Proceedings of the Federated Conference on Computer Science and Information Systems pp. 139–143
DOI: 10.15439/2018F236
ISSN 2300-5963 ACSIS, Vol. 15

Adaptive Supervisor: Method of Reinforcement Learning Fault Elimination by Application of Supervised Learning

Mateusz Krzysztoń
Research and Academic Computer Network (NASK)
ul. Kolska 12, 01-045 Warsaw, Poland
Email: mateusz.krzyszton@nask.pl

***Abstract*—Reinforcement Learning (RL) is a popular approach for solving increasing number of problems. However, standard RL approach has many deficiencies. In this paper multiple approaches for addressing those deficiencies by incorporating Supervised Learning are discussed and a new approach, Reinforcement Learning with Adaptive Supervisor, is proposed. In this model, actions chosen by the RL method are rated by the supervisor and may be replaced with safer ones. The supervisor observes the results of each action and on that basis it learns the knowledge about safety of actions in various states. It helps to overcome one of the Reinforcement Learning deficiencies – risk of wrong action execution. The new approach is designed for domains, where failures are very expensive. The architecture was evaluated on a car intersection model. The proposed method eliminated around 50% of failures.**

I. Introduction

REINFORCEMENT Learning (RL) is a popular approach for solving increasing number of problems. In contrast to Supervised Learning (SL) this type of learning does not require any training data or teacher with prior knowledge. Instead, experimenting with the environemnt is performed to generate knowledge. RL has been successfully used to solve problems in multiple domains: robotics and control [1], game playinng [2] and power systems [3], just to name a few.

However, RL has many deficiencies that hinder applying it to the complex real word problems (e.g. unscalability [4], small data efficiency [5] and low human-readability of generated knowledge). Another defficiency is a risk of failures while searching optimal solution by experimenting with the environment, which requires taking random decisions from time to time. For many domains such risk is justified. However, in some domains any failure can be expensive (e.g. robot control). Thus, chance of failure should be minimized, even at the expense of the exploration. Multiple safe exploration techniques for RL were already proposed in literature [6], [7]. Most of these approaches assumes, that some prior external knowledge exists and can be used in early steps of exploration to avoid failures. However, this assumption is not always valid. Hence, need for techniques that limit risk of failures with no prior knowledge arises. It should be emphasized, however, that all failures in the exploration phase can be eliminated only if a prior knowledge is incorporeted [8].

In the literature multiple successful approaches for combining RL with Supervised Learning (SL) in form of *hybrid methods* were proposed to address various deficiencies of RL [9]–[11]. However, to the best to the Author's knowledge, no hybrid method dedicated to increasing safety of exploration has been proposed yet. In this work such approach by introducing the Adaptive Supervisor to support RL method is proposed. Adaptive Supervisor use SL approach to create knowledge about risky actions and observes states and actions that led to failures during exploration to create training set. The supervisor learns online so it can support RL and limit failures in an early phase of exploration.

The article is organized as follows. Firstly, various concepts for increasing safety of exploration are discussed. Then the novel Reinforcement Learning with Adaptive Supervisor architecture is introduced and the realisation of this architecture is proposed. The approach was verified in SInC domain [12]. Finally, results are presented and discussed. Additionally the knowledge generated by SL is verified.

II. Related Research

The comprehensive survey on Safe Reinforcement Learning can be found in [6]. The survey was recently extended in work [7]. Safe Reinforcement Learning methods can be divided into two groups. The first one involves modifying the risk-neutral optimization criterion to address possibility of failures. The modification can involve adding constraints (based on the external knowledge), optimizing performance for the worst scenario (in case the process is stochastic) or adding factor that makes safe policies more preferable over risky ones (e.g. ones with smaller variability of observed rewards). In the second group the optimization criterion remains risk-neutral, instead the exploration process is modified to avoid failures. Methods in this group can be further divided into methods that incorporates external knowledge (in the form of constraints, a set of demonstrations, a teacher that guides or supervise learning process, initial policy, etc.) and those in which exploration is directed to less risky areas by additional mechanism.

However, none of the presented works verifies possibility to increase safety of exploring process with on-line Supervisor (with no initial knowledge).

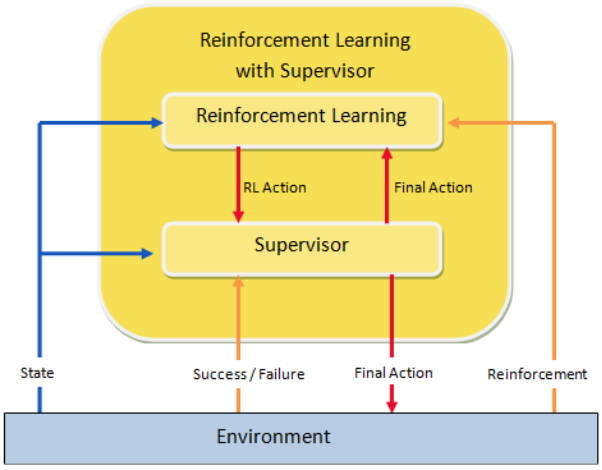


Fig. 1: General scheme of RL with Supervisor (RLS) architecture

Supervisor can be defined as additional mechanism, which role is to support RL in making better and safer decisions or in learning faster. The idea was firstly introduced in [14], where the supervisor is a set of tips, defined by expert. An idea for incorporating the supervisor (this time implemented with SL) to RL was described in [16]. The proposed solution is based on the assumption, that in some domains significant number of available actions in the given state are useless, so there is no sense in performing them. The role of the supervisor is to suggest small set of relevant actions for the given state.

In works [12], [13] SL is used to reduce state space of RL, thus increases efficiency of learning. This *hybrid method* (RLSR) is based on the assumption that two states are similar, if achieving them produces similar reward. The state is described by a set of attributes. The part of the state that is to be reduced is passed to a classifier. The classifier reduces that part of state to a single attribute. That single attribute and not reduced part of state are passed to RL module together as *reduced state*. Based on the *reduced state* the RL module chooses next action to perform. After the action a reinforcement is delivered both to the RL module and classifier for the learning purpose. The approach was tested in two domains: "Hunt the Wumpus" and "Hunter, Preys and Predators", which is the extended version of "Predators and Prey" problem. In the most of cases the conducted experiments proved the approach to be successful in shortening time necessary for learning the best solution, comparing with Q-learning. The proposed method performs well with noisy data.

III. Reinforcement Learning with Adaptive Supervisor – dealing with a risk of failures

The lack of research on applying supervised learning to implement the Supervisor concept was inspiration for developing a novel approach. The Reinforcement Learning with Adaptive Supervisor combines RL with supervisor implemented according to the SL approach.

A. General idea

To address the risk of failures the approach for combining RL with the adaptive supervisor is proposed (RLS). The architecture of the approach is presented in Fig. 1. The concept of the supervisor in this approach is similar to the one proposed in [15], where guard with explicit constraints is introduced. However, in the RLS the supervisor's knowledge is being created simultaneously with RL component. Each time the RL component chooses an action to perform, the selected action is rated by the supervisor. If the supervisor concludes, that the action chosen by RL is not safe enough (may lead to a failure) the supervisor overrides the action with the safest one (according to its current knowledge). The Supervisor observes the result of each action and on that basis it creates the knowledge about the safety of each action in each state in which that action can be preformed.

B. Realization

The proposed architecture was applied for the case where RL is implemented with the hybrid version (RLSR). This version accelerates learning, but the trade off is potentially worse quality of decision. Hence, a supervisor is introduced to minimize number of failures. Scheme of the method is presented in Fig. 2.

The supervisor is implemented as a classifier. In the process of learning the classifier receives training examples in the form: $\langle s, a, e \rangle$, where s is not reduced state, a is performed action and e is evaluation of performing the action a in the state s. e takes one of the following values: *good* or *bad*.

Asking the supervisor if the action is correct corresponds to classifying pair $\langle s, a \rangle$ as *good* or *bad*. If the action is classified as *bad* the supervisor iterates over all possible actions in state s and chooses the action with the highest certainty of being classified as *good* (the safest action). The chosen action is performed and sent back to the RL component as feedback.

To teach the supervisor rating actions, the supervisor has to store all examples gathered during learning process. As the

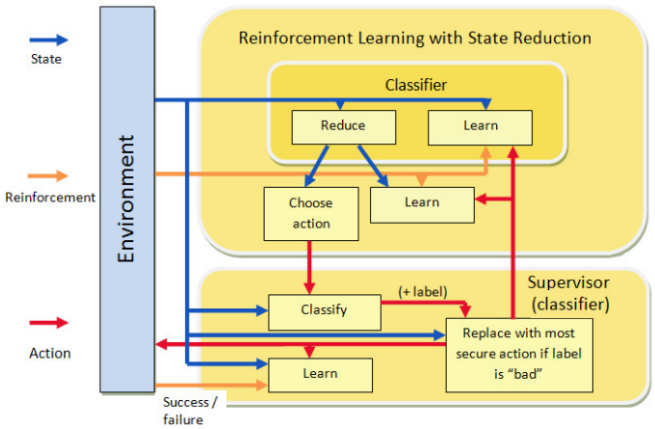


Fig. 2: Realization of RL with Supervisor architecture. First action is chosen by RL with State Reduction (RLSR). Then Supervisor rates action and replace it only if rate is *bad*.

supervisor gets information from the environment that success or failure happened, configured number of last stored examples should be labeled as *good* or *bad* respectively and added to the training set. Additionally weights of examples can be introduced — the closer example is to failure/success state, the bigger weight it should have.

The classifier used in this hybrid method should provide certainties of labels. Hence, rigor r of the supervisor can be configured. To classify action as *good* the certainty of label *good* has to be higher than r. Otherwise, action is classified as *bad*, because there were enough cases (according to the given rigor) which led to failure immediately or few steps after taking this action. It is worth noting, that too high rigor may led to rejecting action that is potentially good (there exists the list of actions following it that lead to success), but in the past that action preceded the incorrect actions that resulted in failure and hence is considered wrong.

C. Experimental domain

The approach was examined in the domain of crossing intersection by autonomous vehicles (SInC) [12]. Crossing intersection is simultaneous, therefore collisions can occur. To avoid them vehicles have to adjust their speed. In the same time vehicles should cross intersection as fast as possible. Hence, in every step of simulation an agent steering car chooses action $a_{\mathrm{i}} \in A$, which corresponds to changing speed by i[3]. The set of actions A is defined as $A := \{-a_{max}, -a_{max}+1, \ldots, a_{max}-1, a_{max}\}$, where a_{max} is the maximal speed change. In case as a result of taking action a_{i} the value of speed should be smaller than 0 or greater than v_{max}, then the speed value is set to 0 or v_{max}, respectively.

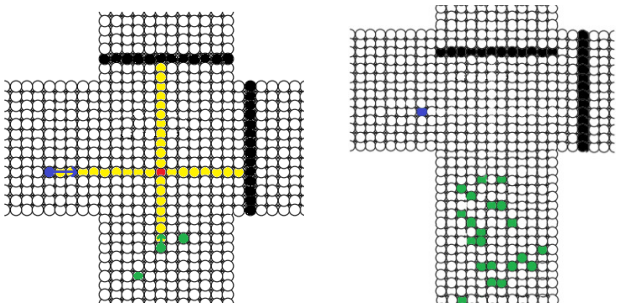
To choose an action the agent can use following information, updated in every step:

- distance to the end of intersection (d_t);
- speed of the car (v_c);
- distance between car and the nearest collision point with a car coming from the crossing road (collision car) (d_{cp});
- distance between the collision car and the collision point (d_{ccp});
- speed of the collision car (v_{cc}).

Below state s is defined as $s = \langle d_t, v_c, d_{cp}, d_{ccp}, v_{cc}\rangle$. In Fig. 3a an exemplary $s = \langle 18, 2, 10, 8, 1\rangle$ is presented.

After each step the agent observes the result of it's action. If the car reached the end of intersection successfully or collision occurred all stored pairs of states and actions ($s = \langle s, a\rangle$) are labeled by the Adaptive Supervisor as $e = good$ or $e = bad$, accordingly. Otherwise (car is still crossing intersection safely) the Adaptive Supervisor continues storing examples.

[3]the decision to change speed influences the speed of the car in the step following the step in which the decision was taken — e.g. if in the step t the speed of the car was equal to 1 field per step and the decision of the agent in that step t is to reduce speed by 1 field per step, than the car will change its position by one field in the step t and than stop (in the step $t+1$ the car won't change its position regardless the decision in the step $t+1$). Postponing result of the speed change makes the considered domain more challenging and realistic (gap between making observation and changing speed is taken into account).



(a) (b)

Fig. 3: (a) The example state at an intersection. Point where collision between the blue car (intelligent one) and the nearest moving car can happen is marked with a red color. With black line is the end of intersection (target); (b) Initial situation at intersection for experiment [12].

D. Verification scenario

Three approaches for taking decision were compared — standard RL, hybrid method RLSR, and the novel hybrid approach RLS (version described in III-B).

The first method (QRL) was based on the standard RL method (Q-learning with ϵ-greedy policy). The following rewards were introduced:

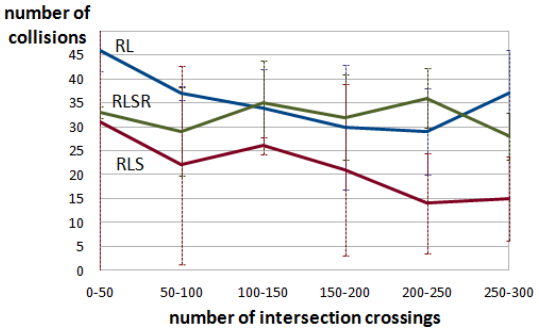
- r_c — negative reward for collision ($r_c = -100$)
- r_t — positive reward for reaching target ($r_t = 100$)
- r_s — negative reward for making move ($r_s = -2$)

Rewards r_t and r_s promote crossing intersection as quickly as possible. Reward r_c teaches the agent to avoid collisions. To implement Q-learning RLPark library was used ($\alpha = 0.25$, $\lambda = 0.4$, $\gamma = 0.55$, $\epsilon = 0.5$). All parameters were chosen with the hill climbing approach.

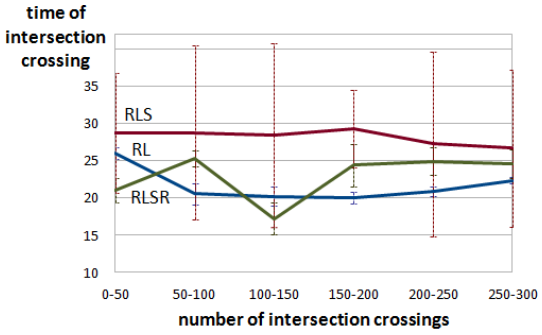
As the second method RL with state reduction with classifier (RLSR) was used. To detect similar states, the state attributes which corresponds to the possibility of collision (v_c, d_{cp}, d_{ccp}, v_{cc}) were reduced to a single bivalent attribute. The value of this attribute can be interpreted as possibility of collision in the given state. Classifier was implemented with C4.5 algorithm supplied by WEKA library. As the RL method the QRL was used.

The third approach was RLS method — to RLSR method supervisor was introduced to increase safety. For the Adaptive Supervisor success was defined as crossing intersection safely (getting to the end of intersection without collision). Any collision during crossing is considered as a failure. To implement the supervisor C4.5 algorithm was used. The rigor r of the supervisor dynamically changes with the development of agent's knowledge and is given for ith simulation of the experiment with formula:

$$r = \begin{cases} 0.5, & i < 10, \\ 0.66, & 10 \leq i < 200, \\ 0.75, & 200 \leq i < 300. \end{cases} \quad (1)$$



(a)



(b)

Fig. 4: A relationship between (a) collision number and (b) intersection crossing time and the agent experience — moving average with window size equal to 50 crossings of intersection.

The performance of methods was compared in the experiment conducted with platform MABICS [12]. The initial situation on the intersection for this experiment is shown in Fig. 3b. On the left side of the intersection a vehicle controlled by the intelligent agent is placed. On the bottom there are 21 vehicles that simulate real traffic. Each car moves with random, constant speed. Maximal speed change $a_{max} = 1$ and maximal speed $v_{max} = 3$. The experiment was repeated three times only, because of efficiency issues of the intersection crossing simulator, integrated with the MABICS. Each experiment consisted of 300 simulations. Between simulations within one experiment all gained experience of the agent was persisted.

E. Experimental results

Obtained average results of crossing time and collision numbers for all three methods are presented in Fig. 4. The RLSR method speeds up learning comparing to RL, but the number of collisions in the last periods is similar. Both RL and RLSR methods have difficulty in exploration of so complex domain, therefore only suboptimal solutions were found. The proposed RLS hybrid method proved to be safer in comparison with both standard RL and RLSR methods ($p < 0.05$ according to t-test[2]). RLS causes about 50% less collisions in the last periods of learning, which is satisfactory in so complex domain.

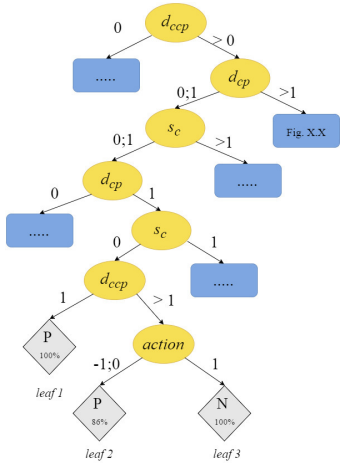
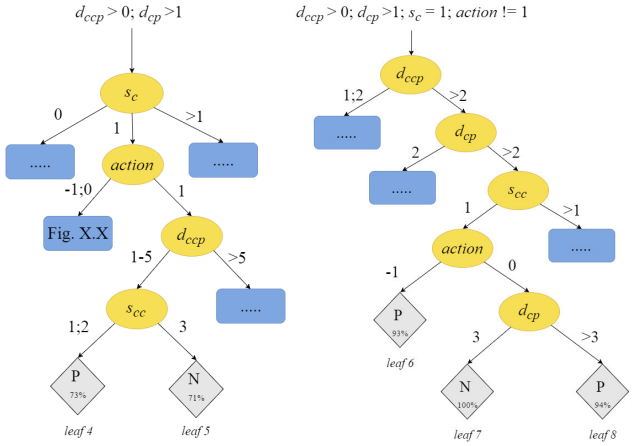


Fig. 5: Decision tree for classifying the given action in the given state as *good* or *bad* — root part



(a) (b)

Fig. 6: Decision tree for classifying the given action in the given state — (a) branch for $d_{ccp} > 0$ and $d_{cp} > 1$ and (b) branch for $d_{ccp} > 0$, $d_{cp} > 1$, $s_c = 1$ and $a < 1$

The decision tree which represents the knowledge created by the supervisor in the end of one of the experiments is presented in Fig. 5 and Fig. 6. Since the obtained tree has about 100 nodes only the most interesting branches are shown in details. With each leaf one rule for accepting or rejecting given action a in the given state(s) can be associated.

In some states any action chosen by the agent is accepted by the supervisor since collision cannot happen after visiting this state (according to the supervisor's experience). The example of such situation is represented by *leaf 1* (Fig. 5). The set of states associated with this leaf is illustrated in Fig. 7a. In this situation the speed of the car controlled by the intelligent

[2]the one sided t-test with equal variance. The equality of variance was verified with the two-sample F-test

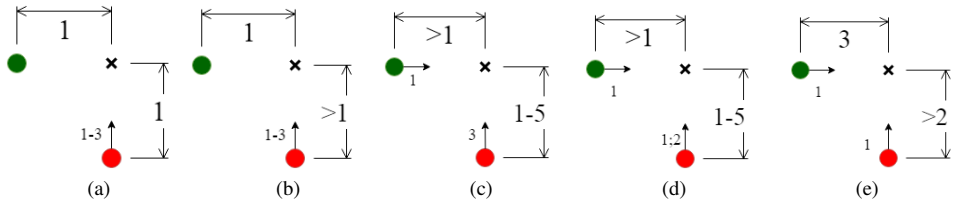


Fig. 7: Illustration of states associated with some leaves of decision tree, respectively (a) *leaf 1*, (b) *leaf 2* and *leaf 3*, (c) *leaf 5*, (d) *leaf 4*, (e) *leaf 7*. As the tree generalize examples each leaf represents set of similar states, e.g. (a) represents states satisfying $s_c = 0$, $s_{cc} \in \{1, 2, 3\}$, $d_{cp} = 1$ and $d_{ccp} = 1$.

agent is equal to zero and distance to collision point of both cars is equal to 1. Accepting each decision by supervisor is correct as the speed of collision car (v_{cc}) is greater than zero (constraint) — even if the agent decides to accelerate the car will move to the collision point in step $t+2$ whereas collision car even in the most pessimistic case ($v_{cc} = 1$) will move to collision point in step $t + 1$ and will leave this point in step $t + 2$.

Similar situation is associated with *leaf 2* and *leaf 3*. (Fig 5), but here the distance of collision car to the collision point (d_{ccp}) is greater than one (Fig. 7b). In this case accelerating is not safe anymore and supervisor will correctly reject decision to increase the speed by one and replace the action with decision to sustain speed or decrease it by one.

leaf 4 and *leaf 5* (Fig. 6a) show situation when the same action can be accepted or rejected depending on only one attribute of the state s, in this case v_{cc}. If $v_{cc} = 3$ (Fig. 7c) the supervisor reject the decision to accelerate. If the value of v_{cc} is smaller (the collision car approach to collision point slower, Fig. 7d) the decision to increase speed will be accepted as long as rigor r is smaller than 73%.

Finally, it is presented how the supervisor selects the best action in case of rejection, taking interesting rule as an example. The rejection rule is associated with *leaf 7* (Fig. 6b) and is illustrated on Fig. 7e. If the selected action is maintaining current velocity ($a = 0$) the supervisor rejects it. For simplification let choose as current state one that match the rule: $d_{ccp} = 3$, $d_{cp} = 3$, $v_c = 1$, $v_{cc} = 1$. Then, to select new action, the tree is searched for $a = 1$ and $a = -1$. For $a = -1$ the *leaf 6* (Fig. 6b) with label P and certainty equal to 93% is found, whereas for $a = 1$ the *leaf 4* (Fig. 6a) with label P and certainty 73%. As both leaves has label P the action with greater certainty is chosen ($action = -1$.). The action will cause that car will stop before reaching collision point, which is the safest option.

IV. Conclusions

This paper presents how Supervised Learning concept can be used to improve safety of the exploration process in RL, when no prior knowledge is available. The proposed method reduces number of failures, that are usually result of the space search, unavoidable part of RL method. Hence, the method should be used in domains where failures are expensive or even intolerable. The conducted experiments made it possible to verify new idea as promising since adding the Adaptive Supervisor eliminated around 50% of failures.

The presented RLS concept should be further examined in various domains, especially ones in which failures are costly and standard RL methods or hybrid methods (e.g. RLSR) are able to find good solution (it is not a case of the SInC domain) to analyze how many failures can be avoided additionally.

References

[1] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv:1511.03791*, 2015.

[2] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *AAAI*, pages 2140–2146, 2017.

[3] Mevludin Glavic, Raphaël Fonteneau, and Damien Ernst. Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine*, 50(1):6918–6927, 2017

[4] Reid, M. Ryan, M. R. K.: Using ILP to Improve Planning in Hierarchical Reinforcement Learning. In: Proceedings of the 10th International Conference on Inductive Logic Programming (ILP '00). Springer-Verlag, London, UK, pp. 174-190, 2000. DOI:10.1007/3-540-44960-4_11

[5] Fachantidis, A.,Partalas, I.,Tsoumakas G.,Vlahavas, I.: Transferring task models in Reinforcement Learning agents. Neurocomput. 107, pp.23-32. May 2013. DOI: 10.1016/j.neucom.2012.08.039

[6] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[7] José M Faria. Machine learning safety: An overview. 2018.

[8] Javier Garcia and Fernando Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012, DOI: 10.1613/jair.3761

[9] Uther, W. T. B.—Veloso, M. M.: Tree based discretization for continuous state space reinforcement learning. Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence (AAAI '98/IAAI '98), American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 769-774, 1998

[10] Kalyanakrishnan, S.—Stone, P.—Liu, Y.: Model-Based Reinforcement Learning in a Complex Domain, RoboCup 2007: Robot Soccer World Cup XI, Springer-Verlag, Berlin, Heidelberg, 2008

[11] Henderson, J.,Lemon, O.,Georgila, K.: Hybrid reinforcement/supervised learning for dialogue policies from communicator data, IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2005

[12] Krzysztoń, M. Sniezynski, B.: Combining Machine Learning and Multi-Agent Approach for Controlling Traffic at Intersection. Computational Collective Intelligence, Springer, 2015, pp 57-66

[13] Wiatrak, Ł.: Hybrid Learning in agent systems, Master Thesis, AGH University of Science and Technology, Cracow, 2012 (in Polish)

[14] Maclin, R.—Shavlik, J. W.: Creating advicetaking einforcement learners, Machine Learning, 22((13)): pp. 251-281, 1996

[15] Benbrahim, H., Franklin, J. A.: Biped dynamic walking using reinforcement learning. Robotics and Autonomous Systems,22,pp.283-302, 1997.

[16] Cetina, V.U.: Supervised reinforcement learning using behavior models, Machine Learning and Applications, 2007. ICMLA 2007, pp.336-341, 13-15 Dec. 2007. DOI: 10.1109/ICMLA.2007.14