Behavior Patterns of adaptive Multi-Joined Robot learned by Multi-Agent Influence Reinforcement Learning

Anton Kabysh[1], Golovko Vladimir[2], Andrei Mikhniayeu[3], Uladzimir Rubanau[4], Arunas Lipnikas [5]

1,2,3) Brest State Technical University, anton.kabysh@gmail.com, gva@brsu.by, ancto@gmail.com

4) Kaunas University of Technology, arunas.lipnickas@ktu.lt.

Abstract: *This paper describes behavior patterns produced by Multi-Joined Robot learned via Influence Reinforcement learning. This learning technique 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. As will show, this learning rule supports positive-reward interactions between agents and does not require any additional information than standard reinforcement learning. The behavior patterns of learned robot shows that optimal behavior strategies differ for various learning techniques. As we will show, every algorithm produces his own behavior's patterns which are optimal for that learning rule to produce a faster convergence.*

Keywords: Multi-Agent Influence Reinforcement Learning, Eligibility Traces, Behavior Patterns.

1. INTRODUCTION

Machine learning being explored an important component in multi-agent systems (MAS). 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 [1]. Learning may also be essential in many non-cooperative domains such as economics and finance, where classical game-theoretic solutions are either infeasible or inappropriate. 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].

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 [5,6]. 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 Pprocess (MDP), so it widely used for learning in game theory, e.g. TD-Gammon [10].

The core of all most Reinforcement Learning methods is a Temporal Difference (TD) learning [4-9]. 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.

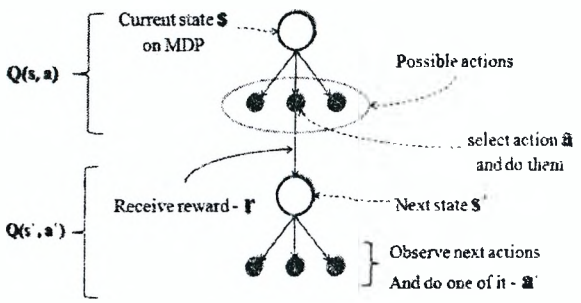

Fig. 1. One iteration of Reinforcement learning

Where 6 - learning rate, r - discount factor, determines the importance of future rewards.

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

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

The natural extension of standard RL algorithm is usage *eligibility traces* to remember previously visited states. Eligibility trace is a temporary records 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.

$$\Delta Q(s,a) = \alpha[r + Q(s',a') - Q(s,a)]e(s) \qquad (2)$$

$$e(s) = \begin{cases} \lambda \gamma e(s) & if\ s \neq s_t \\ \lambda \gamma e(s) + 1 & otherwise \end{cases} \qquad (3)$$

Formula (2) called for every previously visited state if $e(s) > 0$, 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. In case of multi-agents system Reinforcement Learning update rule should be updated for better convergence to reduce several limitations [2, 3, 4, 6]. It is exponentially growing state-action space depending on number of agents, curse of dimensionality, and decentralization of learning process.

3. MULTI AGENT REINFORCEMEN LEARNING

In many articles multi-agent reinforcement learning (MARL) has shown in context of game theory for founding Nash equilibrium point for group of agents. Terms 'coordination' and 'cooperation' used in Multi-Agent Systems is well known in game theory (e.g. zero sum stochastic games and common-payoff). There are several models of Game Theory approaches to multi-agent learning. Works [1,12,13] provided generalized view to this approach, and [14] pointed, that Multi-Agent learning is a different kind of machine learning and still open question. In robotics, multi-agent reinforcement learning has been applied to teams of robots with Q-learning as the algorithm of action choice.

In previous papers [15-17] we purpose *Influences Reinforcement Learning* as a simple reinforcement learning technique for distributed multi-agent system with local update rule and fast convergence.

Influences Reinforcement Learning (I-RL) based on assumption if some problem solved cooperatively by agents so their learning process is related to each other and influences between agents can be used to estimate error value between agents. In this case, actions from one agent may be directed to another agents and change their states.

Let's see to interconnected agents A and B. Agent A at state s_a execute action a over agent B, and set it into new state s_b. Agent in new state B select action b and execute it somewhere (on another agent, or on environment). This situation is shown at Fig. 2.

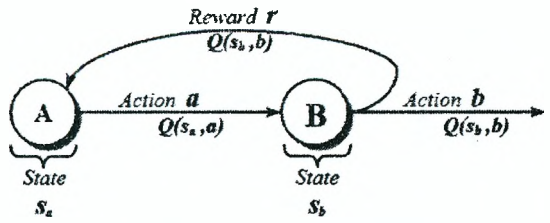

Fig. 2. Influences between two agents. Reinforcement learning view.

Actions a and b has their Q-*values* $Q(s_a, a)$ and $Q(s_b, b)$ respectively. After executing action and receiving reward agent B can sent feedback to A. This feedback include *Q-value* $Q(s_b, b)$ and reward r as a response to action a. Receiving this feedback agent A can calculate Temporal Difference error between them and agent B and learn using standard RL technique. Agent A update their Q-value $Q(s_a,a)$ corresponding to action a using formulas (4, 5).

$$\delta_{AB} = r + \gamma Q(s_b,b) - Q(s_a,a) \quad (4)$$

$$\Delta Q(s_a,a) = \alpha \delta_{AB} \quad (5)$$

Formula (4) defines a *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, and in this case (5) is equal to (1). Using this update rule, the best influences from A to B will be more rewarded. In the end, agent A can build the optimal interaction 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.

Including eligibility traces into agent interactions we can reduce decentralization of learning process and update coherent influences more than between two agents. In influence trace we store history (set) of agent influences to each other, as number of I-RL procedures.

For example, let's see to more complicated and distributed example from previous chapter. Introduce one more agent C. This situation is shown at Fig. 3.

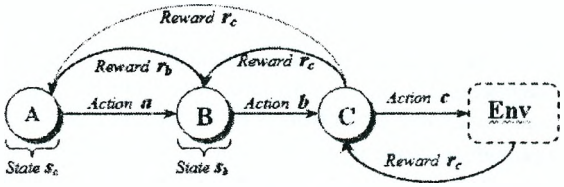

Fig. 3. Influences between two agents (common case). Reinforcement learning view.

Agents interact in following scenario:

1. Agent A acts to agent B with $Q(s_a, a)$. Agent B goes to state s_b.
2. Agent B acts to agent C with $Q(s_b, b)$. Agent C goes to state s_c.
3. Agent C acts with action c to environment *Env* and receive their reward.
4. Agent C *receive* reward r_c and send it to all agents who influence to it.
5. Agent's B receive reward r_c for b and learn. Also, agent B can calculate their own reward r_b ant feedback it to agent A.
6. Agent A receive all reward's and learn.

As we can see at fig. 3, there are direct interaction between A-B, and B-C, and indirect interaction between A and C. Using influence trace we can learn and update these indirect interactions.

For this purposes we introduce parameter *influence value* $i(d)$ into update rule, where d is a distance between agents. This parameter shows how far away *structurally* 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 agent's interactions A-B and B-C the influence distance d equal to 0. For indirect interaction A-C influence distance is equal to 1. If we introduce one more agent D after C, so influence distance between A-D will be 2, and so on. The update rule changed as follows:

$$\delta_{AC} = r_c + \gamma Q(s_c,c) - Q(s_a,a) \quad (6)$$

$$\Delta Q(s_a, a) = \alpha \delta_{AC} i(d) \quad (7)$$

$$i(d) = \lambda^{d-1} \quad (8)$$

Where δ_{AC} - is a influence error between agent A and C, λ - is a coefficient of influence discount factor.

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.

4. MODEL OF MULTI-JOINED ROBOT

Multi-Joined Robot (MJR) learning task is a simple decentralized model, which 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. The goal of experiment is to learn MJR reach some target point. This problem requires synchronization of local agent behaviors to achieve one common goal.

MJR contains one root segment ***R***, several intermediate segments S_1, S_2, ... , S_m, 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 S_1, then second S_2 and so on, until S_m. 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.

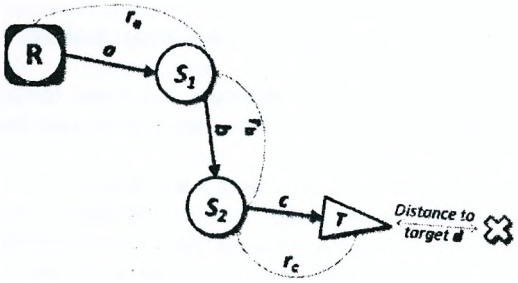

Fig. 6: Multi-Joined Robot with 4 segments *R*, S_1, S_2, *T*. a, b, c - Agent actions. r_a, r_b, r_c – Feedback reward corresponds to actions.

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

Reinforcement Learning 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

5.EXPERIMENTAL RESULTS

Earlier, behavior patterns in multi-agent systems were researched at [18]. Simulation of MJR behavior shows intension to find optimal structure of segments for the robot. During learning synchronization of behaviors between segments (successful learning) produces optimal interactions from which robot can reach the target.

In our experiments was used two basic reinforcement learning technique: *SARSA*, described above, and *Q-Learning*, using the following update formula:

$$\Delta Q(s,a) = \alpha[r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)] \quad (9)$$

One of extensions of *Q-Learning with eligibility traces* is known as *Wankins-Q*(л), where л – it is a eligibility discount factor. This learning rule was also used in experiments.

Convergence. In experiments we used MJR with 5 active segments. It is good compromise between model complexity and decentralization of agent actions. The case of MJR with more than 5 segments is described below. The normal convergence process using *Q-Learning* update rule illustrated at fig 7.

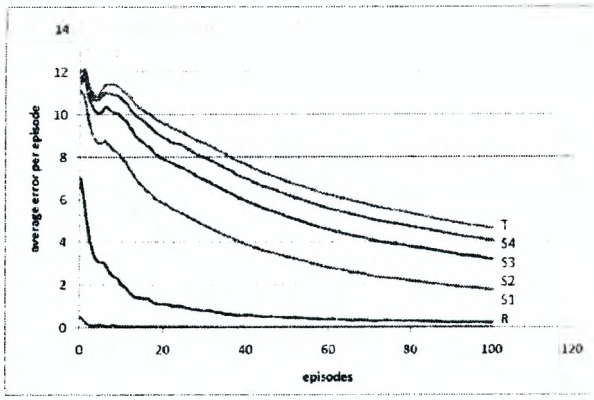

Fig. 7: Average I-RL error per episode for every segment.

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.

Learning decentralization. Quality of convergence depends from number of segments. If MJR have more than 6 segments then probability of convergence is much lower. The main problem in this case is decentralization of agent actions. Fig. 8-a shows situation where actions in the beginning of robot not synchronized with actions in the end of robot. The memory techniques, such as eligibility traces, are not enough for MJR containing more 7

step-by-step interactions between agents, and new techniques of learning for reducing complexity should be used.

Another problem is direct dependencies between agents. 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. Fig. 8-b illustrate situation, where wrong actions from first agent cause for others agents take a wrong action. 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, or another, more complicated and more researched rule should be used [19,20].

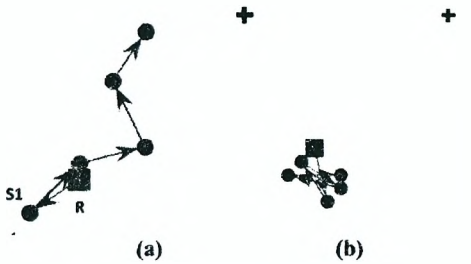

Fig. 8: Two examples where decentralization of actions occur. In the left example, actions of R and S1 segment are inconsistent with action from other segments. The right example shows grouping of agents.

Behavior patterns. Behavior policy variously changed in way of use different algorithms and learning parameters. I-RL algorithms with influence traces such as SARSA(л), Watkins-Q(л) shown more smooth behavior and better synchronization than algorithms without it Q-Learning.

The MJR learning contains with two parts. Fist part – it is a learning of optimal behavior. In this part robot explore the world trying to create (learn) best behavior policy. The second part uses found best behavior policy to finding optimal structure of MJR. Using this structure, MJR should reach target in shortest way with minimum number of actions. Usually, the better strategy is rotating with minimum segments reconfigurations. In this case only one segment does actions.

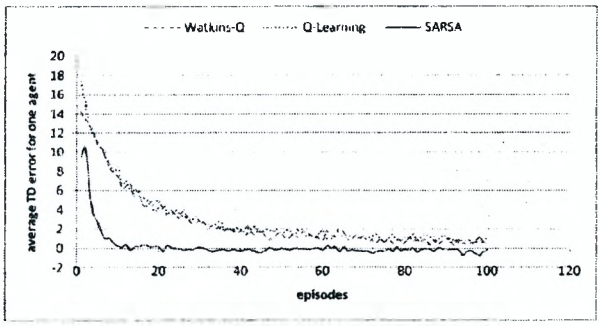

Fig. 9: SARSA, Q-Learning and Watkins-Q(л) I-RL convergence comparison.

SARSA patterns. This algorithm show fast convergence but worse synchronization properties. SARSA(л) with eligibility traces show more smooth behavior in coordination of agent actions. After learning, the following structural/behavior patterns were observed:

1. *Direct.* MJR try to reach target in a straight way. MJR structure reconfigured on-line, agent's actions is compensating each-other in goal to reaching target. Behavior may look as chaotic. Number of actions: High. Illustrated on Fig. 10-a.
2. *Partial-Rotation.* MJR prefer to rotate some part of segments with segment reconfiguration to reach the target. Behavior looks partially synchronized. Number of actions: High-medium. Illustrated on Fig. 10-b.

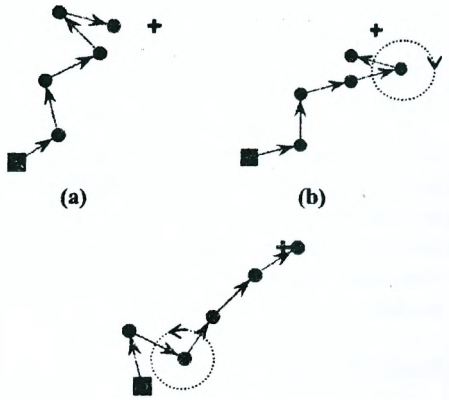

Fig. 10: SARSA/Q-Learning structural/behavior patterns.

Q-Learning patterns. This algorithm show longest, but stable convergence with better synchronization and behavior properties. Q-Learning has better approximation of Q-Function which can produce better actions for agent. The observed patterns are:

1. *Partial-rotation.* The same behavior with partial synchronization. Number of actions: High-medium. Illustrated on Fig. 10-c.
2. *Rotation with fixed points.* An unobvious result was seen in robot behavior at second stage, where learning process was freezed, the learned behavior policy was used to select robot actions. In this stage MJR select a several fixed points for its segments and rotate *without reconfigurations* reaching target every time, at every rotation. This fixed point can be shown as points producing optimal behavior with rotation. Rotation will guarantee, that target will be reached without structural reconfigurations. Behavior looks absolutely synchronized. Number of actions: minimal. Examples of this pattern shown at Fig 11.

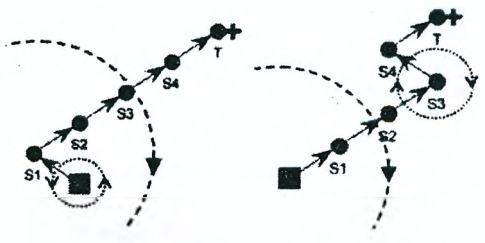

Fig. 11: Rotation behavior strategy with two fixed points.

6. CONCLUSION

This work suggests new approaches to Multi-Agent Reinforcement Learning named Influence Reinforcement Learning. This technique was designed to change standard Reinforcement Learning model in a best essential way to Multi-Agent Learning. Using I-RL we can apply RL model between agents locally.

This technique was used to produce observed behavior patterns during learning multi-joined robot. This patterns was classified as direct reaching, partial rotation and synchronization and rotation around fixed points. Fixed point rotation it is an optimal pattern with minimum number of required actions to reach the target. The state-action space should be explored by learning algorithm to find an optimal fixed point, to build optimal MJR structure.

The I-RL learning 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. The future experiments and extension if I-RL is needed.

7. REFERENCES

[1] Vidal, Hose M. *Fundamentals of Multiagent Systems with Net Logo Examples.* www.multiagent.com, 2009.

[2] Liviu Panait, Sean Luke. "Cooperative Multiagent Learning: The State of Art." *Autonomous Agents and Multiagent Systems, Volume 11*, 2005 : 387-434.

[3] Eduardo Alonso, Mark D'Inverno, Daniel Kudenko, Michael Luck, Jason Noble. "Learning in Multi-Agent Systems." *Science Report, Discussion.* UK's Special Interest Group on Multi-Agent Systems, 2001.

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

[5] Worgotter, F. and Porr, B. "Temporal sequence learning, prediction and control - A review of different models and their relation to biological mechanisms." *Neural Computation, Volume 17*, 2005: 245-319.

[6] Dr. Florentin Woergoetter, Dr. Bernd Porr. "Reinforcement Learning ." *http://www.scholarpedia.org.* 2008. http://www.scholarpedia.org/article/Reinforcement_learning.

[7] Sutton, Richard S. "Learning to Predict by the Methods of Temporal Differences." *Machine Learning, 3*, 1988: 9-44.

[8] Barto, Dr. Andrew G. " Temporal difference learning." *Scholarpedia.org.* 2007. http://www.scholarpedia.org/article/Temporal_difference_learning.

[9] Peter Dayan, Terrence J Sejnowski. "TD(lamda) Converges with Probability 1." 1994.

[10] Tesauro, G. J. "TD-gammon, a self-teaching backgam-mon program, achieves master-level play. ." *Neural Computa-tion, 6(2), (http://www.research.ibm.com/massive/tdl.html)* , 1994: 215-219.

[11] 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: 2635-2675.

[12] Tan, Ming. "Multiagent Reinforcement Learning. Independent vs Cooperative Agents." *Autonomous Agents and Multiagent Systems, v.10 n.3*, 2005: 273-328.

[13] Yoav Shoham, Rob Powers, Trond Grenager. "If multi-agent learning is the answer, what is the question." *Journal of Artificial Intelligence*, 2006.

[14] Stone., Peter. "Multiagent learning is not the answer. It is a question." *Artificial Intelligence, 171*, May 2007 : 402 – 405.

[15] Кабыш А.С., Головко В.А. *Принципы коллективного подкрепляющего обучения* / Конференция Нейроинформатика-2010. // РФ, г. Москва, МИФИ 25-29 января 2010 г. Сборник научных трудов, часть 1, стр. 199-208 (1-е место в конкурсе)

[16] Kabysh, V. Golovko. *Supporting Coherence in Multiagent System: Relational Temporal Difference with Influence Trace.* // The 5th International Conference on Neural Network and Artificial Intelligence (ICNNAI'2010) // June 1 – 4, 2010, Brest State Technical University, Brest, Belarus.

[17] Kabysh A.S. *Coherence Behaviour in Multi-Agent Systems based on Reinforcement Learning* // XI International PhD Workshop – OWD 2010, 23-26 October 2009, Wisla, Poland, p. 453-459

[18] Kabysh A., Golovko V. "Proceedings of the Tenth International Conference "Pattern Recognition and Image Processing" PRIP2009." *Collective Behavior in Multiagent Systems Based on Reinforcement Learning.* Minsk, Republic of Belarus, 2009.

[19] Littman, M. L. (2001). Firend-or-foe Q-learning in general-sum games. *18-tn Conference on Machine Learning* , pages 322 – 328.

[20] Greenwald A., H. K. (2003). Correlated-Q learning. *In Proceedings of the Twentieth International Conference on Machine Learning* , pages 242-249.