Proposal of an Exploitation-oriented Learning Method on Multiple Rewards and Penalties Environments and the Design Guideline

Kazuteru Miyazaki
Research Department, National Institution for Academic Degrees and University Evaluation, Tokyo 187-8587, Japan
Email: teru@niad.ac.jp

***Abstract*— Among machine-learning approaches, reinforcement learning (RL) focuses most on goal-directed learning from interaction. Despite important applications, RL is difficult to design to fit real-world problems because, first, interaction requires too many trial-and-error searches and, second, no guidelines exist on how to design reward and penalty signal values. We are interested in approaches treating reward and penalty signals independently and not assigning them values. We also want to reduce the number of trial-and-error searches by strongly enhancing successful experience — a process known as exploitation-oriented learning (XoL). Though there are many XoL methods, they cannot apply to multiple rewards and penalties environments adequately. In this paper, we propose a new XoL method that can treat multiple rewards and penalties effectively. We present simulation and experimental results to show the effectiveness of our proposal. Furthermore, we describe the design guideline about rewards and penalties for the XoL methods.**

***Index Terms*— reinforcement learning, exploitation-oriented learning XoL, rewards and penalties, design guideline**

I. Introduction

Among machine-learning approaches, reinforcement learning (RL) focuses most on goal-directed learning from interaction [1]. It is very attractive because it can use dynamic programming (DP) to analyze behavior. RL generally treats rewards and penalties as teaching signals in learning. DP-based RL involves optimizing behavior under rewards and penalties signals designed by RL users on the Markov Decision Processes (MDPs).

Despite important applications [2]–[8], RL is difficult to design to fit real-world problems because, first, interaction requires too many trial-and-error searches and, second, no guidelines exist on how to design reward and penalty signal values. While these are essentially neglected in theoretical researches, they become serious issues in real-world applications, e.g., unexpected results arise if inappropriate values are assigned to reward and penalty signals [9].

We are interested in approaches treating reward and penalty signals independently and not assigning them values. We also want to reduce the number of trial-and-error searches by strongly enhancing successful experience — a process known as exploitation-oriented learning (XoL) [10]. In XoL, at the first, we aim to guarantee the rationality by the number of trial-and-error searches as small as possible rather than pursuing the optimality. Second, we do not assign a value for the reward and penalty, unless it is easily for us. We need however a priority among them as a teaching signal. By introducing these unique aspects, XoL has become a new framework placed focus on application (Table I).

TABLE I.
APPROACHES FOR GOAL-DIRECTED LEARNING FROM INTERACTION

	XoL	RL
The number of interaction	Less	More
Rewards and penalties	Priority among them	Values for them
Optimality on MDPs	By multi-start method	Guaranteed
Beyond MDPs	Strong (Bellman-free)	Weak (DP-based)

Examples of XoL learning methods for a type of a reward include profit sharing (PS) [11], the rational policy making algorithm (RPM) [12], PS-r* [13], and PS-r# [10]. Furthermore we know the penalty avoiding rational policy making algorithm (PARP) [9] that can treat a reward and a penalty at the same time. Also RPM and PARP have methods for discretizing states and actions using a basis function [14], [15].

XoL has four features as summarized in Table I: (1) XoL learns more quickly by strongly tracing successful experiences. (2) XoL treats rewards and penalties as independent signals, letting these signals be handled more intuitively and easily than the handling of concrete values [9]. (3) XoL does not pursue optimality efficiently, which can be acquired by multi-start [12] resetting all memory to get a better policy. (4) XoL is strong in the class that exceeds MDPs because it is a Bellman-free [1] method.

On the other hand, in general, we have to do a more careful response to types of rewards. Especially, it is important to avoid the problem that is so called “if you chase after two rabbits, you won’t catch either.” We know two layers model [16]–[18] that has upper and lower layers in order to solve this problem. The lower layer learns a policy to acquire each reward independently and the upper layer learns to switch an appropriate lower layer.

If we select the two layers model, in general, the upper layer uses the learning results that have been learned in

doi:10.4304/jcp.8.7.1683-1690

the lower layer. The learning results in the lower layer, however, do not include information about the need for the reward. We therefore assume that we can observe a desired level that represents the need of each reward. We propose a new XoL method to resolve the problem in which there are multiple rewards and penalties with the desired level. The effectiveness of our proposed method is shown by experimental results.

II. THE DOMAIN

A. Notations

Consider an agent in an unknown environment. After perceiving sensory input from the environment, the agent selects and executes an action. Time is discretized by one input-action cycle. Input from the environment is called a state. The pair consisting of the state and an action selected in a state is called a rule. Rewards and penalties based on a series of actions are provided from the environment, and a reward is given to a state or an action causing transition to a state in which our purpose is achieved, whereas a penalty given to a state or corresponding action in which our purpose is not achieved. Rewards and penalties are treated independently, eliminating the need to design of their values in a sophisticated way, unlike conventional DP-based RL systems that require the sophisticated design.

A rule series that begins from a reward/penalty state or an initial state and ends with the next reward/penalty state is called an episode. For example, when the agent selects S_0A_1, S_0A_0, S_1A_0, S_2A_0, S_1A_1, S_0A_0, S_2A_0 and S_1A_1 in Fig. 1 a), there exist two episodes ($S_0A_1S_0A_0S_1A_0S_2A_0S_1A_1$) and ($S_0A_0S_2A_0S_1A_1$) as shown in Fig. 1 b). If an episode contains rules of the same state but paired with different actions, the partial series from one state to the next is called a detour. For example, the episode ($S_0A_1S_0A_0S_1A_0S_2A_0S_1A_1$) has two detours ($S_0A_1$) and ($S_1A_0S_2A_0$) as shown in Fig. 1 b).

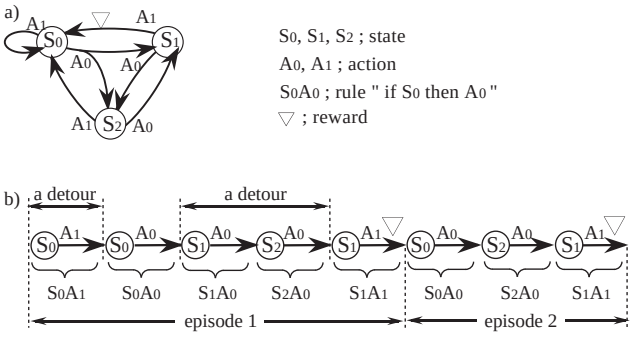

Figure 1. a) An environment consisting of three states and two actions. b) An example of an episode and a detour

A rule always existing on a detour is called an irrational rule, and otherwise called a rational rule. After obtaining the episode 1 in Fig. 1 b), rule S_0A_1, S_1A_0 and S_2A_0 are irrational rules and rule S_0A_0 and S_1A_1 are rational rules. When the episode 2 is experienced furthermore, rule S_2A_0 changes to a rational rule. A rule that directly receives a penalty is called a penalty rule. If all selective rules for a state are penalty or irrational rules, the state is called a penalty state. If a destination resulting after selecting a rule enters a penalty state, the rule is also classified as a penalty rule.

A function that maps states to actions is called a policy. The policy with a positive amount of reward acquisition expectations is called a rational policy, whereas a rational policy receiving no penalty is called a penalty avoiding rational policy. PARP aims to learn a penalty avoiding rational policy by using the Penalty Rule Decision procedure (PRD) [9] in Fig. 2. We can regard the marked rule that is found by PRD as a penalty rule. The agent with PARP selects an action from rule set where there is no irrational and penalty rule. If the agent cannot select any action in a state, it selects an action that has the less probability to transit to a penalty state from the state.

procedure The Penalty Rule Decision procedure (PRD)
begin
 In the experienced episode until now, the rule
 which received a penalty directly is marked
 do
  The state in following condition is marked;
   there is no rational rule or
   there is no rule that can transit to non-marked state
  The rule in following condition is marked;
   there are marks in the states that can be transited by it
 while at least one of the state is newly marked
end

Figure 2. The penalty rule decision procedure [9]

B. The two layers model and a desired level

In this paper, we treat multiple rewards that correspond to each purpose. We use the two layers model in Fig. 3 that has an upper and lower layers in order to solve this problem. The lower layer learns the policy to acquire each reward independently and the upper layer learns to switch an appropriate lower layer. We assume the situation that has been finished the lower layer learning. It means that the lower layer learning should be finished before starting the upper layer learning. The problem when the upper and lower layers learn at the same time is a future work.

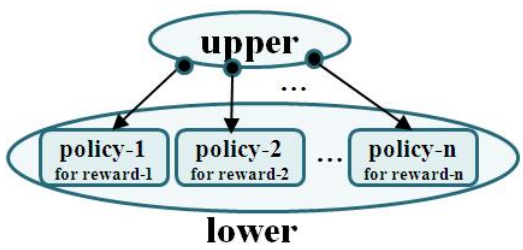

Figure 3. The two layers model

We assume that we can observe a desired level that represents the need of each reward. The higher the value

of the desired level goes up, the more the emergency of the reward rises. We can therefore know priority among rewards by comparing the desired levels. Each value of a desired level increases based on the unknown monotonically increasing function independently for each selecting an action. The case that is not based on monotonically function is a future work.

The upper layer can observe values of all desired levels and aims to switch an appropriate lower layer based on the values. We know Maximum desired level Priority Strategy (MPS) as a simple and an effective upper layer where the reward that has the highest value of a desired value corresponds to the most emergent purpose. MPS will be effective in many cases. However, for example, if the amount of increase in the desired level B is a very faster than the amount in the desired level A, there is a case that we should select the desired level B even if the value of the desired level B is lower than the value of the desired level A at that time.

When at least one value of the desired level exceeds threshold that is defined in advance, the agent receives a penalty, If we do not distinguish the difference of the desired levels whose values exceed threshold, the task is the same as previous works [9], [14] that treat only one type penalty. On the other hand, the agent acquires a reward, when it moves to the state where the value of a desired level can be recovered. We can therefore make new problem on multiple rewards and penalties environments by distinguishing desired levels.

III. Multiple Rewards and Penalties in a Previous Method

We know a method of reference [19] that can treat multiple rewards and penalties. The two layers model, that is used in this paper, is proposed in the paper. In the two layers mode, it is important to memorize the experience correctly in the upper layer by reflecting an action that is learned in the lower layer.

The method of reference [19] memorizes experience by a basis function [14] in Appendix. Center vector of the basis function is constructed by states that have been sensed by the upper layer. In addition, each basis function memorizes the lower layer that was selected when generating the basis function. When the agent acquires a reward, the method memorizes all basis functions that are used to until acquiring the reward.

There is no problem if the agent can acquire a reward continuously by the basis functions. On the other hand, when the agent receives a penalty, it should not select the basis function that is related with the penalty. It is realized by updating the value of f_para that can control the size of a basis function. If we increase the value of f_para, the size of the basis function will be small.

Considering the case that the agent receives a penalty about the desired level B when it aims to acquire a reward about the purpose A, that is, it selects an action based on the lower layer corresponding to the purpose A. If the desired level A is independent of the desired level B, that is, the value of the desired level B is not influenced by acquiring a reward about the purpose A, the basis function that was selected in the previous state should not be selected in that time, since we can consider that the purpose was not need to select in the previous state. In this case, we increase the value of f_para in order not to select the basis function. Otherwise, we increase the value of f_para in the basis function that was selected in the past state where the purpose A did not aimed last in order not to select the basis function in that time, since we can consider that the purpose A should be selected at the more earlier state.

In the upper layer, the agent selects a lower layer memorized in the basis function that is closest to the current state. The method of reference [19] can treat multiple rewards and penalties through the above process.

IV. Proposal of a new XoL method on multiple rewards and penalties environments

A. Ideas

In this paper, we use the two layers model described in Section II-B. The lower layer learns the policy to acquire a reward corresponding to each purpose by PARP independently. The input to the lower layer is the current state and the output of it is an action that should be selected in the state.

The upper layer learns to switch an appropriate lower layer based on three types information, that is, the current state, the values of desired levels, and information given from lower layers, such that, expected values and variances of the number of actions in order to acquire rewards on lower layers. Though, in the upper layer learning, the method of reference [19] uses the basis function that has been proposed in references [14], we do not necessarily need to use the basis function, since the purpose of the learning by a desired level is to find a threshold that defines a switching point of each purpose. We therefore use an avoidance list that is the set of lower layers that should not be selected in each state, instead of the basis function in reference [19].

B. Reconstruction of the desired level

The purpose of the upper layer is to select an appropriate lower layer for problems that cannot be handled by MPS. Among three types information described in section IV-A, the values of desired level and information given from lower layers are inputted to the upper layer only, though the current state is inputted to both lower and upper layers. In our avoidance list learning, we only use the values of desired level and information given from lower layers, since they are characteristic information in the upper layer.

In this paper, we assume the situation that has been finished the lower layer learning. It means that information given from lower layers had been fixed when the upper layer starts the learning. We therefore reconstruct the values of desired levels based on the results of comparison

of the number of actions in order to acquire a reward in each lower layer.

C. The learning based on an avoidance list

The agent, in the upper layer learning of the method of reference [19], selects the lower layer memorized in the basis function that has the best match with current desired level. When the agent acquires a reward, basis functions on the episode are stored. On the other hand, the scope of the basis function is narrowed by a penalty. Therefore, the main part of the learning lies in a reward, and the learning of a penalty is regarded as sub function. It means that the learning has been influenced strongly by the first acquiring reward. Though it is desirable in the point of fast learning, it is difficult to apply to the environment that has a lot of uncertainty.

We therefore construct an avoidance list that is the set of lower layers that should not be selected in each state to put more emphasis on the learning by a penalty. The avoidance list is used to select an appropriate lower layer, and constructed by a penalty as follows;
When the agent receives a penalty, the following information in which the agent had been received the previous penalty is registered in the avoidance list.

- Type of purpose that have aimed to satisfy.
- Values of the desired level of the purpose that have received the penalty.
- Values of the desired level of the purpose that have aimed to satisfy.

This registration should be done by taking into account the interaction between "the purpose that have aimed to satisfy" and "the purpose that have received a penalty" as same as the method on reference [19] in Section III. When the agent, that aims to satisfy the desired level A, receives the penalty about the desired level B and the desired level A is included in the penalty, the avoidance list is updated with the information at that time where the purpose A was not aimed last. The avoidance list is also updated when more wide range list has been constructed.

D. Decision making on the upper layer

The agent refers to the avoidance list when it selects a lower layer in the upper layer learning. If an avoidance list has been matched with the current state, the agent selects the lower layer that aims to pursue the purpose that is not memorized in the avoidance list.

In the case that all purposes cannot be selected due to the avoidance list, taking into account of priority of the penalty, a penalty that has the lowest priority is excluded. In the worst case, there is a possibility that all purposes will be excluded from the selection. In the case and there are multiple lower layers to be selected, the agent selects a lower layer based on assigning the same probabilities among them.

V. Design Guideline of Rewards and Penalties

When a task has multiple purposes, we can treat it by introducing a deadline that is a time until achieving each purpose. If a learning agent missed the deadline, it receives a penalty. Furthermore, when the penalty is assigned for each purpose, priority among them should be given to the agent. It is treated by our XoL methods through the following framework.

- We prepare lower layers for each purpose. An upper layer is prepared in order to switch the lower layers.
- We give the agent desired levels and deadlines that correspond to each purpose in the lower layer. The desired levels are updated for each action independently.
- When a purpose has been achieved, the lower layer concerned with the purpose acquires a reward. On the other hand, when the desired level about the purpose exceeds threshold that is derived from the deadline, a penalty is given to both the upper and the lower layers concerned with the purpose.

The best policy is to acquire a reward without receiving any penalty. If it cannot be learned, the lowest priority penalty is excluded. That is, the upper layer selects the lower layer at the view point of excluding the penalty. It is, for example, equivalent to a human that selects the next best thing.

In our XoL methods, rewards and penalties are designed in terms of signals that are derived from deadlines that correspond to each purpose. On the other hand, RL requires the sophisticated design of rewards and penalties values. It means that XoL is able to handle a wide variety of application more directly and easily than RL.

VI. Numerical Experiments

A. Experimental Setting

We present simulation in order to show the effectiveness of our proposed method. The environment used in our simulation is shown in Fig. 4. The agent has 4 types desired level (A, B, C, D). For each action, the values of the desired level increase based on the amounts that describes later. Though the agent can observe the desired level correctly, it does not have priori knowledge about the function that defines the increasing level.

When the agent reaches to a purpose that is shown at triangle (A, B, C, D) in Fig. 4, it acquires a reward about the corresponding the letter and the desired level about the letter is recovered. On the other hand, if at least one desired level exceeds threshold that is defined in advance, the agent receives a penalty and is forced to return to the initial state s0, and the values of all desired levels are initialized.

In this situation, we consider the problem which purpose should be selected in s0. The solution depends on the amount of change of a desired level when the agent selects an action and acquires a reward. The purpose of the agent is to learn a penalty avoiding rational policy.

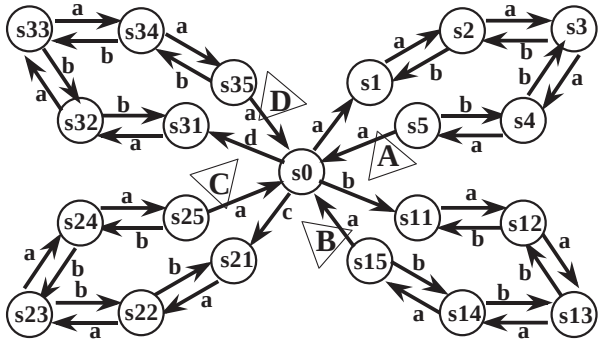

Figure 4. The environment used in numerical experiments

We consider the case that the lower level learning that aims to reach each purpose from each state (s0, s1, $\cdots$, s35) by the shortest number of actions has been finished. The values of the initial desired levels are 0, and the increasing value of a desired level is assigned from 1, 3, 5 or 7 as follows; the values of A and B are 1, C changes from 1 to 7, and D changes from the values of C to 7. The threshold for a desired level is 100. The recovering value of a desired level in case of acquiring a reward is 20, 40, 60, 80 or 100. Therefore, we have done (4+3+2+1) 54 = 6250 experiments.

In this problem, we know Maximum desired level Priority Strategy (MPS) as simple and an effective upper layer. Therefore, the agent tries to use MPS at the first, then, if MPS is not function effectively, our proposed method or the method of reference [19] is used.

B. Experimental Results and Discussion

Table II shows fail times of MPS and success times of the method of reference [19] and our proposed method in the fail times of MPS over 6250 experiments. The success means that the agent does not receive a penalty over one million actions. ST1 means that the shortest steps from s1 to s3 changes to 1 or 3 with a probability of 50%. ST2 is obtained by adding the similar change in the shortest steps from s11 to s13. ST3 and ST4 is the same as ST2.

TABLE II.
EXPERIMENTAL RESULTS

	ST1	ST2	ST3	ST4
Fail times of MPS	4120	4165	4552	4805
Success times of the method of Ref. [19]	208	96	34	13
Success times of our proposed method	342	320	268	253

From Table II, our proposed method is able to handle a wider change than MPS. It means that our proposed method is robust uncertainty. Especially, it is robust to uncertainty compared to the method of reference [19]. Reference [19]'s method is difficult to flexibly adapt to the environment when the degree of uncertainty will be more, since it is strongly affected by the first successful experience. On the other hand, our proposed method has focused on the penalty. Therefore, it is not affected by the first successful experience than the reference [19]'s method and it is robust to uncertainty, though both methods are XoL methods.

Furthermore, we have performed experiments that priority decreases in the order A, B, C, and D. If we cannot get a policy that does not have any penalty, a penalty that has the lowest priority is excluded. Table III shows the result considering only 3 penalties (A, B, C) where the penalty D is excluded. Though MPS does not have such mechanism, we have counted as no failure when the reward D was not acquired.

TABLE III.
EXPERIMENTAL RESULTS CONSIDERING ONLY 3 PENALTIES

	ST1	ST2	ST3	ST4
Fail times of MPS	915	1015	1410	2524
Success times of our proposed method	915	1013	1051	1719

Our proposed method is successful in almost all cases if uncertainty is low. In highly uncertain ST3 or ST4, it has been successful in the order of 2/3 MPS fails. Furthermore, in the case of considering only 2 penalties (A, B), though MPS fails 9 patterns shown in Table IV, our proposed method can success in all cases.

TABLE IV.
FAIL PATTERNS OF MPS IN THE CASE OF CONSIDERING ONLY 2 PENALTIES

	increasing values of desired levels by selecting an action				recovering values of desired levels by acquiring a reward			
	A	B	C	D	A	B	C	D
No.1	1	1	1	1	60	20	20	20
No.2	1	1	1	1	40	20	20	20
No.3	1	1	1	1	20	60	20	20
No.4	1	1	1	1	20	40	20	20
No.5	1	1	1	1	20	20	60	20
No.6	1	1	1	1	20	20	40	20
No.7	1	1	1	1	20	20	20	60
No.8	1	1	1	1	20	20	20	40
No.9	1	1	1	1	20	20	20	20

Case that is difficult to resolve by MPS is that the increasing value of a desired level is intense, or the recovering value of a desired level is small when a reward has been acquired. Our proposed method was able to extend the case in this experiment. In addition, it was possible to achieve a success at all combinations that cannot be learned by MPS through taking into account of priority among penalties. We can confirm the effectiveness of our proposed method through this numerical experiments.

VII. APPLICATION TO THE LEGO ROBOTS

We apply our proposed method to the LEGO robots as shown in Fig. 5, Fig. 7 and Fig. 8. We have performed 3 vs 1 Keepaway task [6] where three robots called keeper turn the path of a ball each other. Each keeper robot located at the apex of an equilateral triangle in Fig. 6. A taker robot interferes the path of a ball among three keepers. The instruction of the prototype model of the robot is given in the reference [20].

We use HiTechnic Infrared Electronic Ball (IRB1005). The taker robot in Fig. 7 is equipped with the same ball

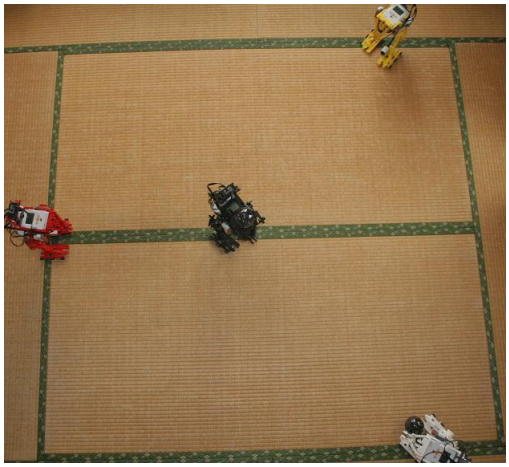

Figure 5. Application to the LEGO robots

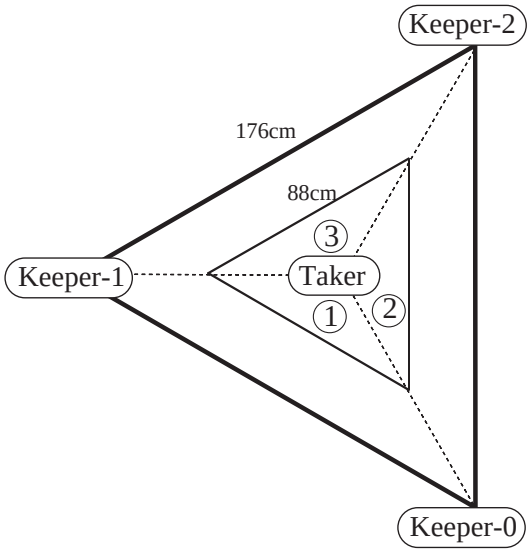

Figure 6. Configuration of each robot

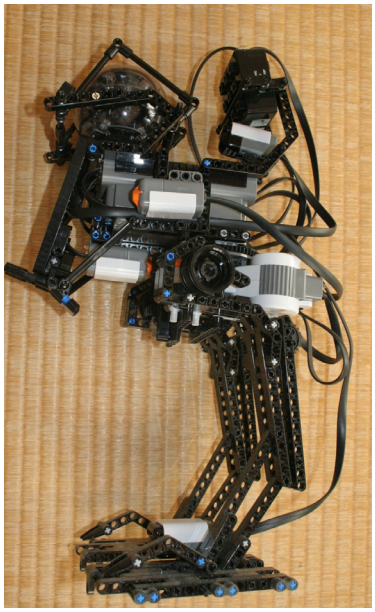

Figure 7. The Taker robot

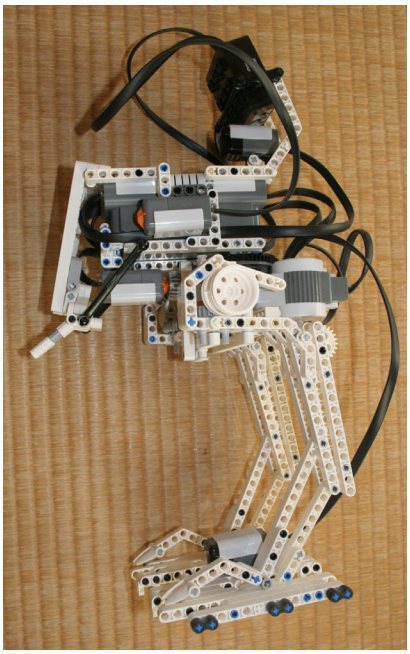

Figure 8. The Keeper robot

at the highest position than a ball on the ground. Each keeper robot in Fig. 8 has two HiTechnic IRSeekers V2 (NSK1042) to locate and provide the direction to their balls. On the other hand, the taker robot has only one IRSeeker to find a ball on the ground.

Each keeper robot is always fixed at the apex of the triangle in Fig. 6. Therefore, the purpose of keeper robot is to select the robot to pass a ball, that is, the action is selected from “Pass a ball to the right keeper robot”, “Pass a ball to the left keeper robot” and “Keep a ball”. The keeper robot learns independently in case of keeping a ball.

The initial position of the taker robot is the center of gravity of the triangle. The taker robot does not learn and moves randomly in a range of an equilateral triangle whose length of the side is 88cm. When the taker robot moves to the out of scope of this triangle, it is forced to return to the initial position by hand.

We prepare three lower layers corresponding to the three actions “Pass a ball to the right keeper robot”, “Pass a ball to the left keeper robot” and “Keep a ball”. The upper layer learns the switching of the lower layers. The learning in the lower layer has been finished in advance. It means that the pass is always success unless there is no taker robot. On the other hand, if there is the taker robot in the position 1 in Fig. 6, the pass from Keeper-0 to Keeper-1 fails. Furthermore, if there is the taker robot in the position 2 in Fig.6, the pass from Keeper-0 to Keeper-2 fails. The pass fails by the same condition in cases of Keeper-1 or Keeper-2 is keeping a ball.

The desired level is designed by the following based on the taker robot position; If Keeper-0 has a ball and the

taker robot lies in the range of position 1 in Fig. 6, the desired level about ”Keeper-0 passes a ball to Keeper-2” is fired. On the other hand, if Keeper-0 has a ball and the taker robot lies in the range of position 2 in Fig. 6, the desired level about ”Keeper-0 passes a ball to Keeper-1” is fired. Either of these two desired levels is fired at random when the taker robot lies in the range of position 3 in Fig. 6. The desired level is designed by the same condition in cases of Keeper-1 or Keeper-2 keeping a ball.

We have compared with our proposed method and MPS. MPS always selects a lower layer corresponding to the fired desired level.

Both our proposed method and MPS can always success when the lower layer learning does not fail. On the other hand, when the lower layer corresponding to “Pass a ball to the right keeper robot” fails at some probability, though our proposed method can always success, MPS fails the learning. In this case, our proposed method can exclude the selection of the lower layer corresponding to “Pass a ball to the right keeper robot”, and always selects the lower layer corresponding to “Pass a ball to the left keeper robot”. We can confirm the effectiveness of our proposed method in this experiment, too.

VIII. Conclusions

In this paper, we have proposed a new XoL method that can treat multiple rewards and penalties effectively. We have shown the effectiveness of our proposal by numerical experiments and application to the LEGO robots.

In the future, we will apply our proposed method to real world problem such that our biped walking robot [2], the course classification support system on NIAD-UE [21], and so on.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 22500143.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. A Bradford Book, MIT Press, 1998.

[2] S. Kuroda, K. Miyazaki, and H. Kobayashi, “Introduction of fixed mode states into online reinforcement learning with penalty and reward and its application to waist trajectory generation of biped robot,” *J. of Advanced Computational Intelligence and Intelligent Informatics*, vol. 16, no. 6, pp. 758–768, 2013.

[3] T. Matsui, T. Goto, and K. Izumi, “Acquiring a government bond trading strategy using reinforcement learning,” *J. of Advanced Computational Intelligence and Intelligent Informatics*, vol. 13, no. 6, pp. 691–696, 2009.

[4] K. Merrick and M. L. Maher, “Motivated reinforcement learning for adaptive characters in open-ended simulation games,” in *Proc. of the Int. Conf. on Advanced in Computer Entertainment Technology*, 2007, pp. 127–134.

[5] J. Randlϕv and P. Alstrϕm, “Learning to drive a bicycle using reinforcement learning and shaping,” in *Proc. of the 15th Int. Conf. on Machine Learning*, 1998, pp. 463–471.

[6] P. Stone, R. S. Sutton, and G. Kuhlamann, “Reinforcement learning toward robocup soccer keepaway,” *Adaptive Behavior*, vol. 13, no. 3, pp. 165–188, 2005.

[7] T. Watanabe, K. Miyazaki, and H. Kobayashi, “A new improved penalty avoiding rational policy making algorithm for keepaay with continuous state spaces,” *J. of Advanced Computational Intelligence and Intelligent Informatics*, vol. 13, no. 6, pp. 675–682, 2009.

[8] J. Yoshimoto, M. Nishimura, Y. Tokita, and S. Ishii, “Acrobot control by learning the switching of multiple controllers,” *J. of Artificial Life and Robotics*, vol. 9, no. 2, pp. 67–71, 2005.

[9] K. Miyazaki and S. Kobayashi, “Reinforcement learning for penalty avoiding policy making,” in *Proc. of the 2000 IEEE Int. Conf. on Systems, Man and Cybernetics*, 2000, pp. 206–211.

[10] ——, “Exploitation-oriented learning ps-r#,” *J. of Advanced Computational Intelligence and Intelligent Informatics*, vol. 13, no. 6, pp. 624–630, 2009.

[11] K. Miyazaki, M. Yamamura, and S. Kobayashi, “On the rationality of profit sharing in reinforcement learning,” in *Proc. of the 3rd Int. Conf. on Fuzzy Logic, Neural Nets and Soft Computing*, 1994, pp. 285–288.

[12] K. Miyazaki and S. Kobayashi, “Learning deterministic policies in partially observable markov decision processes,” in *Proc. of the 5th Int. Conf. on Intelligent Autonomous System*, 1998, pp. 250–257.

[13] ——, “An extension of profit sharing to partially observable markov decision processes: Proposition of ps-r* and its evaluation,” *J. of the Japanese Society for Artificial Intelligence*, vol. 18, no. 5, pp. 285–296, 2003, in Japanese.

[14] ——, “A reinforcement learning system for penalty avoiding in continuous state spaces,” *J. of Advanced Computational Intelligence and Intelligent Informatics*, vol. 11, no. 6, pp. 668–676, 2007.

[15] K. Miyazaki, “Proposal of the continuous-valued penalty avoiding rational policy making algorithm,” *J. of Advanced Computational Intelligence and Intelligent Informatics*, vol. 16, no. 2, pp. 183–190, 2012.

[16] N. Sprague and D. Ballard, “Multiple-goal reinforcement learning with modular sarsa(0),” Tech. Rep., 798, The University of Rochester, Computer Science Department, 2004.

[17] C. K. Tham and R. W. Prager, “A modular q-learning architecture for manipulator task decomposition,” in *Proc. of 11th International Conference on Machine Learning*, 1994, pp. 309–317.

[18] N. Ono and K. Fukumoto, “Multi-agent reinforcement learning: A modular approach,” in *Proc. of 2nd International Conference on Multiagent Systems*, 1996, pp. 252–258.

[19] K. Miyazaki, “Research of a decision making method based on consciousness in multiple rewards environments,” in *Proc. of 39th SICE Symposium on Intelligent systems*, 2012, pp. 95–98, in Japanese.

[20] D. Benedettelli, *Creating Cool MINDSTORMS NXT Robots*. Apress, 2008.

[21] K. Miyazaki and M. Ida, “Proposal and evaluation of the active course classification support system with exploitation-oriented learning,” *Lecture Notes in Computer Science*, vol. 7188, pp. 333–344, 2012.

Kazuteru Miyazaki received his B.S. degree in engineering from Meiji University, Japan in March 1991 and his M.S. degree and Ph.D. in engineering from Tokyo Institute of Technology, Japan in March 1992 and 1996. He is currently Associate Professor in National Institution for Academic Degrees and University Evaluation. His current research interest includes machine learning, especially, reinforcement learning.

APPENDIX

Continuous State Space Discretization

To discretize continuous state space using basis functions [14], let state (n-dimensional vector) at time t be $\mathbf{S}_t$, a selected action by $\mathbf{S}_t$ be a_t, and the resulting transition destination be $\mathbf{S}_{t+1}$, as shown in **Fig. 9**. A basis function is created by an n-dimensional normal distribution function with current state $\mathbf{S}_t$ at the center. The principal axis direction of the function is defined by $\mathbf{S}_{t+1} - \mathbf{S}_t$, the d_1 axis in **Fig. 9**. Directions of other axes $d_2, \ldots, d_n$ are generated using Gram-Schmidt orthonormalization from any linearly independent vectors to mutually intersect at right angles, the $d_2, \ldots, d_n$ axes in **Fig. 9**.

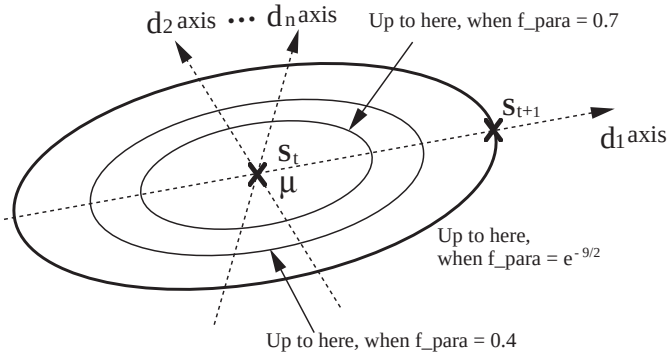

Figure 9. The basis function in the reference [14]

Principal axis range extent σ_1 is given by $3\sigma_1 = |\mathbf{S}_{t+1} - \mathbf{S}_t|$, and the extent of the range of other axes σ_i is given by $3\sigma_i = \frac{|\mathbf{S}_{t+1} - \mathbf{S}_t|}{\sqrt{n}} (i = 2, 3, \ldots, n)$. Because a function biased in an experienced direction is obtained, the range extent of other than the principal axis is multiplied by $\frac{1}{\sqrt{n}}$. $3\sigma_i$ covers 99% of samples.

Center $\mathbf{S}_t$ of the generated basis function is called the basis function state, or μ. The action that generates the basis function is memorized together with the basis function and is used when an action is selected. The basis function corresponds to a rule under discrete state space. The basis function corresponding to the penalty rule is called a penalty basis function, and the other basis function is called a nonpenalty basis function. A basis function is generated initially as an unlabeled basis function. When a learning agent receives a penalty, we can find a penalty basis function by using the Penalty Basis Decision (PBD) procedure [14]. PBD is obtained by replacing a rule within PRD [9] (Fig. 2) a basis function. PARP in continuous state space [14] aims to acquire a reward continuously by selecting a non-penalty basis function.

If values of an observation at time t are given by $\mathbf{y}$, then the returned value is given as:

$$f(\mathbf{d}) = \exp\left\{-\frac{1}{2}(\mu - \mathbf{y})^T T \Sigma T^T (\mu - \mathbf{y})\right\}, \quad (1)$$

T = [$\mathbf{t_1}, \mathbf{t_2}, \ldots, \mathbf{t_n}$], Σ = diag($\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$), and $\mathbf{t_i}$ is a unit vector along the d_i axis for $i = 1, 2, \ldots, n$. Eq(1) becomes 1.0 when $\mathbf{y}$ coincides with μ, and the farther $\mathbf{y}$ is away from μ, the smaller the value of Eq(1). Calculating the value of Eq(1) for each memorized basis function for given observation $\mathbf{y}$ enables us to compare the closeness of the observation to each basis function.

Each basis function scope is controlled by a threshold. The control parameter is called f_para ($e^{-\frac{9}{2}} \leq$ f_para ≤ 1.0) and attached to each basis function. If f_para becomes large, the scope of the function is narrowed and if f_para becomes 1.0, it coincides with center μ. f_para $= e^{-\frac{9}{2}}$ means that the basis function range is expanded to transition destination $\mathbf{S}_{t+1}$.