Reinforcement Learning Considering Worst Case and Equality within Episodes

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

Keywords: Reinforcement Learning, Path-finding, Equality, Leximin.

Abstract: Reinforcement learning has been studied as an unsupervised learning framework. The goal of standard reinforcement learning methods is to minimize the total cost or reward for the optimal policy. In several practical situations, equalization of the cost or reward values within an episode may be required. This class of problems can be considered multi-objective, where each part of an episode has individual costs or rewards that should be separately considered. In a previous study this concept was applied to search algorithms for shortest path problems. We investigate how a similar criterion considering the worst-case and equality of the objectives can be applied to the Q-learning method. Our experimental results demonstrate the effect and influence of the optimization with the criterion.

1 INTRODUCTION

Reinforcement learning has been studied as an unsupervised learning framework (Sutton and Barto, 1998). The goal of standard reinforcement learning methods is to minimize the total cost or reward for the optimal policy. In several practical situations, equalization of the cost or reward values within an episode might be required. For example, in mobile robot navigation or adaptive routing in communication networks, the equalization of lifetime or peak load of individual facilities in a paths might be required. This class of problems can be considered multi-objective, where each part of an episode has individual costs or rewards that should be separately considered. Such episodes relate to situations where a sequence of an agent's behavior should be determined so that its risk or benefit is equalized in the sequence.

As a criterion for multiple objectives, we use leximin, which is defined as a dictionary order on objective vectors sorted in ascending order. The maximization problem with leximin improves the worst-case cost and equality among objectives. Previvous studies show that optimization with leximin can be decomposed in the manner of dynamic programming (Matsui et al., 2014; Matsui et al., 2015).

This concept has been applied to search algorithms based on dynamic programming for shortest path problems with similar criterion (Matsui et al., 2018). Since the problem resembles reinforcement learning for minimization problems that optimize sequences from a start state to a goal state, a similar technique can be applied to a part of the reinforcement learning.

We investigate how a similar criterion considering the worst-case and equality of the objectives can be applied to the Q-learning method. For this goal, we extend the criterion to meet the weighted average operation for different lengths of cost/reward vectors corresponding to episodes, and consider the requirements and limitations of the extended Q-learning. Our experimental results demonstrate the effect and influence of the optimization with our criterion.

The remainder of this paper is organized as follows. In the next section, we describe the preliminary aspects of our study including standard reinforcement learning, a criterion that considers multiple objectives and a shortest path problem with this criterion. Then, we propose a reinforcement learning method with a similar criterion in Section 3. The proposed approach is experimentally evaluated and discussed in Sections 4 and 5. Finally, we conclude our study in Section 6.

2 PRELIMINARY

Below, we describe several properties of reinforcement learning, the leximin criterion for multi-objective problems, and shortest path problems with

Matsui, T.
Reinforcement Learning Considering Worst Case and Equality within Episodes.
DOI: 10.5220/0009178603350342
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 1*, pages 335-342
ISBN: 978-989-758-395-7; ISSN: 2184-433X

a similar criterion quoting definitions and properties from (Matsui et al., 2018).

2.1 Q Learning

Q-learning is a fundamental reinforcement learning method that optimizes a policy to determine a sequence of states and actions in a state transition model. It 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, it causes a state transition with a corresponding reward/cost value based on an environment whose optimal policy should be mapped to Q-values. For the minimization problem, 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. With action selections based on an exploration strategy, Q-values are iteratively updated and propagated in the manner of asynchronous dynamic programming.

Online search algorithms, including the Learning Real-time A* algorithm (Barto et al., 1995), that perform asynchronous dynamic programming for shortest path problems are a base of reinforcement learning algorithms. The algorithms are also designed to perform exploration and exploitation to learn the optimal shortest path on a graph. Here, an agent repeats tours from a start vertex to a goal vertex of the graph with learning and exploration rules.

2.2 Criteria Considering Worst-case and Equality

Since we focus on a class of multi-objective optimization problems, we describe several related concepts below.

Definition 1 (Multi-objective optimization problem)**.** *A multi-objective optimization problem is defined with* $\langle X, D, F \rangle$*. X is a set of variables, D is a set of domains of variables, and F is a set of objective functions. Variable $x_i \in X$ takes value from finite and discrete set $D_i \in D$. For set of variables $X_i \subseteq X$, function $f_i \in F$ is defined as $f_i(x_{i,1}, \cdots, x_{i,k}) : D_{i,1} \times \cdots \times D_{i,k} \to \mathbb{N}$, where $x_{i,1}, \cdots, x_{i,k} \in X_i$. $f_i(x_{i,1}, \cdots, x_{i,k})$ is simply denoted by $f_i(X_i)$. The goal of the problem is to simultaneously optimize the objective functions under a criterion.*

A combination of the values of the objective functions is represented as an objective vector.

Definition 2 (Objective vector)**.** *Objective vector* $\mathbf{v}$ *is defined as* $[v_1, \cdots, v_K]$*. For assignment* $\mathcal{A}$ *to the variables in* X_j*,* v_j *is defined as* $v_j = f_j(\mathcal{A}_{\downarrow X_j})$*.*

Since the goal cannot be achieved because of trade-offs between the objectives in general cases, a Pareto optimal solution is selected based on several criteria (Sen, 1997; Marler and Arora, 2004).

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., 2014; Matsui et al., 2015).

Definition 3 (Sorted-objective vector)**.** *The values of sorted-objective vector* $\mathbf{v}$ *are sorted in ascending order.*

Definition 4 (Leximin)**.** *Let* $\mathbf{v} = [v_1, \cdots, v_K]$ *and* $\mathbf{v}' = [v'_1, \cdots, v'_K]$ *denote the sorted-objective vectors whose length is K. The order relation, denoted with* $\prec_{leximin}$*, is defined as follows:* $\mathbf{v} \prec_{leximin} \mathbf{v}'$ *if and only if* $\exists t, \forall t' < t, v_{t'} = v'_{t'} \wedge v_t < v'_t$*.*

Since leximin is a criterion that repeats the comparison between the minimum values in the vectors, it improves the worst-case values. This maximization also relatively improves the fairness and ensures Pareto optimality.

The addition of two sorted-objective vectors is defined with concatenation and resorting.

Definition 5 (Addition of sorted-objective vectors)**.** *Let* $\mathbf{v}$ *and* $\mathbf{v}'$ *denote vectors* $[v_1, \cdots, v_K]$ *and* $[v'_1, \cdots, v'_{K'}]$*, respectively. The addition of two vectors,* $\mathbf{v} \oplus \mathbf{v}'$*, is represented as* $\mathbf{v}'' = [v''_1, \cdots v''_{K+K'}]$*, where* $\mathbf{v}''$ *consists of all the values in* $\mathbf{v}$ *and* $\mathbf{v}'$*. In addition, the values in* $\mathbf{v}''$ *are sorted in ascending order.*

For the addition of sorted-objective vectors, the following invariance exists that enables dynamic programming to solve optimization problems with the leximin (Matsui et al., 2014).

Proposition 1 (Invariance of leximin on addition)**.** *Let* $\mathbf{v}$ *and* $\mathbf{v}'$ *denote sorted-objective vectors of the same length. In addition,* $\mathbf{v}''$ *denotes another sorted-objective vector. If* $\mathbf{v} \prec_{leximin} \mathbf{v}'$*, then* $\mathbf{v} \oplus \mathbf{v}'' \prec_{leximin} \mathbf{v}' \oplus \mathbf{v}''$*.*

Sorted objective vectors and related operations can be performed with a representation of a sorted histogram or a run-length encoding that is a vector of the sorted pairs of an objective value and the count of the value (Matsui et al., 2014; Matsui et al., 2015).

To evaluate the inequality among different-size populations, we employ the Theil index, a well-known measurement of inequality.

Definition 6 (Theil index)**.** *For n objectives, Theil index T is defined as*

$$T = \frac{1}{n}\sum_i \frac{v_i}{\bar{v}} \log \frac{v_i}{\bar{v}} \tag{2}$$

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

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

2.3 Application to Shortest Path Problems

The concept of leximin has been applied to shortest path problems (Matsui et al., 2018). The original problem is a minimization problem for the total cost values of edges in paths on a graph. In the previous work, the criterion was replaced by a criterion similar to leximin to improve the worst-case cost values and fairness among edges. Since shortest path problems are minimization problems, the maximization on leximin is replaced by the minimization of similar criteria called leximax with sorted-objective vectors whose value ordering is inverted.

Definition 7 (Descending sorted-objective vector)**.** *The values of a descending sorted-objective vector are sorted in descending order.*

Definition 8 (Leximax)**.** *Let $\mathbf{v} = [v_1, \cdots, v_K]$ and $\mathbf{v}' = [v'_1, \cdots, v'_K]$ denote descending objective vectors whose lengths are K. The order relation, denoted with $\prec_{leximax}$, is defined as follows. $\mathbf{v} \prec_{leximax} \mathbf{v}'$ if and only if $\exists t, \forall t' < t, v_{t'} = v'_{t'} \wedge v_t < v'_t$.*

The addition of two descending sorted-objective vectors is defined similarly to the leximin.

In route optimization problems, the paths and corresponding sorted-objective vectors with different lengths must be compared. For this comparison, leximax is extended to variable-length leximax, *vleximax*.

Definition 9 (Vleximax)**.** *Let $\mathbf{v} = [v_1, \cdots, v_K]$ and $\mathbf{v}' = [v'_1, \cdots, v'_{K'}]$ denote descending sorted-objective vectors whose lengths are K and K′, respectively. For $K = K'$, $\prec_{vleximax}$ is the same as $\prec_{leximax}$. In other cases, zero values are appended to one of the vectors so that both vectors have the same number of values. Then, the vectors are compared based on $\prec_{leximax}$.*

This comparison is intuitively based on two modified vectors that have the same sufficient length by padding blanks with zeros. Such padding values can be omitted in actual computation. Moreover, the vector can be represented with a histogram or a run-length encoding with sorted objective values when the cost values are a relatively small set of discrete values. Addition and comparison of the vectors are also performed on the histograms.

With vleximax, path-finding methods based on dynamic programming including the Dijkstra algorithm and the A* algorithm (Hart and Raphael, 1968; Hart and Raphael, 1972; Russell and Norvig, 2003) have been naturally extended. On the other hand, it has been shown that the extension of the Learning Real-time A* algorithm (Barto et al., 1995) is not straightforward, and a heuristic approach to mitigate the problem has been proposed (Matsui et al., 2018).

3 APPLYING LEXIMIN/LEXIMAX TO Q-LEARNING

We investigated the lexmin/leximax criterion with Q-learning. The Q-learning can be considered an extension of the Learning Real-time A* algorithm, the criterion is similarly applied. Below, we address several issues in the case of reinforcement learning.

Similar to the distance values of the shortest path problems with vleximax, we focus on the class of reinforcement learning problems where the cost or reward values take a few discrete values. We mainly address minimization problems where cost values are defined for each state. Moreover, we assume that there are a start state and a goal state that ensure an optimal episode from the start state to the goal state. Since one of our goals is the equalization of cost/reward values within episodes, an agent learns the policy based on complete observation.

3.1 Representation of Sorted-objective Vector

For shortest path problems with vleximax, different lengths of sorted-objective vectors are employed. In actual implementation, the sorted-objective vector can be represented as a histogram or run-length encoded data to reduce its size and computational cost. However, with Q-learning, there is the issue of weighted average among the different lengths of sorted-objective vectors. The original representation of the different lengths of vectors cannot be employed to average any two vectors. Therefore, only the histogram representation can be extended to aggregate two vectors with the weight values. As a result, the definition of the histograms must be modified so that they take real values. While such a representation is not a histogram, we call it a real-valued histogram.

Definition 10 (Real-valued histogram and operations)**.** *A real-valued histogram is an extended representation of a histogram of discrete values, where a count value for a discrete value to be counted can take a real value. Addition* $\mathbf{v}'' = \mathbf{v} \oplus \mathbf{v}'$ *of two real-valued histograms* $\mathbf{v}$ *and* $\mathbf{v}'$ *is defined as:* $v''_i = v_i + v'_i$ *for each* i^{th} *value of the real-valued histograms. Addition* $\mathbf{v}'' = i + \mathbf{v}'$ *of a discrete value* i *and a real-valued histogram* $\mathbf{v}'$ *that contains the count of* i *is defined as:* $v''_i = v'_i + 1$ *and* $v''_j = v'_j$ *for each* j^{th} *value, where* $i \neq j$*. Product* $\mathbf{v}'' = w\mathbf{v}'$ *of a real value* w *and a real-valued histogram* $\mathbf{v}'$ *is defined as:* $v''_i = w \cdot v'_i$ *for each* i^{th} *value.*

The comparison of vleximax with histogram representation can be defined for the count values similarly to the case of integer count values. With the above operations, the weighted average $\mathbf{v}'' = w\mathbf{v} \oplus (1-w)\mathbf{v}$ of two real-valued histograms is represented.

Proposition 2 (Weighted average of real-valued histograms)**.** *A weighted average of two real-valued histograms is an intermediate real-valued histogram of the two real-valued histograms in the ordering of vleximax.*

Proof. The weighed average of two real-valued histograms has weighted average values that are separately aggregated for count values corresponding to discrete values to be counted. Therefore, a count value of each discrete value to be counted is between the count values of the same discrete value in the two real-valued histograms. Since vleximax compares count values from the maximum discrete values, the aggregated real-valued histogram is between two real-valued histograms in the ordering of vleximax. □

The above weighted average is actually a statistical aggregation of populations regarding individual discrete cost values. Other types of aggregation of vectors can be available if such aggregation operations satisfy a similar property.

With the operators for real-valued histograms is the Q-learning is represented as follows:

$$\begin{aligned}\mathbf{Q}(s,a) \leftarrow & (1-\alpha)\mathbf{Q}(s,a) \oplus \\ & \alpha(c + \gamma \min^{vleximax}_{a'} \mathbf{Q}(s',a'))\ ,\end{aligned} \tag{3}$$

where Q-values are replaced by real-valued histograms $\mathbf{Q}(s,a)$. We call these Q-histograms.

For maximization problems, the rule can basically be replaced with rewards and a maximization operator on 'vleximin' that compares real-valued histograms whose discrete values are sorted in ascending order. However, an applicable class of maximization problems can differ from those of minimization problems similar to the cases of the original Q-learning.

3.2 Parameters for Learning and Exploration/Exploitation

Although the settings of parameters for learning and exploration/exploitation resemble the original Q-learning with summation, part of settings affects the case of real-valued histograms and vleximax. The learning rate α can be set to an arbitrary value for deterministic state transition cases, while the values less than one might cause a significant delay of the convergence for real-valued histograms. The settings are related to classes of problems as shown in the following sections.

In the case of Q-learning for minimizing problems that are similar to shortest path problems, the discount rate γ should be set to one because the Q-values/histograms are aggregated to capture the costs for a whole episode. Therefore, the limitation of future information will decrease the accuracy of policies in both cases with the original summation values and the real-valued histograms.

Appropriate exploitation and exploration strategies and parameters such as the epsilon-greedy method can be used.

3.3 Acyclic State Transition Cases

Here, we consider simple cases where the state transition is acyclic and deterministic. A state transition space of this class of problems can be represented with a directed acyclic graph whose vertexes and edges represent states and actions. In addition, we address the case where there are a start sate and a goal state such that there are paths between two states.

Since an episode consists of a sequence of different state-action pairs, the propagation of Q-learning from the goal state to the start state is eventually achieved. Therefore, the Q-learning is correctly works with vleximax.

Like vleximax, both minimization and maximization problems can be learned with Q-learning methods with summation and vleximax/vleximin criteria. In this case, the initial Q-values/histograms can be set with arbitrary values including zero, while it might affect the convergence of the Q-values/histograms.

3.4 Cyclic State Transition Cases

Next, we consider the cases where state transition can be cyclic. We also address the case where there are a start sate and a goal state such that there are paths between two states. Although the original Q-learning with summation is applicable without modifications,

the case of Q-histograms and vleximax requires a few considerations.

With the Learning Real-time A* algorithm for shortest path problems, minimization with vleximax is problematic because it is driven with lower-bound cost values and emphasizes cyclic paths that resemble negative cycles (Matsui et al., 2018). This contrasts with the minimization with the summation that correctly works with both upper and lower bound cost values. Q-learning has the similar issue.

A simple way to perform the learning based on upper bound cost values is to set the initial values of the Q-histograms at a sufficiently large upper bound value similar to the Dijkstra method. Namely, the count values of the Q-histograms are set to a sufficiently large value considering the convergence time of the Q-histograms. Since this is not a complete way, the correct propagation of learning from the goal state must be assured by the setting of parameters and explorations.

With algorithms such as the Learning Real-time A* algorithm, both of learning and exploration rules are driven based on the lower-bound cost values. Therefore, the learning has been modified to episode-based learning with a heuristic exploration strategy. On the other hand, the Q-learning is basically performed with random-walk-based exploration and there might be opportunities to find the goal state. The random walk based algorithm can be available for relatively small problems, while the length of explored episodes might be significantly longer.

In the case of cyclic state transition, the definition of the maximization problem is not straightforward for the real-valued histogram and vleximax. Although minimization problems with summation similar to path-finding problems can be relatively easily replaced by maximization problems with potential fields where only the goal states have rewards, this differs from our aim of equalizing cost/reward values in an episode.

3.5 Stochastic Cases

When the state transition is stochastic, the solution quality of an episode generally decreases. Since minimization with vleximax aims the equalization of future cost/reward values, the noise is serious. In this case, the weighted average of Q-histograms might confuse the learning, since it separately aggregate the count of each discrete cost value.

Therefore, in relatively noisy environments, the aggregated Q-histograms cannot adequately emphasize the probabilistically best episode. In such cases the minimization with vleximax tends toward nonsense and the original Q-learning with summation reduces both the maximum and summation of cost/reward values. While the limitation is substantial for our problem settings, the Q-learning still work if the result of statistic aggregation can hold the episodes in environments with little noise.

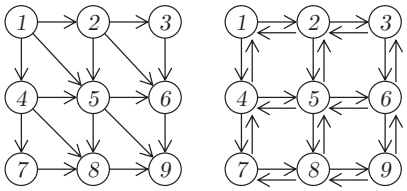

Figure 1: Problems (3×3).

3.6 Properties of Solution Method

Since the proposed approach inherits a major part form the shortest path finding with vleximax shown in (Matsui et al., 2018), it can be optimal as vleximax in the case of deterministic and acyclic state transition. The correctness depends on an exploration and the learning ratio that ensure the propagation of learning from the goal state to the start state. Similar property holds in the case case of minimization problems with deterministic and cyclic state transition.

The weighted average of the real-valued histogram is reasonable in the meaning that it is an intermediate real-valued histogram of the original ones. While it converges to (near) optimal histogram in deterministic transition cases with appropriate exploration and learning, in stochastic transition cases, a weighted average of individual count values might be inaccurate.

The computational and memory cost for the addition, weighted average and vleximax of real-valued histograms is greater than those of scalar cost values. However, it linearly grows with the number of discrete cost values to be counted. Therefore, it is applicable to histograms with relatively small range of discrete cost values.

4 EVALUATION

4.1 Problem Settings

We experimentally evaluated the proposed method with fundamental benchmark problems based on the $w \times h$ lattice directed graphs shown in Figure 1. Vertexes and edges of the graphs represent states and transitions corresponding to actions.

The following two types of state transition models are employed.

- DAG: State transitions are acyclic. To increase different episodes, diagonal-directed edges are added.
- CYC: State transitions are cyclic.

As the initial value of Q-values/histograms, we employed 100 for both DAG and CYC. In addition, two cases of transitions are considered:

- DET: State transitions are deterministic. Therefore, an action and a state transition exactly correspond to a directed edge.
- NDT: State transitions are non-deterministic. Each action a is successfully performed with probability p_a. In this case, the state transition is the same as DET. Otherwise, a state transition corresponding to one of the other actions is randomly performed with uniform distribution. We randomly set p_a of action a from $[p^{\perp}, 1]$ with uniform distribution.

We set the start state and goal state to the left-top and right-bottom states. Each state except the start state gives a cost value. The cost value of each edge was randomly set from the integer values in $[1, c^{\top}]$ based on uniform distribution, where $c^{\top}$ was 5 or 10.

We performed minimization methods with the following two criteria.

- SUM: Minimization of summation for scalar cost values.
- LXM: Minimization with vleximax for real-valued histograms of cost values.

We varied learning rate α and discount rate γ. To eliminate the influence of heuristic exploration, random walk was performed for learning. As the exploitation of learned Q tables, episodes that perform a sequence consisting of each best action in each state were evaluated. In the case of NDT, 100 episodes were averaged for evaluation.

Each episode was performed for at most 2000 steps. In each trial, at most 1,000,000 episodes were performed. For each setting of problems results over ten instances were averaged. The experiment was performed on a computer with a Core i7-3930K CPU (3.20 GHz), 16-GB memory, Linux 2.6.32, and g++ (GCC) 4.4.7.

4.2 Results

Table 1 shows the results of DAG, DET, 10×10 grid, $c^{\top} = 10$, and $\gamma = 1$. Here, the statistic length of objective vector that is calculated from a real-valued histogram (len.), the total cost (sum.), the maximum cost (max.), the Theil index (theil) are evaluated. On average, SUM decreased the total cost value, while LXM decreased the maximum cost and the Theil index. Due to a trade off, the length and the summation in LXM are grater than SUM. While different learning rates affected the convergence, the results resemble in this problem setting.

Table 1: DAG, DET, 10×10, $c^{\top} = 10$, $\gamma = 1$.

alg.	α	len.	sum.	max.	theil
SUM	0.25	10.5	37.3	8.1	0.191
	0.5	10.8	37.3	8	0.189
	1	10.8	37.3	8	0.189
LXM	0.25	11.4	44.7	7.4	0.128
	0.5	11.5	41.3	7.3	0.142
	1	11.6	42.2	7.3	0.135

Table 2: DAG, DET, 10×10, $c^{\top} = 10$, $\gamma = 0.5$.

alg.	α	len.	sum.	max.	theil
SUM	0.25	13.6	44.8	8	0.204
	0.5	13.6	44.8	8	0.204
	1	13.6	44.8	8	0.204
LXM	0.25	15.1	58.4	7.4	0.127
	0.5	15.9	60.4	7.3	0.121
	1	16	60.1	7.3	0.120

Table 3: DAG, DET, 20×20, $c^{\top} = 10$, $\gamma = 1$.

alg.	α	len.	sum.	max.	theil
SUM	0.25	23.6	73.8	8.5	0.216
	0.5	23.8	73.8	8.6	0.212
	1	23.8	73.8	8.6	0.212
LXM	0.25	22.3	100.1	8.2	0.151
	0.5	24.8	82.8	7.3	0.151
	1	25.7	84.3	7.1	0.149

Table 4: DAG, DET, 10×10, $c^{\top} = 5$, $\gamma = 1$.

alg.	α	len.	sum.	max.	theil
SUM	0.25	10.4	21.1	4.3	0.143
	0.5	10.7	21.1	4.2	0.137
	1	10.7	21.1	4.2	0.137
LXM	0.25	11.2	22.4	4	0.117
	0.5	10.8	22.7	4	0.112
	1	11.2	22.4	4	0.117

Table 2 shows the results of DAG, DET, 10×10 grid, $c^{\top} = 10$, and $\gamma = 0.5$. In this case, the accuracy of both SUM and LXM decreased as compared to that of $\gamma = 1$, since the problem resembles shortest path problems as mentioned in Section 3.2. Table 3 shows the results of DAG, DET, 20×20 grid, $c^{\top} = 10$, and $\gamma = 1$. The results resemble the case of 10×10 problems, while the cost values and the length of policies

Table 5: CYC, DET, 5×5, $c^{\top} = 10$, $\gamma = 1$.

alg.	α	len.	sum.	max.	theil
SUM	0.5	8	29	8.2	0.227
	1	8	29	8.2	0.227
LXM	0.5	8	30.1	7.8	0.198
	1	8	30.1	7.8	0.198

Table 6: DAG, NDT, 10×10, $c^{\top} = 10$, $\gamma = 1$.

$p^{\perp}$	alg.	α	len.	sum.	max.	theil
0.7	SUM	0.25	11.7	44.8	8.62	0.202
		0.5	11.6	45.8	8.52	0.189
		1	12.6	51.3	8.69	0.189
	LXM	0.25	13.9	67.3	8.81	0.146
		0.5	13.9	67.8	8.92	0.155
		1	13.5	57.6	8.70	0.149
0.9	SUM	0.25	11.0	40.0	8.28	0.187
		0.5	10.8	39.9	8.10	0.195
		1	11.7	44.2	8.25	0.181
	LXM	0.25	14.0	68.4	8.75	0.157
		0.5	14.3	71.0	8.82	0.143
		1	12.6	48.0	7.99	0.146

Table 7: CYC, NDT, $p^{\perp} = 0.95$, 5×5, $c^{\top} = 10$, $\gamma = 1$.

alg.	α	len.	sum.	max.	theil
SUM	0.5	8.2	30.2	8.142	0.215
	1	31.8	166.1	8.137	0.230
LXM					
	1	8.5	32.5	7.846	0.200

are relatively large for both methods. As shown in Table 4, where the range of cost values is $[1,5]$, the narrow range of cost values decreases the opportunities of differences between SUM and LXM.

Table 5 shows the results of CYC, DET, 5×5 grid, $c^{\top} = 10$, and $\gamma = 1$. Here, we had to set the initial value of Q-histograms to an upper bound value of 100, since the initial value of zero results in incorrect cyclic episodes in the case of LXM, as mentioned in Section 3.4. Indeed, with zero initial values, resulting episodes emphasized incorrect cyclic paths. With the appropriate upper-bound values, the results resemble for the cases of DAG. In the cyclic case, the convergence of LXM takes a number of episodes with the learning rates less than one, while it converged in relatively smaller number of episodes with the learning rate of one even in 20×20 problems.

Table 6 shows the results of DAG, NDT, 10×10 grid, $c^{\top} = 10$, and $\gamma = 1$. In the cases of nondeterministic state transitions, the accuracy of vleximax significantly decreased. As shown in the case of $p^{\perp} = 0.7$, with high randomness of state transition, LXM cannot overcome SUM in any metrics except the Theil index. Since this is a substantial issue of problem definition, use of vleximax should be determined considering the opportunities of the real-valued histograms capturing information within each episode. We found that LXM with the learning rate of one is better in these cases as shown in the case of $p^{\perp} = 0.9$. This reveals the difficulty in the statistic aggregation of objective vectors. Table 7 shows the cases of CYC, NDT, $p^{\perp} = 0.95$, 5×5, $c^{\top} = 10$, and $\gamma = 1$. Although the convergence of LXM was severe in most problem instances, the converged results resemble the cases of DAG and NDT.

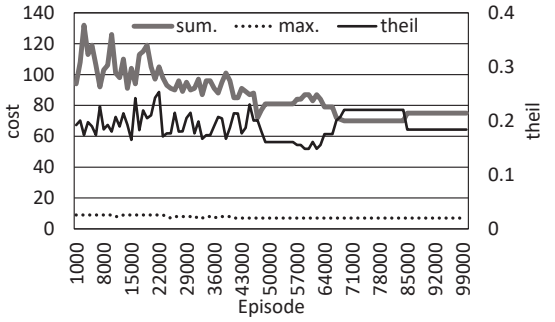

Figure 2: DAG, DET, 20×20, $c^{\top} = 10$, $\alpha = 0.5$, $\gamma = 1$, LXM.

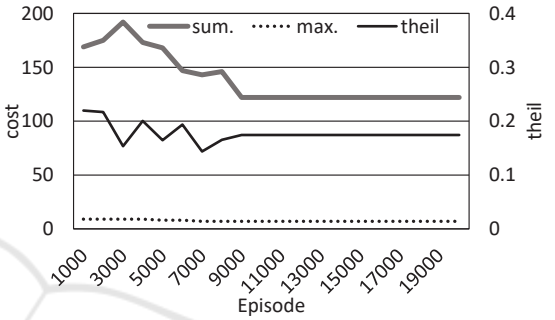

Figure 3: CYC, DET, 20×20, $c^{\top} = 10$, $\alpha = 0.5$, $\gamma = 1$, LXM.

Figures 2 and 3 show examples of anytime curves of learning for LXM in DET. The optimal policies are iteratively evaluated based on learned Q tables. The accuracy gradually improved with the progress of learning.

The averaged computational times including statistic operations for evaluation were 0.936 and 1.645 seconds for SUM and LXM in a DAG, 20×20 grid setting and 100,000 episodes. For the CYC, 20×20 grid, it took 39.782 and 74.359 seconds. The above time of LXM for CYC contains the anytime evaluation in the worst-case of cyclic episodes up to 2000 steps.

5 DISCUSSION

Since the proposed approach resembles an extension of shortest path problems, it can be applied to the class of similar problems in the context of reinforcement learning including the case of positive rewards on acyclic state transition. In the other cases based on a potential field, more consideration is necessary to model such problems.

A shown in (Matsui, 2019), applying the idea of leximin/leximax to optimization of joint policies among multiple agents so that the equality of individual agents is improved is difficult because the aggregation of sorted-objective vectors cannot be well decomposed in a direction of episodes. On the other hand, the class of problems in this study is relatively better, since the decomposition of a real-valued histogram is based on dynamic programming. However, more investigations are necessary to discuss whether this approach can appropriately fit some class of multiagent or multi-objective problems.

Vleximax is based on the assumption that prediction of future optimal episodes is possible. Therefore, it cannot be employed in highly stochastic problems. On the other hand, where statistically optimal policy can be calculated with real-valued histograms, there may be opportunities to improve policies with vleximax.

We employed a simple definition of the weighted average of real-valued histograms with scalar weight values. It may be possibile to use weight vectors to emphasize part of the discrete cost values. In addition, other aggregation operators that differently generate a histogram from the original two histograms could affect solution quality.

6 CONCLUSIONS

We investigated applying a leximin based criterion to the Q-learning method to consider the worst-case and equality among individual cost/reward values in an episode. The experimental results show that the criterion is effective in several cases of problems, while more considerations are necessary to employ it to other problem classes.

Our future work will include, improvement of the proposed method, and investigations applying the criterion to part of the practical multi-objective and multiagent problems.

ACKNOWLEDGEMENTS

This work was supported in part by JSPS KAKENHI Grant Number JP19K12117 and Tatematsu Zaidan.

REFERENCES

Barto, A. G., Bradtke, S. J., and Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138.

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.

Hart, P., N. N. and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Science and Cybernetics*, 4(2):100–107.

Hart, P., N. N. and Raphael, B. (1972). Correction to 'a formal basis for the heuristic determination of minimum-cost paths'. *SIGART Newsletter*, (37):28–29.

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. (2019). A study of joint policies considering bottlenecks and fairness. In *11th International Conference on Agents and Artificial Intelligence*, volume 1, pages 80–90.

Matsui, T., Silaghi, M., Hirayama, K., Yokoo, M., and Matsuo, H. (2014). Leximin multiple objective optimization for preferences of agents. In *17th International Conference on Principles and Practice of Multi-Agent Systems*, pages 423–438.

Matsui, T., Silaghi, M., Hirayama, K., Yokoo, M., and Matsuo, H. (2018). Study of route optimization considering bottlenecks and fairness among partial paths. In *10th International Conference on Agents and Artificial Intelligence*, pages 37–47.

Matsui, T., Silaghi, M., Okimoto, T., Hirayama, K., Yokoo, M., and Matsuo, H. (2015). Leximin asymmetric multiple objective DCOP on factor graph. In *18th International Conference on Principles and Practice of Multi-Agent Systems*, pages 134–151.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall.

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.