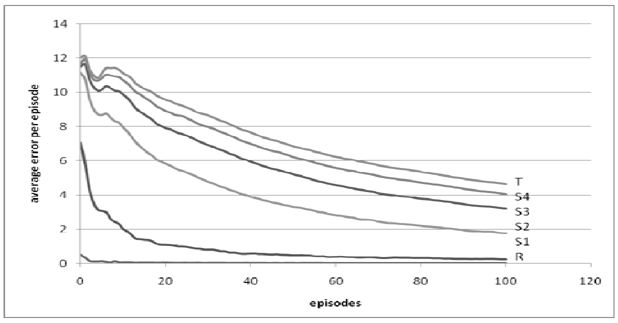

Fig. 5 – Influences between three agents. Reinforcemen learning view

As been shown, there are direct dependencies between agent's actions and errors. Every next segment can be placed at more states (has biggest state-action space), than others, and its error value become higher and convergence is longer.

Compare convergence speed with the overall multi-agent learning technique, *Joint-Action Learners* (JAL) which tries to train all agents at once. In I-RL approach in a single time step the system of learning expression $\{\Delta Q_R(s_r, a_R), \Delta Q_{S1}(s_{S1}, a_{S1}), ..., \Delta Q_{R4}(s_{S4}, a_{S4})\}$ is evaluated. In contrast, Joint-Action learning framework try to build the optimal policy overall multi-agent system approximating one joint Q-function $\Delta Q^*(s^* = \{s_R, s_{S1}, ..., s_{S4}\}, a^* = \{a_R, ..., a_{S4}\})$ composing state and action from all agents.

The advantage of I-RL is that it tries to approximate the number of small policies, one per agent instead of one complex. The fig. 6 shows the convergence result of I-RL in comparison with JAL.

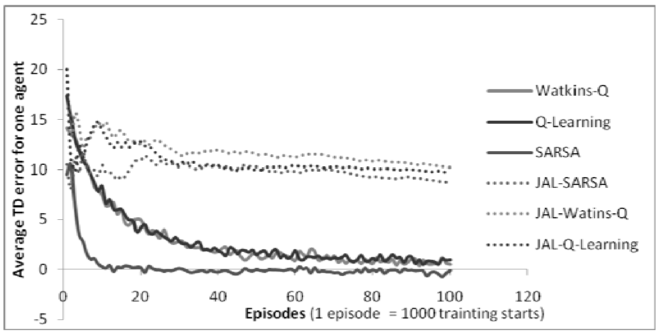

Fig. 6 – Average I-RL error for one agent per episode

Quality of convergence depends from number of segments. If MJR has more than 6 segments then probability of convergence is much lower. The main problem in this case is decentralization of agent actions. For example, if one agent in the beginning of MJR learns unsuccessfully (broken agent), then behavior of all next segments can becomes unstable, if they are can't compensate wrong action from broken agent. Every next agent should be sure that executed on it action is correct, in other words every selected action should guarantee convergence of learning and reaching the target. The described I-RL learning rule should be updated with these properties.

Behavior policy variously changed in way of use different algorithms. RL algorithms with influence tract (SARSA(λ), Watkins-*Q*(λ)) shown more smooth behavior and better synchronization than algorithms without it (*Q*-Learning).

Another unobvious result was seen in robot behavior. For algorithms with eligibility traces robot prefer rotation about a fixed root point with segment reconfiguration on new round to reach the target. Nevertheless, for *Q*-Learning (without eligibility traces) robot prefer reach the target in a straight way.

7. CONCLUSION

It is easy to see that I-RL converges much faster than JAL. This is an illustration of how *the course of dimensionality* affects the convergence of the algorithm. I-RL takes recommendations how to make decomposition of state-action space to much smaller subspaces, where only best influences are rewarded.

This approach helps to solve almost all the difficulties facing the multi-agent learning supportive without losing its simplicity. Also an advantage of this technique is that it can be applied to almost any problem of multi-agent reinforcement learning. The disadvantage of this work is that no comparison was made with more sophisticated multi-agent reinforcement learning algorithms, such us Correlated-Q.

I-RL has two known limitations. One is decentralization problem for long indirect influences, and second one is problem of agent influences insurance. The agent should be sure, that received influences as much optimal, as possible. It is a guarantee of I-RL algorithm convergence.

8. Acknowledgments

This work was supported by a Belorussian-Lithuanian grant F11LIT-003 from Belarusian Republican Foundation for Fundamental Research and Educational Grant GB11/121 from Ministry of Education of Republic of Belarus.

9. REFERENCES

[1] Vidal H.M., *Fundamentals of Multiagent Systems with Net Logo Examples*, E-book: www.multiagent.com, (2008).

[2] Panait L. and Luke S., Cooperative multi-agent learning: the state of the art, *Autonomous Agents and Multi-Agent Systems*, (11) 3 (2005), pp. 387-434.

[3] Alonso E., D'Inverno M., Kudenko D., Luck M., Noble J., *Learning in Multi-Agent Systems. Science Report, Discussion*. UK's Special Interest Group on Multi-Agent Systems, (2001).

[4] Gabel T., *Multi-Agent Reinforcement Learning Approaches for Distributed Job-Shop Scheduling Problems*, PhD Thesis, University of Osnabrueck, 2009.

[5] Bab A., Brafman R.I., Multi-agent reinforcement learning in common interest and fixed sum stochastic games: an experimental study, *Journal of Machine Learning Research*, (9) (2008), pp. 2635-2675.

[6] Richard S. Sutton, Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press., (1998).

[7] Worgotter F., Porr B., Temporal sequence learning, prediction and control – A review of different models and their relation to biological mechanisms, *Neural Computation*, (17) (2005), pp. 245-319.

[8] Tan Ming, Multiagent reinforcement learning. Independent vs cooperative agents, *Autonomous Agents and Multiagent Systems*, (10) 3 (2005), pp. 273-328.

[9] Shoham Y., Powers R., Grenager T., If multi-agent learning is the answer, what is the question, *Journal of Artificial Intelligence*, (2006).

[10] Stone P., Multiagent learning is not the answer. It is a question, *Artificial Intelligence*, (171) (2007), pp. 402-405.

[11] Kabysh A., Golovko V., Mikhniayeu A., Lipnickas A., Behaviour patterns of adaptive multi-joined robot learned by multi-agent influence reinforcement learning, *Proceedings of Pattern Recognition and Image Processing* (PRIP–2011), 2011, 19–21 May, BSU, Minsk, pp. 392-297.

__Anton Kabysh__ Starting Ph.D. "Learning algorithms of Intelligent Multi-Agent Systems". in Brest State Technical University at 2009. Member of teaching staff, Department of Intelligent Information Technologies, Brest State Technical University. Member of: Laboratory of Artificial Neural Networks; Laboratory of Robotics "BrSTU Robotics" www.robobics.bstu.by.

Areas of research interests: Multi-Agent Systems, Multi-Agent Learning, Reinforcement Learning, Intellectual Robotics.

__Prof. Vladimir Golovko,__ received M.E. degree in Computer Engineering in 1984 from the Moscow Bauman State Technical University. In 1990 he received PhD degree from the Belarus State University and in 2003 he received doctor science degree in Computer Science from the United Institute of Informatics problems national Academy of Sciences (Belarus). At present he work as head of Intelligence Information Technologies Department and Laboratory of Artificial Neural Networks of the Brest State Technical University.

His research interests include Artificial Intelligence, neural networks, autonomous learning robot, signal processing, chaotic processes, intrusion and epilepsy detection.

__Dr. Arunas Lipnickas,__ graduated in 1996 the faculty of Electrical Engineering and Control Systems, department of Applied Electronics, Kaunas University of Technology. In 1998 graduated with honours gaining the degree of Master of Science in Electrical Engineering at Kaunas University of Technology. In 2002 he obtained PhD degree in Informatics Engineering. At present is a Senior Assoc. Researcher, Mechatronics Centre for Studies and Research, Head of Laboratory of Robotics, Kaunas University of Technology.

Areas of research interests: artificial intelligence, neural networks, signal processing, decision support systems, fault diagnosis, robotics, mechatronics.