A Study of Joint Policies Considering Bottlenecks and Fairness

Toshihiro Matsui
Nagoya Institute of Technology, Gokiso-cho Showa-ku Nagoya 466-8555, Japan

Keywords: Multiagent System, Multi-objective, Reinforcement Learning, Bottleneck, Fairness.

Abstract: Multi-objective reinforcement learning has been studied as an extension of conventional reinforcement learning approaches. In the primary problem settings of multi-objective reinforcement learning, the objectives represent a trade-off between different types of utilities and costs for a single agent. Here we address a case of multiagent settings where each objective corresponds to an agent to improve bottlenecks and fairness among agents. Our major interest is how learning captures the information about the fairness with a criterion. We employ leximin-based social welfare in a single-policy, multi-objective reinforcement learning method for the joint policy of multiple agents and experimentally evaluate the proposed approach with a pursuit-problem domain.

1 INTRODUCTION

Reinforcement learning (Sutton and Barto, 1998) has been studied for optimization methods of single and multiple agent systems. While conventional reinforcement learning methods address optimization problems with single objectives, multi-objective reinforcement learning (Liu et al., 2015) solves an extended class of problems for optimal policies to simultaneously improve multiple objectives.

In the primary problem settings of multi-objective reinforcement learning, the objectives represent a trade-off between different types of utilities and costs for a single agent. For example, in a setting where a robot collects items from its environment, the objective is the number of collected items, and other objectives are energy consumption and risk avoidance. Those objectives are represented as a multi-objective maximization/minimization problem.

Similar to the solution methods for conventional multi-objective optimization problems, several scalarization functions and social welfare criteria are applied to select one of optimal solutions, since many Pareto optimal (or quasi-optimal) policies exist in general cases. A fundamental scalarization function is the weighted summation for objectives. Other non-linear functions are also applied, including the weighted Tchebycheff function and its variants (Moffaert et al., 2013). While applying scalarization functions in learning processes optimizes a single policy, different approaches apply other minimization filters so that multiple policies are simultaneously optimized with a memory consumption.

Even though most studies of multi-objective reinforcement learning address a single agent domain, the objectives can also be related to multiple agents. A simple representation of such problems is the optimization of the joint policies of multiple agents. In this case, opportunities can be found for employing different scalarization functions and social welfare criteria to improve fairness among the agents. If the learning captures the information of the fairness, there might be additional opportunities for multiagent reinforcement learning with approximate decompositions of state-action space, while different approaches also exist, such as the methods to converge equilibria (Hu and Wellman, 2003; Hu et al., 2015; Awheda and Schwartz, 2016).

Several criteria represent fairness or inequality. The maximization of leximin (Bouveret and Lemaître, 2009; Greco and Scarcello, 2013; Matsui et al., 2018a; Matsui et al., 2018c; Matsui et al., 2018b) is an extension of maximin that improves the worst case utility and fairness. It also slightly improves the total utility. Leximin is based on the lexicographic order on the vectors whose values are sorted in ascending order. Several studies of combinational optimization problems show that the conventional optimization criterion is successfully replaced by the leximin criterion. Other inequality measurements could also be applied to this class of problems, while several modifications to improve the total utility

Matsui, T.
A Study of Joint Policies Considering Bottlenecks and Fairness.
DOI: 10.5220/0007577800800090
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 80-90
ISBN: 978-989-758-350-6

or cost with trade-offs are necessary for pure unfairness criteria.

In this work, we focus on single-policy, multi-objective reinforcement learning with *leximax*, which is a variant of leximin for minimization problems. Our primary interest is how different types of social welfare can be applied to a multi-objective reinforcement learning method. For the first study, we employ a simple class of pursuit problems with multiple hunters and a single target. We first consider a deterministic setting and apply a dynamic programming method. Next we investigate how the approach is affected by a non-deterministic setting and experimentally evaluate the effects and influences of the leximax criterion.

The rest of our paper is organized as follows. In the next section, we describe several backgrounds to our study. We address an approach that applies the leximax criterion to multi-objective Q-learning scheme for joint actions among agents in Section 3. Our proposed approach is experimentally evaluated in Section 4 and several discussions are described in Section 5. We conclude in Section 6.

2 PREPARATION

2.1 Reinforcement Learning and Multiple Objectives

Reinforcement learning is a unsupervised learning method that finds the optimal policy, which is a sequence of actions in a state transition model. In this work, we focus on Q-learning based methods as fundamental reinforcement learning methods. Q-learning consists of set of states S, set of actions A, observed reward/cost values for actions, evaluation values $Q(s,a)$ for each pair of state $s \in S$ and action $a \in A$, and parameters for learning. When action a is performed in state s, a state transition is caused with a corresponding reward/cost value based on an environment whose optimal policy should be mapped to Q-values.

For the minimization problem, a standard Q-learning is represented as follows

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(c + \gamma \min_{a'} Q(s',a')) \ , \tag{1}$$

where s and a are the current state and action, s' and a' are the next state and action, and c is a cost value for action a in state s. α and γ are the parameters of the learning and discount rates.

In basic problems, complete observation of all of the states without confusion is assumed so that the learning process correctly aggregates the Q-values. The Q-values are iteratively updated with action selections based on an exploration strategy, and the values can be sequentially or randomly updated to propagate them by dynamic programming.

Reinforcement learning is extended to find a policy that simultaneously optimizes multiple objectives. For multiple objective problems, cost values and Q-values are defined as the vectors of the objectives. Learning rules are categorized into single and multiple policy learning. In single policy learning, each Q-vector is updated for one optimal policy with a filter that selects a single objective vector. On the other hand, multiple policy learning handles multiple non-dominated objective vectors for each state-action pair and selects a Pareto optimal (or quasi-optimal) policy after the learning, while it generally requires more memory. We focus on single policy learning.

For a single policy with the social welfare of weighted summation, multi-objective Q-learning is represented as follows

$$\boldsymbol{Q}(s,a) \leftarrow (1-\alpha)\boldsymbol{Q}(s,a) + \alpha(\boldsymbol{c} + \gamma \mathrm{minws}(\boldsymbol{v}, \boldsymbol{Q}(s',a'))) \tag{2}$$

'minws' is a minimization operator based on a weighted summation with vector $\boldsymbol{v}$

$$\mathrm{argmin}_{\boldsymbol{Q}(s',a') \text{ for } a'} \boldsymbol{v} \cdot \boldsymbol{Q}(s',a') \tag{3}$$

In general settings, multiple objectives represent several trade-offs, such as the number of collected items and the energy consumption of a robot. Their main issue is how to obtain the Pareto optimal policies. On the other hand, we address a class of multi-objective problems where each objective corresponds to the cost of an agent.

2.2 Criteria of Bottlenecks and Fairness

There are several scalarization functions and social welfare criteria that select an optimal objective vector (Sen, 1997; Marler and Arora, 2004).

Summation $\sum_{i=1}^{K} v_i$ for objective vector $\boldsymbol{v} = [v_1, \cdots, v_K]$ is a fundamental scalarization function that considers the efficiency of the objectives. The minimization of the (weighted) summation is Pareto optimal. However, it does not capture the fairness among the objectives.

The minimization of maximum objective value $\max_{i=1}^{K} v_i$, called minimax, improves the worst case cost value. However, the minimization of the (weighted) maximum value is not Pareto optimal. A variant with the tie-breaking of the weighted maximum value with the weighted summation is called the augmented weighted Tchebycheff function. The minimization with this scalarization is Pareto optimal.

We focus on a criterion called leximax that resembles leximin for maximization problems. Leximin is defined as the dictionary order on objective vectors whose values are sorted in ascending order (Bouveret and Lemaître, 2009; Greco and Scarcello, 2013; Matsui et al., 2018a; Matsui et al., 2018c). The leximax relation is defined as follows with sorted objective vectors $\boldsymbol{v}$ and $\boldsymbol{v}'$ where the values in their original objective vectors are sorted in descending order.

Definition 1 (leximax)**.** *Let $\boldsymbol{v} = [v_1, \cdots, v_K]$ and $\boldsymbol{v}' = [v'_1, \cdots, v'_K]$ denote sorted objective vectors whose length is K. The order relation, denoted with $\succ_{leximax}$, is defined as follows. $\boldsymbol{v} \succ_{leximax} \boldsymbol{v}'$ if and only if $\exists t, \forall t' < t, v_{t'} = v'_{t'} \wedge v_t > v'_t$.*

The minimization on leximax improves the worst case cost value (bottleneck value) and the fairness among the cost values.

We employ the Theil index, which is a well-known measurement of inequality. Although this measurement was originally defined to compare incomes, we use it for cost vectors.

Definition 2 (Theil Index)**.** *For n objectives, Theil index T is defined as*

$$T = \frac{1}{n} \sum_i \frac{v_i}{\bar{v}} \log \frac{v_i}{\bar{v}}, \tag{4}$$

where v_i is the utility or the cost value of an objective and $\bar{v}$ is the mean utility value for all the objectives.

The Theil index takes a value in $[0, \log n]$, since it is an inverted entropy. When all the utilities or cost values are identical, the Theil index value is zero.

3 JOINT POLICIES CONSIDERING BOTTLENECKS AND EQUALITY

3.1 Example Domain

We employ a simplified pursuit game with four hunters and one target in a torus grid world (Figure 1). The hunters should cooperatively capture the target reducing the number of moves, while their individual move cost values are defined as multiple objectives.

To handle joint states and joint actions with a single table and sufficiently scan the state-action space, hunters take only one of two actions: stop or move. When a hunter selects a move, it advances to a neighboring cell in four directions to decrease its distance to the target. If all the hunters move, the target will be captured in relatively few steps. However, the moves of hunters cause costs. The target escapes from hunters. It moves to a neighboring cell to maximize its distance to the nearest hunter.

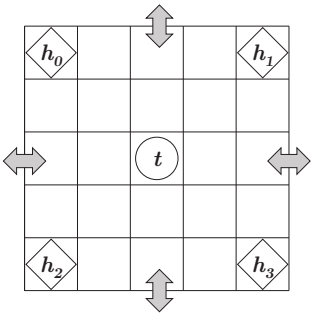

Figure 1: Pursuit game with four hunters h_i and one target t.

To eliminate noise from the stochastic process and the limited sensing, we first choose a deterministic process with a complete observation. The hunters and the target deterministically act with deterministic tie-breaking. The hunters know the location of the target. We address the problem of joint states and joint actions as the first study. The deterministic process is replaced by a non-deterministic process with randomness in the tie-breaking of target actions later.

Targets and hunters stay in the current cell or move to adjoining cells in either vertical or horizontal directions. The goal states are cases where one of hunters and the target are in the same cell. A joint policy is a sequence of joint actions of hunters from an initial state to a goal state.

The target moves to cells to maximize its minimum distance to the hunters. In the deterministic case, the directions are prioritized in the following order: upward, downward, left and right. If there are no improvements, the target stays in the current cell. In the non-deterministic case, ties are randomly broken with uniform distribution.

A hunter moves to cells to minimize its distance to the target. Here vertical directions are prioritized over horizontal directions. The hunters employ this tie-break rule in both the deterministic and non-deterministic cases. A move of a hunter causes a cost value of one, while the hunter can remain in the current cell at a cost value of zero. To avoid cases where no hunter moves, infinity cost values are set to all hunters for such a joint action.

We encoded the joint states and actions as follows. For each hunter, a pair of vertical and horizontal differences between the coordinates of the hunter and the target represent their relative locations. The difference of the minimum distance in the torus grid world is employed. The relative locations are combined for all hunters. Therefore, for grid size g, two types of actions, and four hunters, the number of joint state-

actions is $(g \times g \times 2)^4$.

The goal of the reinforcement learning is to find the optimal (or quasi-optimal) policy that reduces the total cost values with an appropriate criterion. In a common problem setting, the optimal joint policy can be defined as the joint policy which minimizes the total number of moves for all hunters. Namely, the summation is employed as the criterion to aggregate cost values for all hunters. As the first study, we focus on centralized reinforcement learning methods, while there might be opportunities to decompose the problem into multiple approximated ones for agents.

We address the joint policies that improve the fairness among the total cost values of individual hunters with the leximax criterion, while a common policy improves the total cost values for all of them. In the case of the minimization of leximax, the bottleneck is the maximum number of moves among individual hunters, and unfairness can also be defined for the number of moves among the hunters. Both should be reduced with the optimal (or quasi-optimal) policy. Our major interest is how such information is mapped in a Q-table of a reinforcement learning method.

3.2 Multi-objective Deterministic Decision Process

We start from a deterministic domain to consider the fundamental operations of the learning, which is identical as the dynamic programming for path-finding problems. Next we turn to the case of a non-deterministic domain.

3.2.1 Basic Scheme

We focus on the opportunities of the optimal Q-table for the leximax criterion. Therefore, we assume an appropriate exploration strategy and simply employ a propagation method that resembles the Bellman-Ford algorithm as an offline learning method. The update of the Q-table will eventually converge after a sufficient number of propagation operations. We eliminate statistic elements (i.e. the learning and discount rates) for the deterministic process.

A common deterministic update rule for single objective problems is represented as follows

$$Q(s,a) \leftarrow c + \min_a Q(s',a') \ . \qquad (5)$$

A deterministic update rule for multi-objective problems with summation scalarization is similarly represented

$$Q(s,a) \leftarrow \mathrm{sum}(\boldsymbol{c}) + \min_a Q(s',a') \ , \qquad (6)$$

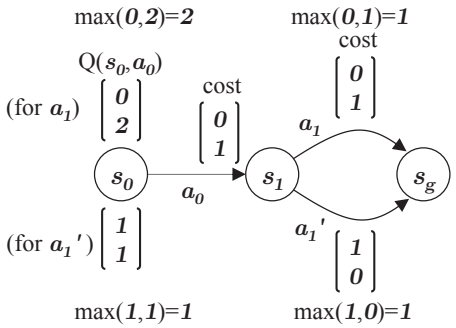

Figure 2: Incorrect minimax aggregation.

where $\mathrm{sum}(\boldsymbol{c})$ is a summation scalarization function that returns the summation of the values in cost vector $\boldsymbol{c}$. We do not use any weighted scalarization functions, because we assume that the hunters have the same priority. This rule is equivalent to Equation (2) with $\alpha = 1$, $\gamma = 1$, and all-one vector $\boldsymbol{v}$, although the Q-values in Equation (6) are scalar. We refer to this rule as single objective learning for multi-objective problems with summation scalarization in experimental evaluations.

On the other hand, Equation (2) with $\alpha = 1$, $\gamma = 1$, and all-one vector $\boldsymbol{v}$,

$$\boldsymbol{Q}(s,a) \leftarrow \boldsymbol{c} + \mathrm{argmin}_{\boldsymbol{Q}(s',a') \text{ for } a'} \boldsymbol{v} \cdot \boldsymbol{Q}(s',a') \ , \qquad (7)$$

is a basic scheme of multi-objective learning for a deterministic process, where the Q-vectors aggregate the cost vectors of the optimal policy. For different criteria, the minimization and the aggregation must be modified.

3.2.2 Applying Scalarization based on leximax

Although one can think that the minimization operator is simply replaced with other scalarization criteria, such an assumption is incorrect in general cases. The summation case is correct. Figure 2 shows an aggregation of two objectives with an incorrect minimax operation. In this example, actions a_1 and a_1' cause transitions from state s_1 to goal state s_g with cost vectors $[0,1]$ and $[1,0]$, respectively. Since the maximum value in these two cost vectors is one, they cannot be distinguished by the minimax. On the other hand, previous action a_0 causes a transition from state s_0 to s_1 with cost vector $[0,1]$. Therefore, $Q(s_0,a_0)$ takes $[0,2]$ for action a_1, while it takes $[1,1]$ for a_1'.

The minimum cost value for $Q(s_0,a_0)$ is different for subsequent actions even though the decision process is deterministic. This also means that there is no information to select the optimal policy. This is a problematic situation, while it might be mitigated in non-deterministic cases. Since leximax is an ex-

tension of maximum scalarization, the same problem exists.

To avoid that problem, we apply a minimization operation to the aggregated cost vectors. For the minimization of leximax, the update rule is represented as follows

$$\boldsymbol{Q}(s,a) \leftarrow \min_{a'}^{\text{leximax}}(\boldsymbol{c}+\boldsymbol{Q}(s',a')) \,, \qquad (8)$$

where $\boldsymbol{c}$ is a cost vector for current action a. The vectors are compared in the manner of leximax.

We must also modify the action selection after the learning process. In the original action selection, the action of the minimum Q-value is simply selected. However, for the modified case, such an action might be incompatible with the previous action, since it is selected without considering the aggregation process. To ensure compatibility among actions, we introduce the following condition

$$\boldsymbol{Q}(s^-,a^-) = \boldsymbol{c}^- + \boldsymbol{Q}(s,a) \,, \qquad (9)$$

where $\boldsymbol{Q}(s^-,a^-)$ is the Q-vector for the previous state and action and $\boldsymbol{Q}(s,a)$ is the Q-vector for the current state and action. $\boldsymbol{c}^-$ is the cost vector for previous action a^-. Namely, the actions in the current state are filtered by this condition, and the optimal action is selected from the filtered ones. In the initial state, the action selection is unfiltered, since there is no previous action. The procedure of action selection is summarized as follows.

1. For initial state s, select optimal action $a = \text{argmin}_{a'}^{\text{leximax}}\boldsymbol{Q}(s,a')$.
2. Perform action a and transit to the next state. Save $\boldsymbol{Q}(s,a)$ and cost $\boldsymbol{c}$ for a as $\boldsymbol{Q}(s^-,a^-)$ and $\boldsymbol{c}^-$.
3. Terminate when the goal state is achieved.
4. For current state s, select optimal action $a = \text{argmin}_{a'}^{\text{leximax}}(\boldsymbol{c}^- + \boldsymbol{Q}(s,a'))$ such that $\boldsymbol{Q}(s^-,a^-) = \boldsymbol{c}^- + \boldsymbol{Q}(s,a')$.
5. Return to 2.

With these modifications, the objectives are aggregated and compared without incorrect decomposition of the summation of vectors, although we need additional computations to maintain compatibility among actions.

3.3 Non-deterministic Domain

The update rules for the deterministic process are extended for the non-deterministic case. For the case of summation, the update rule is the case of Equation (2) when weight vector $\boldsymbol{v}$ is an all-one vector. The rule is equivalent to the following rule with summation scalarization

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(\text{sum}(\boldsymbol{c}) + \gamma \min_{a} Q(s',a'))) \qquad (10)$$

The update rule in Equation (8) is modified for the non-deterministic process.

$$\boldsymbol{Q}(s,a) \leftarrow (1-\alpha)\boldsymbol{Q}(s,a) + \alpha \min_{a'}^{\text{leximax}}(\boldsymbol{c}+\gamma\boldsymbol{Q}(s',a')) \qquad (11)$$

Learning rate α is applied to a cost vector that is minimized for aggregated vectors $\boldsymbol{c}+\gamma\,\boldsymbol{Q}(s',a')$. Discount rate γ can be identically generalized as the original reinforcement learning process. Here we fix the discount rate to one, since we focus on the fairness among individual total cost values for hunters and equally treat all action cost vectors.

We employ action selection similar to that for the deterministic process. However, this is not straightforward, since the condition in Equation (9), which maintains compatibility among actions, cannot be satisfied due to statistic aggregation. Therefore, the condition should be approximately evaluated considering the errors. Here we modify the condition using the leximax operator for the difference between $\boldsymbol{Q}(s^-,a^-)$ and $\boldsymbol{c}^- + \boldsymbol{Q}(s,a)$. Step 4 of the action selection in Section 3.2.2 is replaced as follows.

- For current state s, select action $a = \text{argmin}_{a'}^{\text{leximax}}((\boldsymbol{c}^- + \boldsymbol{Q}(s,a')) - \boldsymbol{Q}(s^-,a^-))$.
- Break ties with $a = \text{argmin}_{a'}^{\text{leximax}}(\boldsymbol{c}^- + \boldsymbol{Q}(s,a'))$.

In general cases, the first rule will usually be applied. The minimization of difference $(\boldsymbol{c}^- + \boldsymbol{Q}(s,a')) - \boldsymbol{Q}(s^-,a^-)$ reduces the cost values that exceed the estimated values in the previous action selection. Although such minimization might cause over-estimations when several cost values are reduced more than their necessary values, we prefer to take a margin for the non-deterministic process. However, such over-estimations might increase the total cost of some hunters due to a mismatch.

3.4 Correctness and Complexity

In the case of a deterministic process with an appropriate minimization criterion and tie-breaking among vectors, the learning process eventually converges to a quasi-optimal vector. The minimum cost vector propagates from the goal states to each state. We note that a non-monotonicity might be found in an update sequence of Q-vector $Q(s,a)$. However, it does not affect the above learning process, since it always overwrites the old Q-vectors. Therefore, each Q-vector converges to the minimum vector in terms of the minimum leximax filtering.

On the other hand, no assurance exists that the obtained policy is globally optimal in terms of the leximax for the individual total cost values for hunters.

Even though leximax aggregation is precisely decomposed with dynamic programming approaches in several cases (Matsui et al., 2018a; Matsui et al., 2018c; Matsui et al., 2018b), this is not the case. We assume that fair policies are more easily improved than unfair policies when additional actions are aggregated. This depends on the freeness of the problem that allows such a greedy approach. Although this approach is only a heuristic, it will reasonably work when there are a number of opportunities to improve the partial costs with previous actions in the manner of leximax.

For most non-deterministic cases, the learning process will not converge. However, after a sufficient number of updates with an appropriate learning rate, the Q-vectors will have some statistic information of the optimal policy similar to the case of a deterministic process.

Since leximax employs operations with sorted vectors, the computational overhead of the proposed approach is significantly large. While our major interest in this work is how the information of bottlenecks and fairness is mapped to Q-values, the overhead should be mitigated with several techniques, such as the caching of sorted vectors in practical implementations.

4 EVALUATION

We experimentally evaluated our proposed approach for deterministic and non-deterministic processes by employing the example domain of the pursuit problem.

4.1 Settings

As shown in Section 3.1, we employ a pursuit problem with four hunters and one target. The grid size of the torus world is 5×5 or 7×7 grids for the deterministic domain, and the size is 5×5 grids for the non-deterministic domain due to the limitations of the computation time of the learning process.

After the learning process, policy selection is performed and the individual total cost values for the hunters are evaluated. Here the total cost value corresponds to the total number of moves of each hunter. In this experiment, the locations of the four hunters are set to four corner cells (Figure 1). In the grid world, it is identical to that the hunters are gathered into an area. On the other hand, the initial locations of the target are set to all the cells except the initial locations of the hunters. We performed ten trials for each setting and averaged their results.

We compared the following three methods.

- sum: a single objective reinforcement learning method shown in Equation (6) and (10) that minimizes the total cost for all the hunters.
- lxm: a multi-objective reinforcement learning method that minimizes the individual total cost values for all the hunters with the minimum leximax filtering.
- sumlxm: a multi-objective reinforcement learning method that minimizes the total cost for all the hunters. However, the policy selection is the same as 'lxm'. Note that we employed update rules that resemble Equations (8) and (11) by replacing the filtering criterion to minimum summation, so that the Q-vectors are compatible with 'lxm'.

Here 'sumlxm' is employed to distinguish the effects of the Q-vectors for the minimum leximax approach from the action selection method.

4.2 Results

4.2.1 Deterministic Domain

We first evaluated the proposed approach for the deterministic decision process without any randomness for the moves of the hunters and the target. The learning process repeatedly updated the Q-vectors for all the joint state-action pairs, similar to the Bellman-Ford algorithm. In the deterministic case, after a few iterations of all the updates for all the Q-vectors, the vectors converged. Due to the infinity cost vector for the all-stop joint actions, there were no infinite cyclic policies.

Table 1 shows the total cost values for different methods in the 5×5 world. The size of the joint state-action space to be learned is $(5 \times 5 \times 2)^4 = 6,250,000$ for this problem. In a policy selection experiment, there are $5 \times 5 - 4 = 21$ initial locations for the target and 210 instances for ten trials. The results show that 'sum' minimizes the total cost for all the hunters which is equivalent to the average cost. On the other hand, 'lxm' minimizes the maximum total cost for the individual hunters, and also reduces the values of the Theil index, which is a criterion of unfairness. Since there are trade-offs between efficiency and fairness, the total (i.e. the average) cost for all hunters of 'lxm' is not less than that of 'sum'.

The result of 'sumlxm', which is a combination of the learning of 'sum' and the action selection of 'lxm', shows mismatches between the learning and the action selection. It also reveals that both the learning and the action selection of 'lxm' have some effects in its improvements.

In addition, the policy length of 'lxm' is relatively greater than that of 'sum'. A possible reason is that

Table 1: Total cost (deterministic, 5×5).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	0.09	1.79	3.50	1.67	5	6.47	8
lxm	1.05	1.94	2.62	0.30	6	7.30	9
sumlxm	0.03	1.79	3.62	1.94	5	6.52	8

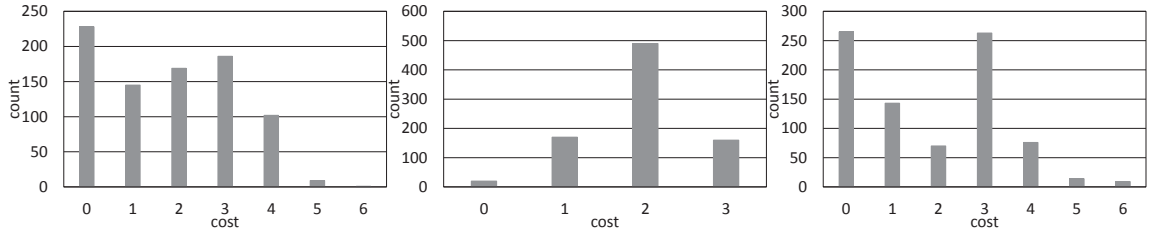

Figure 3: Histogram of total costs (deterministic, 5×5).

Table 2: Total cost (deterministic, 7×7).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	0.09	3	6.15	1.54	8	10.35	13
lxm	2.44	3.51	4.02	0.12	8	11.79	15
sumlxm	0.06	3	6.36	1.59	8	10.53	13

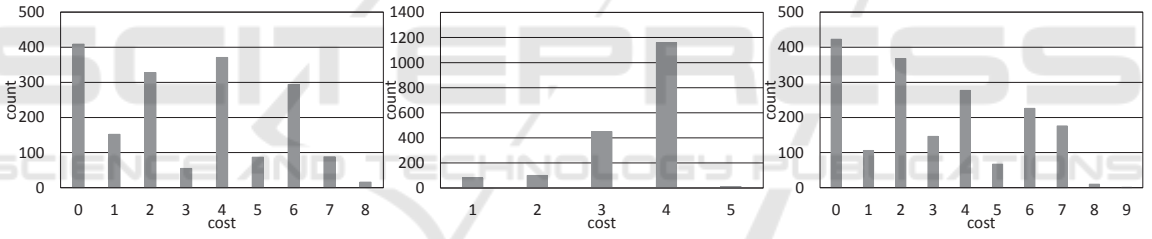

Figure 4: Histogram of total costs (deterministic, 7×7).

the aggressive moves of multiple hunters in 'lxm' might cause noise in target moves in several instances.

Figure 3 shows the histograms of the total cost values for all the hunters and all trials. In the case of 'sum', many hunters did not work at all and caught zero cost values, while several hunters caught relatively large cost values. On the other hand, 'lxm' reduced the number of idle and busy hunters. In the histogram of 'lxm', all the cost values are multiples of ten (i.e. the number of trials for each initial setting). Namely, the histogram is identical for the same initial setting, even though any ties are randomly broken in all action selection steps[1]. The 'lxm' completely maintains the cost values among hunters in the deterministic case, while 'sum' does not capture their individual total cost values.

Table 2 and Figure 4 show the results of the 7×7 world. The size of joint state-action space to be learned is $92,236,816$ for this problem. In the policy selection experiment, 450 instances were evaluated. The results resemble the case of a 5×5 world, while 'sum' shows a variety of cost values due to the larger size of the problems.

We also evaluated the cases where initial locations of the target are limited to a part of the grid as follows.

- inside: the range of vertical/horizontal coordinates of initial locations is $[1, g-2]$ where g is a grid size and the range of all coordinates is $[0, g-1]$.
- border: other locations except goal states.

[1]Note that a hunter moves close to the target with a deterministic tie-break rule when it selects a move. There might be two actions (i.e. a move and a stay) of an identical cost value for the same state in a resulting Q table. We randomly broke such ties with uniform distribution in action selection steps.

Table 3: Total cost (deterministic, 5×5, partial results).

range	method	individual total cost			Theil index	policy length		
		min.	ave.	max.		min.	ave.	max.
inside	sum	0.01	1.75	3.84	2.01	5	6.50	7
	lxm	1.11	1.89	2.44	0.18	6	7.41	8
	sumlxm	0.56	1.92	3	0.83	5	6.97	8
border	sum	0.13	1.8125	3.35	1.42	5	6.48	8
	lxm	1	1.98	2.75	0.39	6	7.13	9
	sumlxm	0.17	1.83	3.08	1.25	5	6.59	8

Table 4: Total cost (deterministic, 7×7, partial results).

range	method	individual total cost			Theil index	policy length		
		min.	ave.	max.		min.	ave.	max.
inside	sum	0.08	3	6.29	1.61	8	10.36	12
	lxm	2.20	3.48	4.04	0.17	8	11.54	15
	sumlxm	0.24	3.12	5.76	1.35	7	10.32	13
border	sum	0.10	3	6.00	1.45	8	10.40	13
	lxm	2.75	3.55	4	0.07	9	12.16	15
	sumlxm	0.30	3.14	5.30	1.16	8	10.59	12

Table 5: Total cost (deterministic, 5×5, comparison with Tchebycheff functions).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	0.09	1.79	3.50	1.67	5	6.47	8
max	1.95	2.39	2.90	0.13	6	8.05	11
lwt	0.71	1.88	2.67	0.59	5	7.16	9
lxm	1.05	1.94	2.62	0.30	6	7.30	9

Table 6: Total cost (deterministic, 7×7, comparison with Tchebycheff functions).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	0.09	3	6.15	1.54	8	10.35	13
max	3.71	4.08	4.27	0.02	8	11.81	17
lwt	2.33	3.46	4.02	0.14	8	11.48	15
lxm	2.44	3.51	4.02	0.12	8	11.79	15

Tables 3 and 4 show the results for both cases. The relationship among results resembles the cases for all initial locations.

In addition to the summation and leximax criteria, we compared the results with the Tchebycheff functions as follows.

- max: the Tchebycheff function.
- lwt: a variant of the augmented weighted Tchebycheff function. Here we logically prioritized the comparison of maximum cost values.

Tables 5 and 6 show the comparison with the Tchebycheff functions. Although the maximum cost of 'max' is less than that of 'sum', it is greater than the results of 'lwt' and 'lxm'. It reveals that the proposed approach is an approximate heuristic. Moreover, 'max' does not maintain the total (average) cost value and fairness. The low Theil index value of 'max' relates to the high total cost values. The results of 'lwt' relatively resemble the cases of 'lxm'. However, the unfairness (i.e. the Theil index) of 'lwt' is greater than 'lxm', since its tie-break is based on the summation criterion.

4.2.2 Non-deterministic Domain

Next we evaluated the non-deterministic process for a 5×5 world. We updated all the Q-vectors, similar to the case of the deterministic process. Since the computation is statistic and does not converge, the iterations of the whole updates for all the Q-vectors was terminated at the tenth iteration. We set learning rate α to $1, 0.5, 0.25, 0.125$ and fixed discount rate γ to one, as shown in Section 3.3.

Table 7: Total cost (non-deterministic, 5×5, $\alpha = 1$).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	2.54	5.94	10.43	0.85	5	22.99	110
lxm	4.38	6.72	9.04	0.35	5	23.94	102
sumlxm	4.81	7.22	9.65	0.25	5	24.86	86

Table 8: Total cost (non-deterministic, 5×5, $\alpha = 0.5$).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	1.48	3.67	6.10	0.81	5	12.59	55
lxm	3.19	4.57	5.88	0.16	5	12.40	40
sumlxm	3.58	5.1	6.80	0.21	4	14.12	60

Table 9: Total cost (non-deterministic, 5×5, $\alpha = 0.25$).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	0.95	3.09	5.73	1.13	5	10.39	33
lxm	2.91	4.20	5.45	0.16	4	10.26	33
sumlxm	3.31	4.76	6.37	0.22	4	12.72	49

Table 10: Total cost (non-deterministic, 5×5, $\alpha = 0.125$).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	0.98	3.10	5.58	1.05	5	10.19	23
lxm	2.77	3.89	4.93	0.15	5	9.19	32
sumlxm	3.03	4.58	6.15	0.23	4	12.6	45

Tables 7-10 show the total cost values for different methods and learning rates. The comparison among the methods resembles that of the deterministic case, although there is the influence of stochastic target moves. However, we found that matching the selected actions is difficult in this case. For example, in many aspects, the result of the different action selection method for 'lxm' with the Euclidean distance between $(\boldsymbol{c}^- + \boldsymbol{Q}(s,a'))$ and $\boldsymbol{Q}(s^-,a^-)$, or with the leximax for the vectors of the absolute values in difference $(\boldsymbol{c}^- + \boldsymbol{Q}(s,a')) - \boldsymbol{Q}(s^-,a^-)$, was often worse than 'sum'.

Figure 5 shows the histograms of the total cost values for all the hunters over all the instances in the case of $\alpha = 0.125$. Although the average maximum cost value of 'lxm' is less than 'sum' as shown in Table 10, the histogram shows that the maximum cost of 'lxm' for all the instances exceeds that of 'sum'. In addition, the cost values of 'lxm' are distributed in a wider and higher range than the deterministic case, revealing the difficulty of capturing the maximum cost value in inexact computations.

Table 11 shows the results for different initial locations of the target. It resembles the deterministic cases. Table 12 shows the comparison with the Tchebycheff functions. In this experiment we replaced all leximax operators by 'max'/'lwt' including the ones to estimate compatible actions in Section 3.3. While the results totally resemble the deterministic cases, the cost values of 'lwt' are relatively better than others. It is considered that the combination of maximization and summation relatively well worked in this case. However, 'lxm' still reduced the unfairness of cost values.

5 DISCUSSIONS

In this work, we employed leximax as a fundamental criterion for fairness. Although the leximin/leximax based approach requires more computational cost than the conventional operation, our primary interest is how different criteria affect joint actions. Efficient methods that reduce computational overheads must be addressed in practical domains. To reduce computational overheads, other criteria without sorted vectors might be employed, such as the augmented weighted Tchebycheff function, which is an extension of max-

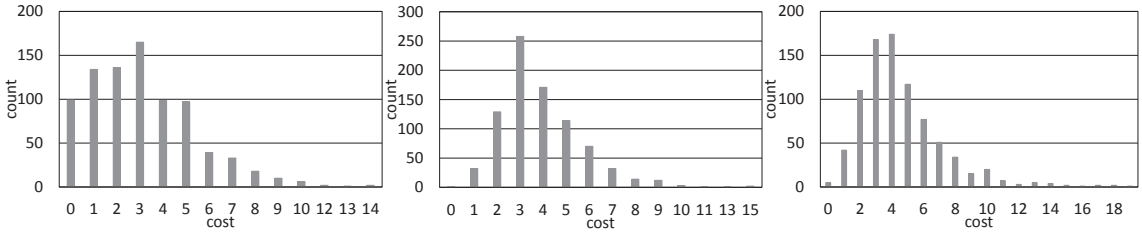

Figure 5: Histogram of total costs (non-deterministic, 5×5, $\alpha = 0.125$).

Table 11: Total cost (non-deterministic, 5×5, $\alpha = 0.125$, partial results).

range	method	individual total cost			Theil index	policy length		
		min.	ave.	max.		min.	ave.	max.
inside	sum	0.83	2.93	5.44	1.17	6	9.24	25
	lxm	2.77	3.91	4.94	0.14	6	9.37	22
	sumlxm	2.98	4.46	6.08	0.22	6	12.08	35
border	sum	0.91	2.99	5.41	1.11	5	10.08	35
	lxm	2.67	3.78	4.82	0.15	5	8.79	22
	sumlxm	2.88	4.53	6.26	0.25	4	12.33	33

Table 12: Total cost (non-deterministic, 5×5, $\alpha = 0.125$, comparison with Tchebycheff functions).

method	individual total cost			Theil index	policy length		
	min.	ave.	max.		min.	ave.	max.
sum	0.98	3.10	5.58	1.05	5	10.19	23
max	2.83	3.90	4.98	0.15	4	9.35	39
lwt	2.7	3.75	4.77	0.16	4	8.90	21
lxm	2.77	3.89	4.93	0.15	5	9.19	32

imum scalarization where the ties are broken by the summation. However, it only improves the worst-case and the total cost.

Since we focused on the costs for pairs of joint state and joint action that require a huge state-action space, we had to employ simplified actions even in small scale problems. To handle more types of actions, approximate decompositions of state-action spaces to multiple agents or partial observation are necessary. These modifications will decrease the accuracy of the proposed approach, and additional techniques are necessary.

We investigated how fairness is mapped into joint state-action space with multiple objectives. On the other hand, action shaping is a promising technique to reactively maintain the fairness. Indeed, our proposed method partially employs an action shaping approach, since it tries to match corresponding actions. In cases with more stochastic processes with noise, the proposed approach will be ineffective, and reactive approach will be the only solution. Although the rotation of roles among agents is a simple solution to distribute unfairness, we did not focus on this approach.

The proposed approach is heuristic and depends on some kind of freeness of the problem to greedily construct fair policies. In the cases where such a greedy approach does not meet, the solution quality of the proposed method will be decreased.

Exploration strategies for leximax might not be straightforward. In general cases of criteria based on minimax, the minimization operation emphasizes the minimum-maximum cost value. Therefore, the best first exploration based on lower bound vectors might fall into a kind of cyclic path that resembles negative cycles in path-finding problems.

We did not address multiple policy learning, since its space complexity is impractical for joint state-actions. On the other hand, if state-actions are approximately decomposed, multiple policies might be addressed.

The proposed approach employs a single joint state/action space and addresses bottlenecks and fairness among agents. A major class of related solution methods is the approach to converge equilibria (Hu and Wellman, 2003; Hu et al., 2015; Awheda and Schwartz, 2016). The comparison with equilibrium based methods with shared and individual learning tables will be a future work.

6 CONCLUSIONS

We addressed single-policy multi-objective reinforcement learning with leximax to improve bottlenecks and fairness among agents. We first investigated our proposed approach for the deterministic process and then extended it to the non-deterministic case. Our initial results with a pursuit-problem domain show that the learning and the action selection worked reasonably well. On the other hand, the noise in the decision process reduced its efficiency.

Our future work will include theoretical analysis, more detailed evaluations in various problem domains with some noise and partial observations, and the exploration strategies for on-line learning. The integration of reactive action shaping methods using the history of performed actions, and the investigation of approximate decompositions of joint state-action spaces to multiple agents will also be issues.

ACKNOWLEDGEMENTS

This work was supported in part by JSPS KAKENHI Grant Number JP16K00301 and Tatematsu Zaidan.

REFERENCES

Awheda, M. D. and Schwartz, H. M. (2016). Exponential moving average based multiagent reinforcement learning algorithms. *Artificial Intelligence Review*, 45(3):299–332.

Bouveret, S. and Lemaître, M. (2009). Computing leximin-optimal solutions in constraint networks. *Artificial Intelligence*, 173(2):343–364.

Greco, G. and Scarcello, F. (2013). Constraint satisfaction and fair multi-objective optimization problems: Foundations, complexity, and islands of tractability. In *Proc. 23rd International Joint Conference on Artificial Intelligence*, pages 545–551.

Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 4:1039–1069.

Hu, Y., Gao, Y., and An, B. (2015). Multiagent reinforcement learning with unshared value functions. *IEEE Transactions on Cybernetics*, 45(4):647–662.

Liu, C., Xu, X., and Hu, D. (2015). Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):385–398.

Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395.

Matsui, T., Matsuo, H., Silaghi, M., Hirayama, K., and Yokoo, M. (2018a). Leximin asymmetric multiple objective distributed constraint optimization problem. *Computational Intelligence*, 34(1):49–84.

Matsui, T., Silaghi, M., Hirayama, K., Yokoo, M., and Matsuo, H. (2018b). Study of route optimization considering bottlenecks and fairness among partial paths. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence, ICAART 2018, Volume 1, Funchal, Madeira, Portugal, January 16-18, 2018.*, pages 37–47.

Matsui, T., Silaghi, M., Okimoto, T., Hirayama, K., Yokoo, M., and Matsuo, H. (2018c). Leximin multiple objective dcops on factor graphs for preferences of agents. *Fundam. Inform.*, 158(1-3):63–91.

Moffaert, K. V., Drugan, M. M., and Nowé, A. (2013). Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 191–199.

Sen, A. K. (1997). *Choice, Welfare and Measurement*. Harvard University Press.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning : an introduction*. MIT Press.