Doctoral Dissertation

Bottom-Up Multi-Agent Reinforcement Learning for Complexity and Uncertainty

Takumi Aotani
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Professor Kenji Sugimoto
Intelligent System Control Lab. (Division of Information Science)

Submitted on March 17, 2022

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Takumi Aotani

Thesis Committee:
Supervisor Kenji Sugimoto
(Professor, Division of Information Science)
Takahiro Wada
(Professor, Division of Information Science)
Takamitsu Matsubara
(Associate Professor, Division of Information Science)
Taisuke Kobayashi
(Assistant Professor, Division of Information Science)

Bottom-Up Multi-Agent Reinforcement Learning for Complexity and Uncertainty*

Takumi Aotani

Abstract

Multi-agent systems (MASs) are expected to be applied to various real-world problems where a single agent cannot accomplish given tasks. Due to the inherent complexity and uncertainty in the real-world MASs, the manual design of group behaviors of agents is intractable. Multi-agent reinforcement learning (MARL), which is a framework for multiple agents in the same environment to learn their policies adaptively, would be a promising methodology. The conventional MARL methods, however, target a limited class of MASs assuming a common task and a centralized system.

In this study, we propose "bottom-up MARL" as an autonomous distributed framework in which agents have their own tasks. Compared to the conventional MARL, the agents' own tasks increase the complexity of group behaviors, and the autonomous distributed manner increases the uncertainty that threatens safe learning. Hence, this study addresses these issues by developing i) a reward shaping algorithm for the group behaviors, and ii) a meta-optimization method for the bias-variance trade-off in the model learning. These are elemental techniques for learning the group behaviors safely, using the predictive models of the rewards and the dynamics, and the reshaped reward, in model-based reinforcement learning.

i) In order to learn the group behaviors, the rewards of all agents are shared and predicted. Each agent obtains a reshaped reward for the group behaviors based on the prediction for its own state. The proposed algorithm

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 17, 2022.

consists of four components: reward prediction, promotion of exploration, classification of interests, and reward shaping. The effectiveness of the proposed method is verified by the simulations and the experiments using real robots.

ii) For long-term safety assurance in uncertain environments, avoiding the predictions with large errors by the dynamics model, is necessary. In this study, we formulate the bias-variance trade-off as a multi-objective optimization problem and develop a meta-optimization method to adjust the trade-off simultaneously with model training. The effectiveness of the proposed method is verified by the simulations.

Keywords:

Multi-agent reinforcement learning, distributed autonomous system, reward shaping, model learning, bias-variance trade-off, meta-optimization

Contents

1 Introduction **1**
1.1 Background 1
1.2 Contribution 3
1.3 Dissertation outline 4

2 Bottom-up Multi-agent Reinforcement Learning by Reward Shaping **5**
2.1 Introduction 5
2.2 Related work 8
2.3 Preliminaries 12
2.3.1 Reinforcement learning 12
2.3.2 Multi-agent reinforcement learning 12
2.4 Reward Shaping Algorithm for Bottom-up MARL 14
2.4.1 Overview 14
2.4.2 i) Reward prediction 16
2.4.3 ii) Exploration bonus 16
2.4.4 iii) Classification of interests 18
2.4.5 iv) Reward shaping 20
2.5 Numerical simulations 21
2.5.1 Verification of ability to classify static interests 21
2.5.1.1 Conditions 21
2.5.1.2 Results 24
2.5.2 Valification of effectiveness to learn state-dependent interests 25
2.5.2.1 Conditions 25
2.5.2.2 Results 26
2.5.3 Verification of versatility 26
2.5.3.1 Conditions 27

2.5.3.2 Results 28
2.6 Experiments 29
2.6.1 Environment 29
2.6.2 Task settings 31
2.6.2.1 Cooperative task 31
2.6.2.2 Competitive task 32
2.6.3 Results 32
2.6.3.1 Cooperative task 32
2.6.3.2 Competitive task 33
2.6.3.3 Discussion 34
2.7 Conclusion 35

3 Meta-Optimization of Bias-Variance Trade-off in Stochastic Model Learning 48
3.1 Introduction 48
3.2 Related work 51
3.2.1 Bias-variance decomposition 51
3.2.2 Meta-optimization 52
3.3 Preliminaries 56
3.3.1 Stochastic model learning in Markov decision processes . . 56
3.3.2 Augmented weighted Tchebycheff scalarization 57
3.4 Meta-optimization of bias-variance trade-off 59
3.4.1 Overview 59
3.4.2 Formulation of MOO problem 61
3.4.2.1 Inter-data MOO: IDMO 61
3.4.2.2 Statistics-perspective MOO: SPMO 62
3.4.2.3 Summary of proposed losses 62
3.4.3 Meta-optimization of hyperparameter 63
3.5 Experiments 64
3.5.1 Common conditions 64
3.5.1.1 Human-operated single-agent environment 67
3.5.1.2 Multi-agent environment 67
3.5.2 Meta-objective: linear weighted sum of mean and worst losses 69
3.5.2.1 Human-operated single-agent environment 69

3.5.2.2 Multi-agent environment 71
3.5.3 Meta-objective: accuracy of long-term prediction 72
3.5.3.1 Human-operated single-agent environment 73
3.5.3.2 Multi-agent environment 75
3.5.4 Discussion 78
3.6 Conclusion 80

4 Conclusion **86**
4.1 Summary 86
4.2 Future work 87
4.2.1 Improvements in the proposed reward shaping algorithm . 87
4.2.2 Improvements in the proposed meta-optimization algorithm 88
4.2.3 Safety bottom-up multi-agent reinforcement learning . . . 88

Acknowledgements **90**

References **91**

Publication List **102**

List of Figures

1.1 Overview of contribution 3

2.1 Concept of the proposed algorithm: the interests among all the agents are assumed to be cooperative (orange lines); in the proposal, they are unknown (dashed gray lines) at first, but are estimated online as correlation coefficients of the agents' rewards; as a result, it is expected that all the agents correctly classify their interests from competitive (green lines) to cooperative (orange lines) ones. 8

2.2 Two types of configurations of MARL: (a) in top-down MARL, a reward generation system is centralized to achieve a common task explicitly; (b) in bottom-up MARL, all the agents have their own reward functions corresponding to individual tasks; the bottom-up MARL is a scalable configuration due to no centralized systems, although it requires to achieve group behaviors implicitly. 13

2.3 Reward shaping algorithm: rewards for all the agents are predicted at first with uncertainty, which is a key factor to represent the subsets; to improve the reward prediction, an exploration bonus is added; to avoid interference with irrelevant and competitive agents, interests among the agents are classified; finally, all the values in the above are integrated as a new reward to be maximized, which would imply the group behaviors. 15

2.4 Concept of learning the state-dependent interests: by learning state-dependent interests, a method will be developed that can be applied not only to tasks with only static interests, but also to more general tasks where interests change depending on the situation. 20

2.5 Simulation environment: four agents work in the same environment; a ball bounces when it hits walls on four edges of the environment or the agents. 22
2.6 Learning curves summarizing fifty trials: (a) without the classification of the interests among the agents, all the agents could not distinguish the enemies and the allies, thereby failing to learn the competitive task; (b) thanks to the classification of the interests, they found the team organization correctly, and as a consequence, the left team defeated the right team by making full use of their body sizes. 36
2.7 Scale parameters in the policies at the final episode summarizing fifty trials: the dashed line means the initial policy scale; (a) the scales of all the agents kept large values to explore the optimal policy that does not exist; (b) thanks to the classification of the interests, the left team could find the deterministic policies for winning, on the other hand, the right team could not. 37
2.8 Correlation matrices: they were obtained by merging the agents' estimations; (a) the accurate correlation matrix was not found due to insufficient data; (b) finally, all the agents correctly distinguished the enemies and the allies. 37
2.9 Simulation environment: two agents work in the same environment; both agents move on the orbiter; the agent 2 has a personal space set. 38
2.10 Learning curves summarizing ten trials: the reward of the agent 1 decreases with the increase of rewards of the agent 2. 38
2.11 Group behaviors after learning: (1) Classified as a cooperative relationship; (1) Classified as a competitive relationship; (1) Classified as a irrelevant. 39
2.12 Simulation environment: four agents work in the same environment; the task of agents 1-3 (predators) is to catch agent 4 (prey); the task of agent 4 is to escape from agent 1-3 on the screen. . . 39
2.13 Learning curves summarizing twenty trials: the reward of the agent 4 decreases with the increase of rewards of the agents 1-3. 40

2.14 Snapshots of the demonstration task: the agents 1-3 were moving to surround the agent 4; thanks to that behaviors, escaping from the agents 1-3 is difficult for the agent 4. 40
2.15 Correlation matrices before and after learning: they were obtained by merging the agents' estimations; (a) the accurate correlation matrix was not found due to insufficient data; (b) finally, all the agents correctly distinguished the allies, and calculated the small correlation coefficients with the enemies. 41
2.16 Experimental environment: two agents are in this environment; a mobile robot can move along the front-back direction while stopping in front of an obstacle; a four-axis manipulator is connected to the obstacle by a string, so it can move the obstacle nonlinearly; each agent has a web camera to recognize AR markers. 41
2.17 Robots as agents: a two-wheeled mobile robot and a four-axis manipulator are employed as the agents in this experiment; they have web cameras to recognize AR markers, which provide distance information. 42
2.18 Obstacle design: it can be pulled by the manipulator via the string; AR markers on both sides allow the agents to detect this obstacle. 42
2.19 Snapshots of the cooperative task: the manipulator pulled the obstacle to make the mobile robot close to itself; according to that behavior, the mobile robot approached to the manipulator rapidly. 43
2.20 Snapshots of the competitive task: the manipulator prohibited the mobile robot approaching by not pulling the obstacle; the mobile robot repeated exploratory actions to find new states with high rewards, even though they did not exist. 43
2.21 Learning curves of the cooperative task: (a) the average rewards of both agents were increased as the episode went on (in particular, after 20 episodes); as a result, they emerged the cooperative behaviors as shown in Fig. 2.19; (b) the average loss of the reward predictors in both agents were decreased and became negative, that is, the reward predictors acquired the high accuracy with low variance. 44

2.22 The correlation coefficients of the cooperative task: in both agents, the correlation coefficients between themselves and the others converged around 1 within 20 episodes. 45
2.23 Learning curves of the competitive task: (a) the average rewards of only the agent 2 were increased as the episode went on (in particular, after 20 episodes); as a result, they emerged the competitive behaviors as shown in Fig. 2.20; (b) the average loss of the reward predictors in both agents were decreased and became negative, that is, the reward predictors acquired the high accuracy with low variance. 46
2.24 The correlation coefficients of the competitive task: in both agents, the correlation coefficients between themselves and the others converged around -1 within 20 episodes. 47

3.1 Illustration of the contour and the Pareto solution obtained by the linearly weighted sum: the shape of the contour line is linear; (a) the Pareto solution of the convex part of the Pareto frontier can be obtained; (b) however, the Pareto solution of the non-convex part of the Pareto frontier cannot be obtained. 58
3.2 Illustration of the contour and the Pareto solution obtained by the augmented weighted Tchebycheff scalarization: The shape of the contour line is a linear combination of the L-shaped line and the linear line; (a) the Pareto solution of the convex part of the Pareto frontier can be obtained; (b) and, the Pareto solution of the non-convex part of the Pareto frontier can also be obtained. . 59
3.3 Schematic of the proposed method: Two stochastic models are learned using the training dataset, and based on the differences between them, meta policy is simultaneously optimized under the meta objective using the validation dataset. 60
3.4 Human-operated single-agent environment: The robot is operated by an expert (human); The expert aims to land the robot on the landing field at zero speed. 65

3.5 Multi-agent environment: four agents work in the same environment; the task of agents 1–3 (predators) is to catch an agent 4 (prey); the task of the agent 4 is to run away from the agent 1-3 on the screen. 66
3.6 Learning results of α on human-operated single-agent environment and eq. (3.17): as w_{wm} increased, the α approached 1, and vice versa; the positive correlation between w_{wm} and α was captured. . 70
3.7 Learning results of the mean and worst losses on human-operated single-agent environment and eq. (3.17): (a) the mean loss was decreased as w_{wm} increased; (b) the worst loss was increased as w_{wm} increased; as a result, a model suitable for the meta-objective was learned. 71
3.8 Learning results of α on multi-agent environment and eq. (3.17): as w_{wm} increased, the α approached 1, and vice versa; the positive correlation between w_{wm} and α was captured. 72
3.9 Learning results of the mean and worst losses on multi-agent environment and eq. (3.17): (a) the mean loss was decreased as w_{wm} increased; (b) the worst loss was increased as w_{wm} increased; as a result, a model suitable for the meta-objective was learned. 73
3.10 Learning results of α on human-operated single-agent environment and eq. (3.18) with $H = 10$: α was learned to be close to 1; the worst loss was emphasized in the long-term prediction in this environment. 74
3.11 The results of the scores on human-operated single-agent environment and eq. (3.18) with $H = 10$: the score was reduced more in the case of SPMO+MO than in the case of IDMO+MO; namely, SPMO+MO improved the meta-objective. 75

3.12 Learning results of the mean and worst losses on human-operated single-agent environment and eq. (3.18) with $H = 10$: IDMO+MO obtained the large difference between the mean and worst losses, which means that the variance of the learned stochastic model was large; in contrast, SPMO+MO succeeded in keeping the difference between the mean and worst losses small, resulting in that fatal errors in long-term prediction were less likely to occur, as indicated in Fig. 3.11. 76
3.13 The comparison of the scores on human-operated single-agent environment and eq. (3.18) with $H = 10$: the scores of both SPMO+MO and IDMO+MO were reduced more in the cases of the maen loss, and only SPMO yielded results comparable to the case of the worst loss; namely, the proposed methods leads to a better Pareto solution than the conventional method, and the results suggest that convergence to the worst loss was the optimal solution. 77
3.14 Learning results of α on human-operated single-agent environment and eq. (3.18) with $H = 1$: α was adjusted to be close to 0; the mean loss was emphasized to predict only the next step ($H = 1$ with the validation dataset. 78
3.15 The results of the scores on human-operated single-agent environment and eq. (3.18) with $H = 1$: The scores obtained from the meta-optimization were comparable for both methods. 79
3.16 Learning results of the mean and worst losses on human-operated single-agent environment and eq. (3.18) with $H = 1$: since the worst loss was not needed to be minimized in this configuration, SPMO+MO ignored it to prioritize the mean loss. 80
3.17 The comparison of the scores on human-operated single-agent environment and eq. (3.18) with $H = 1$: the scores of both SPMO+MO and IDMO+MO were reduced more in the cases of the maen loss and the worst loss; namely, the proposed methods leads to a better Pareto solution than the conventional method. 81

3.18 Learning results of α on multi-agent environment and eq. (3.18) with $H = 10$: α was trained to be close to 1; the worst loss was emphasized in the long-term prediction in this environment, as in the case of the human-operated single-agent environment. 82
3.19 The results of the scores on multi-agent environment and eq. (3.18) with $H = 10$: the score was reduced more in the case of SPMO+MO than in the case of IDMO+MO, as in the case of the human-operated single-agent environment. 82
3.20 Learning results of the mean and worst losses on multi-agent environment and eq. (3.18) with $H = 10$: SPMO+MO succeeded in keeping the difference between the mean and worst losses smaller than that of IDMO+MO, as in the case of the human-operated single-agent environment. 83
3.21 The comparison of the scores on multi-agent environment and eq. (3.18) with $H = 10$: the score of SPMO+MO was almost the same level as that of in the cases of the worst loss; namely, the results suggest that convergence to the worst loss was the optimal solution. 83
3.22 Learning results of α on multi-agent environment and eq. (3.18) with $H = 1$: α was optimized towards 1 unlike the case of the human-operated single-agent environment; this result suggested that the multi-agent environment was with high uncertainty and required the larger variance to cover it. 84
3.23 The results of the scores on multi-agent environment and eq. (3.18) with $H = 1$: the scores obtained from the meta-optimization were comparable for both methods. 84
3.24 Learning results of the mean and worst losses on multi-agent environment and eq. (3.18) with $H = 1$: (a) the mean loss was kept at the same level for both methods; (b) the worst loss was also comparable for both methods. 85

3.25 The comparison of the scores on multi-agent environment and eq. (3.18) with $H = 1$: the scores of both SPMO+MO and IDMO+MO were reduced slightly in the cases of the maen loss and the worst loss; namely, the proposed methods leads to a little better Pareto solution than the conventional method. 85

List of Tables

2.1 Comparison of recent MARL methods with indices for decentralized autonomous systems: only our method satisfies all the indices introduced here. 11
2.2 Hyperparameters for each agent 23
2.3 Hyperparameters for each agent 25
2.4 Hyperparameters for each agent 29
2.5 Specifications of robot systems 30

3.1 Comparison of recent meta-optimization methods with four indices: only our method satisfies all the indices introduced here. . . 53
3.2 Reachability to non-convex solutions 62
3.3 Hyperparameters in human-operated single-agent environment . . 66
3.4 Hyperparameters in multi-agent environment 67

1 Introduction

1.1 Background

The field of distributed intelligence or distributed robotics has been studied since the late 1980s [1–3]. This trend is due to the fact that multi-agent systems (MASs), in which multiple agents work together, have several advantages over single-agent systems. MASs can be expected to solve many problems efficiently in terms of time efficiency, flexibility, and robustness [2]. In parallel with the development of such theoretical research, the application of MASs is being considered in various domains due to the increasing complexity of the real-world problems in recent years [4].

In many cases, the practical application of MASs requires learning the policies for the group behaviors. When considering a MAS such as a robot team with large scale or physical interactions, the environment becomes complex, and designing the appropriate group behaviors in advance, is difficult [5]. With the recent development in the field of machine learning, there has been a lot of research on multi-agent reinforcement learning (MARL) for MASs to learn the group behaviors based on reinforcement learning [5]. Although MARL is a promising learning method for acquiring the group behaviors in an unknown environment, an analysis of conventional methods shows that in many cases it implicitly assumes a "top-down" structure. Specifically, a common task is set for the entire MAS, or rewards, which are reinforcement signals, are distributed from a centralized system to each agent. When considering general MASs, each agent is, however, considered to have a primitive task to solve the problem in an autonomous distributed manner. In addition, a centralized system that monitors the entire system and communicates with all agents is contrary to the ideal of distributed problem solving, and is unrealistic in terms of computation and communication costs for the

large-scale MAS.

Hence, this dissertation focuses on the development of a general-purpose, highly autonomous and decentralized MARL that avoids the common task and the centralized systems. The assumption of a common task means that the learning of the group behavior is aimed at a clear goal for the entire MAS. On the other hand, the group behaviors emerged by assuming a primitive task for each agent is not necessarily clear. Depending on the contents of the primitive task given to each agent, assuming not only the case where cooperation with other agents is necessary but also the case where competitive behavior should be selected, is possible. Since the MAS is only a system whose overall capability can be improved by taking advantage of the fact that multiple agents exist in the same environment, each agent should not pursue only the primitive task. The relationships between agents, therefore, need to be acquired adaptively. Although to handle a general class of MAS with mixed cooperative and competitive relationships, is possible, the complexity of the group behaviors is expected to increase. An algorithm is needed that can take into account such the complex group behaviors, under MARL.

In addition, the absence of a centralized system poses inherent challenges in MARL. When a centralized system is not assumed, each agent has only its own state and action space, and to determine whether rewards and state transitions are due to its own state and actions, is difficult. This condition is meant to increase the uncertainty of the environment for the agent. Since multiple agents exist in the same environment in the MAS, the increase in uncertainty is directly related to safety issues. Each agent needs to execute MARL while considering the uncertainty and satisfying the safety constraints. Ensuring safety (or risk avoidance) during learning has been the focus of attention not only in MARL but also in the context of reinforcement learning. In particular, we focus on model-base reinforcement learning, which can be used for planning with the state transition model because of the need to account for uncertainty in the state transitions.

As described above, in order to abandon the common task and centralized system and propose an autonomous decentralized MARL, to cope with complex collective behavior and to ensure safety considering uncertainty, are necessary.

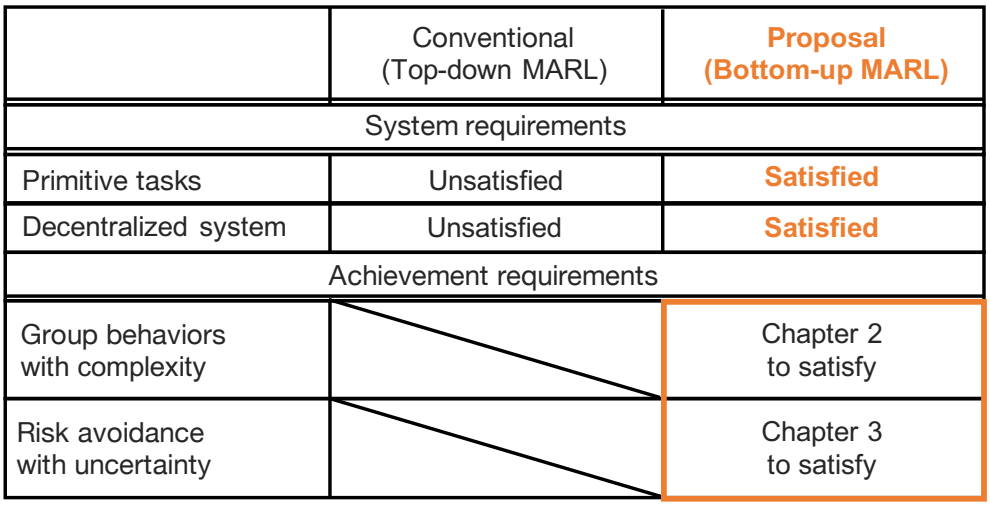


Figure 1.1: Overview of contribution

1.2 Contribution

This dissertation proposes “bottom-up MARL”, which is an algorithm that avoids a centralized system to obtain the group behaviors under the assumption that each agent has its own task. Fig. 1.1 shows the overview of the contributions of the proposal. In the bottom-up MARL framework, by definition, the system requirements "primitive tasks" and "distributed systems" are satisfied. As mentioned above, coping with complex group behaviors and risk avoidance considering uncertainty are the requirements to be satisfied. These issues are resolved step by step in Chapter 2 and Chapter 3.

The proposed framework, bottom-up MARL, intrinsically satisfies the requirements of the premitive tasks and the autonomous distributed system. However, if each agent learns a policy to accomplish only its own primitive task, the group behaviors of the MAS will not be able to achieve. As described in detail in Chapter 2, this study develops a reward shaping algorithm that allows each agent to obtain a learning reward for the group behaviors. Agents share rewards with each other and learn a reward prediction model for each agent’s own state. The proposed algorithm specifically includes a component that distinguishes between interests among the agents, which emerges by assuming the primitive tasks. One of the major contributions of this research is that the proposed algorithm en-

ables the acquisition of complex group behaviors that are emerged bottom-up based on primitive tasks. The effectiveness of the proposed method is verified by simulations and experiments with cooperative and competitive tasks.

In addition, this study considers how to deal with the uncertainty that becomes more pronounced with the elimination of centralized systems from the perspective of safety learning. Since each agent learns the policy by reinforcement learning based only on its own state and action space, the risks such as collision is high, especially in the exploration phase. Although the risk in the exploration has been pointed out in reinforcement learning for single-agent systems, it appears as a more intrinsic uncertainty in MASs where other learning agents exist in the same environment. In this study, we focus on model-based reinforcement learning, which is expected to provide safe learning by using dynamics models for planning. We specifically aims to explicitly consider the bias-variance trade-off in learning the stochastic model using state transition data. An increase in bias will result in a deterioration of the average prediction accuracy, while an increase in variance may produce outliers in the long-term prediction. As described in detail in Chapter 3, this study proposes a method to optimize the trade-off simultaneously with stochastic model learning, under a meta-objective. The multi-objective optimization problem developed in the proposed method provides a new formulation of the bias-variance decomposition, which is significant in the theoretical aspect of model learning. Another major contribution of this research is the development of a general-purpose and efficient meta-optimization method. The effectiveness of the proposed method is verified by two simulations for the environments with uncertainty due to human operation and presence of other agents.

1.3 Dissertation outline

The rest of the dissertation is as follows. Chapter 2 provides details on bottom-up MARL and the proposed reward shaping for learning the group behaviors. Chapter 3 gives the proposed meta-optimization method for adjusting the bias-variance trade-off in stochastic model learning. Finally, we summarize the dissertation and describe the future work in Chapter 4.

2 Bottom-up Multi-agent Reinforcement Learning by Reward Shaping

2.1 Introduction

As systems, which humans have to handle recently, become large-scale and complicated, their tasks to develop/achieve new outcomes are basically tackled by teams consisting of so many people. In such scenes of the real world, multiple robots are expected to alternate with humans or help us accomplish dangerous tasks as multi-agent systems (MASs) [6]: for example, coordinated transportation in a warehouse [7]; and satellite constellations for earth observation [8]. A purpose of the MAS is to resolve the tasks, where even a high-performance single robot would fail, by cooperation between agents with simple limited functions. As can be imagined, however, predesign of the MAS is infeasible due to the wide range of possibilities caused by interactions between the agents, although the cases with a few agents have been addressed [9].

To make the MAS feasible, multi-agent reinforcement learning (MARL) is a promising methodology [10], in which each agent in the MAS adaptively learns its own policy according to reinforcement learning (RL) [11], since it estimates the interactions between the agents from their experiences (not from our insufficient domain knowledge). MARL has open problems different from the normal RL with a single agent, such as how to cooperate with each other, and therefore, various algorithms have been developed and applied to simple robot systems [12, 13]. With the recent remarkable development of deep RL [14, 15], many new ideas have been further proposed to overcome the complexity of MARL: e.g., learning

communication signals between agents [16]; extending the state-of-the-art RL algorithms to the scalable MARL [17]; sharing rewards to find best one [18]; and inferring others' hidden states [19]. In particular, scalability has an important role in recent MARL so as to allow it to apply real large-scale systems as mentioned in the above.

However, such previous work basically assumes that a "top-down" MARL, where all the agents have a common task given by designers/operators. In that case, reward should be distributed to all the agents by judging their contributions (i.e., their state and action). Such a reward generator/distributor is naturally under a centralized system, which is the most fundamental obstacle to scalability of MARL. Although Lowe et al. [20] proposed a quasi-decentralized MARL by providing a server-side value function estimator, which is utilized only during learning phase, truly decentralized MAS (and MARL) would be difficult as long as the common task is explicitly given. Applying this method to agents that have their own tasks, is difficult. As another approach, shaping the reward of each agent in a form suitable for group behavior, is conceivable. Devlin et al. [21] show that the optimal group behaviors can be realized without prior knowledge of a problem domain, by using potential-based reward shaping [22], which is one of the typical reward shaping methods and the difference rewards [23] as contribution degree of the global reward. This method does not change the Nash equilibrium for the set global reward theoretically, however, the common task is still assumed.

To resolve this scalability problem, we have proposed to convert the framework of MARL from "top-down" to a "bottom-up" [24]. The "bottom-up" framework gives all the agents primitive tasks, and has no centralized systems to generate/distribute reward based on the common task, even though it does not necessarily promise to improve learning performance compared to "top-down". It is therefore regarded as a decentralized system potentially, while each agent would learn selfish policy if no tricks are implemented. As a novel trick to emerge group behaviors, the previous work has proposed a reward shaping algorithm from rewards communicated with each other [24]. The idea of this trick is to specify the part of the social reward that depends on the agent's own state, which is the same as difference rewards [23]. However, the framework of the proposed

method differs from the conventional research in that the global task is acquired by a bottom-up manner, instead of the common task given by a top-down manner. Since the proposed reward shaping algorithm does not intervene in the RL algorithm, our framework allows each agent to use arbitrary single-agent RL algorithms. Autonomy and decentralization are therefore guaranteed in that the learning algorithm used in our framework does not need to be specialized in MAS. As a further advantage, since the proposed method has higher autonomous decentralization than the conventional methods, it can be applied to tasks for which the conventional methods cannot be applied. The proposed algorithm has been verified in dynamical simulations with the heterogeneous agents under cooperative relationships.

As a fatal problem that limits applications of the MAS, the previous work implicitly assumed that it is utilized only the cooperative tasks (and agents). Only when each agent know the interests between agent in advance, the previous work on the top-down MARL succeeded in handling both cooperative and competitive tasks [20, 25]. This chapter therefore addresses the extended reward shaping algorithm to allow all the agents not to know their relationships (i.e., interests). Here, we suppose that the interests among the agents are hidden in their rewards, as implied in the literature [25]. All the agents therefore estimate their interests as correlation coefficients of the agents' rewards in the scalable manner, as illustrated in Fig. 2.1. The estimated interests can be classified into the cooperative, competitive, and irrelevant relationships, and they are integrated to the reward shaping algorithm numerically to emphasize the reward of the cooperative agents. Here, competitive relationships do not have the meaning of hostility. The reward sharing with competing agents can be assumed, since interests mean relationships between tasks that cannot be known in advance.

The effectiveness of the proposed algorithm is verified by both simulations and real robot experiments without knowledge of the interests between all the agents. In the simulations, the proposed algorithm can distinguish the enemies (i.e., the competitive agents) and the allies (i.e., the cooperative agents) simultaneously regardless of RL algorithm. In the real experiments, it enables two types of behaviors to emerge according to the given primitive tasks. As consequence, we show that the bottom-up MARL with the proposed reward shaping algorithm

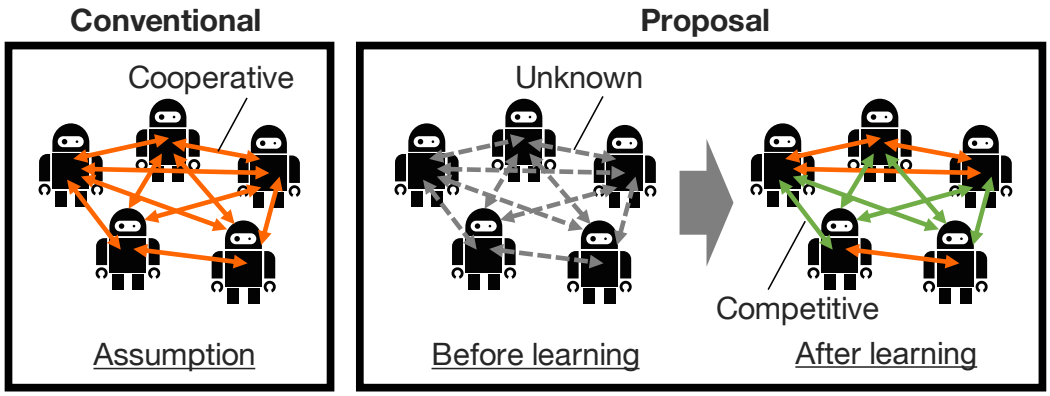


Figure 2.1: Concept of the proposed algorithm: the interests among all the agents are assumed to be cooperative (orange lines); in the proposal, they are unknown (dashed gray lines) at first, but are estimated online as correlation coefficients of the agents' rewards; as a result, it is expected that all the agents correctly classify their interests from competitive (green lines) to cooperative (orange lines) ones.

allows all the agents to acquire the selectively cooperative group behaviors in the decentralized manner.

2.2 Related work

Using a decentralized autonomous method is necessary to resolve large-scale and complex multi-agent problems in the real world. In the recent MARL domain, new methods have been developed that aim to resolve such problems. However, they have partially satisfied autonomy and/or decentralization. We consider "heterogeneous agents", "partial observation", "primitive tasks", and "decentralized learning" as indices for the decentralized autonomous systems, in this study. The importance of these indicators for decentralization of MARL has been pointed out in related work [5, 26, 27]. Based on these indices, the methods recently proposed are summarized in Table 2.1.

Heterogeneous agents: The essential issues of MARL on the decentralized autonomous systems are to handle various RL algorithms under a partially observed

Markov decision process (POMDP). Many of the conventional MARL methods only consider POMDP. The MARL methods in [19,27–30], however, allow respective agents to use multiple kinds of RL algorithms, and can handle heterogeneous agents in terms of the heterogeneity of action space. In these methods, no common parameters are assumed for agents, and only the basic framework of RL are shared. They share high-dimensional vectors (e.g. actions [19], parameters of the value network [27]), that is, in terms of communication cost, they are not scalable enough.

Partial observation: The causes of POMDP include partial state observation and probabilistic policies of other agents. The non-stationarity of the reward for the agent's own state and action is also one of the causes, in the case of each agent has a primitive task. Many conventional methods simply assume a complete observation of the state in order to avoid POMDP [19, 25, 27–31]. POMDP can therefore be relaxed even when handling individual tasks. However, complete observation of the state means observation of the environment by a centralized system, and lacks decentralization.

Primitive tasks: Instead, as shown in Table 2.1, all of the conventional methods that do not assume complete observation of the state, assumes a common task [16, 26, 32]. This assumption of a common task implicitly assumes a centralized system that sets a common reward somehow. Hence, they are also not suitable for the decentralized autonomous systems.

Decentralized learning: Finally, learning mechanism needs to be decentralized, not just during action execution, to adapt agents to the non-stationary real world. Some methods achieve decentralized learning [25–28, 31]. Zhang et al. [27] and Yang et al. [31], however, used values and actions of other agents as input of value functions, respectively. Each agent learns own policy by itself, but the cost of sharing high-dimensional data is high. Foester et al. [28] and Silva et al. [26] realized decentralized learning under the condition where the MAS contained two homogeneous agents. Tampuu et al. [25] investigated decentralized learning when interests between agents were adjusted by hand. In this study, however, no algorithm for improving the result has been presented.

Proposal: All of the recent MARL methods mentioned above have problems from the viewpoint of both autonomy and decentralization. This study therefore proposes a new MARL method for MAS that needs to satisfy all the indices listed in Table 2.1.

Our proposed method can handle heterogeneous agents by only sharing rewards that always appear in the RL algorithm and are scalar values with low communication cost [11,24]. Indeed, the total communication cost is a few bytes multiplied by the number of agents, which is smaller and more tractable than the case with state communication that may be consist of high-dimensional images. Since each agent find the parts of rewards related to itself, POMDP can be relaxed. Each agent naturally has specific primitive task. In addition, no centralized learning mechanism is required, namely, each agent optimizes own policy according to reshaped reward, which includes the other agents' rewards if they are related to it.

Our method is a kind of reward shaping and independent on RL algorithm. In other words, the latest RL algorithm can be employed easily. In fact, we have adopted one of the state-of-the-art algorithms, called Proximal Policy Optimization (PPO) [33] for RL.

Table 2.1: Comparison of recent MARL methods with indices for decentralized autonomous systems: only our method satisfies all the indices introduced here.

	Heterogeneous agent	Partial observation	Primitive tasks	Decentralized learning
Our method	✓	✓	✓	✓
Lowe et al. [20]	✓	✓	✓	
Foerster et al. [28]	✓		✓	✓
Zhang et al. [27]	✓		✓	✓
Iqbal et al. [29]	✓		✓	
Raileanu et al. [19]	✓		✓	
Lanctot et al. [30]	✓			
Foerster et al. [16]		✓		
Omidshafiei et al. [32]		✓		
Silva et al. [26]		✓		✓
Tampuu et al. [25]				✓
Yang et al. [31]				✓

2.3 Preliminaries

2.3.1 Reinforcement learning

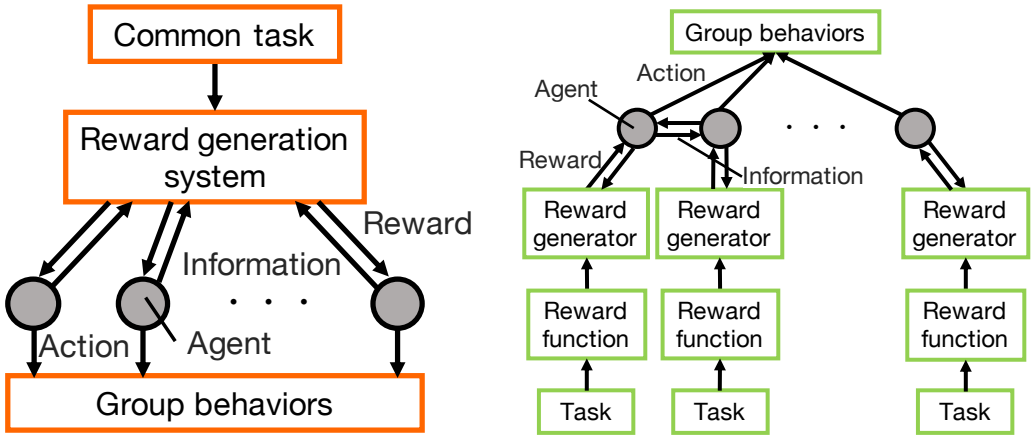
Reinforcement learning (RL) is a framework for learning the optimal policy, which maximizes the sum of rewards given over the future [11]. In reinforcement learning, an agent interacts with environment under Markov decision process (MDP) as a precondition to guarantee the agent's learning. MDP can be represented by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{S}$ is state space, $\mathcal{A}$ is action space, $\mathcal{P}$ is a set of state transition probabilities, and $\mathcal{R}$ is a reward function. The reward function is defined as $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$. In this study, the reward set mapped from $\mathcal{S}$ and $\mathcal{A}$ is represented as $\mathcal{R}(\mathcal{S}, \mathcal{A})$.

Specifically, when the agent observes a state s_t at time t, it samples an action a_t from the policy $\pi(a_t \mid s_t)$ according to s_t. The agent acts a_t on the environment, then the state transits to a new one s_{t+1} over the state transition probability $p_T(s_{t+1} \mid s_t, a_t)$, and a reward r_t is given from a reward function $r(s_t, a_t, s_{t+1})$ at the same time. The sum of rewards given over the future, so-called the return, is defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ with a discount factor $\gamma \in [0, 1)$. The agent aims to maximize R_t by optimizing the policy to π^*.

In this research, we focus on the reward shaping, which only controls $\mathcal{R}$ in MDP. That is, all the RL algorithms are acceptable. For simplicity, an actor-critic algorithm [34] combined with true online TD(λ) [35] (see the literature [36]), which is one of the representative algorithms with continuous state and action spaces, is employed.

2.3.2 Multi-agent reinforcement learning

Instead of MDP, multi-agent reinforcement learning (MARL) assumes a Markov game, where N agents exist in the same environment. That is, the Markov game can be represented by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}_1, \ldots, \mathcal{A}_N, \mathcal{P}, \mathcal{R}_1, \ldots, \mathcal{R}_N \rangle$. Here, $\mathcal{N} = \{1, \ldots, N\}$ is a set of agents, $\mathcal{A}_i$ denotes the subspace of $\mathcal{A}$ that can be handled by the agent i, and $\mathcal{R}_i$ also denotes the reward function for the agent i. $\mathcal{R}_i$ maps a subset of the reward set mapped by $\mathcal{R}$. Although the basic definition of the Markov game is given in the above tuple, each agent



(a) Top-down MARL

(b) Bottom-up MARL

Figure 2.2: Two types of configurations of MARL: (a) in top-down MARL, a reward generation system is centralized to achieve a common task explicitly; (b) in bottom-up MARL, all the agents have their own reward functions corresponding to individual tasks; the bottom-up MARL is a scalable configuration due to no centralized systems, although it requires to achieve group behaviors implicitly.

cannot observe the states of other agents generally. That is, MARL would essentially be a partial observation problem as represented by a following tuple $\langle \mathcal{S}_1, \ldots, \mathcal{S}_n, \mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{P}, \mathcal{R}_1, \ldots, \mathcal{R}_N \rangle$ with $\mathcal{S}_i$ the subspace of $\mathcal{S}$ that can be observed by the agent i.

From the point of view of each agent, in this case, the environment (i.e., the state transition $\mathcal{P}$) is regarded as non-stationary until the policies of the other agents converge on the deterministic ones, and the environment is considered non-MDP for each agent. Hence, each agent cannot learn its policy efficiently since it cannot judge whether the new state and reward is yielded by its own action. This problem is well known as concurrent learning problem [5, 26], which is one of the causes that adversely affect of MARL.

To allow the policies to converge on the deterministic ones, it is practically important for all the agent to be under the assumption of a partially observable MDP (POMDP). That is, an appropriate reward function $\mathcal{R}_i : \mathcal{S}_i \times \mathcal{A}_i \times \mathcal{S}_i \mapsto \mathbb{R}$ should be designed to understand relationship between the rewards gained and

the agent's behaviors.

In the top-down MARL as shown in Fig. 2.2(a), a centralized reward generation system distributes the appropriate reward $r_i \in \mathcal{R}_i(\mathcal{S}_i, \mathcal{A}_i)$ from a set of common rewards $\mathcal{R}(\mathcal{S}, \mathcal{A})$ indicating the group behaviors explicitly. To this end, the way to extract $\mathcal{R}_i(\mathcal{S}_i, \mathcal{A}_i)$ from $\mathcal{R}(\mathcal{S}, \mathcal{A})$ is main difficulty. This configuration is not suitable for large and complicated problems since they would need intractable calculation (exploration) time in $\mathcal{R}(\mathcal{S}, \mathcal{A})$ to extract $\mathcal{R}_i(\mathcal{S}_i, \mathcal{A}_i)$.

In contrast, our bottom-up MARL [24] as shown in Fig. 2.2(b) has $\mathcal{R}_i$ that maps $\mathcal{S}_i$ and $\mathcal{A}_i$, explicitly. It is regarded as an individual task (or objective) for the agent i. Only with this configuration, however, each agent selfishly pursues only its own reward, thereby not achieving the group behaviors. The agent therefore extracts the subset $\mathcal{R}_j^i(\mathcal{S}_i, \mathcal{A}_i) \subseteq \mathcal{R}_j(\mathcal{S}_j, \mathcal{A}_j)$, and aims to maximize the sum of rewards from $\bigcup_{j=1}^{N} \mathcal{R}_j^i(\mathcal{S}_i, \mathcal{A}_i)$. This configuration is still in POMDP and $\mathcal{R}_j^i(\mathcal{S}_i, \mathcal{A}_i)$ can be extracted in the decentralized manner with tractable calculation (exploration) time in $\mathcal{R}_j(\mathcal{S}_j, \mathcal{A}_j)$.

In summary, the bottom-up MARL would be scalable to the state and action space (i.e., the number of agents) due to its decentralized system. As a drawback, however, it has no common task for generating the group behaviors explicitly. From the next section, therefore, we propose the way to extract $\mathcal{R}_j^i(\mathcal{S}_i, \mathcal{A}_i)$ numerically and to reshape the reward for each agent, which implies the group behaviors.

2.4 Reward Shaping Algorithm for Bottom-up MARL

2.4.1 Overview

As mentioned in the above, the way to extract $\mathcal{R}_j^i(\mathcal{S}_i, \mathcal{A}_i)$ is the key issue for the bottom-up MARL. Analysis of all the subsets in offline is however undesired from a practical point of view. All the agent needs to extract them numerically from a few communication with each other to maintain the decentralized system. We have proposed a reward shaping algorithm to extract $\mathcal{R}_j^i(\mathcal{S}_i, \mathcal{A}_i)$, which is utilized to reshape the reward to be maximized [24]. This study further improves

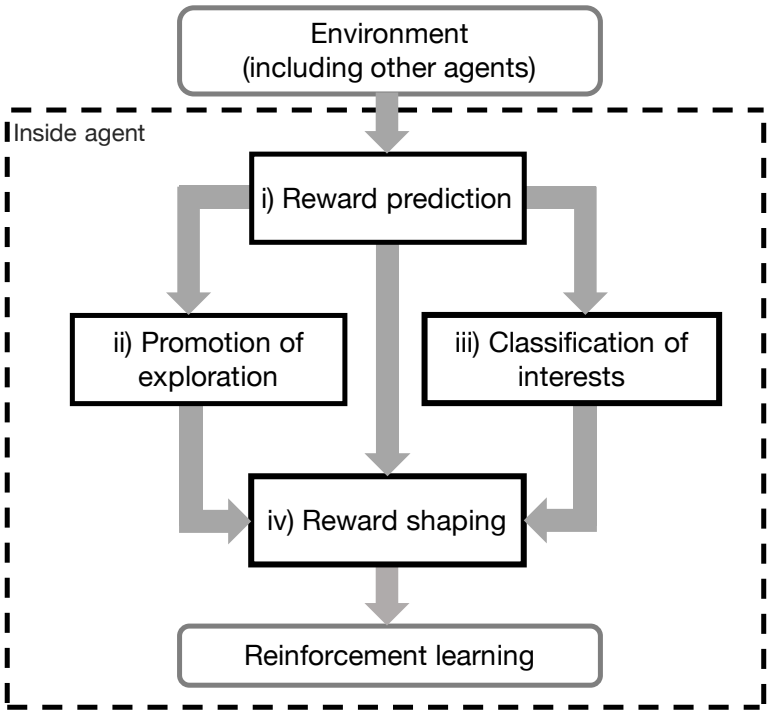


Figure 2.3: Reward shaping algorithm: rewards for all the agents are predicted at first with uncertainty, which is a key factor to represent the subsets; to improve the reward prediction, an exploration bonus is added; to avoid interference with irrelevant and competitive agents, interests among the agents are classified; finally, all the values in the above are integrated as a new reward to be maximized, which would imply the group behaviors.

the previous algorithm by considering interests among the other agents to avoid interference with irrelevant and competitive objectives. The improved algorithm is illustrated in Fig. 2.3 and its pseudo code is summarized in Alg. 1.

Here, let us briefly follow roles of respective components in the algorithm in order (details are from the next section). As a prerequisite, all the agents can communicate only their rewards to each other (although not always necessary). According to this communication, the agent i learns *i) a stochastic predictor of all the agents' rewards* with regard to its state (and its action). Here, the prediction uncertainty is revealed as variance, so if the predicted reward is sampled

from $\mathcal{R}_j^i(\mathcal{S}_i, \mathcal{A}_i)$, it should be with low variance; otherwise, with high variance. This uncertainty however consists of a lack of learning and the information from which the reward is sampled. To eliminate adverse effects of the lack of learning, *ii) an exploration bonus* is added in practice. Next, the rewards from irrelevant and competitive agents would be obstacles to learn the group behaviors. They are therefore identified based on *iii) correlations between the predicted rewards*. Finally, *iv) a reshaped reward for the agent i* is derived according to the above three types of information. This reward shaping implicitly and numerically extracts (or prioritizes) $\mathcal{R}_j^i(\mathcal{S}_i, \mathcal{A}_i)$ while avoiding the adverse effects by (ii) and (iii).

2.4.2 i) Reward prediction

In this component, the agent i predicts the rewards of all the agents $\{1, \ldots, N\}$ at every time t, $\boldsymbol{r}_t = [r_{1,t}, \ldots, r_{N,t}]^\top$, as $\hat{\boldsymbol{r}}_{i,t}$ using a neural network (NN) with regard to its own state $s_{i,t} \in \mathcal{S}_i$ (and action $a_{i,t} \in \mathcal{A}_i$). As a stochastic output from NN, student-t distribution $\mathcal{T}$ parameterized by location $\boldsymbol{\mu} \in \mathbb{R}^N$, scale $\boldsymbol{\sigma} \in \mathbb{R}_+^N$, and degree of freedom $\nu \in \mathbb{R}_+$ is employed as follows:

$$\begin{aligned} \hat{\boldsymbol{r}}_{i,t} &\sim p(\boldsymbol{r} \mid s_{i,t}; \eta_i) \\ &= \mathcal{T}(\boldsymbol{r} \mid \boldsymbol{\mu}(s_{i,t}; \eta_i), \mathrm{diag}(\boldsymbol{\sigma}^2(s_{i,t}; \eta_i)), \nu(s_{i,t}; \eta_i)) \end{aligned} \tag{2.1}$$

where η_i denotes parameter set of NN.

All the agents can communicate the rewards with communication noise $\boldsymbol{\epsilon}$, $\tilde{\boldsymbol{r}}_{i,t} = \boldsymbol{r}_t + \boldsymbol{\epsilon}$, with each other. $\tilde{\boldsymbol{r}}_{i,t}$ are utilized as supervisory signals to update NN. A loss function of NN $\mathcal{L}_i$ to be minimized by updating η_i is given by negative log likelihood.

$$\mathcal{L}_{i,t} = \sum_k -\log p(\tilde{\boldsymbol{r}}_{i,k} \mid s_{i,k}; \eta_i) \tag{2.2}$$

Although it is acceptable to store all the communicated rewards from the beginning, in this study, only the latest ones are used as streaming data.

2.4.3 ii) Exploration bonus

This component aims to eliminate the variance caused by the lack of learning. To this end, we focus on the fact that the parameter to be optimized would be largely

Algorithm 1 Learning algorithm for agent i

Initialize reward predictor η_i and policy π_i
for $n \leftarrow 1, N_{episodes}$ **do**
    Initialize state $s_{i,0}$ and time $t = 0$
    while NOT meet end conditions **do**
        Sample action $a_{i,t} \sim \pi_i(a \mid s_{i,t})$
        Execute action $a_{i,t}$
        Observe state $s_{i,t+1}$ and reward $r_{i,t}$
        Publish reward $r_{i,t}$ to other agents
        if $t = 0$ **then**
            $\tilde{\boldsymbol{r}}_{i,t-1} \leftarrow \mathbf{0}$
        else
            Subscribe rewards $\tilde{\boldsymbol{r}}_{i,t-1}$
        end if
        i) Predict rewards $\hat{\boldsymbol{r}}_{i,t}$ according to eq. (2.1)
        ii) Calculate exploration bonus $\bar{\rho}_{i,t}$ according to eqs. (2.3)–(2.5)
        iii) Calculate correlation coefficients $\boldsymbol{\zeta}_{i,t}$ according to eq. (2.6)
        iv) Calculate utility $u_{i,t}$ according to eq. (2.8)
        Update policy π_i using $(s_{i,t}, a_{i,t}, s_{i,t+1}, u_{i,t})$
        Update reward predictor η_i using $\tilde{\boldsymbol{r}}_{i,t-1}$ and eq. (2.2)
        Increment time $t+ = 1$
    end while
end for

changed before convergence of learning. In other words, divergence between the stochastic models represented by η_i^{old} the parameter set one step before and η_i would be large during such periods. By giving an exploration bonus to such periods (and such states), the lack of learning would be resolved.

According to this fact, we define the following exploration bonus $\bar{\rho}_{i,t}$ based on

Kullback-Leibler divergence $\mathrm{KL}(\cdot \mid \cdot)$ like other literature [37].

$$\bar{\rho}_{i,t} = \frac{\alpha_1}{\max(1, \beta_{i,t})} \rho_{i,t} \tag{2.3}$$

$$\rho_{i,t} = \mathrm{KL}(p(\cdot \mid s_{i,t}; \eta_i) \mid p(\cdot \mid s_{i,t}; \eta_i^{\mathrm{old}})) \tag{2.4}$$

$$\beta_{i,t} = \alpha_2 \beta_{i,t-1} + (1 - \alpha_2) \rho_{i,t} \tag{2.5}$$

where β means the scaling factor to suppress the too much large exploration bonus. α_1 and α_2 denote hyperparameters that represent the scale of the exploration bonus and the ratio of the exponential moving average, respectively.

2.4.4 iii) Classification of interests

The tasks of all the agents are not necessarily cooperative, and there is the possibility that they are irrelevant or competitive and would interfere with each other. Alternatively, competitive tasks such as sports competition are also considered. To success in achieving the group behaviors even in such cases, active consideration of interests among the agents is effective as proposed in this component.

Here, the tasks of all the agents are represented by their rewards, although all the agents do not necessarily gain them via communication. Instead, all the agents have the predicted rewards, which can be used to calculate the correlation coefficients between them.

Static interests: At time t, therefore, the agent i performs online calculation of correlation coefficients between its reward and j-th agent's one according to the following equation [38].

$$\zeta_{i,j,t} = \frac{\bar{\sigma}^2_{i,j,t}}{\bar{\sigma}^2_{i,i,t} \bar{\sigma}^2_{j,j,t}} \tag{2.6}$$

where,

$$\bar{\sigma}^2_{i,j,t} = \frac{1}{\bar{t}} \{(\bar{t} - 1) \bar{\sigma}^2_{i,j,t-1} + (\mu_j(s_{i,t}; \eta_i) - \bar{\mu}_{i,j,t-1})^2\}$$

$$\bar{\mu}_{i,j,t} = \frac{1}{\bar{t}} \{(\bar{t} - 1) \mu_{i,j,t-1} + \mu_j(s_{i,t}; \eta_i)\}$$

$$\bar{t} = \min(t, t_{\max})$$

$\bar{t}$ restricts the maximum number of samples (i.e., $t_{\max}$) for the above calculation to mitigate adverse effects of the bad prediction accuracy at the early stage of learning.

According to $\zeta_{i,j,t}$, we can easily classify the interest of the agent j from the viewpoint of the agent i.

1. $\zeta_{i,j,t} \simeq 1$: cooperative relationship
2. $\zeta_{i,j,t} \simeq 0$: irrelevant relationship
3. $\zeta_{i,j,t} \simeq -1$: competitive relationship

Note that, due to the use of the predicted reward, $\zeta_{i,j,t}$ is not equal to $\zeta_{j,i,t}$, although it is natural even for human to have asymmetry in the interest from limited perception.

State-dependent interests: The interests are not only static, but can also change depending on the situation (state). In the above method, the correlation coefficients are calculated based on the predicted rewards for all states reached during learning, and the interests of each state are not classified.

Therefore, a method is proposed to classify state-dependent interests in each agent, as shown in Fig. 2.4. The same neural network as the reward predictor shown in Section 2.4.2 is used to learn the correlation coefficients between each agent with respect to the agent's own state. The correlation coefficients using the mean $\mu_j(s_{i,t};\eta_i)$ $(j = 1, \ldots, N)$ of predicted reward by the agent i can be calculated sequentially with eq. 2.6. By adjusting $t_{\max}$, emphasizing the prediction $\mu_j(s_{i,t};\eta_i)$ for the current state is possible. Furthermore, by using the exponential moving average under the parameter c_ζ, the teacher signal $\bar{\zeta}_{i,j,t}$ of the correlation coefficient can be obtained using the previous information, as shown in the following equation.

$$\bar{\zeta}_{i,j,t} = c_\zeta \cdot \bar{\zeta}_{i,j,t-1} + (1 - c_\zeta) \cdot \zeta_j(s_{i,t};\eta_i) \tag{2.7}$$

In order to use the outputs of the neural network to be trained as the state-dependent correlation coefficients, the outputs are normalized by the tanh function.

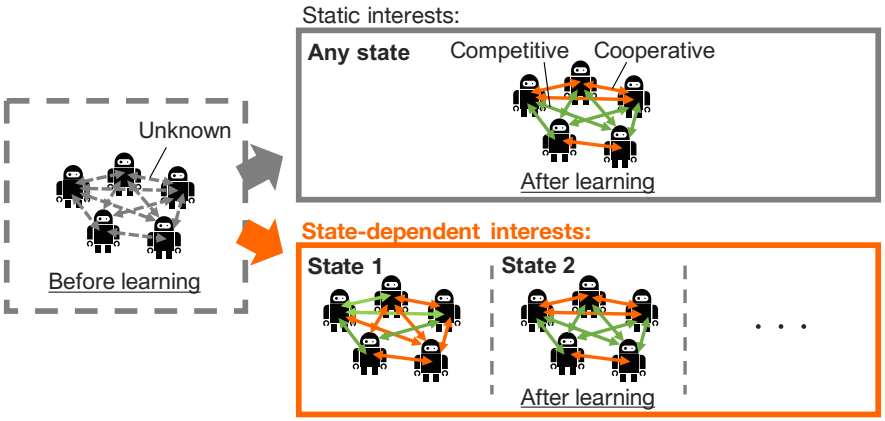


Figure 2.4: Concept of learning the state-dependent interests: by learning state-dependent interests, a method will be developed that can be applied not only to tasks with only static interests, but also to more general tasks where interests change depending on the situation.

2.4.5 iv) Reward shaping

This component integrates all the return values from the above three components: i) the predicted rewards $\boldsymbol{\mu}(s_{i,t};\eta_i)$ with their uncertainties $\boldsymbol{\sigma}(s_{i,t};\eta_i)$ (and $\nu(s_{i,t};\eta_i)$); ii) the exploration bonus $\bar{\rho}_{i,t}$; and iii) the correlation coefficients $\boldsymbol{\zeta_{i,t}} = [\zeta_{i,1,t},\ldots,\zeta_{i,N,t}]^\top$. In particular, $\boldsymbol{\sigma}(s_{i,t};\eta_i)$ would have an important role to numerically extract (or prioritize) the subsets of the j-th ($j = 1,\ldots,N$) agent's reward set for the agent i, $\mathcal{R}^i_j(\mathcal{S}_i,\mathcal{A}_i)$. This is because the reward sampled from $\mathcal{R}^i_j(\mathcal{S}_i,\mathcal{A}_i)$ has low variance; otherwise, high variance.

This concept can be implemented according to a random effects model [39], which is a method to generate a new mean that takes into account the differences in the populations of samples and gives priority to the means with the small variances. Combining it and the other values, $\bar{\rho}_{i,t}$ and $\boldsymbol{\zeta_{i,t}}$, a new reward to be maximized by RL, so-called utility $u_{i,t}$, is given as follows:

$$u_{i,t} = \frac{(\boldsymbol{w}^*_{i,t})^\top \boldsymbol{\mu}^*_{i,t}}{\mathbf{1}^\top \boldsymbol{w}^*_{i,t}} + \bar{\rho}_{i,t} \tag{2.8}$$

where,

$$
\begin{aligned}
\boldsymbol{\mu}^*_{i,t} &= \boldsymbol{\zeta_{i,t}} \odot \boldsymbol{\mu}(s_{i,t};\eta_i) \\
\boldsymbol{w}^*_{i,t} &= \frac{1}{(\boldsymbol{w}'_{i,t})^{-1} + \tau^2_{i,t}} \\
\boldsymbol{w}'_{i,t} &= \left\{ \frac{\nu(s_{i,t};\eta_i)}{\nu(s_{i,t};\eta_i) - 2} \boldsymbol{\sigma}^2(s_{i,t};\eta_i) \right\}^{-1} \\
\tau_{i,t} &= \frac{(\boldsymbol{w}'_{i,t})^\top \left(\boldsymbol{\mu}^*_{i,t} - \frac{\mathbf{1}^\top \boldsymbol{\mu}^*_{i,t}}{N} \right)^2 - (N-1)}{\mathbf{1}^\top \boldsymbol{w}'_{i,t} - \frac{\|\boldsymbol{w}'_{i,t}\|_2^2}{\mathbf{1}^\top \boldsymbol{w}'_{i,t}}}
\end{aligned}
$$

Please note that $\odot$ denotes element-wise multiplication and $\mathbf{1}^\top \boldsymbol{x}$ denotes the summation of vector $\boldsymbol{x}$. In short, $u_{i,t}$ prioritizes the rewards of the agents with clear interests and high confidence, and ignores others.

2.5 Numerical simulations

Since this study assumes application to the real-world MAS, decentralized learning is supposed for all the following tasks.

2.5.1 Verification of ability to classify static interests

From this section, the effectiveness of the proposed reward shaping algorithm is verified. First, a simulation environment with four agents (robots) and a ball (see Fig. 2.5) is developed based on OpenAI Gym [40]. It can be downloaded from Github: `https://github.com/kbys-t/gym_MA`. Note that the components except iii) the classification of the interests among the agents have been verified in previous work [24], and therefore, this study particularly focuses on it. This section deals with static interests.

2.5.1.1 Conditions

The state and action spaces for all the agents are the same for simplicity, although heterogeneous spaces are acceptable. The state space is six-dimensional: two-dimensional position of the agent; and two-dimensional position and velocity

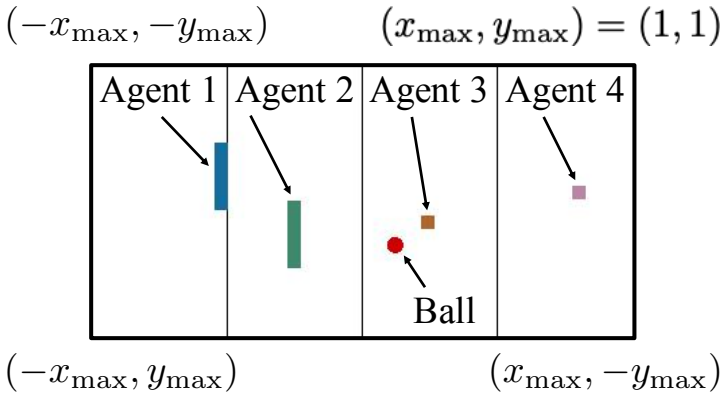


Figure 2.5: Simulation environment: four agents work in the same environment; a ball bounces when it hits walls on four edges of the environment or the agents.

of the ball. The action space is two-dimensional velocity of the agent. The movable ranges of agents 1–4 in the x-axis direction are set to $[-x_{\max}, -x_{\max}/2]$, $[-x_{\max}/2, 0]$, $[0, x_{\max}/2]$, and $[x_{\max}/2, x_{\max}]$, respectively. Since the positions of the other agents cannot be observed, this task is under "partial observation" (see Table 2.1).

For "primitive tasks" shown in Table 2.1, the agents' tasks and the corresponding reward functions are respectively given as follows:

- A left team (i.e., agents 1 and 2) aims to push a ball to the right side, namely, its reward function is defined using x-axis position of the ball, x_{ball}, and a half width of the simulation area, $x_{\max}$.

$$r_{\text{left}} = \frac{x_{\text{ball}}}{x_{\max}} \tag{2.9}$$

- A right team (i.e., agents 3 and 4) aims to push the ball to the left side with the following reward:

$$r_{\text{right}} = -\frac{x_{\text{ball}}}{x_{\max}} \tag{2.10}$$

As can be seen in the above, the intra-team agents are in cooperative relationships and the inter-team agents are in competitive relationship, although all the agents do not know such relationships in advance. In addition, such reward design

Table 2.2: Hyperparameters for each agent

Symbol	Meaning	Value
N_{res}	Number of neurons in reservoir computing	700
α	Learning rate of reservoir computing	0.0005
γ	Discount factor of the return	0.99
α_1	Scale of the exploration bonus	0.5
α_2	Moving average weight of the exploration bonus	0.9
$t_{\max}$	Maximum time for calculation of correlation coefficients	100

yields zero-sum game, so they will fail to learn special policies if they cannot consider/find the interests among them.

In addition to the above conditions, the sizes of the left team's agents are larger than ones of the right team's agents. Therefore, MAS should satisfy "heterogeneous agents" in Table 2.1 in this sense. Since the left team is easier to hit the ball than the right team, we expect that the left team will win the right team if they appropriately learn their policies from the utility while considering the interests among the agents.

As shown above, this task is one of the simple examples to which the conventional methods cannot be applied because it requires three indicators in Table 2.1 for MARL, and only our method satisfies all of them.

Each agent learns its own policy to maximize the return (i.e., the sum of the utility). To this end, each agent approximate its policy with θ and value function with ϕ according to the actor-critic algorithm combined with true online TD(λ) [36] in addition to the reward shaping algorithm with η. The parameter sets θ, ϕ, and η are given as reservoir computing [41] with N_{res} neurons. In this study, the reservoir computing updates its parameters by stochastic gradient decent with α learning rate to minimize the loss function. Table 2.2 summarizes the hyperparameters used for the implemented algorithms. They were designed based on the general knowledge about reinforcement learning [11], reservoir computing [41], and our previous work [24]. Only $t_{\max}$ is additionally given in this study (the effect of its setting is discussed later).

2.5.1.2 Results

To mainly verify the component for the classification of the interests, two kinds of simulations are performed: with/without that component. In both cases, the maximum time in every episode is 100, and the number of total episodes is 500. If the ball position is $\mid x_{\text{ball}} \mid > 0.9x_{\max}$, the episode will end even if the time is less than the maximum time. That is of one trial, and in total, fifty trials are performed for each.

The learning curves and the 95% confidence intervals of respective agents were depicted in Figs. 2.6(a) and (b). The average of the policy scale in the final episode were shown in Figs. 2.7(a) and (b). As can be seen in the case without the classification of the interests (see Fig. 2.6(a)), all the agents did not gain the positive rewards stably. This is because the reward functions given to the agents represent a zero-sum game, and therefore, the utility without the correlation coefficients would always be almost zero. In other words, all the agents attempted to make the others including the enemy team win, although there were no optimal policies satisfying such objectives. As a result, the policy of each agent did not converge to deterministic one as shown in Fig. 2.7(a).

In contrast, thanks to the classification of the interests, the average rewards of the agents 1 and 2 (i.e., the left team) converged to higher values than the agents 3 and 4 (i.e., the right team), as shown in Fig. 2.6(b). This might be due to the correct estimation of the interests among the agents. Actually, Figs. 2.8(a) and (b) showed the matrix that combined the correlation coefficients estimated by the agents at the first and final episodes, respectively. We found that all the agents successfully distinguished the enemies (i.e., $\zeta_{i,j,t} \simeq -1$) and the allies (i.e., $\zeta_{i,j,t} \simeq 1$). That is, according to the utility defined in eq. (2.8), the agents learned their policies in cooperation with the allies to defeat the enemies, and the left team succeeded in doing so by making full use of their body sizes. As a result, the policies of the left team converged to the deterministic one for winning, but the right team were still exploring the policies, as shown in Fig. 2.7(b). For agents 1 and 2, the differences between the scales of the cases without and with the classification of the interests were verified by t-test. As a result, the p-values were sufficiently smaller than 0.05 (agent 1: $p \simeq 3 \times 10^{-25}$, agent 2: $p \simeq 1 \times 10^{-26}$), and a significant differences were confirmed.

Table 2.3: Hyperparameters for each agent

Symbol	Meaning	Value
N_{res}	Number of neurons in reservoir computing	700
α	Learning rate of reservoir computing	0.0005
γ	Discount factor of the return	0.99
α_1	Scale of the exploration bonus	0.5
α_2	Moving average weight of the exploration bonus	0.9
$t_{\max}$	Maximum time for calculation of correlation coefficients	3
c_ζ	Hyperparameter for exponential moving averages	0.99

In summary, the component for the classification of the interests among the agents allows the agents without any knowledge about the other agents to numerically estimate the interests between themselves and the others as the correlation coefficients. Such classification is reflected to the utility to be maximized by RL. As a result, the agents emerge the group behaviors in cooperation with the cooperative agents while defeating/ignoring the competitive/irrelevant agents.

2.5.2 Valification of effectiveness to learn state-dependent interests

Next, the effectiveness of the learning method for state-dependent interests is valified. In the conventional context of MARL, the common task are mostly assumed, and state-dependent interests have not been sufficiently considered. In the original simulation environment shown below, the task that require selective cooperation is set. Therefore, to distinguish the interests appropriately depending on the state, is necessary.

2.5.2.1 Conditions

The simulation environment is shown in Fig. 2.9. Each agent's state is its own absolute position and velocity and the relative position of the other, and its action is acceleration. The absolute values of the maximum velocity and acceleration of the agent 2 are larger than those of the agent 1. The agent 1's task is to

reduce the distance to agent 2. The agent 2's task is also basically to reduce the distance between agents, but in personal space, the smaller the distance, the lower the reward. Therefore, the interests of both agents are cooperative outside the personal space and competitive inside. Without a state-dependent distinction of interests, the MAS cannot achieve a group behaviors that takes into account each task, with both agents approaching each other outside the personal space and the agent 2 moving away inside. Table 2.3 summarizes the hyperparameters used for the implemented algorithms.

2.5.2.2 Results

The maximum time in every episode is 700, and the number of total episodes is 500. This one trial are repeated ten times.

The learning curves and the 95% confidence intervals of respective agents were depicted in Fig. 2.10. Fig. 2.10 shows that the agent 2's reward has improved, and the two agents are able to keep close to the outer loop of the personal space. On the other hand, the agent 1's reward is decreasing, which confirms that the agent 2 with higher ability is prioritizing its own task.

The transition between the environment and the prediction of the correlation coefficient by the agent 2 in the final episode is shown in Fig.2.11. Fig.2.11 shows that the correlation coefficient is predicted to be close to $+1$ when the agent 1 is outside the personal space (as shown in the scene (1)), and close to -1 when it is inside the personal space (as shown in the scene (2)). In the steady state, the distance is about the same as the outer loop of the personal space, as shown in the scene (3).

These results show numerically that the proposed method can predict the interests that change depending on the state.

2.5.3 Verification of versatility

Since the proposed method is for reshaping rewards in the decentralized manner, it is versatile for MAS problems (e.g., whether state-action spaces are discrete or continuous) and the algorithms to learn the agents' policies. In this section, therefore, the proposed method is additionally applied to Predetor-Prey problem,

which is one of the benchmarks for MAS [20, 42–44]. This problem has discrete action space, and therefore, Deep Q-Network (DQN) [14] can be applied instead of the actor-critic algorithm we used in the above.

2.5.3.1 Conditions

This environment based on [20] consists of four agents and two objects, which are randomly placed at the beginning of each episode, under our setup (see Fig. 2.12). The source codes of the environment can be downloaded from Github: `https://github.com/openai/multiagent-particle-envs`). The state space of the agents 1-3 (predators) is sixteen-dimensional: absolute position and velocity of itself; relative positions of other agents and objects; and velocity of the agent 4 (prey). The state space of the agent 4 (prey) is fourteen-dimensional: absolute position and velocity of itself; relative positions of other agents and objects. Each agent has a discrete action space that determines the moving direction (up/down/left/right) of each step. Partially observations are assumed because each agent does not share the actions of all agents and the agent 4 does not know the objects' positions.

The respective tasks and the corresponding reward functions are given as follows:

- Predators (i.e., the agents 1-3) aims to to approach and collide with agent4.

$$r_{\text{predator}} = -0.1 \cdot \sum_{i=1}^{3} \sqrt{(x_4 - x_i)^2 + (y_4 - y_i)^2} + 10 \cdot \sum_{i=1}^{3} c_i \tag{2.11}$$

  where, x_i, y_i ($i = 1, \ \ldots, \ 4$) are x-axis and y-axis position of i-th agent and c_i takes 1 when the agent 4 collides with i-th agent (otherwise, 0).

- Prey (i.e., the agent 4) aims to avoid the collisions with the predators and

to stay in the screen.

$$
\begin{aligned}
r_{\text{prey}} = & -10 \cdot \sum_{i=1}^{3} c_i \\
& - \begin{cases} 0 & (|x_4| < 0.9) \\ 10 \cdot (|x_4| - 0.9) & (|x_4| < 1) \\ \min(\exp(2|x_4| - 2), 10) & (|x_4| \geq 1) \end{cases} \\
& - \begin{cases} 0 & (|y_4| < 0.9) \\ 10 \cdot (|y_4| - 0.9) & (|y_4| < 1) \\ \min(\exp(2|y_4| - 2), 10) & (|y_4| \geq 1) \end{cases}
\end{aligned} \tag{2.12}
$$

where, the second and third terms are for the penalty if the prey goes outside of the screen.

To solve this task without the knowledge of these primitive tasks, only our method satisfies the conditions in Table 2.1.

Table 2.4 summarizes the hyperparameters used for the learning. The other hyperparameters are the same as the cases for the above simulation summarized in Table 2.2. Reservoir computing is used for the reward prediction, and DQN is used for the learning the action-value function, which can be converted to the policy. The rewards in the batch for DQN are obtained by the latest reward prediction network.

2.5.3.2 Results

The learning results are shown in Fig. 2.13. The maximum time in every episode is 30, and the number of total episodes is 250. As shown in Fig. 2.13, the rewards of the agents 1-3 (i.e., the predators) were increased. The reward of the agent 4, the prey, was decreased. That is, the predators could collide with the prey somehow. Indeed, the snapshots of the environment in the final episode are shown in Fig. 2.14. The predators seemed to cooperate with each other well and finally caught the prey.

To confirm whether the predators were in cooperation, the correlation matrices before and after learning are illustrated in Figs. 2.15(a) and (b). As expected,

Table 2.4: Hyperparameters for each agent

Symbol	Meaning	Value
N_{layer}	Number of DQN layers	3
N_{dqn}	Number of neurons in each DQN hidden layer	100
α_{dqn}	Learning rate of DQN	0.0005

the predators were recognized as cooperative relationships with each other. In contrast, the the correlation coefficients between the predators and the prey were enough small to ignore the predators' rewards, but still positive. That is because the reward does not always have a negative correlation in this task. For example, when either of the predators and the prey are close enough without collision in the screen, both of them get no rewards. The correlation would be positive when the prey moves outside of the screen while being away from the predators.

2.6 Experiments

We verify that the proposed algorithm works properly and learns the target tasks even in the MAS with real robots. Since this study assumes application to the real-world MAS, decentralized learning is supposed for all the following tasks.

2.6.1 Environment

A whole image of the experimental environment is shown in Fig. 2.16. Two robots are prepared as the agents summarized in Table 2.5. The agent 1 is a two-wheeled mobile robot, named TurtleBot 2 (or Kobuki), developed by Yujin Robot Co. (see Fig. 2.17(a)); and the agent 2 is a four-axis manipulator, named Dobot Magician, manufactured by Dobot (see Fig. 2.17(b)). Their movement direction is restricted to one dimension (i.e., longitudinal direction) by hardware and software settings. They can detect AR markers (and their distances) via web cameras.

The state and action spaces of each agent are specified as follows:

- Agent 1: the two-wheeled mobile robot
  It has three-dimensional state space: distances from its base to the agent 2

Table 2.5: Specifications of robot systems

Robot name (Company name)	TurtleBot 2 (YUJIN ROBOT)	Dobot Magician (DOBOT)
Type	Two-wheeled mobile base	Four-axis manipulator
CPU	Intel Core i5-2557M (1.70 GHz)	Intel Core i7-7700K (4.20 GHz)
RAM	4 GB	16 GB
OS	Ubuntu 16.04	Ubuntu 16.04
Software platform	ROS Kinetic	ROS Kinetic
Sensor	Web camera (640×480 pixel, 30 fps)	Web camera (1920×1080 pixel, 30 fps)

and the obstacle; and longitudinal velocity. It controls longitudinal acceleration, namely, it has one-dimensional action space.

- Agent 2: the four-axis manipulator
  It has three-dimensional state space: distances from its base to the agent 1 and the obstacle; and position of its end-effector. The end-effector is controlled by adding movement amount as one-dimensional action space.

Note that noise would be included in the measured distance using the web camera, and therefore, exponential moving average with 0.8 a gain is applied as its low-pass filter.

Both robots are controlled using robot operating system (ROS) [45], which is a middleware making software easily communicate with hardware. ROS also allows the robots to communicate rewards with each other. That is, both robots can learn the reward predictor and the correlation coefficients in online. Note that their hyperparameters are the same as the cases for the above simulation summarized in Table 2.2 except the number of neurons in learning rate: $\alpha = 0.001$ and reservoir computing: $N_{\text{res}} = 1000$. This increase in N_{res} is because the real experiment is basically more difficult than the simulations.

An obstacle is placed between the two agents. It prohibits the mobile robot from approaching the manipulator since the robot is restricted not to push it. In contrast, it is connected to the manipulator via a string. The manipulator is

therefore able to pull it, but not to push it. This irreversibility makes given tasks introduced in the next section difficult.

2.6.2 Task settings

By giving two kinds of reward functions, following cooperative and competitive tasks are performed. Note that these tasks also need to be learned by the MARL algorithm, which satisfies all the indicators in Table 2.1, as in Section 2.5.

2.6.2.1 Cooperative task

In this case, the agents 1 and 2 aim to achieve the following tasks independently.

- The agent 1's task is to approach the agent 2, that is represented by the following reward function r^1_{coop}.

$$r^1_{\text{coop}} = 2\exp\left(-0.15d_{1,2}\right) - 1 \tag{2.13}$$

  where $d_{1,2}$ is the distance from the agent 1 to the agent 2 observed by the agent 1.

- The agent 2's task is to make the agent 1 close to itself, that is represented by the following reward function r^2_{coop}.

$$r^2_{\text{coop}} = 2\exp\left(-0.15d_{2,1}\right) - 1 \tag{2.14}$$

  where $d_{2,1}$ is the distance from the agent 2 to the agent 1 observed by the agent 2.

Note that $d_{1,2}$ and $d_{2,1}$ are theoretically in symmetric, but in practice, they would not match due to the observation noise.

The agent 1 cannot approach to the agent 2 due to the obstacle in the initial state. In addition, the agent 2 cannot gain high reward unless the agent 1 is closer to it than the initial position of the obstacle. That is, both agents will gain high rewards only if the agent 2 pulls the obstacle and the agent 1 moves forward to approach the agent 2.

2.6.2.2 Competitive task

In this case, the tasks for the agents 1 and 2 are designed as follows:

- As well as the cooperative task, the agent 1's task is to approach the agent 2, that is represented by the following reward function r^1_{comp}.

$$r^1_{\text{comp}} = 2\exp\left(-0.15d_{1,2}\right) - 1 = r^1_{\text{coop}} \tag{2.15}$$

- As oppose to the cooperative task, the agent 2's task is to prohibit the agent 1 from being close to itself, that is represented by the following reward function r^2_{comp}.

$$r^2_{\text{comp}} = r = 1 - 2\exp\left(-0.15d_{2,1}\right) = -r^2_{\text{coop}} \tag{2.16}$$

As can be seen in the above, only the reward function for the agent 2 is inverted from the cooperative task. The objectives of both agents are absolutely in conflict, but the agent 2 takes the lead since only it can move the obstacle that prohibits the agent 1 approaching. That is, it is expected that the agent 2 gain higher reward than the agent 1 due to its initiative in this competition.

2.6.3 Results

Both cooperative and competitive tasks terminate every episode after 100 time steps, and conduct 100 episodes in total. This one trial are repeated three times for each. The behaviors of the two robots after learning is illustrated in an attached video and Figs. 2.19 and 2.20, which are extracted scenes from the video.

2.6.3.1 Cooperative task

First, the experimental results for the cooperative task are shown below. The learning curves and the 95% confidence intervals of respective agents were depicted in Fig. 2.21: (a) denotes the average reward indicating the performance of RL; and (b) denotes the average loss indicating the accuracy of the reward predictor. As can be seen in Fig. 2.21(a), both agents gained the high average

reward finally, that means, they could find the cooperative way. In addition, their reward predictors gained negative losses, that means, they were with good accuracy to predict their rewards with small variance (see Fig. 2.21(b)).

These two results suggested that both robots could correctly estimate the correlation coefficients between them and consider each other to be in a cooperative relationship. To confirm this suggestion, the correlation coefficients with the episodes as x-axis were plotted in Fig. 2.22. Note again that they can be in asymmetric since each agent estimates them from the output of each reward predictor, although the true ones should be absolutely symmetric. In each agent, the correlation coefficients for itself and the other agent converged around 1, which means the cooperative relationship, within 20 episodes. Actually, after 20 episodes, the learning curves in Fig. 2.21(a) started to be increased since the group objective to be learned became clear.

2.6.3.2 Competitive task

Next, the experimental results for the competitive task are shown below. The learning curves and the 95% confidence intervals of respective agents were depicted in Fig. 2.23: as well as Fig. 2.21, (a) and (b) denote the average reward and loss, respectively. Apart from the results of the cooperative task, only the agent 2 (i.e., the four-axis manipulator) gained the high reward, and the agent 1's reward were decreased rather than the early stage of learning. The reward predictors of both agents were however with good accuracy sufficiently.

From this results, as expected in advance, it is suggested that the agent 2 noticed their competitive relationship at least, and actively blocked the approach of the agent 1. To confirm this suggestion, the correlation coefficients were again plotted in Fig. 2.24. In each agent, the correlation coefficients for itself and the other agent converged around -1, which means the competitive relationship, within 20 episodes. Note that, in the first 20 episodes, the wide confidence intervals of the average rewards imply that both agents confirmed their relationship. That is, not only the agent 1 but also the agent 2 were aware that they are in the competitive relationship. The agent 1 however took the lead as expected, and as a consequence, only the agent 1 could acquire the optimal policy to block the agent 2. In contrast, the agent 2 never gained high reward no matter how

it moves, so it repeated back and forth movement to find new states (see the attached video and Fig. 2.20).

2.6.3.3 Discussion

In summary, we verified that the proposed reward shaping algorithm can find the interests among the agents autonomously even in the experiments with real robots, and using the estimated results, all the agents can finally emerge the group behaviors suitable for the tasks given to them independently (and environment). This task is one of the cases, where the conventional methods with insufficient satisfaction of the indicators in Table 2.1 cannot be applied. In other words, it is also an example of belonging to a class that only the proposed algorithm can tackle. These results claim that the proposed reward shaping algorithm significantly improved the previous work without the classification of the interests [24]. In addition, this improvement would naturally extend the feasible applications by the MAS with the bottom-up manner to the cases even without knowledge about the interests among the agents in the MAS. This extension certainly reduces the burden on users when building large MASs.

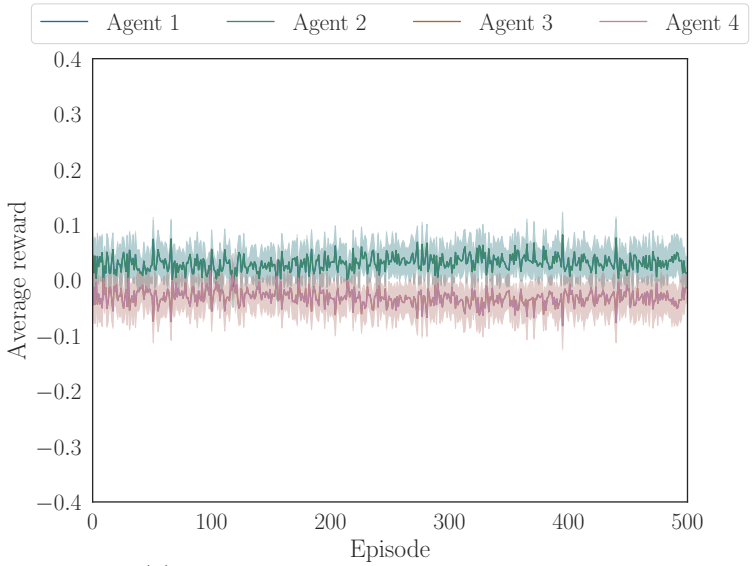
However, our proposal still have two open problems: state-dependent interests and; learning convergence and stability. Regarding the first problem, the interests between the agents in the above experiments (and the simulations) are stationary and can be regarded as constant correlation coefficients when the environment and tasks are set. Under such the stationary environments, fine-tuning of the hyperparameter $t_{\max}$ existing in the proposed method on the correlation coefficient prediction, is not needed. In real problems, however, the interests would be changed according to interactions between the agents and their situations. In such the non-stationary environments, if $t_{\max}$ is larger than the rate of change of the correlation, the change cannot be expressed; and vice versa. Assuming in advance the rate of change in the interests, is necessary in the proposed method. An idea to resolve this open problem is the classification of the interests with regard to the observed state (and the performed action), although it is difficult to learn due to no supervisory signals.

Regarding the second problem, analyzing the convergence and stability in MARL for non-stationary environments is generally difficult. The spaces used

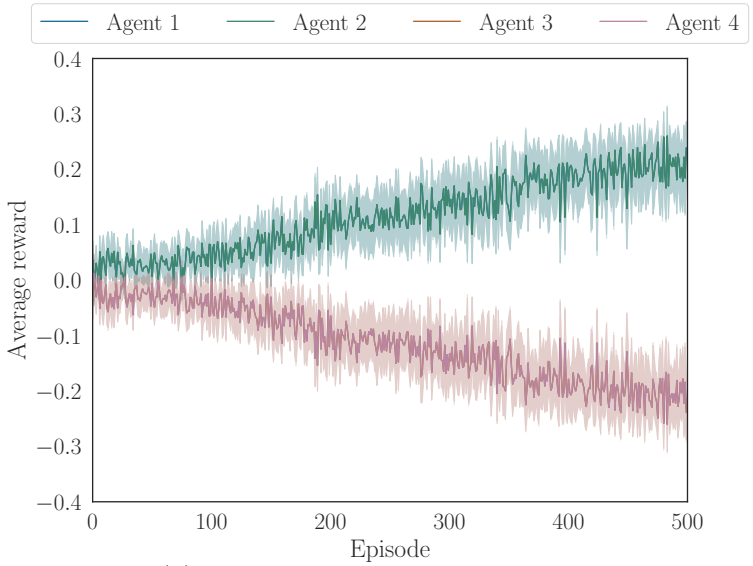
by each agent in RL are MDP (POMDP until the probabilistic policies of the other agents converge on the deterministic ones) since the reshaped reward by the proposed algorithm is finally mapped only by $\mathcal{S}_i \times \mathcal{A}_i \times \mathcal{S}_i$. At that time, it would be theoretically proved that the policy of each agent converges to the optimal policy for the reshaped reward. The proposed method, however, learns the reward prediction and the policy simultaneously. The agents, therefore, are likely to learn the policies at the stage where the accuracy of reward prediction is insufficient. This process has the risk to update the policies towards wrong directions. At least, the reward prediction should be acquired with high accuracy before making the policy get stuck in the wrong local optima. Although the reward prediction is acquired in the supervised manner, which would be faster than the case of reinforcement learning, we have to regulate the order of learning.

2.7 Conclusion

This chapter proposed the reward shaping algorithm for bottom-up MARL to achieve the group behaviors from the agents' own tasks independently given. The proposed algorithm is designed as the decentralized system as long as possible except the communication of rewards (scalar real values) between the agents, which can be omitted after learning phase. Specifically, each agent predicts the rewards of the other agents as a stochastic model with location (i.e., mean) and scale (i.e, variance) parameters by sharing the rewards during the learning. By learning this predictor, each agent knows the dependencies of the other agents' tasks on its own state (and action) from the variance of the predicted rewards, which is revealed by the exploration bonus. As a further improvement from the previous work [24] to manage the MAS with unknown interests among the agents, they are classified online based on the correlation coefficients between the means of predicted rewards. Finally, the new reward, named utility, is derived by integrating the stochastic model of the agents' rewards and the correlation coefficients. The above implementation was verified through the simulations and the real experiments. In all the cases, all the agents found the interests among the others and acquired the cooperative/competitive tasks, although the previous algorithm failed to do so.

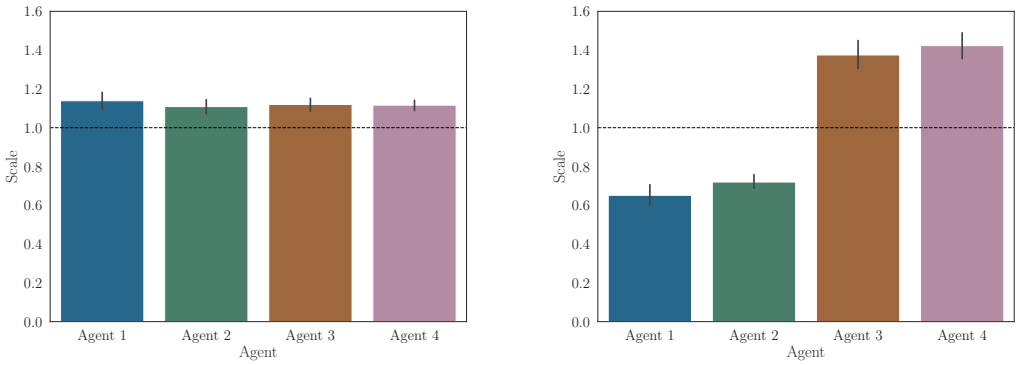


(a) Without the classification of interests



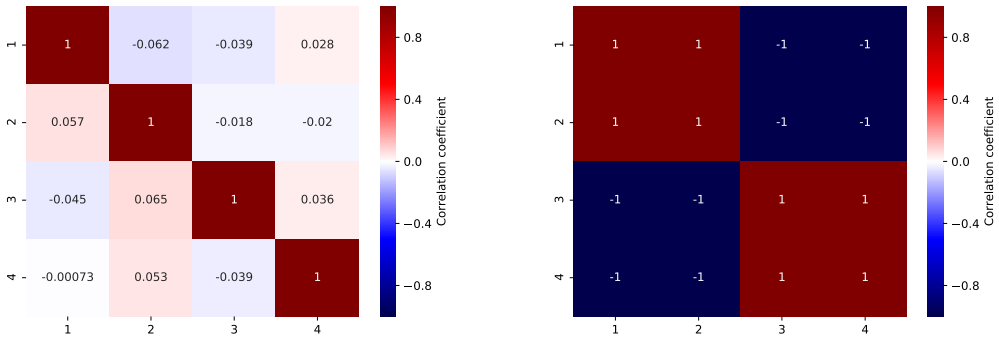
(b) With the classification of interests

Figure 2.6: Learning curves summarizing fifty trials: (a) without the classification of the interests among the agents, all the agents could not distinguish the enemies and the allies, thereby failing to learn the competitive task; (b) thanks to the classification of the interests, they found the team organization correctly, and as a consequence, the left team defeated the right team by making full use of their body sizes.



(a) Without the classification of interests (b) With the classification of interests

Figure 2.7: Scale parameters in the policies at the final episode summarizing fifty trials: the dashed line means the initial policy scale; (a) the scales of all the agents kept large values to explore the optimal policy that does not exist; (b) thanks to the classification of the interests, the left team could find the deterministic policies for winning, on the other hand, the right team could not.



(a) After first episode (b) After final episode

Figure 2.8: Correlation matrices: they were obtained by merging the agents' estimations; (a) the accurate correlation matrix was not found due to insufficient data; (b) finally, all the agents correctly distinguished the enemies and the allies.

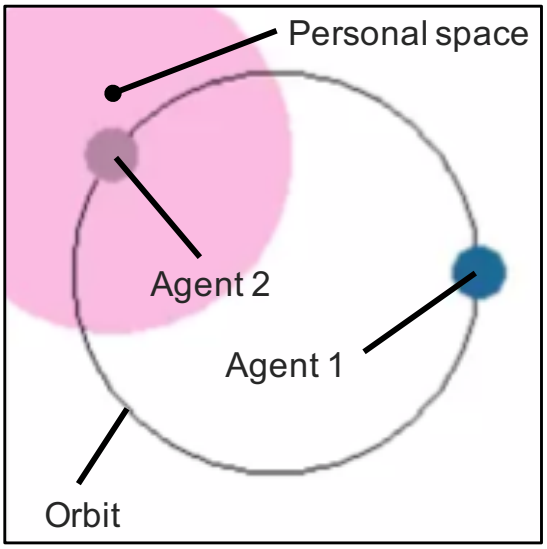


Figure 2.9: Simulation environment: two agents work in the same environment; both agents move on the orbiter; the agent 2 has a personal space set.

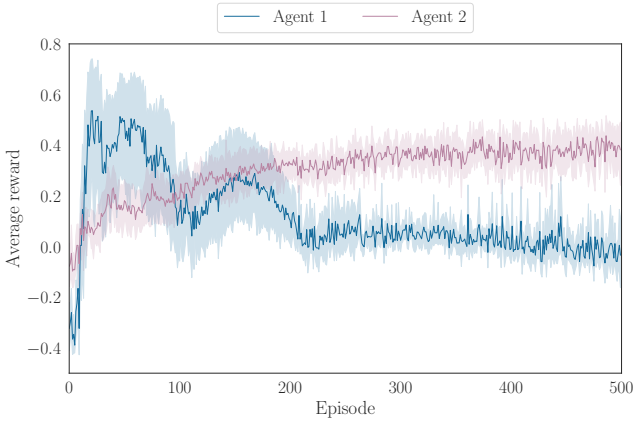


Figure 2.10: Learning curves summarizing ten trials: the reward of the agent 1 decreases with the increase of rewards of the agent 2.

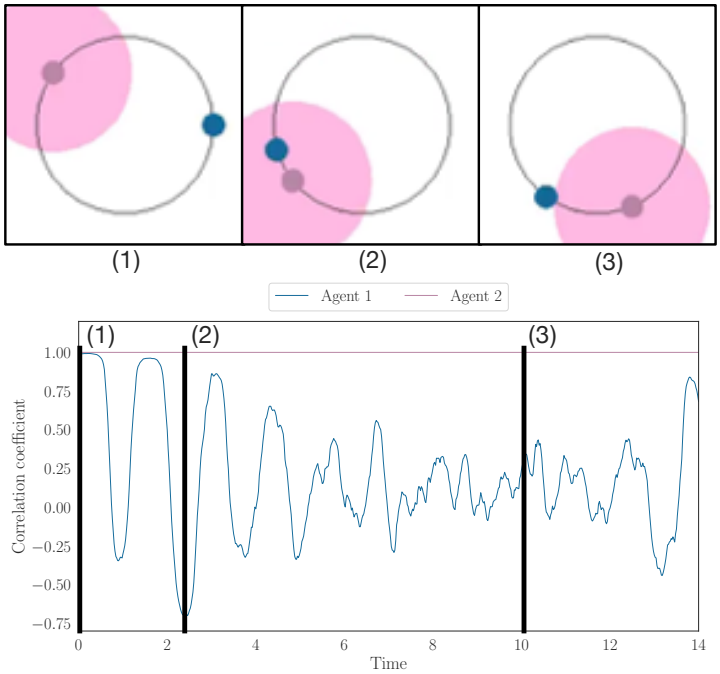


Figure 2.11: Group behaviors after learning: (1) Classified as a cooperative relationship; (1) Classified as a competitive relationship; (1) Classified as a irrelevant.

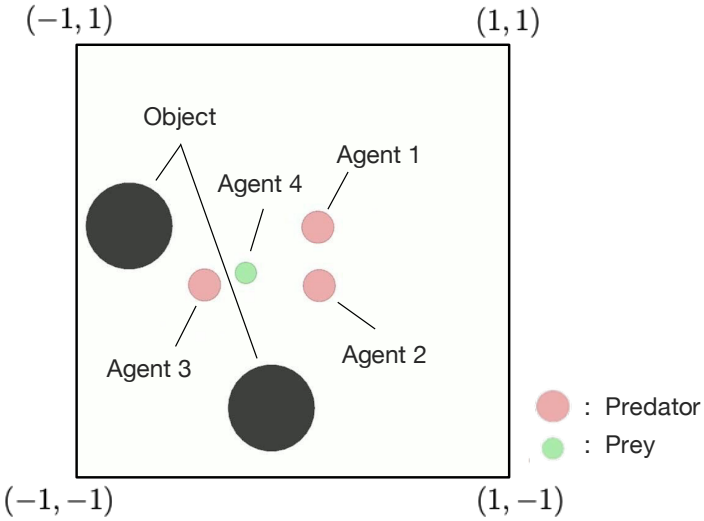


Figure 2.12: Simulation environment: four agents work in the same environment; the task of agents 1-3 (predators) is to catch agent 4 (prey); the task of agent 4 is to escape from agent 1-3 on the screen.

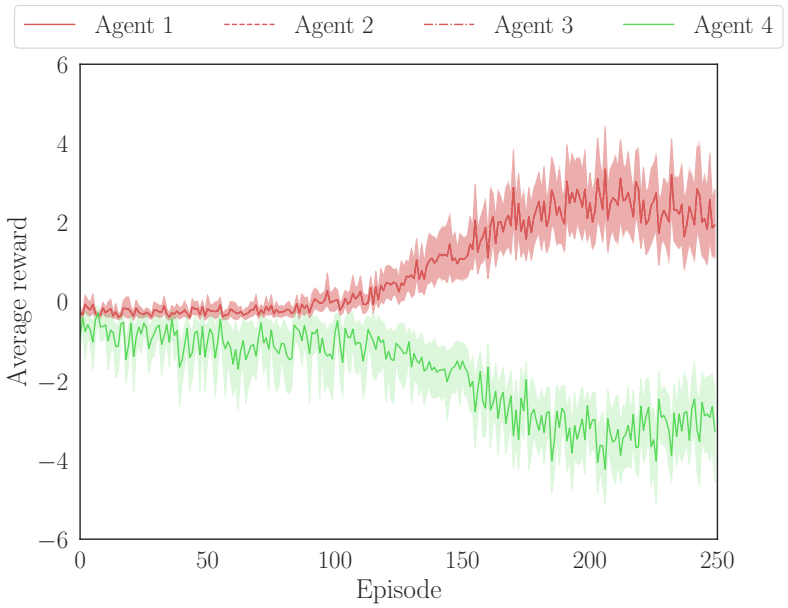


Figure 2.13: Learning curves summarizing twenty trials: the reward of the agent 4 decreases with the increase of rewards of the agents 1-3.

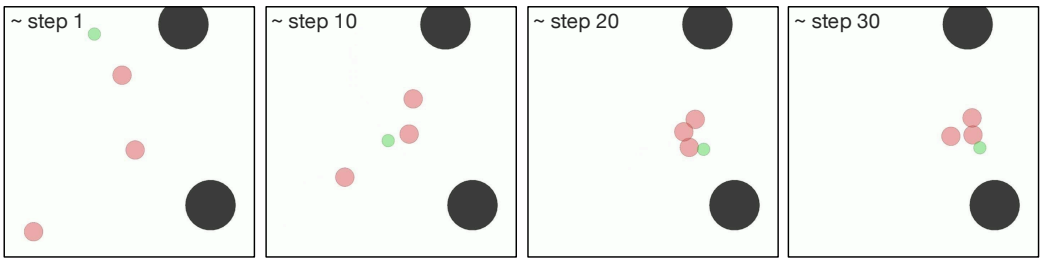
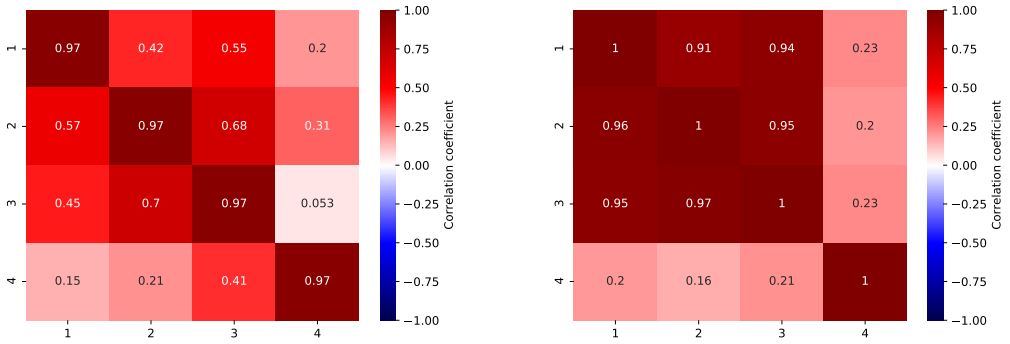


Figure 2.14: Snapshots of the demonstration task: the agents 1-3 were moving to surround the agent 4; thanks to that behaviors, escaping from the agents 1-3 is difficult for the agent 4.



(a) After first episode

(b) After final episode

Figure 2.15: Correlation matrices before and after learning: they were obtained by merging the agents' estimations; (a) the accurate correlation matrix was not found due to insufficient data; (b) finally, all the agents correctly distinguished the allies, and calculated the small correlation coefficients with the enemies.

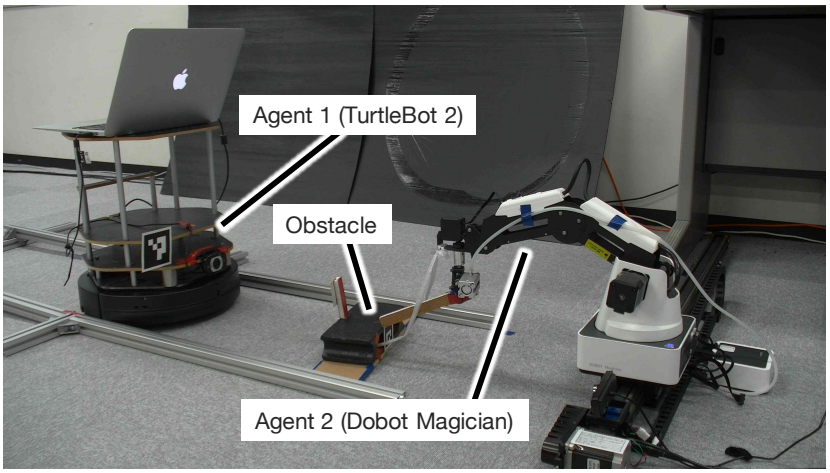
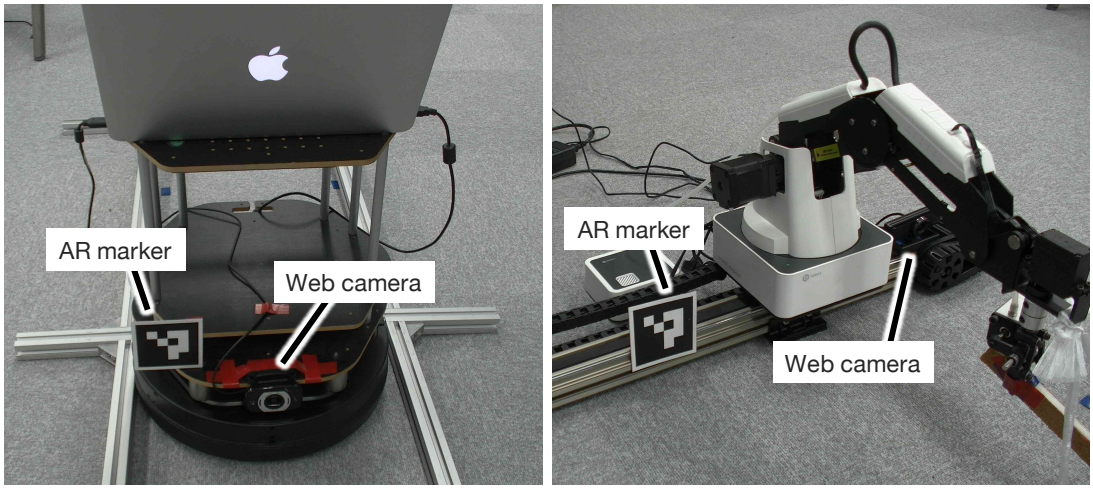
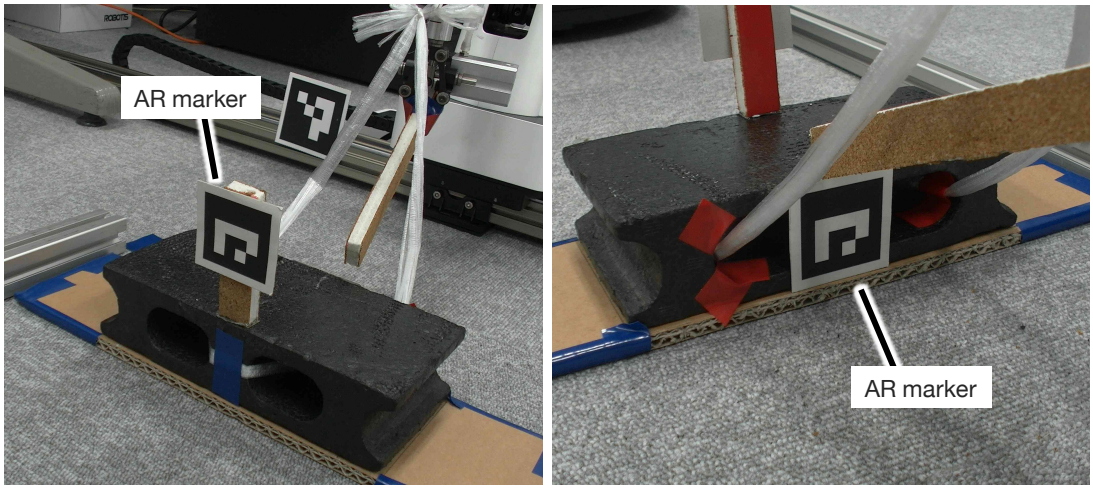


Figure 2.16: Experimental environment: two agents are in this environment; a mobile robot can move along the front-back direction while stopping in front of an obstacle; a four-axis manipulator is connected to the obstacle by a string, so it can move the obstacle nonlinearly; each agent has a web camera to recognize AR markers.



(a) Agent 1: TurtleBot 2 (b) Agent 2: Dobot Magician

Figure 2.17: Robots as agents: a two-wheeled mobile robot and a four-axis manipulator are employed as the agents in this experiment; they have web cameras to recognize AR markers, which provide distance information.



(a) View from the agent 1 (b) View from the agent 2

Figure 2.18: Obstacle design: it can be pulled by the manipulator via the string; AR markers on both sides allow the agents to detect this obstacle.

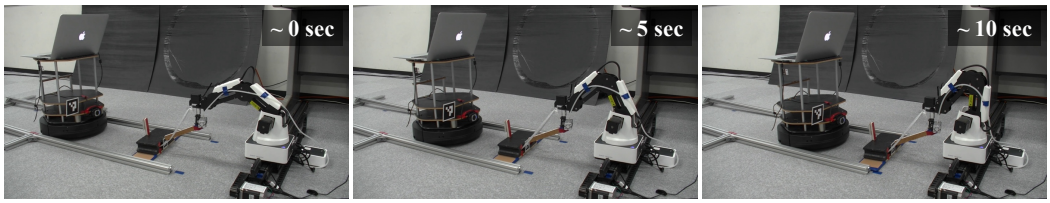


Figure 2.19: Snapshots of the cooperative task: the manipulator pulled the obstacle to make the mobile robot close to itself; according to that behavior, the mobile robot approached to the manipulator rapidly.

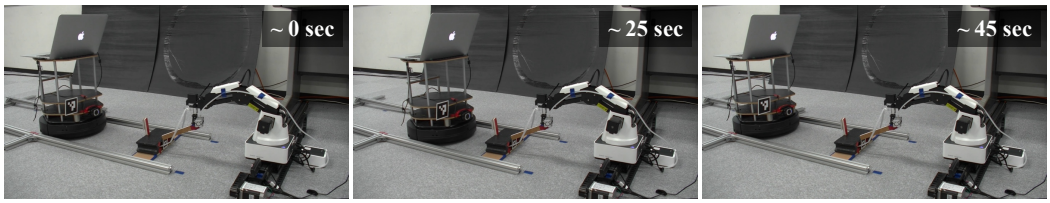
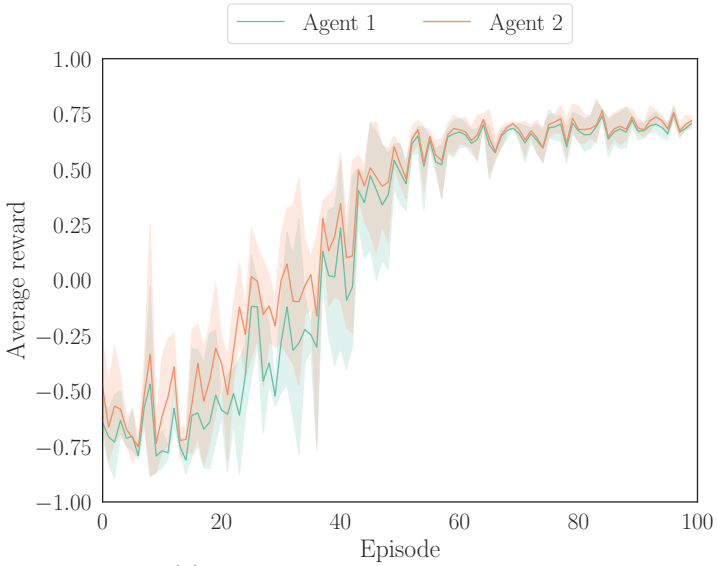
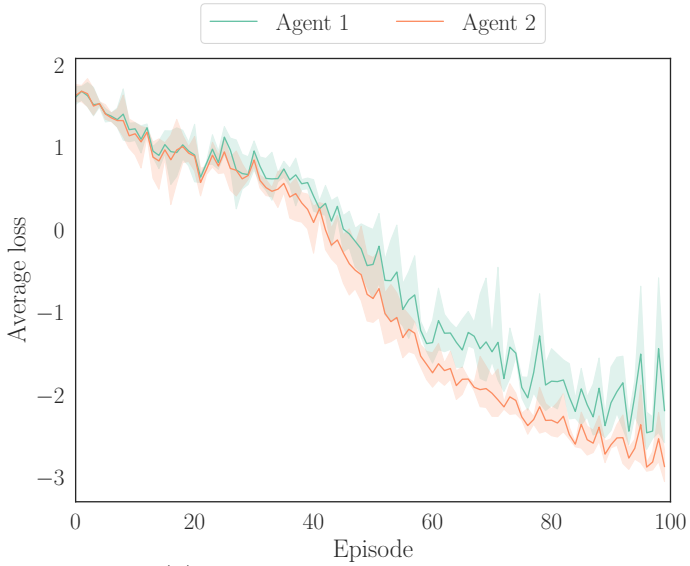


Figure 2.20: Snapshots of the competitive task: the manipulator prohibited the mobile robot approaching by not pulling the obstacle; the mobile robot repeated exploratory actions to find new states with high rewards, even though they did not exist.

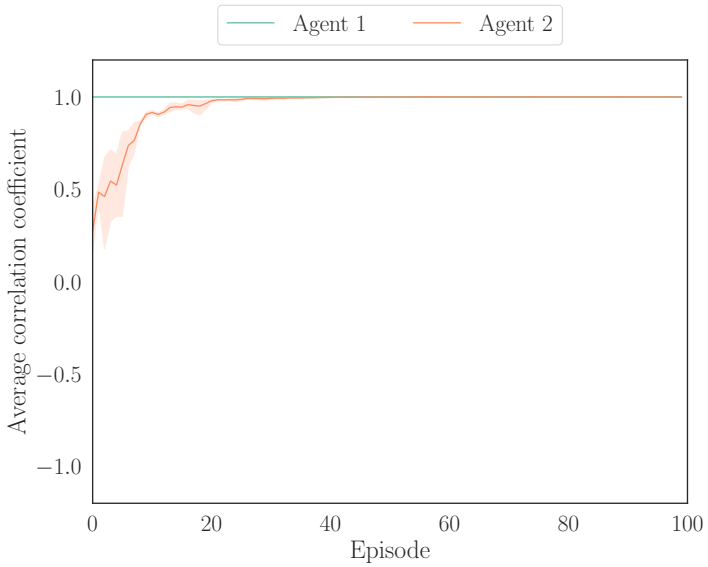


(a) Learning curves of average reward

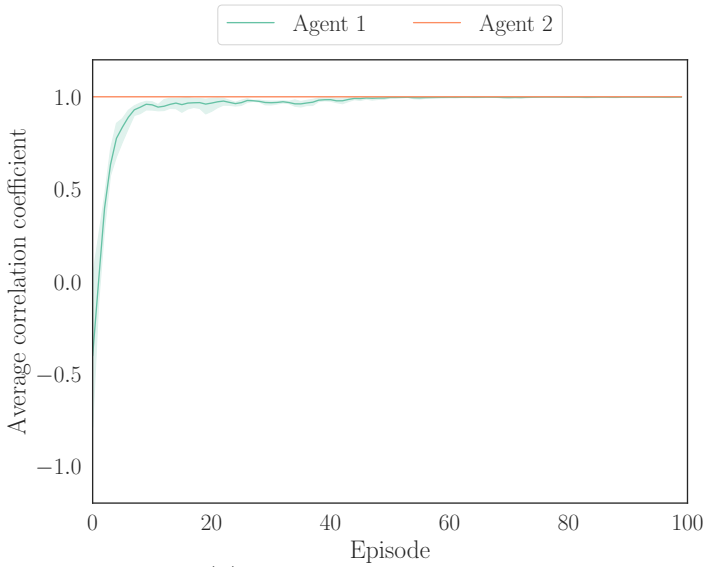


(b) Learning curves of average loss

Figure 2.21: Learning curves of the cooperative task: (a) the average rewards of both agents were increased as the episode went on (in particular, after 20 episodes); as a result, they emerged the cooperative behaviors as shown in Fig. 2.19; (b) the average loss of the reward predictors in both agents were decreased and became negative, that is, the reward predictors acquired the high accuracy with low variance.

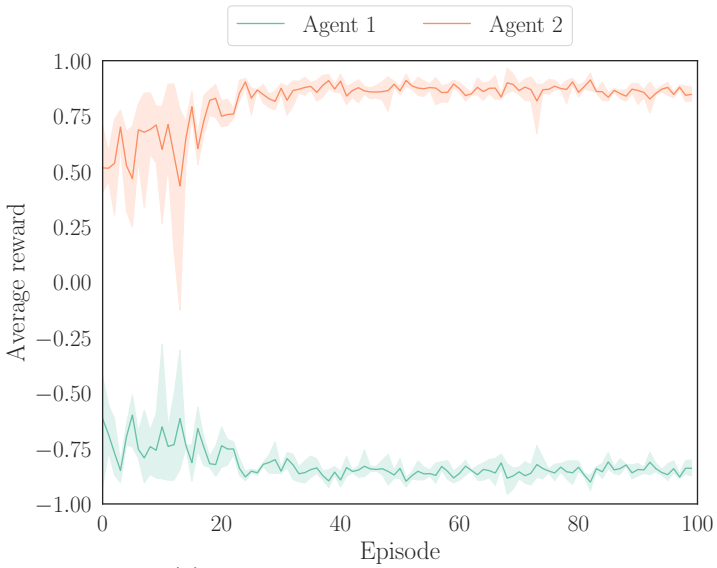


(a) Estimation by the agent 1

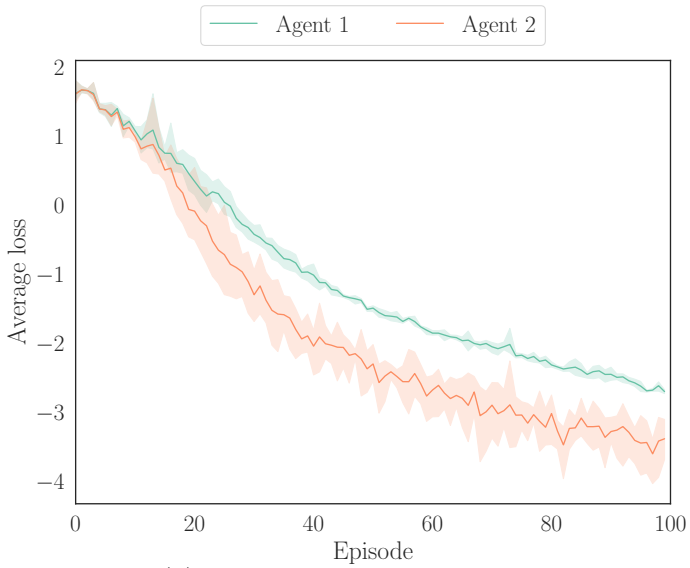


(b) Estimation by the agent 2

Figure 2.22: The correlation coefficients of the cooperative task: in both agents, the correlation coefficients between themselves and the others converged around 1 within 20 episodes.

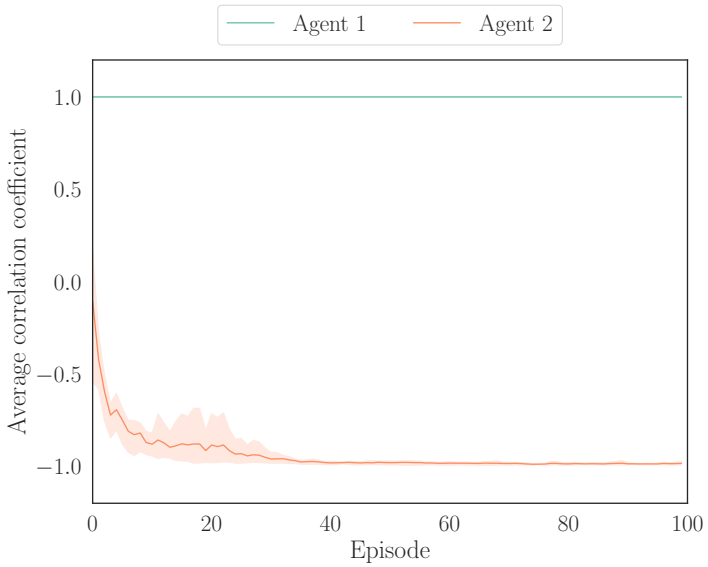


(a) Learning curves of average reward

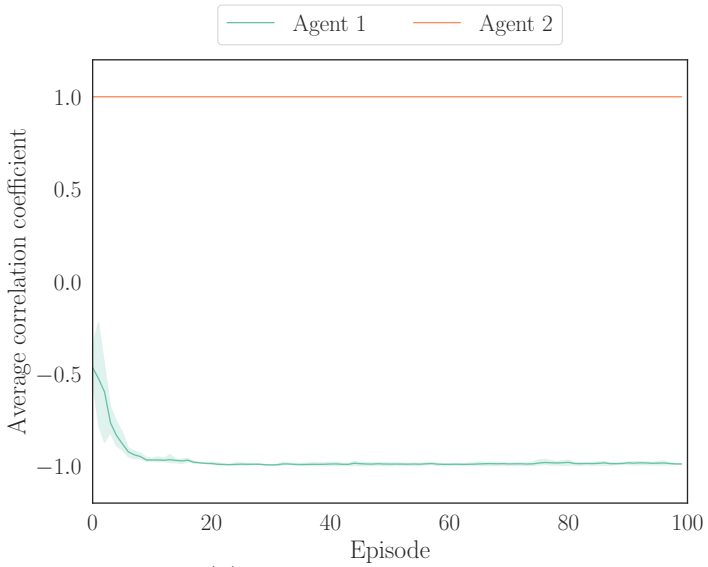


(b) Learning curves of average loss

Figure 2.23: Learning curves of the competitive task: (a) the average rewards of only the agent 2 were increased as the episode went on (in particular, after 20 episodes); as a result, they emerged the competitive behaviors as shown in Fig. 2.20; (b) the average loss of the reward predictors in both agents were decreased and became negative, that is, the reward predictors acquired the high accuracy with low variance.



(a) Estimation by the agent 1



(b) Estimation by the agent 2

Figure 2.24: The correlation coefficients of the competitive task: in both agents, the correlation coefficients between themselves and the others converged around -1 within 20 episodes.

3 Meta-Optimization of Bias-Variance Trade-off in Stochastic Model Learning

3.1 Introduction

Reinforcement learning (RL) [46] is one of the promising methods for robots to adaptively acquire their own policies in the real world. In recent years, RL has been applied in environments with high uncertainty, where there are multiple actors (eg. human-containing systems [47, 48] and multi-agent systems [49, 50]). RL-based agents attempt stochastic actions during exploration, which may have a negative impact on the environment. Safe learning, which mitigates risk during exploration such as collision, is required in these environments.

Model-based RL is expected to take safety into account by using stochastic dynamics models in planning. One such approach is model predictive shielding (MPS), which utilizes an idea called *shielding* [51, 52]. MPS uses the stochastic dynamics model, and *shielding* intervenes in the agent's action to ensure that state transitions satisfy safety constraints. Intervention by *shielding* is triggered when the agent is prone to go into the states outside of the safety constraints (predicted by the dynamics model). Similarly, tube model predictive control (tube MPC) is another model-based planning method that explicitly considers safety. Uncertainty in dynamics prediction propagates in a time-evolving manner. The region surrounding a possible transition state is called *tube*, and planning within this *tube* is performed in tube MPC [53–55]. However, these methods do not provide how to obtain the accurate dynamics model for the target environment, while that is mandatory in them.

Learning with the deep neural networks (DNNs) is widely applied to achieve the stochastic model prediction in recent years. The objective function for training is generally formulated as the minimization problem of expected prediction loss for the next state. The expectation is regarded as a first-order moment, namely, this approach does not optimize higher-order moments, such as variance. Hence, if a large prediction error occurs even once during the prediction of a long-term trajectory, all subsequent states will become outliers. With the fact that the prediction model can be given as a probability distribution, Bayesian theory has been appropriately utilized to consider the uncertainty of the model [56, 57]. In particular, Chua et al. [57] has proven a simple ensemble method, in which multiple models are prepared and trained simultaneously. In their work for the latest model-based RL, the learned models made stable planning possible. The models are however approximated with DNNs, hence the number of parameters would be huge if multiple models are used. In addition, the number of models required for the target environment must be determined empirically. Although robust control theory that explicitly considers model uncertainty or input uncertainty have been proposed [58, 59], they assume linearity and cannot be directly applied to nonlinear stochastic models. A learning method that predicts the stochastic nonlinear dynamics by DNNs with limited size is required for practical use.

If DNNs with limited size try to reduce the uncertainty of model, a well-known problem in regression, called the bias-variance trade-off [60], cannot be ignored. While the bias can be reduced by the conventional minimization problem of the expected loss, that raises the risk degrading the generalization performance caused by the increase of the variance. On the other hand, if the variance is somehow reduced excessively, the average prediction performance would be deteriorated. Even though both bias and variance can be reduced in DNNs with sufficiently large size [61,62], the bias-variance trade-off still need to be considered for practical use. We notice the important fact that the optimal balance of the trade-off is task-dependent and basically non-trivial. With the same awareness of the issue, various bias-variance decompositions for regression problems have been proposed [63–66]. The conventional methods, however, cannot be applied to model training with DNNs for model-based RL. Therefore, a new decomposition suitable for sequential training of the deep neural networks is needed.

In this study, we propose a comprehensive algorithm to obtain an optimal balance between bias and variance for the meta-objective required in model-based RL. We first attempt to formulate the bias-variance trade-off as a multi-objective optimization (MOO) problem. From a statistical point of view on the loss of the entire dataset, we note that the bias and variance can be represented by the mean loss and the worst loss. The argument begins with the fact that the expected loss function of the minimization problem for a given dataset is an equally-weighted sum of the losses for each data. In other words, the conventional method can be interpreted as a method to obtain a Pareto solution by evaluating the loss of each data equivalently. The optimization based on scalarization with the linear weighted sum, however, cannot obtain Pareto solutions on the non-convex part. Therefore, we apply the augmented weighted Tchebycheff scalarization [67, 68], which can effectively find non-convex Pareto frontiers, to each data loss. The weighted sum of the mean loss and the worst loss is derived as a new minimization target in this scalarization. That is, the next step is to apply the augmented weighted Tchebycheff scalarization again so that arbitrary Pareto solutions among the statistics (i.e. mean and worst) can be found.

The balance between the bias and the variance can be adjusted using a hyperparameter given by the above process. The Pareto solution to be used, called the preferred solution, is therefore selected by tuning this hyperparameter from the set of Pareto solutions according to the higher-level objective in general. The simplest way to find a preferred solution is brute-force exploration of the related hyperparameter(s), although this approach is computationally expensive as a matter of course. As a more advanced method, meta-optimization of parameters included in lower-level objectives [69] has been developed with several forms: e.g. gradient descent (GD) [70–73]; RL [74–76]; evolutionary search (ES) [77–79]; and Bayesian optimization (BO) [80, 81]. However, these conventional methods have the limitation of assuming the differentiability of the meta objective and/or requiring multiple lower-level learning trials.

Hence, we propose a general-purpose and efficient meta-optimization method based on a policy gradient method [82]. Specifically, the proposed method learns a policy that outputs hyperparameters stochastically. In each epoch, twin models are trained using the mean and sampled values of the policy, respectively, and

the trained models are validated against the meta objective. The difference in the validation results would be related to only the sampled hyperparameters, not the training results, and therefore, the log-likelihood of the policy with the sampled hyperparameters, weighted by the difference in the validation results, can be maximized so as to optimize the meta objective. Before starting the next epoch, the twin models are remade from either of the old twins. In this method, the meta objective is not differentiated, and multiple trials are not necessary since the hyperparameters are optimized at the same time as the DNNs parameters.

The contributions in this study are three folds:

1. Formulation of the bias-variance trade-off as a MOO problem
2. Development of a general-purpose and efficient meta-optimization method
3. Numerical verification of the proposed formulation with the meta-optimization on two simulations for the environments with uncertainty due to human operation and presence of other agents

3.2 Related work

3.2.1 Bias-variance decomposition

Traditionally, the problem of bias-variance trade-off has been pointed out in data-driven learning. Various bias-variance decompositions are presented for several loss functions (e.g. mean-squared loss [60,83], zero-one loss [83], and log-likelihood type loss [84]) used in regression. For the selection of regression models to avoid overfitting, the bias-variance have been decomposed as accuracy and complexity according to the information criterion [85]. In recent years, several bias-variance decompositions have been proposed to treat the trade-off as a MOO problem. In this section, we characterize the proposed decomposition method by comparing it with conventional methods.

A semi-parametric Gaussian copula regression that is robust to multiple datasets is proposed [65]. In generating the cumulative distribution function used for the prior distribution, the parameters of the quantile estimate adjust the bias and

variance. However, the idea is not directly applicable to model training with neural networks.

A decomposition method has been proposed for model selection of Bayesian networks, where the evaluation function is defined by the accuracy and complexity using the minimal description length (MDL) [63]. Applying MDL to general DNNs where each node (neuron) has a real number of outputs and is large in scale, is, however, difficult. Since the data is sampled online, selecting the best model in advance, is also not suitable for the model-based RL.

Several methods have been presented for RL, focusing on the bias-variance trade-off of the policy gradient estimation. The method of using regularization by a Kullback–Leibler divergence for variance reduction [64] is discussed by restricting the problem to hyperparameters used in RL. A method that deals with the merge of gradients appearing in off-policy and on-policy learning [66] is also a decomposition method unique to RL.

Although various bias-variance decompositions have been proposed as described above, the methods suitable for safe model learning used in model-based RL, which is the target of this method, have not been well investigated. The proposed method does not analyze existing loss functions such as [60, 83, 84], but defines a new loss function by providing a decomposition that deals with the bias and variance of the loss values themselves. By focusing on DNNs, which have been traditionally used for learning stochastic models of dynamics, the loss function is defined in a form that is easy to handle in model-based RL. Furthermore, the proposed loss function is naturally derived by interpreting the conventional loss function as a MOO problem, and is not applied at the model selection stage as in [63]. This feature is an essential condition for model-based RL, which assumes online learning.

3.2.2 Meta-optimization

Table 3.1: Comparison of recent meta-optimization methods with four indices: only our method satisfies all the indices introduced here.

Approach	Method	High efficiency	High scalability	Arbitrariness of target	No use of gradient
	Our method	✓	✓	✓	✓
Gradient descent	[70–72]	✓	✓		
	[73]	✓	✓	✓	
Reinforcement learning	[74]		✓	✓	✓
	[75, 76]	✓	✓		
Evolutionary search	[77–79]			✓	✓
Bayesian optimization	[80, 81]	✓		✓	✓

A learning algorithm trains DNNs based on a task-specific (low-level) loss function. According to a user-desired (high-level) meta-objective (e.g. generalizing across different tasks and long-term prediction accuracy like our setting), meta-optimization methods aim to optimize hyperparameters in the learning algorithm and/or the low-level loss function. The reason why various methods have been proposed is that the conditions to be satisfied differ depending on the problem. In this section, we qualitatively check the performance of the conventional and proposed meta-optimization methods while summarizing the necessary requirements for general meta-optimization. The comparison results are summarized in Table 3.1.

First, minimization of the low-level loss function is generally with high computational cost due to large dataset for training DNNs. The meta-optimization methods should be, therefore, highly efficient. The number of hyperparmeters to be optimized is problem-dependent (e.g. one in our case and hundreds in optimization of the architecture of DNNs), and therefore, scalability is important. Furthermore, versatility is also important to employ arbitrary meta-objective, loss function, architecture of DNNs, and so on. In particular, differentiability of meta-objective function over hyperparameters cannot be assumed since it absolutely limits the applicable problems. Hence, the following four requirements are raised: i) *high efficiency*; ii) *high scalability*; iii) *arbitrariness of target*; and iv) *no use of gradient*.

High efficiency Since meta-optimization is performed at a higher layer of the low-level learning, using the results of low-level learning is generally necessary [69]. While meta-optimization may aim to improve the efficiency of low-level learning, improving the efficiency of meta-optimization itself is also important, in order to reduce time cost and computational resources. In this evaluation, we examine whether or not to complete meta-optimization is possible from a single trial of a limited number of low-level learners.

GD-based methods [70–73] directly optimize the target hyperparameters, and thus have higher efficiency. RL-based methods are fundamentally less efficient because they require wide exploration and many trials [74]. Some methods [75, 76], however, achieved high efficiency with using the gradient of meta-objective

or low-level loss, by limiting the application to RL. BO can also achieved high efficiency by finding the points to be explored. On the other hand, ES-based methods [77–79] are well known as less efficient because they use multiple trials or many low-level learners.

High scalability When meta-optimization methods are applied to optimize a large number of hyperparameters, the complexity of search space increases combinatorially. Since the order of computational complexity with respect to the number of hyperparameters is directly related to the versatility, a scalable method is desired to be developed.

Methods that aim for local optima based on direct information, such as hyperparameter-dependent gradients, generally have high scalability [70–76] by not using global information in the search space. On the other hand, in heuristic methods [77–79], it is intractable to find the optimal solution without many search points on large search space. The convergence is sacrificed in exchange for not limiting the search space, which is the reason for the reduced scalability. In addition, BO [80,81] uses the Gaussian process [86] to estimate the model, the computational cost explodes with respect to the number of search points because samples of various values need to obtain the global shape of the objective function.

Arbitrariness of target When dealing with MOO problems such as the bias-variance trade-off, the meta-objective for selecting one of the Pareto solution sets cannot be assumed in advance.. A method that can handle arbitrary meta-objective, rather than a method requiring the specific meta-objective format, is essential.

Many methods specialize in the typical meta-objectives: domain generalization [70, 71]; surrogate loss [72]; RL [75, 76, 78]; winning in games [79]; and low-level learning loss itself [77]. The meta-heuristics used in ES-based methods (e.g. CMA-ES [87] and evolution strategies [88]), however, can potentially be extended to arbitrary targets. BO [80,81] is also suitable for handling the arbitrary targets due to its statistically-generalized design. Some methods [73, 74] have been proposed that can also handle a relatively wide range of meta-objectives, although they still restrict the format of the meta-objectives and the information required.

No use of gradient A meta-objective for extracting a preferred solution to a MOO problem may be given only an evaluation value, and differentiability in the hyperparameters of interest cannot be assumed. This metric is marked whether the gradient with the target hyperparameter is used for meta-optimization.

In GD-based methods [70–73], the use of gradient is the key to meta-optimization. One method [74] based on RL, however, avoids the differentiation of meta-objective by using stochastic policy. ES-based methods [77–79] and BO [80,81] are sampling-based and do not require gradient information.

Proposal The proposed method performs meta-optimization simultaneously with low-level learning, and the low-level learning is limited to a single trial (but with two learners). Both the conventional method [74] and the proposed method use the stochastic policy for meta-optimization. However, the proposed method avoids state-dependent exploration by using only a policy-gradient method instead of RL. Another advantage of using only the policy-gradient method is that there is no need to design the state on which the meta-objective depends. Two ideas provide these advantages. The first is that the proposed method identifies the local gradient direction from the difference in evaluation between the baseline and the sample values. The other is to match the states of the two low-level learners at the beginning of each epoch, thus eliminating the need to take the states into account for the difference in learning results. In addition, the meta-objective only needs to be given a numerical scalar value as an evaluation of the low-level learners, and there is no need to assume either type or differentiability.

3.3 Preliminaries

3.3.1 Stochastic model learning in Markov decision processes

The stochastic dynamics of the environment including the agent is formulated by a Markov decision process (MDP) in RL. Given a state $s_t \in \mathcal{S} \subset \mathbb{R}^{d_s}$ and an action $a_t \in \mathcal{A} \subset \mathbb{R}^{d_a}$ at time t, the next state s_{t+1} is assumed to be stochastically sampled from the environment-specific state transition probability $p_e(s_{t+1} \mid s_t,\ a_t)$. Here,

d_s and d_a are the dimension sizes of the state and the action, respectively. p_e is, however, generally unknown, and model-based RL approximates it to be a model constructed DNNs parameterized by θ, $p_m(s_{t+1} \mid s_t,\ a_t;\ \theta)$. If p_m is accurately acquired, the agent can predict the future states according to the performed actions, hence, can plan the best actions to maximize rewards from the environment.

Therefore, the goal of a stochastic model learning is usually to fit p_m to p_e through minimization of expectation of negative log-likelihood, $\ln p_m$, w.r.t. p_e. Since p_e is a black-box, the expectation is replaced by Monte Carlo method as sample mean over the dataset obtained from p_e, $D = \{(s_n,\ a_n),\ s_{n+1}\}_{n=1}^{N}$, with N tuples. More specifically, θ is optimized toward θ^*, to minimize the following formula.

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \mathcal{L}_i(\theta),\ \forall i \in \mathbb{N} \\ \mathcal{L}_i(\theta) &= \frac{1}{N_i} \sum_{s_i, a_i, s_{i+1} \in D_i} -\ln p_m(s_{i+1} \mid s_i,\ a_i;\ \theta) \end{aligned} \tag{3.1}$$

where $D_i \subset D$ denotes i-th mini-batch with batch size N_i extracted from D.

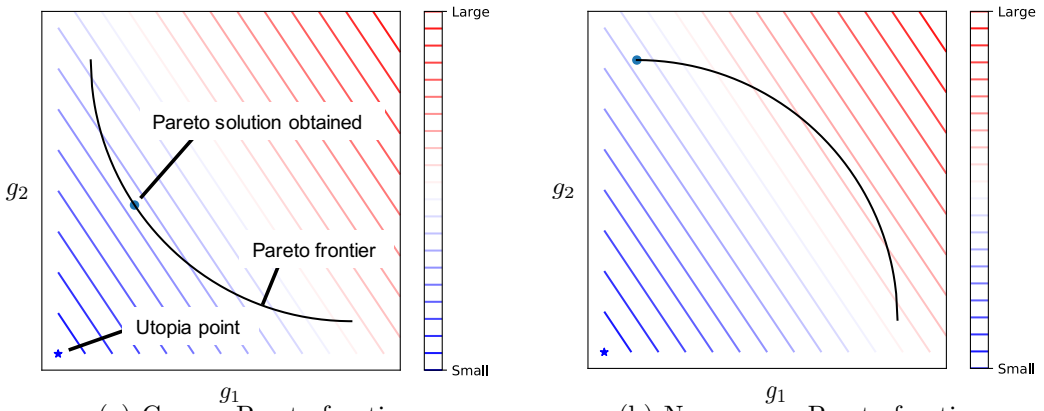
3.3.2 Augmented weighted Tchebycheff scalarization

When considering the minimization of M objective functions $g_1, \ldots, g_M$ as a MOO problem, the optimality of the solution is defined by dominance. The solution x that satisfies the following formula dominates the solution x' and is expressed as $x \prec x'$.

$$\forall m, g_m(x) \leq g_m(x') \wedge \exists m, g_m(x) < g_m(x') \tag{3.2}$$

The solution that is not dominated by all other solutions is called the Pareto solution. The set of Pareto solutions is called the Pareto frontier.

The goal of MOO is to find the Pareto solution or the Pareto frontier while taking into account the trade-offs among the objective functions. To this end, in most cases, a scalarization function $h : \mathbb{R}^M \to \mathbb{R}$ with a weight vector $\boldsymbol{w} \in \mathbb{R}^M$ makes the objective function vector $\boldsymbol{g} = [g_1, \ldots, g_M]^\top$ scalar in order to transform a MOO problem into a set of single-objective optimization problems. The simplest



(a) Convex Pareto frontier (b) Non-convex Pareto frontier

Figure 3.1: Illustration of the contour and the Pareto solution obtained by the linearly weighted sum: the shape of the contour line is linear; (a) the Pareto solution of the convex part of the Pareto frontier can be obtained; (b) however, the Pareto solution of the non-convex part of the Pareto frontier cannot be obtained.

scalarization function is the linear weighted sum in the following equation.

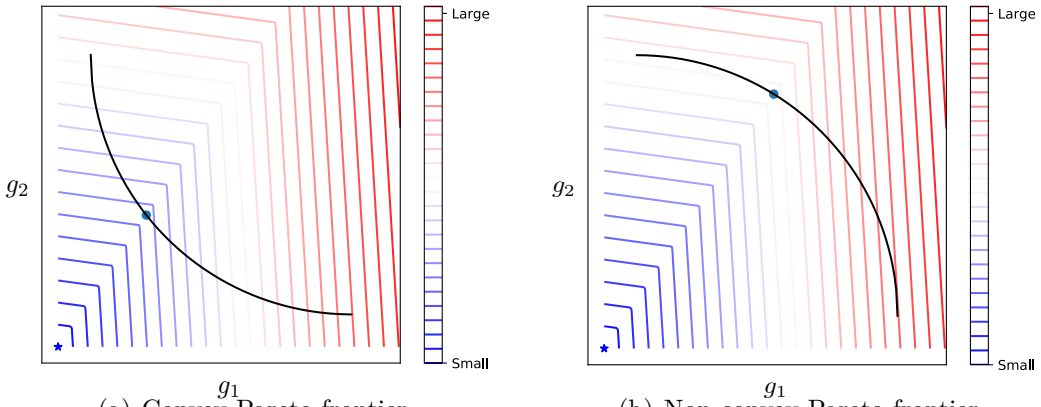
$$h(x) = \sum_{m=1}^{M} w_m g_m(x) \tag{3.3}$$

In the case of scalarization by linear weighted sum, the contour line in the search space is just a linear line. Therefore, the Pareto solution in the non-convex part of the Pareto frontier cannot be obtained (see Figs. 3.1 (a), (b)).

The augmented weighted Tchebycheff scalarization, defined by the following equation, is widely used as one of the scalarization functions that can deal with non-convex Pareto frontiers [67, 89].

$$\begin{aligned} h(x) = & \max_{1 \leq m \leq M} w_m (g_m(x) - u_m) \\ & + \alpha \sum_{m=1}^{M} w_m (g_m(x) - u_m) \end{aligned} \tag{3.4}$$

where u_m is called a utopia point that strictly dominates g_m. The contour line is a linear combination of the L-shaped line from the first term and the linear line from the second term with the hyperparameter $\alpha > 0$ (see Figs. 3.2 (a), (b)). As



(a) Convex Pareto frontier (b) Non-convex Pareto frontier

Figure 3.2: Illustration of the contour and the Pareto solution obtained by the augmented weighted Tchebycheff scalarization: The shape of the contour line is a linear combination of the L-shaped line and the linear line; (a) the Pareto solution of the convex part of the Pareto frontier can be obtained; (b) and, the Pareto solution of the non-convex part of the Pareto frontier can also be obtained.

for the choice of α, the effects are noted that too small α would cause a weak Pareto solution because the effect of the second term is relatively insignificant, and too large α would make the non-convex solutions unreachable because the effect of the first term is weakened [68, 90].

3.4 Meta-optimization of bias-variance trade-off

3.4.1 Overview

The state transition model with the parameter θ is optimized to minimize the loss function, i.e. the expected value of the negative log-likelihood, defined in eq. (3.1). Although this approach is effective when the stochastic behavior is relatively small, it does not take into account the variance of losses and worst-case scenarios, which are represented by higher-order moments, and thus can lead to large prediction errors in reality. Particularly in the case of RL and MPC applications, the resulting model is used for long-term prediction, and if

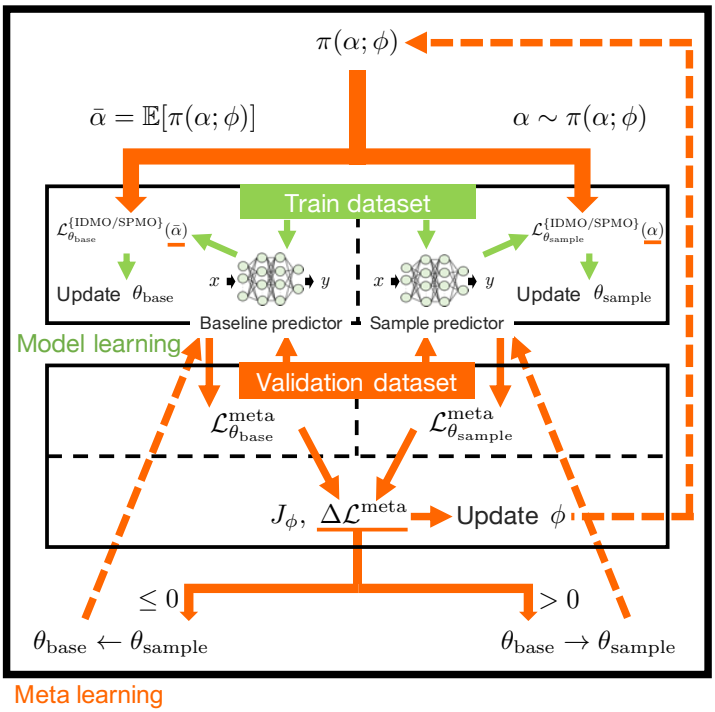


Figure 3.3: Schematic of the proposed method: Two stochastic models are learned using the training dataset, and based on the differences between them, meta policy is simultaneously optimized under the meta objective using the validation dataset.

the prediction fails even once during the period, subsequent predictions from that point may fail. This problem is caused mainly by a bias-variance trade-off.

This study proposes a method to adjust the balance of the bias-variance trade-off by simultaneously minimizing the expected loss and the worst loss, which are theoretically derived later. In this case, since learning the state transition model becomes a MOO problem, a loss function scalarized by the augmented weighted Tchebycheff scalarization can be applied. In light of the fact that the size of the Pareto solution set is generally innumerable, a general-purpose meta-optimization method is also proposed to obtain the preferred solution depending on the given meta-objective. A schematic diagram of the entire proposed method is shown in Fig. 3.3.

3.4.2 Formulation of MOO problem

3.4.2.1 Inter-data MOO: IDMO

To avoid complication, the loss function of the conventional method, eq. (3.1), is redefined as follows:

$$\mathcal{L}_\theta^{\text{mean}} = \frac{1}{N}\sum_{i=1}^{N} l_{i,\theta} \tag{3.5}$$

$$l_{i,\theta} = -\ln p_m(s_{i+1} \mid s_i,\ a_i;\ \theta) - u \tag{3.6}$$

where u is a utopia point, which is given commonly to all data as $u = \min -\ln p_m(s_{i+1} \mid s_i,\ a_i;\ \theta)$. The above objective function can be interpreted as the linear weighted sum with objectives for each data and equivalent weights $w_i = 1/N$. That is, the conventional way can be regarded as an inter-data multi-objective (IDMO) optimization problem.

According to this interpretation, we formulate this IDMO optimization problem based on the augmented weighted Tchebycheff scalarization that can obtain all Pareto solutions.

$$\begin{aligned}\theta^* &= \arg\min_\theta \mathcal{L}_\theta^{\text{IDMO}} \\ \mathcal{L}_\theta^{\text{IDMO}} &= \frac{\tilde{\alpha}}{N}\sum_{i=1}^{N} l_{i,\theta} + \frac{1}{N}\max_i l_{i,\theta} \\ &\propto \mathcal{L}_\theta^{\text{mean}} + \alpha\mathcal{L}_\theta^{\text{worst}}\end{aligned} \tag{3.7}$$

where $\mathcal{L}_\theta^{\text{worst}} = \max_i l_{i,\theta}$ is the worst loss, and $\tilde{\alpha}$ is the hyperparameter in the augmented weighted Tchebycheff scalarization. Since the solution is not changed by constant multiplication of the loss function, eq. (3.7) can be used under $\alpha = 1/(\tilde{\alpha}N)$.

As shown in the above formula, the application of the augmented weighted Tchebycheff scalarization to the IDMO optimization naturally leads to a loss function that explicitly considers the mean loss and the worst loss, which correspond to the bias and the variance, respectively. Therefore, the bias-variance trade-off can be adjusted by setting α appropriately.

Table 3.2: Reachability to non-convex solutions

Loss	Non-convex solutions	
	Inter data	Statistics perspective
Vanilla loss $\mathcal{L}_\theta^{\text{mean}}$		
Proposed loss $\mathcal{L}_\theta^{\text{IDMO}}$	✓	
Proposed loss $\mathcal{L}_\theta^{\text{SPMO}}$	✓	✓

3.4.2.2 Statistics-perspective MOO: SPMO

The loss $\mathcal{L}_\theta^{\text{IDMO}}$ obtained above can be again interpreted as a linear weighted sum of the statistics $\mathcal{L}_\theta^{\text{mean}}$ and $\mathcal{L}_\theta^{\text{worst}}$ weighted by the ratio $1 : \alpha$. When the Pareto frontier for $\mathcal{L}_\theta^{\text{mean}}$ and $\mathcal{L}_\theta^{\text{worst}}$ is non-convex, the Pareto solutions that cannot be obtained at the statistical level would be worth considering.

We, therefore, apply the augmented weighted Tchebycheff scalarization again to such a statistics-perspective multi-objective (SPMO) optimization problem as follows:

$$\begin{aligned}\mathcal{L}_\theta^{\text{SPMO}} = &\max\left(\mathcal{L}_\theta^{\text{mean}}, \alpha\mathcal{L}_\theta^{\text{worst}}\right) \\ &+ \beta\left(\mathcal{L}_\theta^{\text{mean}} + \alpha\mathcal{L}_\theta^{\text{worst}}\right)\end{aligned} \tag{3.8}$$

where β denotes the hyperparameter in the augmented weighted Tchebycheff scalarization. Since each loss is non-negative, its statistics, $\mathcal{L}_\theta^{\text{mean}}$ and $\mathcal{L}_\theta^{\text{worst}}$, are also non-negative, and no utopia point is needed. Note that, in the case of $\alpha \geq 1$, the above formula is essentially equivalent to eq. (3.7) since the first term is always $\mathcal{L}_\theta^{\text{worst}}$.

3.4.2.3 Summary of proposed losses

The properties of the loss functions based on the proposed multi-objective optimization problems are summarized in Table 3.2.

First, from the perspective of IDMO, the conventional loss function, the mean loss $\mathcal{L}_\theta^{\text{mean}}$, can be interpreted as a linear weighted sum of the objectives. Since the linear weighted sum cannot yield non-convex solutions, a loss $\mathcal{L}_\theta^{\text{IDMO}}$ was formulated based on the augmented weighted Tchebycheff scalarization.

Furthermore, since $\mathcal{L}_\theta^{\mathrm{IDMO}}$ was derived as a linear sum of the mean loss and the worst loss, a loss $\mathcal{L}_\theta^{\mathrm{SPMO}}$ was formulated by applying the augmented weighted Tchebycheff scalarization to those statistics again. The loss $\mathcal{L}_\theta^{\mathrm{SPMO}}$ implicitly includes $\mathcal{L}_\theta^{\mathrm{IDMO}}$, and the non-convex solutions can be obtained even in IDMO.

3.4.3 Meta-optimization of hyperparameter

Even though the hyperparameter α of the loss functions ($\mathcal{L}_\theta^{\mathrm{IDMO}}$ and $\mathcal{L}_\theta^{\mathrm{SPMO}}$) shown in the previous section contributes significantly to the bias-variance trade-off, no clear metric has been defined to determine its value. In this section, we propose a general-purpose meta-optimization method for α under an arbitrary meta-objective, $\mathcal{L}^{\mathrm{meta}}$, given as a high-level design metric.

This meta-optimization problem can be formulated as follows:

$$\begin{aligned} &\alpha^* = \arg\min_{\alpha} \mathcal{L}^{\mathrm{meta}}(\theta^*(\alpha), D^{\mathrm{val}}) \\ &s.t.\ \theta^*(\alpha) = \arg\min_{\theta} \mathcal{L}_\theta^{\{\mathrm{IDMO,SPMO}\}}(\alpha, D^{\mathrm{trn}}) \end{aligned} \tag{3.9}$$

where D^{trn} and D^{val} are the training and validation datasets, respectively, and are generated such that $D^{\mathrm{trn}} \cap D^{\mathrm{val}} = \emptyset,\ D^{\mathrm{trn}} \cup D^{\mathrm{val}} = D$ are satisfied.

To solve this meta-optimization problem, we first suppose that $\alpha \in [0, 1]$ (this restricted range is for distinguishing IDMO and SPMO) is sampled from a meta-policy $\pi(\alpha;\ \phi)$ constructed as a probability distribution parameterized by ϕ. The purpose is converted to optimize ϕ to minimize $\mathcal{L}^{\mathrm{meta}}$ stochastically.

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{\alpha\sim\pi(\alpha;\ \phi)}[\mathcal{L}^{\mathrm{meta}}(\theta^*(\alpha), D^{\mathrm{val}})] \tag{3.10}$$

The gradient of the above objective function over ϕ can be computed following the policy gradient method.

$$\begin{aligned} &\nabla_\phi \mathbb{E}_{\alpha\sim\pi(\alpha;\ \phi)}[\mathcal{L}^{\mathrm{meta}}(\theta^*(\alpha), D^{\mathrm{val}})] \\ &= \mathbb{E}_{\alpha\sim\pi(\alpha;\ \phi)}[\mathcal{L}^{\mathrm{meta}}(\theta^*(\alpha), D^{\mathrm{val}}) \nabla_\phi \ln \pi(\alpha;\ \phi)] \\ &= \mathbb{E}_{\alpha\sim\pi(\alpha;\ \phi)}[\{\mathcal{L}^{\mathrm{meta}}(\theta^*(\alpha), D^{\mathrm{val}}) - b\} \nabla_\phi \ln \pi(\alpha;\ \phi)] \end{aligned} \tag{3.11}$$

where b denotes the baseline, which is not related to α. Since the expectation of the term for b is zero, it can be added freely as long as it does not depend on α, which greatly reduces the variance of the learning results.

To design b, we provide twin models, p_m^{base} and p_m^{sample}, which are with exactly the same θ before each epoch. In each epoch, they are trained with $\bar{\alpha} = \mathbb{E}[\pi(\alpha;\ \phi)]$ and $\alpha \sim \pi(\alpha;\ \phi)$, resulting in θ_{base} and θ_{sample}, respectively. Since p_m^{base} is not involved in α, $\mathcal{L}^{\text{meta}}$ with θ_{base} can be utilized as the baseline. In addition, we can separate whether the variation in $\mathcal{L}^{\text{meta}}$ comes from α or training, and extract only the contribution of α by subtracting this baseline from $\mathcal{L}^{\text{meta}}$ with θ_{sample}.

Hence, with $\mathcal{L}^{\text{meta}}(\theta_{\text{base,sample}}, D^{\text{val}}) =: \mathcal{L}^{\text{meta}}_{\theta_{\text{base,sample}}}$, the meta-objective function for ϕ, J_ϕ, can be given as follows:

$$J_\phi = \Delta\mathcal{L}^{\text{meta}} \ln \pi(\alpha;\ \phi) \tag{3.12}$$

$$\Delta\mathcal{L}^{\text{meta}} = \mathcal{L}^{\text{meta}}_{\theta_{\text{sample}}} - \mathcal{L}^{\text{meta}}_{\theta_{\text{base}}} \tag{3.13}$$

where the expectation operation in eq. (3.11) is eliminated by one-sample Monte Carlo approximation as well as the standard policy gradient method.

Afterwards, to start the new epoch with the twin models parameterized by the same θ, they are renewed from the superior model.

$$\begin{cases} \theta_{\text{base}} \leftarrow \theta_{\text{sample}} & (\Delta\mathcal{L}^{\text{meta}} \leq 0) \\ \theta_{\text{sample}} \leftarrow \theta_{\text{base}} & (\Delta\mathcal{L}^{\text{meta}} > 0) \end{cases} \tag{3.14}$$

The proposed method can perform the model learning and the meta-optimization simultaneously at each epoch. In addition, the additional computational cost for the meta-optimization is only to learn the twin model, resulting in sufficiently low computational cost and high efficiency. There are no requirements on the learning algorithm, the meta-objectives, and so on. That is, the proposed method is sufficiently versatile. Even if the number of hyperparameters is increased, the computational cost of the policy gradient method is merely proportional to it, keeping high scalability.

3.5 Experiments

3.5.1 Common conditions

In order to validate the effectiveness of the proposed method, model learning with two types of meta-objective is conducted with datasets collected in numerical simulations. One meta-objective is simply the minimization of the linear

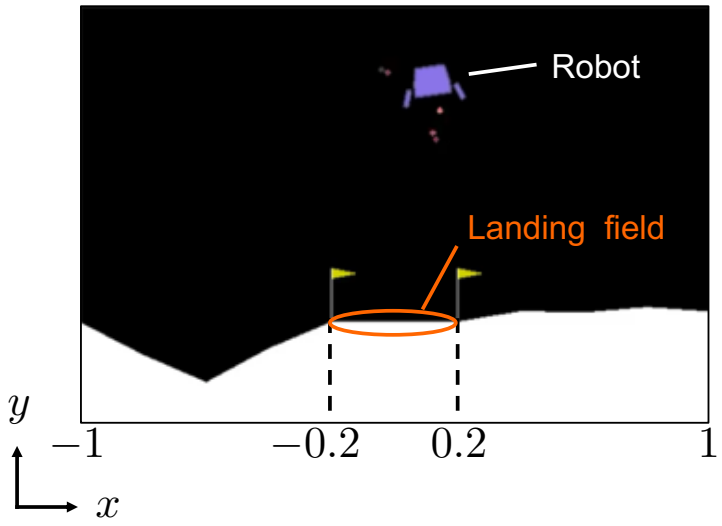


Figure 3.4: Human-operated single-agent environment: The robot is operated by an expert (human); The expert aims to land the robot on the landing field at zero speed.

weighted sum of the mean and worst losses to verify whether the proposed meta-optimization method succeeds in making reasonable adjustments. Another is the minimization of the negative log-likelihood in the long-term prediction as more practical case. In the following, the proposed method combining IDMO/SPMO and meta-optimization will be referred to as IDMO+MO/SPMO+MO, respectively.

The model to learn the stochastic dynamics is configured as a three-layered neural network with 100 neurons in each hidden layer, and the meta-policy is configured as a one-layered neural network with 100 neurons in hidden layer, in all trials. All hidden layers are configured as fully connected layers, and the activation function is the ReLU (Rectified Linear Unit). These networks output a multivariate diagonal Gaussian distribution and a Beta distribution, respectively, and implemented by Pytorch. They are optimized with one of the state-of-the-art stochastic GD optimizer, t-Adam [91], which is robust to noise and outliers in dataset.

Two types of simulation environments are prepared: 1) a human-operated single-agent environment; and 2) a multi-agent environment. Details of each

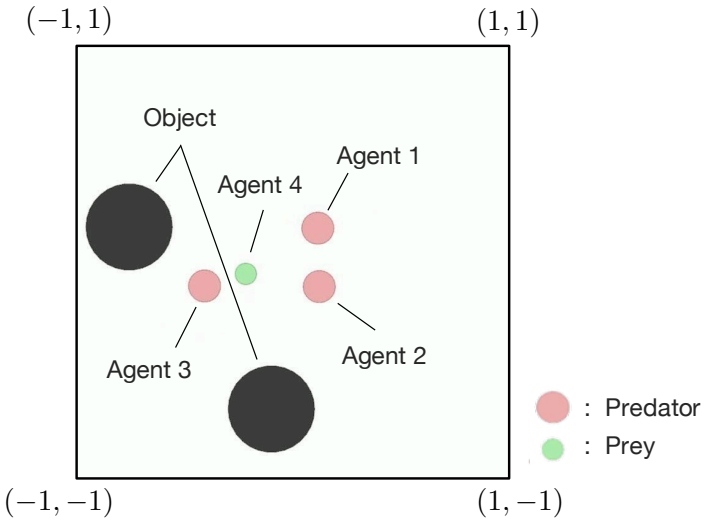


Figure 3.5: Multi-agent environment: four agents work in the same environment; the task of agents 1–3 (predators) is to catch an agent 4 (prey); the task of the agent 4 is to run away from the agent 1-3 on the screen.

Table 3.3: Hyperparameters in human-operated single-agent environment

Hyperparameter	Value
Learning rate of model predictor	0.0001
Batch size of model predictor	64
β for $\mathcal{L}_\theta^{\mathrm{SPMO}}$	0.0001
Learning rate of meta-policy	0.0001

environment is described below.

In both environments, their dataset were randomly divided in proportions such that $N_{\mathrm{trn}} : (N_{\mathrm{val}} + N_{\mathrm{tst}}) = 7 : 3$ and $N_{\mathrm{val}} : N_{\mathrm{tst}} = 7 : 3$, where N_{trn}, N_{val}, and N_{tst} are the numbers of training, validation, and test data, respectively. For each training condition, 10 trials of model learning in 200 epochs are performed with different random seeds to confirm the statistical performance.

Table 3.4: Hyperparameters in multi-agent environment

Hyperparameter	Value
Learning rate of model predictor	0.00001
Batch size of model predictor	32
β for $\mathcal{L}_\theta^{\mathrm{SPMO}}$	0.0001
Learning rate of meta-policy	0.0001

3.5.1.1 Human-operated single-agent environment

This environment is "LunarLanderContinuou-v2" provided by OpenAI Gym [92] (see Fig. 3.4). The robot moves in the environment by the thrust of the main engine and the left and right engines. The state space of the robot is eight-dimensional: two-dimensional absolute position and velocity; attitude and angular velocity; and two states of contact between the ground and each foot. The action space of the robot is two-dimensional: thrust of main engine; and thrust of left and right engines. The task to be accomplished is to softly land on the landing field and stop.

To train the stochastic dynamics model, we manually collected state transition data $\{(s_t,\ a_t),\ s_{t+1}\}$. The data collector generated the action sequences as an expert with the aim of realizing the task. The number of data in this dataset D is $N = 70,394$. The manually-collected dataset would contain bias in the states visited, and such a dataset is prone to bias and/or variance of the trained model.

In this environments, all the following trials are conducted with the hyperparameters shown in Table 3.3.

3.5.1.2 Multi-agent environment

This environment based on [93] consists of four agents and two objects, which are randomly placed at the beginning of each episode (see Fig. 3.5). The source codes of this environment can be downloaded from Github: `https://github.com/openai/multiagent-particle-envs`). The state space of the agents 1–3 (predators) is 16-dimensional: two-dimensional absolute position and velocity of itself; two-dimensional relative positions of the other agents and objects; and two-

dimensional velocity of the agent 4 (prey). The state space of the agent 4 (prey) is fourteen-dimensional: two-dimensional absolute position and velocity of itself; and two-dimensional relative positions of other agents. Each agent has a discrete action space for determining the moving direction (up/down/left/right).

This experiment can be regarded as a partially observable MDP (POMDP), where each agent does not share the actions of all the agents and the agent 4 does not know the positions of two objects. As a result, it is difficult to infer the true dynamics p_e from the observable states of each agent, and the model p_m will always contain uncertainty. Therefore, higher-order moments (the worst loss in our case) must be properly considered.

To make the dataset for this environment, the action of each agent at each time was designed as follows:

- Predator (i.e., the agents $i \in \{1, 2, 3\}$) pursues the agent 4.

$$a_i = \begin{cases} a_{\text{left}} & |d^{\text{h}}_{i,4}| \geq |d^{\text{v}}_{i,4}| \ \& \ d^{\text{h}}_{i,4} \leq 0 \\ a_{\text{right}} & |d^{\text{h}}_{i,4}| \geq |d^{\text{v}}_{i,4}| \ \& \ d^{\text{h}}_{i,4} > 0 \\ a_{\text{down}} & |d^{\text{h}}_{i,4}| < |d^{\text{v}}_{i,4}| \ \& \ d^{\text{v}}_{i,4} \leq 0 \\ a_{\text{up}} & |d^{\text{h}}_{i,4}| < |d^{\text{v}}_{i,4}| \ \& \ d^{\text{v}}_{i,4} > 0 \end{cases} \tag{3.15}$$

- Prey (i.e., the agent 4) run away from the agents 1–3.

$$a_4 = \begin{cases} a_{\text{left}} & |d^{\text{h}}_{4,i^{\text{h}}_{\text{min}}}| \leq |d^{\text{v}}_{4,i^{\text{v}}_{\text{min}}}| \ \& \ d^{\text{h}}_{4,i^{\text{h}}_{\text{min}}} > 0 \\ a_{\text{right}} & |d^{\text{h}}_{4,i^{\text{h}}_{\text{min}}}| \leq |d^{\text{v}}_{4,i^{\text{v}}_{\text{min}}}| \ \& \ d^{\text{h}}_{4,i^{\text{h}}_{\text{min}}} \leq 0 \\ a_{\text{down}} & |d^{\text{h}}_{4,i^{\text{h}}_{\text{min}}}| > |d^{\text{v}}_{4,i^{\text{v}}_{\text{min}}}| \ \& \ d^{\text{v}}_{4,i^{\text{v}}_{\text{min}}} > 0 \\ a_{\text{up}} & |d^{\text{h}}_{4,i^{\text{h}}_{\text{min}}}| > |d^{\text{v}}_{4,i^{\text{v}}_{\text{min}}}| \ \& \ d^{\text{v}}_{4,i^{\text{v}}_{\text{min}}} \leq 0 \end{cases}, \tag{3.16}$$

$$i^{\text{h}}_{\text{min}} = \arg\min_i |d^{\text{h}}_{4,i}|, \ i^{\text{v}}_{\text{min}} = \arg\min_i |d^{\text{v}}_{4,i}|$$

where $a_{\{left,right,down,up\}}$ represents left, right, down, up movement, $d^{\{\text{h,v}\}}_{i,j}$ is the relative position of the agent j in the horizontal, vertical direction from the agent i. The action of each agent is collected to keep each agent within 90% of the vertical and horizontal limits of the screen. With such ad-hoc controllers, the dataset D is collected with $N = 30,000$.

In this environment, all the following trials are conducted with the hyperparameters shown in Table 3.4. Note that all the agents are with the same hyperparameters, although the results of random initialization are different.

3.5.2 Meta-objective: linear weighted sum of mean and worst losses

In this validation, the meta objective is defined as the linear weighted sum of the mean and worst losses for the validation dataset D^{val}, as shown in the following equation.

$$\mathcal{L}_\theta^{\mathrm{meta}} = (1 - w_{\mathrm{wm}})\mathcal{L}_\theta^{\mathrm{mean}}(D^{\mathrm{val}}) + w_{\mathrm{wm}}\mathcal{L}_\theta^{\mathrm{worst}}(D^{\mathrm{val}}) \tag{3.17}$$

where $w_{\mathrm{wm}} \in [0, 1]$ denotes the priority of the worst loss. Bias and variance must be adjusted for optimization of this meta-objective. The proposed method is applied to each of the 21 loss functions generated by varying w_{wm} with 0.05 increments in the range $[0, 1]$, thereby validating the meta-optimization for α, which should be positively correlated with (but not linearly proportional to) w_{wm}.

3.5.2.1 Human-operated single-agent environment

The learning results by the proposed method in human-operated single-agent environment are shown below. Each value in the following graphs represents the mean and 95 % confidence interval over 10 trials of the values obtained in the final five epochs.

The transition of the learned α with evenly-spaced w_{wm} is shown in Fig. 3.6. As can be seen in Fig. 3.6, α tends to increase with the increase of w_{wm} for both IDMO+MO and SPMO+MO methods. In other words, the proposed meta-optimization was able to capture the positive correlation between the two variables α and w_{wm}.

The transitions of mean and worst losses shown in Figs. 3.7 (a) and (b), show that the learning results converge to different values depending on the change of α. Since minimizing the mean loss of the validation data is close to the meta-objective with small w_{wm}, the stochastic model was trained to focus on the mean

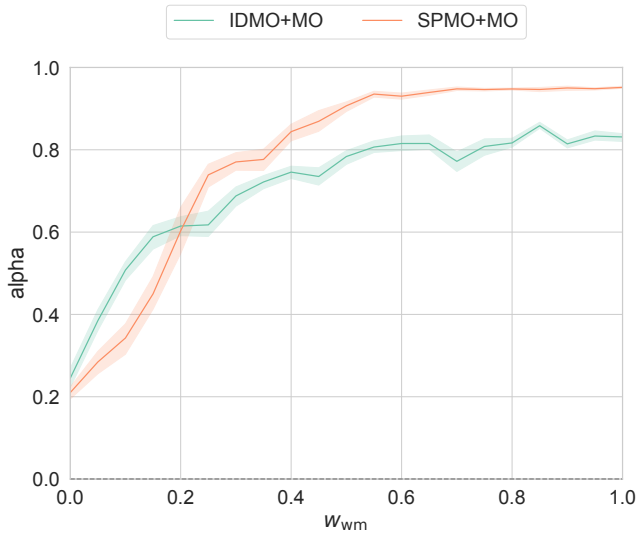
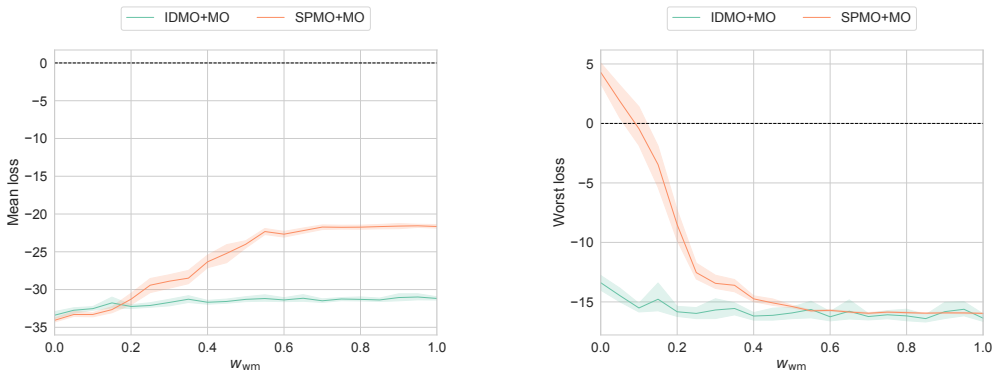


Figure 3.6: Learning results of α on human-operated single-agent environment and eq. (3.17): as w_{wm} increased, the α approached 1, and vice versa; the positive correlation between w_{wm} and α was captured.

loss at the expense of the worst loss, and vice versa. Compared to SPMO+MO, IDMO+MO shows less variation in the learning results, even though α varies in a similar range. This is because SPMO can cover all Pareto solutions in $\alpha \in [0, 1]$, whereas IDMO is affected by the mean loss even in $\alpha = 1$. Probably thanks to this capability of SPMO, it succeeded in minimizing the mean loss with small w_{wm} less than one of IDMO, and the worst loss with large w_{wm} more stably than IDMO (i.e. smaller confidence interval).

A simple simulation with the meta-objective of minimizing the weighted sum of the mean and worst losses for the validation data shows numerically that the proposed meta-optimization method can adaptively adjust the bias-variance trade-off at the same as model learning. Since the simulation environment is human-operated single-agent environment, the proposed method would be effective in dealing with uncertainty in human-involved biased datasets. In particular, the comparison results show that SPMO+MO can handle a wider range of trade-off than IDMO+MO.



(a) Mean loss (b) Worst loss

Figure 3.7: Learning results of the mean and worst losses on human-operated single-agent environment and eq. (3.17): (a) the mean loss was decreased as w_{wm} increased; (b) the worst loss was increased as w_{wm} increased; as a result, a model suitable for the meta-objective was learned.

3.5.2.2 Multi-agent environment

The results of applying the proposed method into the multi-agent environment are shown below, as well in the previous section.

The transition of the learned α with evenly-spaced w_{wm} is shown in Fig. 3.6. As well in the human-operated single-agent environment, the proposed meta-optimization method was able to captured the positive correlation between α and w_{wm} in both model learning methods. However, α saturated near 1 even when w_{wm} was relatively small compared to Fig. 3.6, suggesting that this environment is prone to large variance.

The mean and worst losses of the learning results (see Fig. 3.9) also show the similar tendency to Fig. 3.7, although the range of change in the case with IDMO was increased since the variance would be dominant. In addition, it can be seen that the model accuracy of SPMO was inferior to that of IDMO in both mean and worst losses when prioritizing the worst loss with large w_{wm}. This is probably because IDMO always uses the gradients of both the mean and worst losses, while SPMO uses either of them per batch depending on the max operator. Therefore, even with the same epochs, the number of uses of data to update the parameters

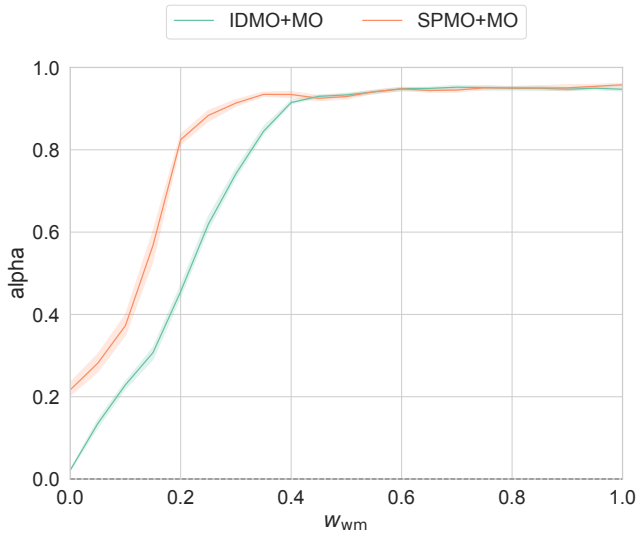


Figure 3.8: Learning results of α on multi-agent environment and eq. (3.17): as w_{wm} increased, the α approached 1, and vice versa; the positive correlation between w_{wm} and α was captured.

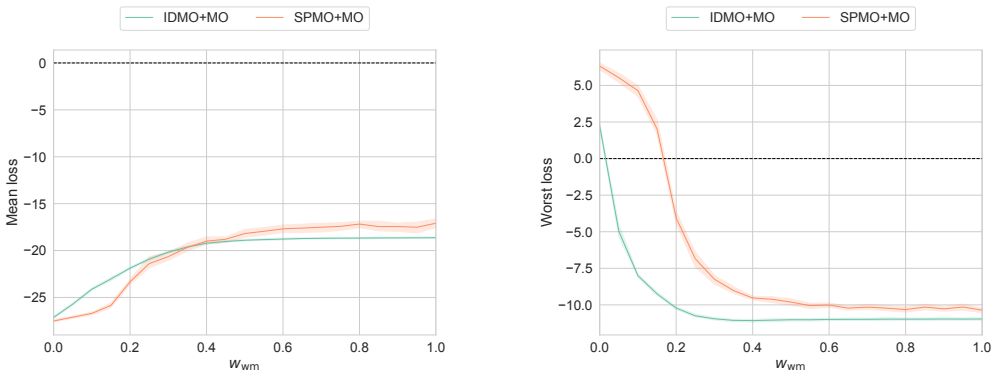
is reduced, resulting in delaying the model learning itself. This drawback may be mitigated by annealing β from the large initial value, for example.

3.5.3 Meta-objective: accuracy of long-term prediction

In this validation, the meta-objective is defined as the mean loss in long-term prediction for the validation dataset D^{val}, as shown in the following equation.

$$
\begin{aligned}
\mathcal{L}_\theta^{\mathrm{meta}} &= \frac{1}{K}\sum_{k=1}^{K} l_\theta^{(H,k)} \qquad (3.18)\\
l_\theta^{(h,k)} &= -\ln p_m(s_{t_k+h+1} \mid \bar{s}_{t_k+h},\ a_{t_k+h};\ \theta) - u\\
\bar{s}_{t_k+h} &= \begin{cases} s_{t_k} & h = 0 \\ \mathbb{E}[p_m(\cdot \mid \bar{s}_{t_k+h-1},\ a_{t_k+h-1};\ \theta)] & \text{otherwise} \end{cases}
\end{aligned}
$$

where H denotes the horizon of the prediction period and K denotes the number of sequences. Namely, it predicts the state sequence based on the state at t and the action sequence from t in the dataset, and the prediction accuracy at the end of the prediction period is employed as the meta-objective.



(a) Mean loss (b) Worst loss

Figure 3.9: Learning results of the mean and worst losses on multi-agent environment and eq. (3.17): (a) the mean loss was decreased as w_{wm} increased; (b) the worst loss was increased as w_{wm} increased; as a result, a model suitable for the meta-objective was learned.

In such a long-term prediction, the stochastic model is desired to be trained not only to improve the accuracy of the one-step prediction, but also to avoid outliers during the prediction period. This meta-objective, therefore, requires the optimal balance of the bias-variance trade-off, which is difficult to be revealed analytically. In order to verify whether the proposed meta-optimization method on SPMO can properly find the optimal balance, we experiment with $H = \{1, 10\}$ as below.

3.5.3.1 Human-operated single-agent environment

The learning results as box plots for the values obtained in the final five epochs in the human-operated single-agent environment are shown below.

First, the results for the time-horizon $H = 10$ are shown. Fig. 3.10 shows the results for meta-learned α. In both the SPMO+MO and IDMO+MO cases, α converged to around 1, indicating that the meta-optimization was done in a way that emphasized the worst loss.

The meta-objective obtained by the baseline model is shown in Fig. 3.11. Fig. 3.11 shows that smaller meta-objective is obtained by using SPMO+MO. The reason for this can be understood from the results of the mean and worst losses for each method shown in Figs. 3.12 (a) and (b). In the case of IDMO+MO,

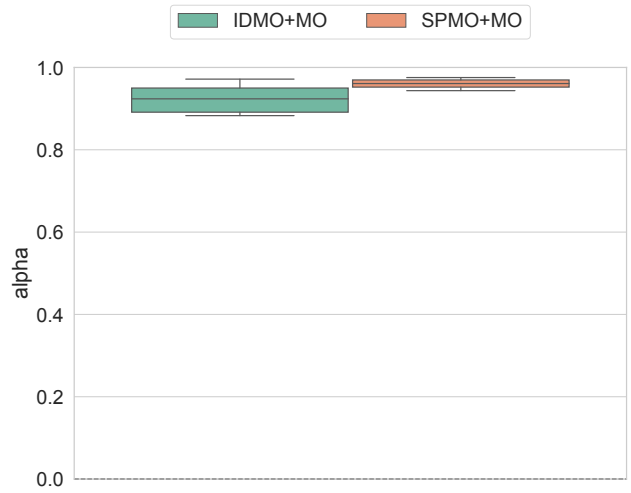


Figure 3.10: Learning results of α on human-operated single-agent environment and eq. (3.18) with $H = 10$: α was learned to be close to 1; the worst loss was emphasized in the long-term prediction in this environment.

the mean loss was still minimized even under $\alpha \simeq 1$, and the gap between it and the worst loss was enlarged. This gap would cause overlearning to the mean loss, and induced outliers during the long-term prediction. In contrast, SPMO+MO obtained the larger mean loss than that of IDMO+MO, but the gap between it and the worst loss and the variance of their losses were smaller, thereby achieving the stable prediction without overlearning. In addition, Fig. 3.13 shows that, both proposed methods reduce the meta-objective more than the conventional method using mean loss (Mean). Only SPMO+MO resulted in the same degree of reduction as the case of using the worst loss (Worst).

Next, the results for the time-horizon $H = 1$ are described below. According to the minimization results for α shown in Fig. 3.14, the smaller α, i.e. prioritizing the mean loss, would be better for this setting. This is a reasonable result because the meta-objective is the same as the low-level loss for model training when $\alpha = 0$, namely the worst loss is no longer considered. SPMO+MO succeeded in obtaining the smaller α than one of IDMO+MO, which may yield a slightly better result in the meta-objective shown in Fig. 3.15.

It is clear from Fig. 3.16, which shows the mean and worst losses with $H = 1$,

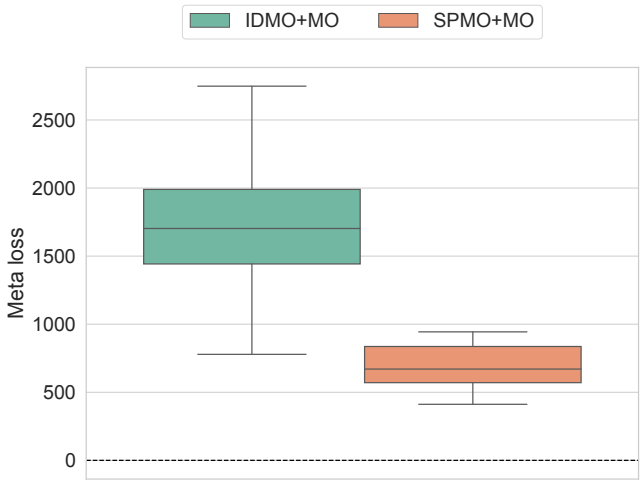
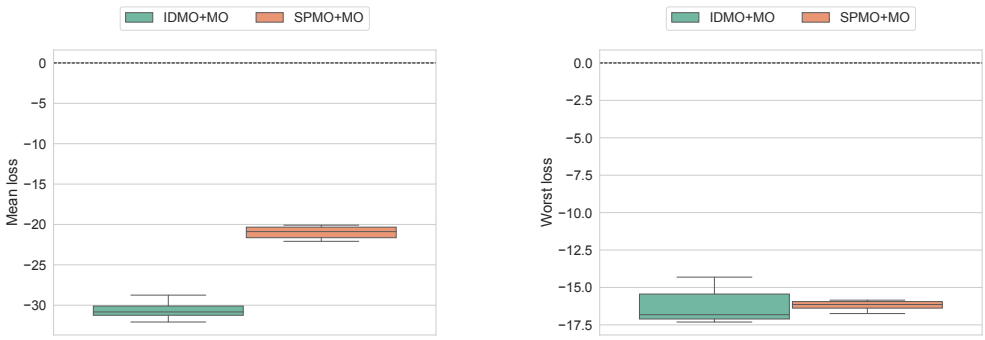


Figure 3.11: The results of the scores on human-operated single-agent environment and eq. (3.18) with $H = 10$: the score was reduced more in the case of SPMO+MO than in the case of IDMO+MO; namely, SPMO+MO improved the meta-objective.

and Fig. 3.12 why α was not sufficiently small in IDMO+MO. That is, although IDMO+MO obtained the different α, the mean and worst losses were almost the same, indicating a low dependency on α. On the other hand, in SPMO+MO, the mean loss was minimized to the same level as in IDMO+MO, and the worst loss was explicitly ignored instead. Such a high dependency on α enables SPMO+MO to find the preferred solution that satisfies the meta-objective as much as possible. Fig. 3.17 shows that, both proposed methods reduce the meta-objective more than Mean and Worst. This indicates that the proposed methods can obtain the learning results with less meta loss by using the intermediate Pareto solution between Mean and Worst. Since Mean lowers the meta loss more than Worst, α converged to a value that emphasizes mean loss, is reasonable.

3.5.3.2 Multi-agent environment

The learning results in the multi-agent environment are shown below. As well as the case of the human-operated environment, the results for the time-horizon $H = 10$ depicted in Fig. 3.18 obtained $\alpha \simeq 1$. In addition, as shown in Fig.3.19,



(a) Mean loss

(b) Worst loss

Figure 3.12: Learning results of the mean and worst losses on human-operated single-agent environment and eq. (3.18) with $H = 10$: IDMO+MO obtained the large difference between the mean and worst losses, which means that the variance of the learned stochastic model was large; in contrast, SPMO+MO succeeded in keeping the difference between the mean and worst losses small, resulting in that fatal errors in long-term prediction were less likely to occur, as indicated in Fig. 3.11.

SPMO+MO could minimize the meta-objective much more than IDMO+MO. The reason for this result is also the same as for the previous environment: i.e. SPMO+MO acquired the generalized model by appropriate suppression of overlearning confirmed from a small gap between the mean and worse losses in Fig.3.20, while IDMO+MO did not. Note that both losses were smaller in IDMO+MO, but they were computed for the training data. In addition, Fig. 3.21 shows that, only SPMO+MO reduces the meta-objective to the same level as Worst.

On the other hand, even under the meta-objective with $H = 1$, α was adjusted toward 1 to emphasize the worst loss in both methods (see Fig. 3.22). As a result, Figs. 3.23 and 3.24 indicate that SPMO+MO and IDMO+MO obtained comparable performance. Fig. 3.25 shows that, both proposed methods reduce the meta-objective slightly more than Mean and Worst. Contrary to the case of the human-operated single-agent environment, since Worst lowers the meta loss

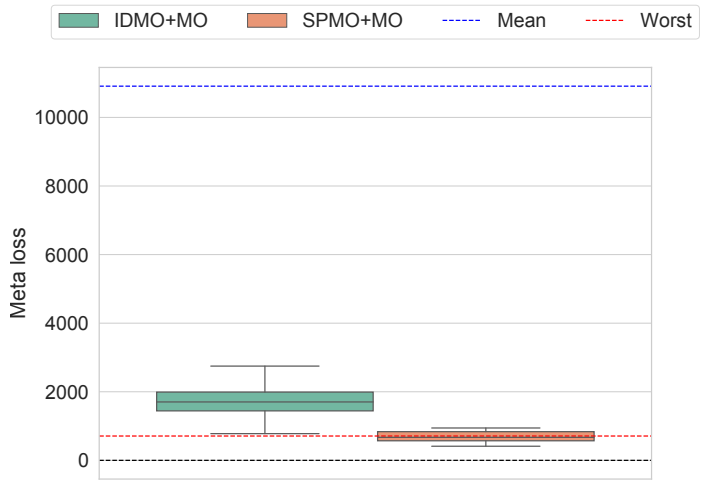


Figure 3.13: The comparison of the scores on human-operated single-agent environment and eq. (3.18) with $H = 10$: the scores of both SPMO+MO and IDMO+MO were reduced more in the cases of the maen loss, and only SPMO yielded results comparable to the case of the worst loss; namely, the proposed methods leads to a better Pareto solution than the conventional method, and the results suggest that convergence to the worst loss was the optimal solution.

more than Mean, α converged to a value that emphasizes worst loss, is reasonable.

The reason why α was optimized to be close to 1 even with $H = 1$ comes from the fact that this environment is partially observable (i.e. POMDP). Specifically, state transitions that occur in response to unobservable states are unavoidably expressed as uncertainty, hence the uncertainty of state transitions is inherently large in POMDP. The model trained with the mean loss cannot capture this uncertainty, and therefore it lacks generality to the validation and test data even if it is consistent with the meta-objective. To reduce the number of unexpected events as much as possible, the worst loss can be useful to make the variance of the model wider.

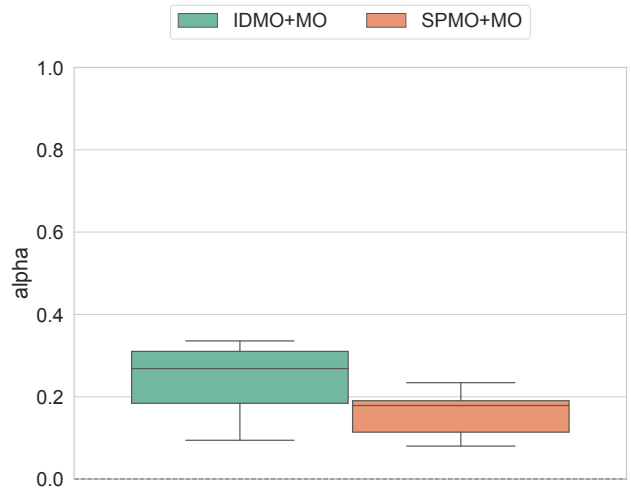


Figure 3.14: Learning results of α on human-operated single-agent environment and eq. (3.18) with $H = 1$: α was adjusted to be close to 0; the mean loss was emphasized to predict only the next step ($H = 1$ with the validation dataset.

3.5.4 Discussion

As investigated above, the optimal hyperparameters that lead to the preferred solution depend not only on the meta-objective but also on the contents of the dataset and the model architecture, hence it is not infeasible to give them analytically in advance. The proposed meta-optimization method based on the policy gradient allows us to obtain the preferred solution by adjusting the hyperparameters in a data-driven manner at the same time as learning the model. In addition, adjusting all hyperparameters will have little effect, namely we must make sure that which hyperparameters have the capability to find the Pareto frontier, as like α in SPMO.

One of the concerns is the exploration performance of the meta-optimization methods. Although the augmented weighted Tchebycheff scalarization theoretically guarantees the reachability of all Pareto solutions, optimizing the hyperparameters with the policy gradient method may lead to local solutions in the meta-objective. To address this open issue, adding an auxiliary term to the meta-objective function defined in eq. (3.12) to facilitate the exploration (e.g. entropy

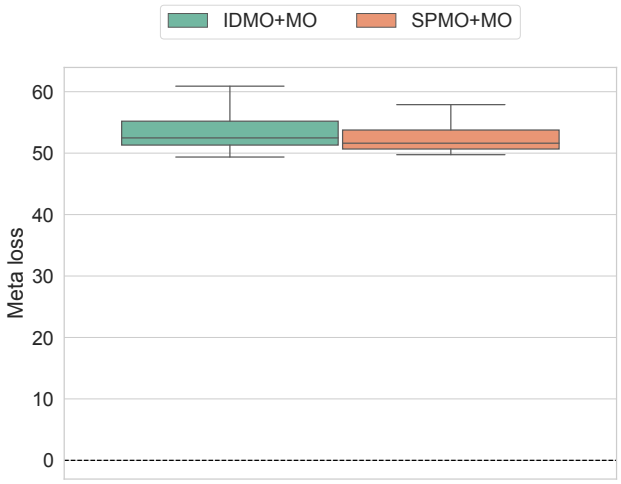
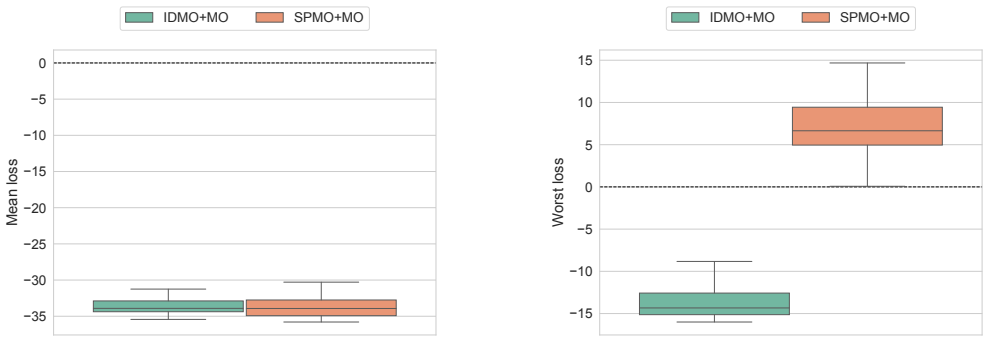


Figure 3.15: The results of the scores on human-operated single-agent environment and eq. (3.18) with $H = 1$: The scores obtained from the meta-optimization were comparable for both methods.

of the meta-policy [94]) may increase the reachability of the global optimal solution.

Another concern is the effect of the variation of the optimization target on the stochastic model learning. In the proposed method, the loss function, which is defined as a MOO problem used in stochastic model learning, is modified at each epoch due to the simultaneous low-level learning and meta-optimization. As in curriculum learning [95], adaptive changes in the optimization target sometimes provide opportunity to escape from the local solutions, but vice versa. In combination with the exploration facilitation described above, this problem may be solved in practice, but deeper investigation is necessary.

Finally, this study has developed the meta-optimization method starting from model learning for the model-based RL. Although this model learning is done in an offline manner with datasets already constructed, the model-based RL often involves planning and adding data using the model even in the process of learning [96, 97]. How the proposed method affects such online applications remains an open issue.



(a) Mean loss

(b) Worst loss

Figure 3.16: Learning results of the mean and worst losses on human-operated single-agent environment and eq. (3.18) with $H = 1$: since the worst loss was not needed to be minimized in this configuration, SPMO+MO ignored it to prioritize the mean loss.

3.6 Conclusion

This chapter proposed a stochastic model learning method that is adjustable the bias-variance trade-off of the stochastic model according to higher-level objective. The proposed method consists of the loss function derived from the two-step MOO problem with inter-data and statistic-perspective objectives, and the meta-optimization of the hyperparameter in the loss function. Specifically, we first pointed out that the conventional loss for model learning is described as the inter-data MOO problem. The inter-data MOO was reformulated as the multiple single objective optimizations using the augmented weighted Tchebycheff scalarization. Furthermore, by applying the augmented weighted Tchebycheff scalarization again to the weighted sum of the mean and worst losses naturally obtained above, we defined the loss function as the stochastic-perspective MOO problem. The meta-optimization method was newly developed to balance the bias and the variance of the resulting stochastic model by adjusting the hyperparameter in the proposed loss function. Inspired by the policy-gradient method, that can be accomplished simultaneously with model learning only during a single trial with two model learners. The proposed method was applied to the human-operated single-agent and multi-agent environments with different types

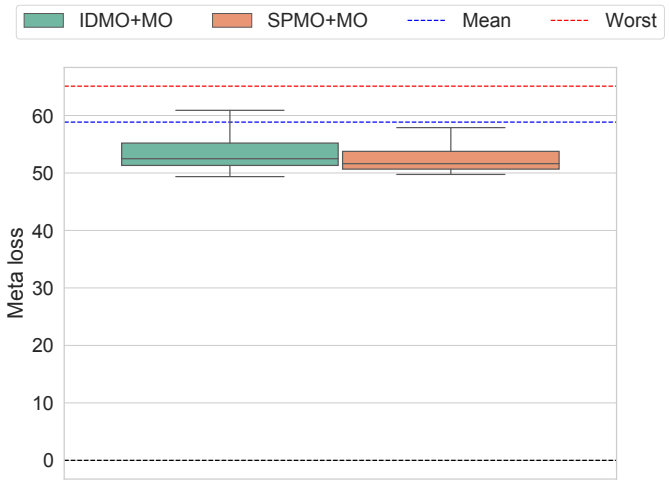


Figure 3.17: The comparison of the scores on human-operated single-agent environment and eq. (3.18) with $H = 1$: the scores of both SPMO+MO and IDMO+MO were reduced more in the cases of the maen loss and the worst loss; namely, the proposed methods leads to a better Pareto solution than the conventional method.

of uncertainty. First, the weighted sum of the mean loss and the worst loss was used as the meta-objective. The results showed that the hyperparameter was able to be adjusted according to the weight between the mean and worst loss with positive correlation. Next, the long-term prediction accuracy was used as another practical meta-objective. We found that the proposed method can improve the long-term prediction accuracy by revealing the best balance of the bias-variance trade-off and avoiding overfitting to the training data.

Figure 3.18: Learning results of α on multi-agent environment and eq. (3.18) with $H = 10$: α was trained to be close to 1; the worst loss was emphasized in the long-term prediction in this environment, as in the case of the human-operated single-agent environment.

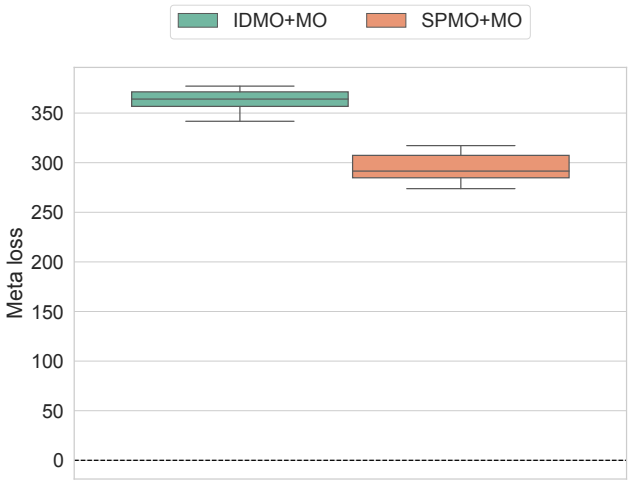
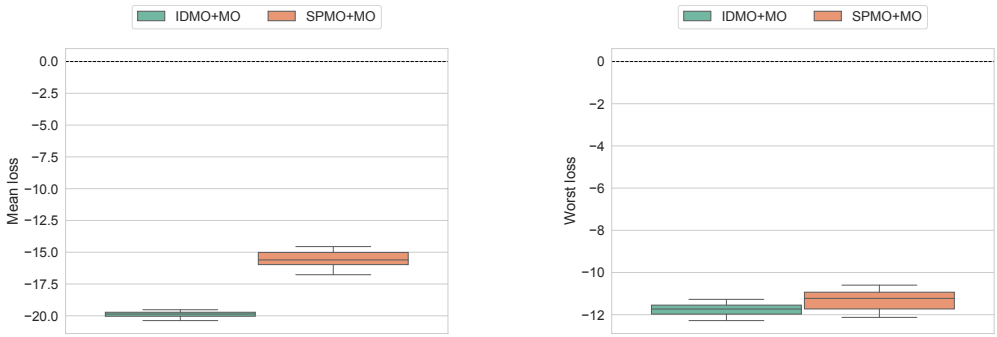


Figure 3.19: The results of the scores on multi-agent environment and eq. (3.18) with $H = 10$: the score was reduced more in the case of SPMO+MO than in the case of IDMO+MO, as in the case of the human-operated single-agent environment.



(a) Mean loss (b) Worst loss

Figure 3.20: Learning results of the mean and worst losses on multi-agent environment and eq. (3.18) with $H = 10$: SPMO+MO succeeded in keeping the difference between the mean and worst losses smaller than that of IDMO+MO, as in the case of the human-operated single-agent environment.

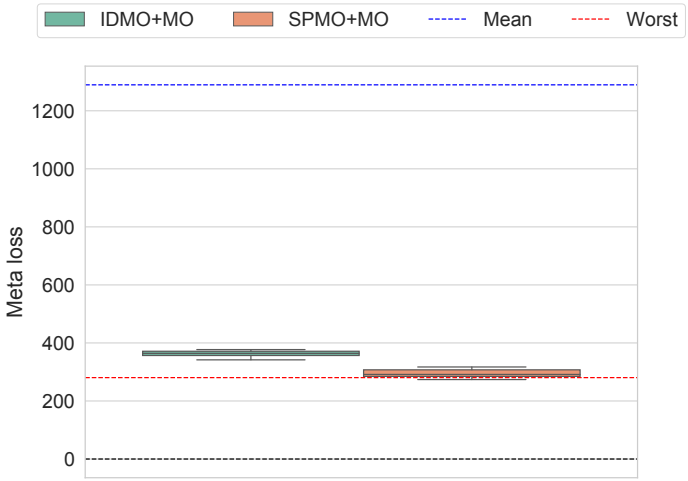


Figure 3.21: The comparison of the scores on multi-agent environment and eq. (3.18) with $H = 10$: the score of SPMO+MO was almost the same level as that of in the cases of the worst loss; namely, the results suggest that convergence to the worst loss was the optimal solution.

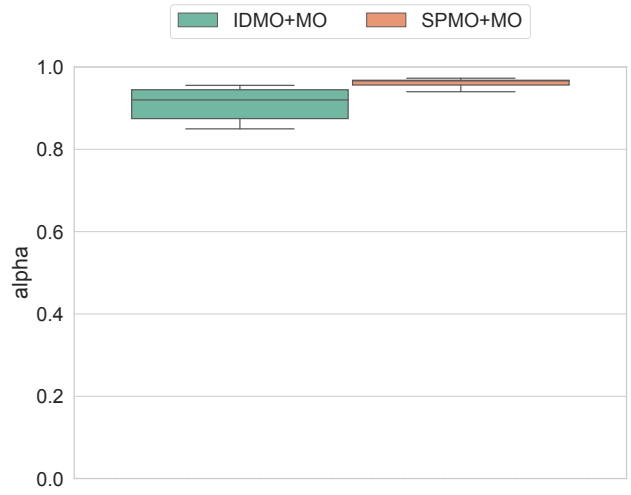


Figure 3.22: Learning results of α on multi-agent environment and eq. (3.18) with $H = 1$: α was optimized towards 1 unlike the case of the human-operated single-agent environment; this result suggested that the multi-agent environment was with high uncertainty and required the larger variance to cover it.

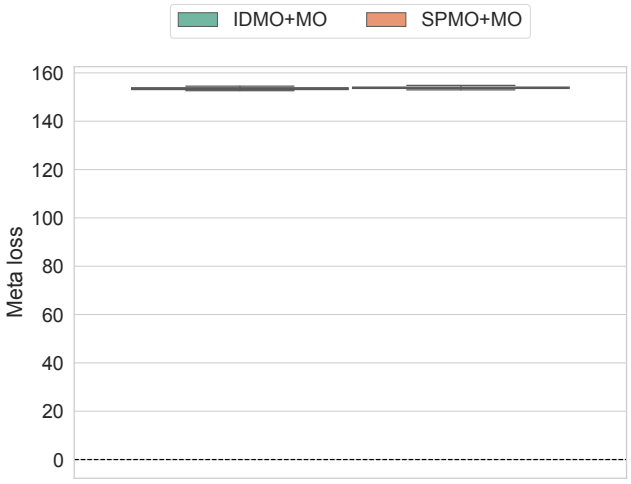
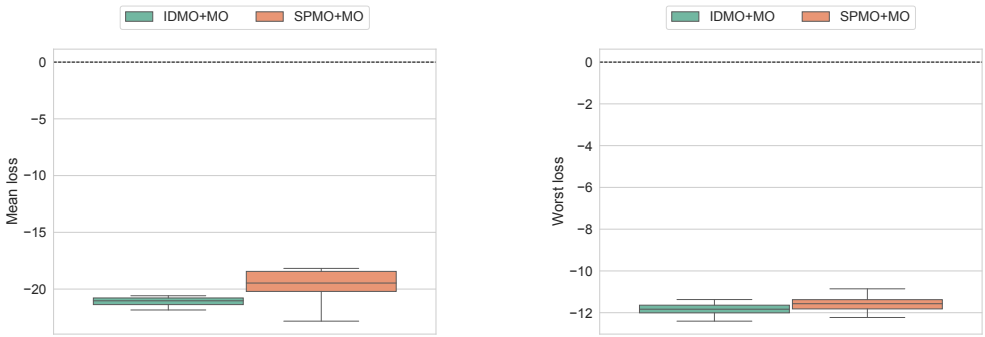


Figure 3.23: The results of the scores on multi-agent environment and eq. (3.18) with $H = 1$: the scores obtained from the meta-optimization were comparable for both methods.



(a) Mean loss (b) Worst loss

Figure 3.24: Learning results of the mean and worst losses on multi-agent environment and eq. (3.18) with $H = 1$: (a) the mean loss was kept at the same level for both methods; (b) the worst loss was also comparable for both methods.

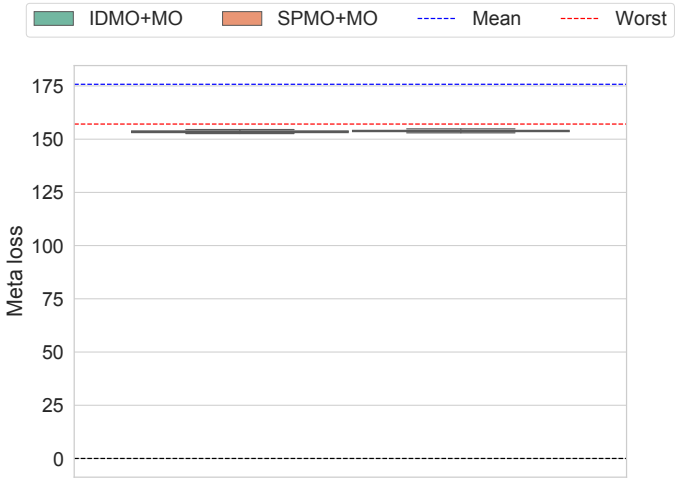


Figure 3.25: The comparison of the scores on multi-agent environment and eq. (3.18) with $H = 1$: the scores of both SPMO+MO and IDMO+MO were reduced slightly in the cases of the maen loss and the worst loss; namely, the proposed methods leads to a little better Pareto solution than the conventional method.

4 Conclusion

4.1 Summary

This dissertation focuses on the development of bottom-up MARL, which is an autonomous distributed MARL in which each agent has a primitive task. By assuming the primitive tasks, the relationship between agents can take various forms, and therefore the algorithm that can handle the complex group behaviors is needed. In addition, the absence of a centralized system requires risk avoidance that takes into account the uncertainty of state transitions. We proposed the reward-shaping algorithm for acquiring the complex group behaviors, and the meta-optimization method for the bias-variance trade-off in the model learning for safety learning under the uncertainty of state transitions.

In Chapter 2, the reward shaping algorithm for the group behaviors based on the rewards shared among agents in bottom-up MARL, is proposed. The algorithm includes a component that classifies interests from the correlation coefficients of the rewards between agents. This enabled the agents to adaptively acquire complex group behaviors in which cooperative and competitive relationships are mixed. The Simulations and the experiments show that the proposed method enables each agent to acquire the appropriate group behaviors in both cooperative and competitive task settings.

In Chapter 3, the meta-optimization method to adjust the bias-variance tradeoff is proposed for application to model-based reinforcement learning. The proposed method consists of a formulation of the bias-variance tradeoff as a multi-objective optimization problem and a versatile and efficient meta-optimization method for its Pareto solution. The meta-optimization method is realized based on the measure gradient method. The proposed method can optimize the balance between bias and variance according to the uncertainty of the environment under

a meta-objective, is numerically demonstrated by the simulations.

For the application of bottom-up MARL to real-world MAS, fundamental techniques have been developed to account for the complexity and the uncertainty.

4.2 Future work

4.2.1 Improvements in the proposed reward shaping algorithm

In the proposed method, training of the reward predictor and reinforcement learning are performed simultaneously. When the learning of the reward predictor is slow, the policy becomes deterministic under uncertain predicted rewards. To adjust the learning speed of the reward prediction and/or the policy, the regularization techniques are also required for stable convergence of the proposed method.

Although the proposed method aims at a decentralized system, the reward sharing network is still assumed to be fully coupled. Designing a sparse network for communication of reward is important to apply the proposed algorithm to a truly large-scale MAS, in terms of communication and calculation costs. For example, agents with uncorrelated tasks can be identified from the variance of predicted rewards, and communication with such agents is not required.

Furthermore, the current algorithm does not have a mechanism to consider the task accomplishment of each agent or the entire the MAS. The problem may be that agents with tasks that are easy to accomplish are given preferential treatment, or that agents who should contribute to the overall system prioritize their own tasks. To solve such problems, developping a bottom-up MARL method that can incorporate constraints on the minimum tasks to be accomplished in the entire MAS as conditions, is expected.

We will challenge to control really complicated MASs by resolving the above issues.

4.2.2 Improvements in the proposed meta-optimization algorithm

As mentioned in Section 3.5.4, the exploration performance should be guaranteed in order to acquire global solution.

The analysis of the learning dynamics during meta-optimization is not completed yet. Since the proposed method performs model learning and meta-learning simultaneously, balancing the convergence speed of each, is important. If the model learning is too fast, the predictive model will become deterministic before arriving at the meta-optimal Pareto solution. On the other hand, if the meta-optimization is too fast, not enough samples can be considered.

Furthermore, by extending the proposed method in an online learning manner, it can be integrated with model-based RL. However, when sampling is simply based on reinforcement learning, to achieve risk avoidance until a sufficient model is trained, is difficult. For example, pre-collection of samples by experts should be used to ensure the security of learning from the initial stage.

4.2.3 Safety bottom-up multi-agent reinforcement learning

An important task is to apply the proposed methods in Chapter 3 to model-based reinforcement learning and to validate its effectiveness. In particular, to verify whether agents can safely design planning for learning in uncertain environments, is important. Then, by applying it to bottom-up MARL including reward shaping as described in Chapter 2, we develop a safety MARL method under the framework of model-based reinforcement learning. Model-based reinforcement learning requires the reward prediction as well as the state prediction, and as shown in Chapter 2, the agents learn a predictive model of reward in the process of the proposed reward shaping. By applying meta-optimization methods to this reward prediction, considering the bias-variance trade-off for all predicted values is possible.

In addition, and related to Sec. 4.2.1, constraints on rewards could be imposed in order to take into account the minimum tasks to be accomplished by the entire MAS. The reward in reinforcement learning is only a reinforcement signal,

but by using a reward predictor, for example, a threshold can be given as a condition to be satisfied. By planning the predictive rewards corresponding to agents with tasks to be cooperated so that they retain sufficient values during long-term prediction, we believe that model-based MARL that takes into account the known cooperation conditions is possible.

Acknowledgements

I would like to express my sincere gratitude to all those who helped me in completing this doctoral dissertation.

I would like to express my utmost appreciation to my supervisor, Prof. Kenji Sugimoto. During my five years of graduate school life, Prof. Sugimoto gave me a lot of guidance, including not only the content of my research but also the attitude of a researcher. I would like to express my deepest gratitude to Prof. Takahiro Wada for accepting the position of co-supervisor and for providing me with guidance from the peripheral fields of robotics. I would like to express my deepest gratitude to my co-supervisor, Associate Prof. Takamitsu Matsubara, for his many valuable insights, mainly in the field of machine learning. I would like to express my sincere gratitude to my co-supervisor, Assistant Prof. Taisuke Kobayashi for his comprehensive guidance in all aspects of my research and encouragement in my research activities. I would like to express my deepest gratitude to Dr. Tsukasa Ogasawara, Executive Director/Vice President of Nara Institute of Science and Technology, for his suggestive advices from the viewpoint of robotics and practical applications.

I would like to express my appreciation to Assistant Prof. Kenta Hanada for his valuable comments at the workshop. I would like to express my gratitude to Ms. Hideko Hayashi, the secretary of the Intelligent Systems Control Laboratory, for her help in various aspects of my graduate school life. I would also like to thank the students of Intelligent System Control Laboratory for improving the quality of my research through daily discussions.

I would like to thank my parents, Kaoru and Yae Aotani, from the bottom of my heart.

This research was partly supported by JST PRESTO Grant Number: JPMJPR20C3.

References

[1] L. E. Parker, "Distributed intelligence: Overview of the field and its application in multi-robot systems." in *AAAI fall symposium: regarding the intelligence in distributed intelligent systems*, 2007, pp. 1–6.

[2] R. N. Darmanin and M. K. Bugeja, "A review on multi-robot systems categorised by application domain," in *2017 25th mediterranean conference on control and automation (MED)*. IEEE, 2017, pp. 701–706.

[3] T. Arai, E. Pagello, L. E. Parker *et al.*, "Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.

[4] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.

[5] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in multi-agent systems and applications-1*. Springer, 2010, pp. 183–221.

[6] R. N. Darmanin and M. K. Bugeja, "A review on multi-robot systems categorised by application domain," in *Mediterranean Conference on Control and Automation*. IEEE, 2017, pp. 701–706.

[7] H. Bai and J. T. Wen, "Cooperative load transport: A formation-control perspective," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 742–750, 2010.

[8] R. Sandau, K. Brieß, and M. D'Errico, "Small satellites for global coverage: Potential and limits," *ISPRS Journal of photogrammetry and Remote Sensing*, vol. 65, no. 6, pp. 492–504, 2010.

[9] K. M. Wurm, C. Dornhege, B. Nebel, W. Burgard, and C. Stachniss, "Coordinating heterogeneous teams of robots using temporal symbolic planning," *Autonomous Robots*, vol. 34, no. 4, pp. 277–294, 2013.

[10] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings*. Elsevier, 1994, pp. 157–163.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[12] S. Sen, M. Sekaran, J. Hale *et al.*, "Learning to coordinate without sharing information," in *AAAI Conference on Artificial Intelligence*, vol. 94, 1994, pp. 426–431.

[13] M. J. Matarić, "Reinforcement learning in the multi-robot domain," *Autonomous Robots*, vol. 4, no. 1, pp. 73–83, 1997.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[16] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137–2145.

[17] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[18] D. M. Guisi, R. Ribeiro, M. Teixeira, A. P. Borges, and F. Enembreck, "Reinforcement learning with multiple shared rewards," *Procedia Computer Science*, vol. 80, pp. 855–864, 2016.

[19] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling others using oneself in multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2018, pp. 4254–4263.

[20] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[21] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer, "Potential-based difference rewards for multiagent reinforcement learning," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 2014, pp. 165–172.

[22] S. Devlin and D. Kudenko, "Theoretical considerations of potential-based reward shaping for multi-agent systems," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 225–232.

[23] A. K. Agogino and K. Tumer, "Analyzing and visualizing multiagent rewards in dynamic and stochastic domains," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 2, pp. 320–338, 2008.

[24] T. Aotani, T. Kobayashi, and K. Sugimoto, "Bottom-up multi-agent reinforcement learning for selective cooperation," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2018, pp. 3590–3595.

[25] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[26] F. L. Da Silva, R. Glatt, and A. H. R. Costa, "Simultaneously learning and advising in multiagent reinforcement learning," in *International Conference on Autonomous Agents and MultiAgent Systems*, 2017, pp. 1100–1108.

[27] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv preprint arXiv:1802.08757*, 2018.

[28] J. N. Foerster, F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. Bowling, "Bayesian action decoder for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1811.01458*, 2018.

[29] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," *arXiv preprint arXiv:1810.02912*, 2018.

[30] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multiagent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4190–4203.

[31] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," *arXiv preprint arXiv:1802.05438*, 2018.

[32] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2681–2690.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[34] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[35] H. Van Seijen, A. R. Mahmood, P. M. Pilarski, M. C. Machado, and R. S. Sutton, "True online temporal-difference learning," *Journal of Machine Learning Research*, vol. 17, no. 145, pp. 1–40, 2016.

[36] T. Kobayashi, "Student-t policy in reinforcement learning to acquire global optimum of robot control," *Applied Intelligence*, vol. 49, no. 12, pp. 4335–4347, 2019.

[37] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," *arXiv preprint arXiv:1703.01732*, 2017.

[38] D. E. Knuth, *Seminumerical Algorithms, Vol. 2: The Art of the Computer Programming.* Addison-Wesley, 1981.

[39] R. DerSimonian and N. Laird, "Meta-analysis in clinical trials," *Controlled clinical trials*, vol. 7, no. 3, pp. 177–188, 1986.

[40] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[41] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[42] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[43] A. Khan, C. Zhang, D. D. Lee, V. Kumar, and A. Ribeiro, "Scalable centralized deep multi-agent reinforcement learning via policy gradients," *arXiv preprint arXiv:1805.08776*, 2018.

[44] A. Malysheva, D. Kudenko, and A. Shpilman, "Magnet: Multi-agent graph network for deep multi-agent reinforcement learning," in *2019 XVI International Symposium" Problems of Redundancy in Information and Control Systems"(REDUNDANCY).* IEEE, 2019, pp. 171–176.

[45] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[46] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[47] G. Bingjing, H. Jianhai, L. Xiangpan, and Y. Lin, "Human–robot interactive control based on reinforcement learning for gait rehabilitation training robot," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839584, 2019.

[48] H. Oliff, Y. Liu, M. Kumar, M. Williams, and M. Ryan, "Reinforcement learning for facilitating human-robot-interaction in manufacturing," *Journal of Manufacturing Systems*, vol. 56, pp. 326–340, 2020.

[49] K. Zhang, T. Sun, Y. Tao, S. Genc, S. Mallya, and T. Basar, "Robust multi-agent reinforcement learning with model uncertainty." in *NeurIPS*, 2020.

[50] T. Aotani, T. Kobayashi, and K. Sugimoto, "Bottom-up multi-agent reinforcement learning by reward shaping for cooperative-competitive tasks," *Applied Intelligence*, vol. 51, no. 7, pp. 4434–4452, 2021.

[51] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[52] W. Zhang, O. Bastani, and V. Kumar, "Mamps: Safe multi-agent reinforcement learning via model predictive shielding," *arXiv preprint arXiv:1910.12639*, 2019.

[53] D. D. Fan, A.-a. Agha-mohammadi, and E. A. Theodorou, "Deep learning tubes for tube mpc," *arXiv preprint arXiv:2002.01587*, 2020.

[54] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Dynamic tube mpc for nonlinear systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1655–1662.

[55] S. Gros and M. Zanon, "Safe reinforcement learning with stability & safety guarantees using robust mpc," *arXiv preprint arXiv:2012.07369*, 2020.

[56] R. Pasquier and I. F. Smith, "Robust system identification and model predictions in the presence of systematic uncertainty," *Advanced Engineering Informatics*, vol. 29, no. 4, pp. 1096–1109, 2015.

[57] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems*, 2018, pp. 4754–4765.

[58] H. Xu, C. Caramanis, and S. Mannor, "Robust regression and lasso," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3561–3574, 2010.

[59] D. Bertsimas and O. Nohadani, "Robust maximum likelihood estimation," *INFORMS Journal on Computing*, vol. 31, no. 3, pp. 445–458, 2019.

[60] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[61] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas, "A modern take on the bias-variance tradeoff in neural networks," *arXiv preprint arXiv:1810.08591*, 2018.

[62] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, 2019.

[63] V.-J. Aguilera-Rueda, N. Cruz-Ramírez, and E. Mezura-Montes, "Data-driven bayesian network learning: A bi-objective approach to address the bias-variance decomposition," *Mathematical and Computational Applications*, vol. 25, no. 2, p. 37, 2020.

[64] S. Paul, V. Kurin, and S. Whiteson, "Fast efficient hyperparameter tuning for policy gradients," *arXiv preprint arXiv:1902.06583*, 2019.

[65] D. Salinas, H. Shen, and V. Perrone, "A quantile-based approach for hyperparameter transfer learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8438–8448.

[66] G. Chen, "Merging deterministic policy gradient estimations with varied bias-variance tradeoff for effective deep reinforcement learning," *arXiv preprint arXiv:1911.10527*, 2019.

[67] R. E. Steuer and E.-U. Choo, "An interactive weighted tchebycheff procedure for multiple objective programming," *Mathematical programming*, vol. 26, no. 3, pp. 326–344, 1983.

[68] K. Dächert, J. Gorski, and K. Klamroth, "An adaptive augmented weighted tchebycheff method to solve discrete, integer-valued bicriteria optimization problems," Technical Report BUWAMNA-OPAP 10/06, University of Wuppertal, FB Mathematik ⋯, Tech. Rep., 2010.

[69] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint arXiv:1810.03548*, 2018.

[70] Y. Li, Y. Yang, W. Zhou, and T. Hospedales, "Feature-critic networks for heterogeneous domain generalization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3915–3924.

[71] Y. Balaji, S. Sankaranarayanan, and R. Chellappa, "Metareg: Towards domain generalization using meta-regularization," *Advances in Neural Information Processing Systems*, vol. 31, pp. 998–1008, 2018.

[72] J. Grabocka, R. Scholz, and L. Schmidt-Thieme, "Learning surrogate losses," *arXiv preprint arXiv:1905.10108*, 2019.

[73] S. Bechtle, A. Molchanov, Y. Chebotar, E. Grefenstette, L. Righetti, G. Sukhatme, and F. Meier, "Meta learning via learned loss," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 4161–4168.

[74] C. Huang, S. Zhai, W. Talbott, M. B. Martin, S.-Y. Sun, C. Guestrin, and J. Susskind, "Addressing the loss-metric mismatch with adaptive loss alignment," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2891–2900.

[75] W. Zhou, Y. Li, Y. Yang, H. Wang, and T. M. Hospedales, "Online meta-critic learning for off-policy actor-critic methods," *arXiv preprint arXiv:2003.05334*, 2020.

[76] V. Veeriah, M. Hessel, Z. Xu, J. Rajendran, R. L. Lewis, J. Oh, H. van Hasselt, D. Silver, and S. Singh, "Discovery of useful questions as auxiliary tasks," in *NeurIPS*, 2019.

[77] S. Gonzalez and R. Miikkulainen, "Improved training speed, accuracy, and data utilization through loss function optimization," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.

[78] R. Houthooft, R. Y. Chen, P. Isola, B. C. Stadie, F. Wolski, J. Ho, and P. Abbeel, "Evolved policy gradients," *arXiv preprint arXiv:1802.04821*, 2018.

[79] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman *et al.*, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019.

[80] M. Feurer, J. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[81] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[82] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[83] P. Domingos, "A unified bias-variance decomposition," in *Proceedings of 17th International Conference on Machine Learning*, 2000, pp. 231–238.

[84] T. Heskes, "Bias/variance decompositions for likelihood-based estimators," *Neural Computation*, vol. 10, no. 6, pp. 1425–1433, 1998.

[85] H. J. Bierens, "Information criteria and model selection," *Manuscript, Penn State University*, 2004.

[86] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[87] N. Hansen and S. Kern, "Evaluating the cma evolution strategy on multimodal test functions," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2004, pp. 282–291.

[88] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.

[89] A. Engau, "Proper efficiency and tradeoffs in multiple criteria and stochastic optimization," *Mathematics of operations research*, vol. 42, no. 1, pp. 119–134, 2017.

[90] T. K. Ralphs, M. J. Saltzman, and M. M. Wiecek, "An improved algorithm for solving biobjective integer programs," *Annals of Operations Research*, vol. 147, no. 1, pp. 43–70, 2006.

[91] W. E. L. Ilboudo, T. Kobayashi, and K. Sugimoto, "Robust stochastic gradient descent with student-t distribution based first-order momentum," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[92] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[93] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[94] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," *arXiv preprint arXiv:1703.01732*, 2017.

[95] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[96] A. Strehl and M. Littman, "Online linear regression and its application to model-based reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1417–1424, 2007.

[97] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in *Conference on Robot Learning*. PMLR, 2018, pp. 617–629.

Publication List

Journal Papers

[1] Takumi Aotani, Taisuke Kobayashi, and Kenji Sugimoto, “Meta-Optimization of Bias-Variance Trade-off in Stochastic Model Learning,” IEEE Access, Vol. 9, pp. 148783-148799, 2021.

[2] Takumi Aotani, Taisuke Kobayashi, and Kenji Sugimoto, “Bottom-up Multi-agent Reinforcement Learning by Reward Shaping for Cooperative-Competitive Tasks,” Applied Intelligence, Vol. 51, No. 7, pp. 4434-4452, 2021.

Conference Papers

[1] Taisuke Kobayashi, Takumi Aotani, Julio Rogelio Guadarrama-Olvera, Emmanuel Dean-Leon, and Gordon Cheng, “Reward-Punishment Actor-Critic Algorithm Applying to Robotic Non-grasping Manipulation,” 2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics, pp. 37-42, Oslo, Norway, Aug. 2019.

Domestic Conference

[1] 青谷拓海，小林泰介，杉本謙二，“確率モデル学習のためのバイアス・バリアンス を調整する方策勾配型メタ最適化”，日本ロボット学会学術講演会，1I2-05，Zoom 開催，2021 年 9 月.（査読無し，口頭）

[2] 青谷拓海，小林泰介，杉本謙二，“バイアス・バリアンスのトレードオフを考慮可能な確率モデル学習”，第 33 回自律分散システム・シンポジウム，1A1-2，Zoom 開催，2021 年 3 月.（査読無し，口頭）

[3] 青谷拓海，小林泰介，杉本謙二，“多変量分布を用いた報酬予測による利害関係を考慮したマルチエージェント強化学習”，第 32 回自律分散システム・シンポジウム，1C1-3，東京，2020 年 1 月.（査読無し，口頭）

[4] 青谷拓海，小林泰介，杉本謙二，“状況により変化する利害関係の推定に基づくマルチエージェント強化学習”，日本ロボット学会学術講演会，3B3-04，東京，2019 年 9 月.（査読無し，口頭）