CMIX: Deep Multi-agent Reinforcement Learning with Peak and Average Constraints

Chenyi Liu(✉)[1,4,5], Nan Geng[1,4,5], Vaneet Aggarwal[2], Tian Lan[3], Yuan Yang[1,4,5], and Mingwei Xu[1,4,5]

[1] Department of Computer Science and Technology, Tsinghua University, China
[2] School of Industrial Engineering and School of Electrical and Computer Engineering, Purdue University, USA
[3] School of Engineering and Applied Science, George Washington University, USA
[4] Beijing National Research Center for Information Science and Technology, China
[5] Peng Cheng Laboratory (PCL), China
{liucheny19,gn16}@mails.tsinghua.edu.cn, tlan@gwu.edu, vaneet@purdue.edu,
yangyuan_thu@mail.tsinghua.edu.cn, xumw@tsinghua.edu.cn

Abstract. In many real-world tasks, a team of learning agents must ensure that their optimized policies collectively satisfy required peak and average constraints, while acting in a decentralized manner. In this paper, we consider the problem of multi-agent reinforcement learning for a constrained, partially observable Markov decision process – where the agents need to maximize a global reward function subject to both peak and average constraints. We propose a novel algorithm, CMIX, to enable centralized training and decentralized execution (CTDE) under those constraints. In particular, CMIX amends the reward function to take peak constraint violations into account and then transforms the resulting problem under average constraints to a max-min optimization problem. We leverage the value function factorization method to develop a CTDE algorithm for solving the max-min optimization problem, and two gap loss functions are proposed to eliminate the bias of learned solutions. We evaluate our CMIX algorithm on a blocker game with travel cost and a large-scale vehicular network routing problem. The results show that CMIX outperforms existing algorithms including IQL, VDN, and QMIX, in that it optimizes the global reward objective while satisfying both peak and average constraints. To the best of our knowledge, this is the first proposal of a CTDE learning algorithm subject to both peak and average constraints.

Keywords: Multi-agent reinforcement learning · peak constraint · average constraint

1 Introduction

Multi-agent reinforcement learning (MARL) has shown great promise in many cooperative tasks such as intelligent transportation system [2,16], network optimization [28], and robot swarms [13]. Existing MARL algorithms – e.g., IQL [27],

VDN [26], and QMIX [24] – often leverage centralized training with decentralized execution (CTDE). However, in many real-world decision making problems, a team of distributed learning agents must also ensure that their optimized policies collectively satisfy required instantaneous peak constraints [3] and long-term average constraints [10]. For instance, optimizing a vehicular network must meet the requirements of both peak network performance (e.g., the maximum latency constraint [25]) and average network performance (the average transmission rate [15] or average latency [1] constraint). We note that since the exact action space satisfying these constraints cannot be determined prior to training, new CTDE algorithms are needed to simultaneously address both peak and average constraints.

In this paper, we propose a new MARL algorithm with CTDE, called CMIX, for maximizing the discounted cumulative reward subject to both peak and average constraints. We note that while the problem of constrained reinforcement learning (RL) has been studied under either peak constraints [3,9,11,12] or average constraints [4,6,10,22], most of the existing algorithms tackle only one type of the two constraints – but not both – and focus on single-agent tasks with centralized execution. To the best of our knowledge, there are no MARL learning algorithms that can address both peak and average constraints especially in the realm of CTDE algorithms for multiple agents. In contrast, CMIX is able to optimize distributed agents' policies under both peak and average constraints, while leveraging value function factorization to allow distributed executions.

More precisely, CMIX casts a Markov decision process (MDP) under both peak and average constraints into an equivalent max-min optimization problem. First, by setting a lower bound on the MDP's objective value, we can replace the objective with a new average constraint. Second, we introduce a penalty into the reward functions of the average constraints so that the peak constraints can be absorbed into the set of average constraints. These allow us to obtain a multi-objective constrained problem with only average constraints, whose solution can be solved through an equivalent max-min optimization problem. Finally, a search algorithm is designed to find an appropriate lower bound, such that an optimal solution of the original constrained problem can be obtained by solving the max-min optimization problem.

We further leverage value function factorization and propose a CTDE algorithm to solve the max-min optimization problem, thus providing a solution to the MDP under both peak and average constraints. To this end, individual agent networks are used to estimate the per-agent action-values with respect to each average constraint and conditioned on only local observations, while a neural mixing module consisting of multiple mixing networks combines the outputs of the agent networks to produce an estimate of the joint action-values with respect to the average constraints. Two structures of the mixing module are developed, i.e., a module with mixing networks having independent parameters (CMIX-M) and a module with all mixing networks sharing the same parameters (CMIX-S). The neural parameters can be updated through an end-to-end method similar to QMIX [24]. However, directly applying the TD-error loss function

is likely resulting in a biased solution due to the fact that CMIX solves the original max-min optimization problem approximately in a CTDE paradigm. To address the issue, we propose two gap loss functions for CMIX-M and CMIX-S, respectively, to minimize the gap between the original max-min optimization problem and the approximated one solved by CMIX. The neural parameters are updated by minimizing a linear combination of TD-error loss and gap loss. To the best of our knowledge, this is the first MARL algorithm that enables CTDE under both peak and average constraints.

We conduct extensive evaluations on a blocker game with travel cost and on a large-scale vehicular network routing problem. Evaluation results compares CMIX with state-of-the-art CTDE learning algorithms – including IQL [27], VDN [26], and QMIX [24] – as well as algorithms for constrained MDP such as C-IQL, which is implemented by extending IQL in a similar way to CMIX. The results show that CMIX outperforms state-of-the-art CTDE learning algorithms in the sense that it can optimize the global objective while satisfying both peak and average constraints. Further evaluations validate that gap loss improves the objective value of CMIX while enabling CMIX to comply with both peak and average constraints.

2 Background

A cooperative multi-agent sequential decision-making task in a stochastic, partially observable environment can be modeled as a *decentralized partially observable Markov decision process (Dec-POMDP)* [21], denoted by a tuple $G = \langle \mathcal{S}, \mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, P, r, \{\mathcal{Z}_i\}_{i\in\mathcal{N}}, \gamma \rangle$. $s \in \mathcal{S}$ denotes the global state of the environment. At each time step t, each agent $i \in \mathcal{N} \equiv \{1, \ldots, N\}$ gets a local observation $z_i \in \mathcal{Z}_i$ and chooses a local action $a_i \in \mathcal{A}_i$, forming a joint action $\boldsymbol{a} \in \mathcal{A} \equiv \Pi_{i=1}^N \mathcal{A}_i$. Then the environment evolves transforms from the current state to a new state according to the state transition function $P(s'|s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ and returns a global reward $r(s, \boldsymbol{a})$ to the agents. Given a joint policy $\boldsymbol{\pi} := (\pi_i)_{i\in\mathcal{N}}$, the joint action-value function at time step t is defined as $Q^{\boldsymbol{\pi}}(s^t, \boldsymbol{a}^t) := \mathbb{E}_{\boldsymbol{\pi}} [R^t|s^t, \boldsymbol{a}^t]$, where $R^t = \sum_{\tau=0}^{\infty} \gamma^\tau r^{t+\tau}$ is the discounted cumulative reward. The goal is to find an optimal policy $\boldsymbol{\pi}^*$ which results in the optimal value function $Q^* = \max_{\boldsymbol{\pi}} Q^{\boldsymbol{\pi}}(s^t, \boldsymbol{a}^t)$.

2.1 QMIX

CTDE paradigm [23] is promising in solving the optimization problems of Dec-POMDP. During training, the learning algorithm of CTDE trains agents centrally and gets access to the global state s. During execution, each agent i can only make decisions according to its local observation z_i. There have recently been many MARL algorithms proposed in the CTDE paradigm, among which value-decomposition methods attract much attention recently.

QMIX [24,23] is one of the representative MARL algorithms belonging to value-decomposition methods, which factorizes joint action-value function Q_{tot}

into a combination of local action-value functions. QMIX combines the per-agent action-value function via a state-dependent, differentiable monotonic function: $Q_{tot}(s, \boldsymbol{a}) = f(Q_1(z_i, a_1), \ldots, Q_N(z_N, a_N); s)$, where $\frac{\partial f}{\partial Q_i} \geq 0, \forall i \in \mathcal{N}$. Since the mixer $f(\cdot)$ is a monotonic function to per-agent action-value input, we can maximize the joint action value by maximizing per-agent action values locally.

QMIX is trained much like deep Q-network (DQN) [19]. A buffer replay mechanism and a TD-error loss function are taken for training agents. Particularly, for a mini-batch of B samples $(s_b, \boldsymbol{a}_b, r_b, s_b')$ $(b = 1, \ldots, B)$, QMIX learns parameters θ by minimizing $\mathcal{L}_{TD-error} = \frac{1}{B} \sum_{b=1}^{B} (Q_{tot}(s_b, \boldsymbol{a}_b; \theta) - y_b)^2$, where $y_b = r_b + \gamma \max_{\boldsymbol{a}'} Q_{tot}(s_b', \boldsymbol{a}'; \theta^-)$ is the target value of the b-th sample and θ^- denotes the parameters of a target network which are periodically copied from θ. The monotonic mixing function f is parameterized as a feed-forward network, whose non-negative weights are generated by hypernetworks [14] with the global state as input.

2.2   Constrained Reinforcement Learning

Constrained reinforcement learning tries to find a policy $\boldsymbol{\pi}^*$ to maximize the global discounted cumulative reward subject to some constraints such as *peak* constraints [3,9,11,12] and *average* constraints [4,6,10,22]. Formally, the objective function of these constrained problems is $\max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s^t, \mathbf{a}^t) \right]$. Without loss of generality, we assume that there exist J peak constraints and K average constraints. Peak constraints limit instantaneous returns, which can be formulated as $c_j(s^t, \boldsymbol{a}^t) \geq 0, \quad \forall t, j = 1, \ldots, J$. While average constraints focus on long-term limitations, which can be stated as $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_k(s^t, \boldsymbol{a}^t) \right] \geq 0, k = 1, \ldots, K$. $r, \{c_j\}_{j=1}^{J}$, and $\{r_k\}_{k=1}^{K}$ are unknown functions, but their returned values can be acquired at each time step.

Existing researchers usually develop constrained RL algorithms for dealing with the two kinds of constraints separately. For the problems with only peak constraints, many approaches [3,9] choose to introduce a penalty into the original reward function, which has been validated to be effective.

There are also some researches focusing on the problems with only average constraints. [10] converts the optimization problem with average constraints to the multi-objective constrained problem by adding a lower bound δ to the objective function. The multi-objective constrained problem aims to find a feasible policy while satisfying the original average constraints as well as the new average constraint on the objective function. Such a constraint satisfaction problem can be solved through a max-min optimization problem which maximizes the minimum margin of average constraints.[6] Therefore, with an appropriate δ-search algorithm, we can approximate the optimal solution that maximizes the discounted cumulative reward under the original average constraints.

However, most of the constrained RL algorithms are for single-agent settings, and the algorithms considering both peak and average constraints are missing especially in the paradigm of MARL.

---

[6] The margin of an average constraint represents the left hand of the average constraint.

3   Problem Formulation

In this paper, we consider the problem of the Dec-POMDP under both *peak* and *average* constraints. The problem can be formulated as

$$\max_{\boldsymbol{\pi}} \quad \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r(s^t, \boldsymbol{a}^t) \right] \tag{1}$$

$$\text{s.t.} \quad c_j(s^t, \boldsymbol{a}^t) \geq 0, \quad \forall t, j = 1, \ldots, J \tag{2}$$

$$\mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r_k(s^t, \boldsymbol{a}^t) \right] \geq 0, \quad k = 1, \ldots, K \tag{3}$$

Next, we need to find an optimal joint policy $\boldsymbol{\pi}^*$ which optimizes the discounted cumulative reward in Eq.(1) and satisfies both peak constraints on instantaneous returns in Eq.(2) and average constraints on long-term returns in Eq.(3).

4   CMIX

We propose CMIX to solve the Dec-POMDP problem under both peak and average constraints. First, we convert the original MDP to a multi-objective constrained problem, whose solution can be obtained by solving an equivalent max-min optimization problem. Second, we leverage value function factorization to develop a CTDE algorithm for solving the max-min optimization problem approximately. We further analyze the gap between the original max-min optimization problem and the approximated one solved by CMIX and propose two gap loss functions to eliminate the bias of learned solutions.

4.1   Multi-objective Constrained Problem

The constrained Dec-POMDP problem can be converted into a multi-objective constrained problem by setting a lower bound δ for the objective function [10]. The bounded objective function becomes a new average constraint, which is equivalent to

$$\mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t (r(s^t, \boldsymbol{a}^t) - \delta(1 - \gamma)) \right] \geq 0. \tag{4}$$

Then, we obtain a multi-objective constrained problem which aims to find a feasible policy while satisfying the peak constraints of Eq. (2) as well as the average constraints of Eq. (3) and Eq. (4).

Consider that peak and average constraints have very different forms. So directly dealing with the two kinds of constraints together is much challenging. To address the issue, we design that the peak constraints of Eq. (2) can be absorbed into the average constraints of Eq. (3) and Eq. (4) by importing a penalty to reward functions. In particular, we define a penalty function as $p(s^t, \boldsymbol{a}^t) = \sum_{j=1}^{J} \min_{\lambda_j \geq 0} \lambda_j c_j(s^t, \boldsymbol{a}^t)$. When computing rewards, a penalty $p(s^t, \boldsymbol{a}^t)$ will be

added to the global reward r and the rewards for average constraints $\{r_k\}_{k=1}^K$. The penalty $p(s^t, \boldsymbol{a}^t)$ equals zero when all the peak constraints are satisfied; otherwise, a large minus value will be added to the computed reward values.

Now, we obtain a multi-objective constrained problem with only average constraints, i.e.,

$$\text{Find} \quad \boldsymbol{\pi}$$

$$\text{s.t.} \quad \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t (r(s^t, \boldsymbol{a}^t) - \delta(1-\gamma) + p(s^t, \boldsymbol{a}^t)) \right] \geq 0,$$

$$\mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t (r_k(s^t, \boldsymbol{a}^t) + p(s^t, \boldsymbol{a}^t)) \right] \geq 0, k = 1, \ldots, K$$

Since a violation of any peak constraints will result in very small rewards, the above problem with only average constraints is equivalent to the multi-objective constrained problem with both peak and average constraints. Also note that the penalty $p(s^t, \boldsymbol{a}^t)$ can not be realized directly but there exists a large body of literature on reward engineering for the penalty design[8,9]. In practice, we use a simple approximator to $p(s^t, \boldsymbol{a}^t)$ as $\hat{p}(s^t, \boldsymbol{a}^t) = -C \sum_{j=1}^{J} \mathbf{1}_{c^j(s^t, \mathbf{a}^t) < 0}$, where C is a positive constant bounding the variance of global reward function r and the reward functions for average constraints $\{r_k\}_{k=1}^K$, and $\mathbf{1}_{c^j(s^t, \mathbf{a}^t) < 0}$ equals 1 when $c^j(s^t, \mathbf{a}^t) < 0$ otherwise 0. We can find that the approximator equals zero when all the peak constraints are satisfied and a large minus value otherwise, which is similar to the original penalty $p(s^t, \mathbf{a}^t)$.

The above multi-objective constrained problem can be solved through an equivalent max-min optimization problem, noted as a zero-sum Markov-Bandit game in [10]. Particularly, the max-min optimization problem focuses on finding a feasible policy to maximize the minimum margin of the average constraints. It is easy to find that if the original multi-objective constrained problem is feasible, the policy, that maximizes the minimum margin of the average constraints, should be a feasible solution to the original problem. We define action-value functions for representing the margins of the average constraints. Formally, let o indicates the index of average constraints. Then, we define

$$\text{if } o = 0: \ Q(s, \boldsymbol{a}, o) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t (r(s^t, \boldsymbol{a}^t) - \delta(1-\gamma) + p(s^t, \boldsymbol{a}^t)) \right], \text{ and}$$

$$\text{for } o = 1, \ldots, K: \ Q(s, \boldsymbol{a}, o) := Q(s, \boldsymbol{a}, k) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \left(r_k(s^t, \boldsymbol{a}^t) + p(s^t, \boldsymbol{a}^t) \right) \right].$$

The Bellman equation of the max-min optimization problem is given as follows

$$\begin{aligned} Q(s, \boldsymbol{a}, o) &= \ r(s, \boldsymbol{a}, o) + \gamma \cdot \mathbb{E}_{\boldsymbol{\pi}(s')} \left[Q(s', \boldsymbol{\pi}(s'), o) \right], \\ \boldsymbol{\pi}(s) &= \ \underset{\boldsymbol{\pi}(s)}{\arg\max} \, \underset{o \in O}{\min} \, \mathbb{E}_{\boldsymbol{\pi}(s)} \left[Q(s, \boldsymbol{\pi}(s), o) \right], \end{aligned} \quad (5)$$

where $r(s, \boldsymbol{a}, o) := r(s^t, \boldsymbol{a}^t) - \delta(1-\gamma) + p(s^t, \boldsymbol{a}^t)$ for $o = 0$ and $r(s, \boldsymbol{a}, o) := r(s, \boldsymbol{a}, k) = r_k(s^t, \boldsymbol{a}^t) + p(s^t, \boldsymbol{a}^t)$ for $o = k = 1, \cdots, K$. Note that $\boldsymbol{\pi}(s)$ is a

distribution over the joint action space and can be obtained by solving a max-min optimization problem. By solving Eq. (5), we can maximize the minimum margin of the average constraints and find the feasible policy for the original multi-objective constrained problem.

This max-min optimization problem aims to provide a feasible solution to the multi-objective constrained problem for a given δ. It is easy to prove that if δ equals the optimal objective value of the original constrained Dec-POMDP problem, the obtained feasible solution is also an optimal solution to the original constrained Dec-POMDP problem. To find the optimal δ, we can set $\delta = \mathbb{E}\left[r(s, \boldsymbol{a})\right]/(1 - \gamma)$ in the max-min optimization problem learning process. In this way, the learning algorithm can learn the optimal δ and the corresponding policy $\boldsymbol{\pi}^*$ self-adaptively. In our proposed algorithm, we take an adaptive δ-search algorithm by setting $\delta_t = \varepsilon \cdot \delta_{t-1} + (1 - \varepsilon) \cdot r(s^t, \boldsymbol{a}^t)/(1 - \gamma)$ at each time step, where $\varepsilon \in [0, 1]$ is an adjustable parameter.

4.2 CMIX Architecture

To efficiently solve the constrained Dec-POMDP problem, we propose a neural architecture called CMIX by extending the original QMIX.

When applying QMIX in solving the Bellman equation in Eq.(5), we need to solve a max-min problem

$$\boldsymbol{\pi}^*(s) = \arg\max_{\boldsymbol{\pi}(s)} \min_{o \in O} \mathbb{E}_{\boldsymbol{\pi}(s)}[Q_{tot}(s, \boldsymbol{\pi}(s), o)], \tag{6}$$

where $O = \{0, 1, \ldots, K\}$ is the set of constraints and $Q_{tot}(s, \boldsymbol{\pi}(s), o)$ is the joint action value calculated by the monotonic function f_o: $Q_{tot}(s, \boldsymbol{\pi}(s), o) = f_o(Q_1^o, Q_2^o, ..., Q_N^o)$. So, there are total of $1 + K$ mixing functions. For brevity, we omit the notations of each agent's local observations and actions, and let Q_i^o denote the i-th agent's action-value function with respect to the o-th average constraint.

Recall that each agent takes actions independently in the original QMIX. So naturally we need to make each agent solve a local max-min problem for taking an action. Formally, the local max-min problem of agent i can be stated as

$$\hat{\pi}_i(z_i) = \arg\max_{\pi_i(z_i)} \min_{o \in O} \mathbb{E}_{\pi_i(z_i)}[Q_i^o]. \tag{7}$$

It is easy to prove that by solving Eq.(7) we optimize a similar but not same problem to Eq.(6). We define function g as

$$g(x_1, x_2, \ldots, x_n) := \min_{o \in O} f_o(x_1, x_2, \ldots, x_n).$$

Since $\{f_o\}$ are monotonic functions, g is also monotonic. Then the actual max-min problem solved by Eq.(7) is

$$\hat{\boldsymbol{\pi}}(s) = \arg\max_{\boldsymbol{\pi}(s)} \mathbb{E}_{\boldsymbol{\pi}(s)}[g(\min_{o \in O} Q_1^o, \ldots, \min_{o \in O} Q_N^o)]. \tag{8}$$
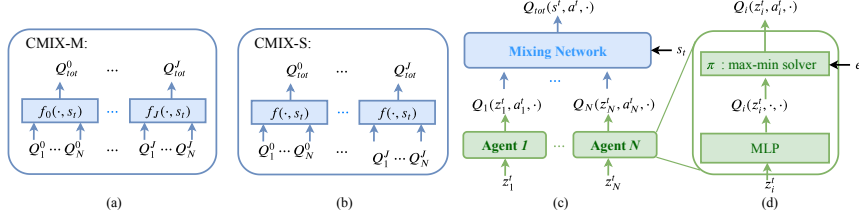
Fig. 1. (a) The mixing network structure of CMIX-M. (b) The mixing network structure of CMIX-S. (c) The overall CMIX architecture. (d) The agent network structure.

According to Eq. (8), we propose a CMIX architecture with two kinds of mixing module structures. In particular, we propose CMIX-M, the CMIX architecture with multiple mixing networks. That is to say, the mixing functions f_o ($\forall o \in O$) have independent parameters. Besides, we propose CMIX-S, the CMIX architecture with a single mixing network. The mixing functions in CMIX-S share parameters, i.e., $f_o = f$ ($\forall o \in O$). Fig. 1 shows the overall CMIX architecture with the two kinds of mixing module structures. Agent network i adopts DQN [18] and stores Q_i^o ($\forall o \in O$). Each agent takes local state z_i^t as input and outputs the corresponding Q values $Q_i^o(s_i^t, a_i^t)$ ($\forall o \in O$) after taking a local action a_i^t. The mixing network module combines the outputs of the agents monotonically, producing the values of $Q_{tot}^o(\forall o \in O)$. To guarantee the monotonicity, hypernetworks with absolute activation functions in output layers are used to generate the weights of mixing networks f_o, which follows [24].[7]

4.3 Gap Loss Function

Given the CMIX architecture, the neural parameters θ can be learned in an end-to-end fashion. For a transition $(s, \boldsymbol{a}, r, \{r_k\}_{k=1,\ldots,K}, s')$, the TD-error loss can be computed by

$$\mathcal{L}_{TD-error} = \sum_o (Q_{tot}(s, \boldsymbol{a}, o; \theta) - y^o)^2, \qquad (9)$$

where $y^o = r(s, \boldsymbol{a}, o) + \gamma \cdot \mathbb{E}_{\hat{\boldsymbol{\pi}}(s')}[Q(s', \hat{\boldsymbol{\pi}}(s'), o; \theta^-)]$ is the target value and $\hat{\boldsymbol{\pi}}(s')$ can be obtained with Eq.(7).

However, applying the TD-error loss function may be likely to result in a biased solution. As analyzed previously, the problem solved by CMIX is actually the max-min problem of Eq. (8). Since $\{f_o\}$ are monotonic functions, and g is also monotonic. We can get

$$\min_{o \in O} f_o(Q_1^o, \ldots, Q_N^o) \geq \min_{o \in O} g(Q_1^o, \ldots, Q_N^o) \geq g(\min_{o \in O} Q_1^o, \ldots, \min_{o \in O} Q_N^o), \ \forall \boldsymbol{\pi}(s) \tag{10}$$

---

[7] For brevity, we do not show the hypernetwork part of the mixing module in Fig. 1.

The left hand of Eq. (10) is the true objective function to be maximized in Eq. (6) while the right hand is the objective function we actually maximized in CMIX. We can see that CMIX may lead to a biased solution $\hat{\boldsymbol{\pi}}(s)$ to the original global max-min problem of Eq. (6) due to the gap in objective functions.

To address the issue, we design two loss functions for CMIX-M and CMIX-S, respectively.

Loss function for CMIX-M: To eliminate the bias caused by the gap between true objective function $\min_{o \in O} f_o(Q_1^o, \ldots, Q_N^o)$ and the one actually used in CMIX-M $g(\min_{o \in O} Q_1^o, \ldots, \min_{o \in O} Q_N^o)$, we propose a loss function called gap loss $\mathcal{L}_{gap}$ to minimize the gap, i.e.,

$$\mathcal{L}_{gap} = \left(\min_{o \in O} f_o(Q_1^o, \ldots, Q_N^o) - g(\min_{o \in O} Q_1^o, \ldots, \min_{o \in O} Q_N^o) \right)^2. \tag{11}$$

In the training phase, we combine the TD-error loss of Eq. (9) with the gap loss to update parameters. The final loss is computed by

$$\mathcal{L}_{final} = \mathcal{L}_{TD-error} + \beta \mathcal{L}_{gap}, \tag{12}$$

where β is a coefficient for adjusting the weight of the gap loss.

Loss function for CMIX-S: In CMIX-S we assume that all the mixing functions f_o share parameters, i.e., $g = f_o = f$ ($\forall o \in O$). Then the original max-min problem becomes $\arg \max_{\boldsymbol{\pi}(s)} \min_{o \in O} \mathbb{E}_{\boldsymbol{\pi}(s)}[f(Q_1^o, \ldots, Q_N^o)]$, and the actual max-min problem solved by CMIX is $\hat{\boldsymbol{\pi}}(s) = \arg \max_{\boldsymbol{\pi}(s)} \mathbb{E}[f(\min_{o \in O} Q_1^o, \ldots, \min_{o \in O} Q_N^o)]$. According to the above equations, we design the gap loss $\mathcal{L}_{gap}$ for CMIX-S as

$$\mathcal{L}_{gap} = \left(\min_{o \in O} f(Q_1^o, \ldots, Q_N^o) - f(\min_{o \in O} Q_1^o, \ldots, \min_{o \in O} Q_N^o) \right)^2. \tag{13}$$

The final loss can be computed in the way same as Eq. (12).

Note that the gap loss functions in Eq.(11) and (13) are fully differentiable and can be easily optimized by existing gradient descent methods.

4.4   CMIX Algorithm

In Algorithm 1, we outline the pseudocode of CMIX. In the beginning, we initialize the neural parameters of θ and θ^-, and the replay buffer $\mathcal{D}$. From line 2 to line 16, we train the agents for a number of epoch$_{\max}$ epochs, and each epoch contains a total of step$_{\max}$ update steps. The state will be initialized at the beginning of each epoch (line 3). In line 5, the agents' local observations are collected. From line 6 to line 9, each agent takes an action through an $\epsilon-$greedy method. In line 10, the joint action $\boldsymbol{a}^t$ is executed, and the transition $(s^t, \boldsymbol{a}^t, r, \{r_k^t\}_{k=1,\ldots,K}, s^{t+1})$ is stored in the buffer $\mathcal{D}$. In line 11, a mini-batch B of samples are selected from $\mathcal{D}$ randomly. Then, the loss $\mathcal{L}_{final}$ of Eq. (12) is computed with respect to each sample in line 12. In line 13, the computed loss guides the update of θ. Finally, the target θ^- will be updated from θ after a specified interval.

Algorithm 1: CMIX Algorithm

1: Initialize parameters θ, target $\theta^- = \theta$, and the replay buffer $\mathcal{D} = \emptyset$

2: **for** epoch $= 1, \cdots, \text{epoch}_{max}$ **do**

3:     Initialize state s^0

4:     **for** $t = 0, \cdots, \text{step}_{max}$ **do**

5:         Collect observations $\{z_i^t\}_{i \in \mathcal{N}}$ for all agents

6:         **for** each agent i **do**

7:             $\hat{\pi}_i(z_i^t) = \underset{\pi_i \in \Pi_i}{\arg\max} \underset{o \in O}{\min} \mathbb{E}[Q_i(z_i^t, \pi_i(z_i^t), o)]$

8:             $a_i^t = \begin{cases} sample(\hat{\pi}_i(z_i^t)) & \text{with prob. } 1 - \epsilon \\ randint(1, |\mathcal{A}_i|) & \text{with prob. } \epsilon \end{cases}$

9:         **end for**

10:        Execute the joint action $\boldsymbol{a}^t$ and store the transition
           $(s^t, \boldsymbol{a}^t, r, \{r_k^t\}_{k=1,\ldots,K}, s^{t+1})$ in $\mathcal{D}$

11:        Sample a mini-batch B from $\mathcal{D}$

12:        Compute $\mathcal{L}_{final}$ of Eq. (12) for each sample in B

13:        Update θ by minimizing the average loss of $\mathcal{L}_{final}$ with respect to B

14:        Update target $\theta^- = \theta$ periodically

15:    **end for**

16: **end for**

---

5   Experiments

We evaluate CMIX on two different tasks: a blocker game with travel cost and a cooperative routing optimization task in large-scale vehicular networks. In our numerical results, CMIX is compared with the state-of-the-art CTDE learning algorithms, including IQL [27], VDN [26], and QMIX [23]. Since these algorithms do not consider constraints during training, we also compare performance with C-IQL where each agent independently optimizes the global objective with the consideration of peak and average constraints. C-IQL is implemented by extending IQL in a way similar to CMIX.

5.1   Blocker Game with Travel Cost

We consider a non-trivial blocker game with travel cost, which is an extension of the blocker game in [29]. The agents should cooperate to reach the bottom row of the map as quickly as possible, while the blockers move left or right to block the agents. Each cell on the map is assigned a cost, and an agent will take the cost when it moves into the cell. We also place some traps on the map, which the agents are not allowed to move into.

Fig. 2 shows the blocker game with travel cost in our evaluation. In the game, it costs -1 reward per time-step before the agents reach the destination, and an extra small reward will be returned to the agents when they are getting closer to the bottom. The agents try to minimize the winning step, i.e., the movement steps needed by the agents for reaching the bottom, subject to a peak constraint
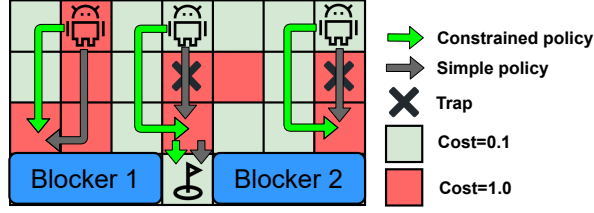
Fig. 2. Blocker game with travel cost.

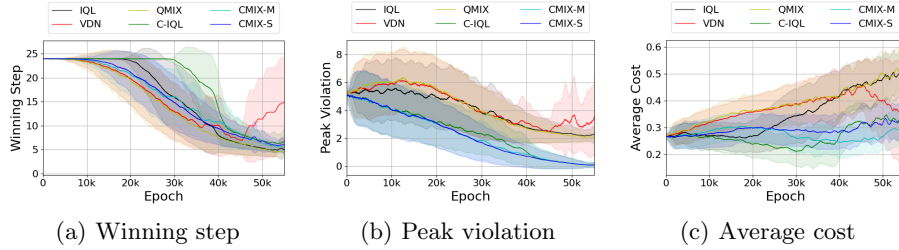(a) Winning step   (b) Peak violation   (c) Average cost

Fig. 3. Convergence results over training epoch in the blocker game of Fig. 2. Each curve with shadow shows the mean values and the variants of the observed metric. The average cost no larger than 0.3 represents that the average constraint is fulfilled.

and an average constraint. The agents should not move into traps (i.e., peak constraint). Besides, the average cost taken by these agents in one game should be bounded. The upper bound of average cost is set to 0.3 in our evaluation. With the optimal strategy labeled by green arrows, the agents can win the game in 5 steps, satisfying both peak and average constraints. The blocker game is challenging for the agents in the sense of cooperation with only decentralized policy and local observations, and the peak and average constraints make the task even more difficult to complete. The convergence results of different algorithms in the blocker game with travel cost are presented in Fig. 3.

Winning step: Fig. 3 (a) shows the convergence results of winning step. We can see that the winning step decreases with the increment of the epoch. QMIX gets the smallest winning step finally without considering constraints. CMIX-M, CMIX-S, and IQL get similar performance on winning step and outperform VDN and C-IQL which either have larger variance or take more training epochs to converge. We note that CMIX-M and CMIX-S optimize winning step under peak and average constraints, while IQL does not consider constraints.

Peak violation: Fig. 3 (b) shows the convergence results of peak violation, i.e., the number of violated peak constraints in each epoch. We can see that CMIX-M, CMIX-S, and C-IQL get results very close to zero after convergence. That is to say, the peak constraint is mostly satisfied by CMIX-M, CMIX-S, and C-IQL. However, the other algorithms, i.e., QMIX, IQL, and VDN, receive significant peak violations due to the ignorance of peak constraints.

Average cost: Fig. 3 (c) shows the convergence results of average cost, i.e., the average cost with respect to each agent and each step. We can see only CMIX-M satisfies the average cost after convergence, while CMIX-S and C-IQL violate the average constraints slightly. Note that compared with CMIX-M, CMIX-S using a single mixing network has a relatively limited representation ability.

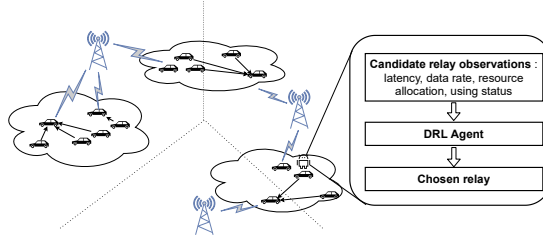## 5.2   Vehicular Network Routing Optimization

Fig. 4. The cooperative routing optimization problem in a vehicular network.

We apply the MARL algorithms in a cooperative routing optimization problem in vehicular networks shown in Fig. 4. In the scenario with three cells, each vehicle can establish a V2I (Vehicle-to-Infrastructure) link with the base station in the local cell or V2V (Vehicle-to-Vehicle) links with neighboring vehicles. These V2I and V2V links have different transmission data rates and link latencies. We attach an RL agent on each vehicle and consider downlink data transmission, i.e., data needs to be delivered from base stations to destination vehicles. For a destination vehicle, its downlink data can be delivered through i) the path of the direct V2I link, or ii) the path consisting of a V2V link with a neighboring vehicle (i.e., a relay) and a V2I link between the relay and the corresponding base station. The routing decision, that an agent on the vehicle needs to make, is choosing a proper path for downlink data transmission according to local observations of the candidate relays. All the vehicles need to be coordinated to maximize the total transmission rate with the consideration of proportional fairness, while satisfying some peak and average latency constraints. Particularly, the path latency of each vehicle should not exceed a threshold, i.e., peak constraints, and the average path latency of all the vehicles in the network is bounded by a soft upper bound.

In our evaluation, we consider a vehicular network with three cells (also three base stations) with tens of randomly generated vehicles (30-60 vehicles). Link latencies and link transmission rates are also set randomly.

Besides the baselines in the blocker game, we also consider three other baselines. i) Only-V2I: Downlink data is delivered to each vehicle through the directly connected V2I link. ii) Data-greedy: Each vehicle chooses the path with the largest data rate. iii) DCRA [15]: An iterative vehicular routing optimization
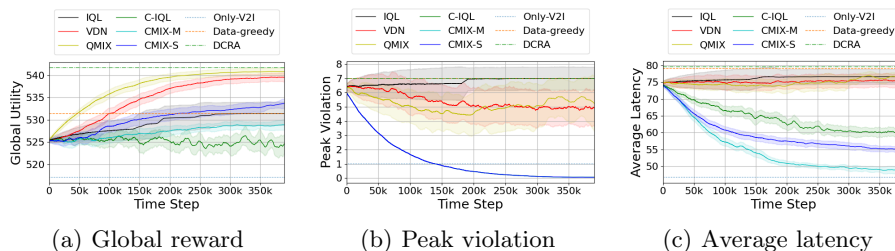
(a) Global reward   (b) Peak violation   (c) Average latency

Fig. 5. Convergence results over training step in the vehicular routing problem. The average latency no larger than 60 represents that the average constraint is fulfilled.

algorithm that can improve the transmission rate with the consideration of proportional fairness.

Global utility: Fig. 5 (a) shows the convergence of global utility. We can see that QMIX and VDN outperform others because they do not consider constraints. Then, both CMIX-S and CMIX-M outperform Only-V2I significantly, and CMIX-S provides even larger global utility than Data-greedy, which validates the effectiveness of CMIX. Note that both IQL and C-IQL suffer performance degradation since it is difficult to learn a consistent strategy for the agents without a mixing architecture. Also, note that CMIX-S shows a better performance than CMIX-M. This is because CMIX-M having more parameters is more challenging to converge than CMIX-S in such a large-scale and complicated task.

Peak violation: Fig. 5 (b) shows the convergence of peak violation. Note that, the curves of C-IQL, CMIX-M, and CMIX-S are overlapped. We can see the three algorithms can fulfill the peak constraint after convergence. Note that Only-V2I using the one-hop path of the directly connected V2I link still violates the peak constraint sometimes.

Average latency: Fig. 5 (c) shows the convergence of average latency. We can see only CMIX-M and CMIX-S satisfy the average constraint. The other baselines do not meet the average constraint.

5.3   Gap Loss Coefficient

We evaluate the performance of CMIX under different settings of the gap loss coefficient weight β in the blocker game with travel cost. $\beta = 0$ means not using the gap loss function. Fig. 6 shows the convergence result over training epoch. We find that the CMIX algorithm converges more stably and to a better result when β is large enough, which illustrates the positive effect of gap loss in bias elimination. Moreover, the results indicate that gap loss plays a more important role in the CMIX-M algorithm.

We also evaluate the effect of β in the vehicular network routing optimization task. Fig. 7 shows the convergence result over training step. We can find that CMIX-S with gap loss gives better results, which illustrates the effectiveness of gap loss for CMIX in large-scale tasks. Also, note that CMIX-M performances
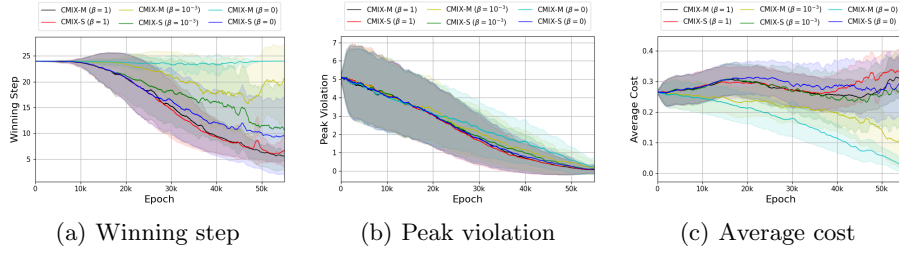
(a) Winning step   (b) Peak violation   (c) Average cost

Fig. 6. Convergence results of CMIX with different coefficient β over training epoch in the blocker game of Fig. 2. The average cost bound is set to 0.3.
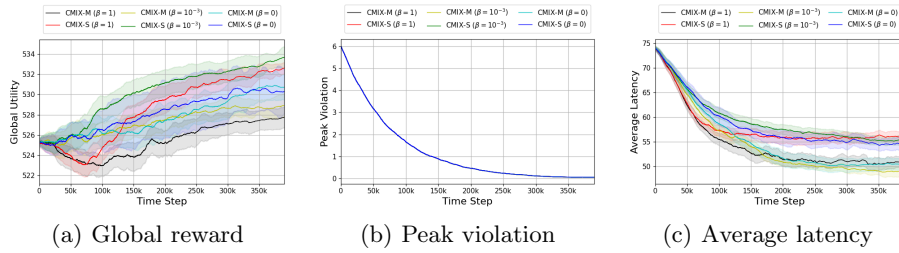

(a) Global reward   (b) Peak violation   (c) Average latency

Fig. 7. Convergence results of CMIX with different coefficient β over training step in the vehicular routing problem. The average latency bound is set to 60.

worse with larger β since CMIX-M has more parameters and the gap loss possiblily makes it more difficult to converge in large-scale tasks.

6 Related Work

Reinforcement learning under peak or average constraints. There are lots of approaches [9,3,11,12,6,10,22,4] focusing on RL under either peak or average constraints. However, most of the constrained RL algorithms tackle only one type of the two constraints – but not both – and focus on single-agent setting.

    Multi-agent reinforcement learning under constraints. Existing MARL algorithms are often developed in the paradigm of CTDE, i.e., centralized training with decentralized execution. Independent learning algorithms like IQL [27] treat other agents as part of the environment, which usually do not converge well. Another kind of approaches called centralized learning [7,13] learn a fully centralized state-action value function and then use it to guide the optimization of policies for decentralized agents. However, centralized learning suffers bad scalability due to combinatorial complexity, especially in large-scale cooperative tasks. Recently, another paradigm lying between independent learning and centralized learning has attracted much attention. A typical method called QMIX [24] as well as its extensions [23,17,29] combines the Q-values of agents through a mixing module,

which can coordinate agents efficiently and result in good scalability. MARL under constraints has already attracted some attentions [5,20]. However, to the best of our knowledge, there are no MARL learning algorithms that can address both peak and average constraints especially in the realm of CTDE algorithms.

7 Conclusion

In this paper, we propose the CMIX algorithm for Dec-POMDP problem under both peak and average constraints. To this end, the original problem is converted into a multi-objective constrained problem, which can be solved through an equivalent max-min optimization problem. We leverage the value function factorization to develop a novel neural architecture in the CTDE paradigm for solving the max-min optimization problem approximately. We further analyze the gap between the original max-min optimization problem and the approximated ones solved by CMIX and propose two gap loss functions to eliminate the bias of learned solutions. Evaluations on a blocker game with travel cost and a large-scale vehicular network routing problem validate the effectiveness of CMIX. We note that the proposed CMIX approach can be integrated with other algorithms using value function factorization, e.g., [29,17], which will be left to our future work.

Acknowledgment

References

1. Abedi, A., Ghaderi, M., Williamson, C.: Distributed routing for vehicular ad hoc networks: Throughput-delay tradeoff. In: IEEE MASCOTS (2010)
2. Al-Abbasi, A.O., Ghosh, A., Aggarwal, V.: Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. IEEE TITS (2019)
3. Bai, Q., Gattami, A., Aggarwal, V.: Provably efficient model-free algorithm for MDPs with peak constraints. arXiv preprint arXiv:2003.05555 (2021)
4. Chow, Y., Nachum, O., Duenez-Guzman, E., Ghavamzadeh, M.: A Lyapunov-based approach to safe reinforcement learning. In: NeurIPS (2018)
5. Diddigi, R.B., Danda, S.K.R., Bhatnagar, S., et al.: Actor-critic algorithms for constrained multi-agent reinforcement learning. arXiv preprint arXiv:1905.02907 (2019)
6. Ding, D., Wei, X., Yang, Z., Wang, Z., Jovanović, M.R.: Provably efficient safe exploration via primal-dual policy optimization. In: AISTATS (2021)
7. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: AAAI (2018)
8. Garcıa, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. JMLR (2015)

9. Gattami, A.: Reinforcement learning of markov decision processes with peak constraints. arXiv preprint arXiv:1901.07839 (2019)
10. Gattami, A., Bai, Q., Aggarwal, V.: Reinforcement learning for multi-objective and constrained Markov decision processes. In: AISTATS (2021)
11. Geibel, P.: Reinforcement learning for MDPs with constraints. In: Springer ECML (2006)
12. Geibel, P., Wysotzki, F.: Risk-sensitive reinforcement learning applied to control under constraints. JAIR (2005)
13. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multi-agent control using deep reinforcement learning. In: Springer AAMAS (2017)
14. Ha, D., Dai, A., Le, Q.V.: Hypernetworks. arXiv preprint arXiv:1609.09106 (2016)
15. Kassir, S., de Veciana, G., Wang, N., Wang, X., Palacharla, P.: Enhancing cellular performance via vehicular-based opportunistic relaying and load balancing. In: IEEE INFOCOM (2019)
16. Li, Z., Wang, C., Jiang, C.J.: User association for load balancing in vehicular networks: An online reinforcement learning approach. IEEE TITS (2017)
17. Mahajan, A., Rashid, T., Samvelyan, M., Whiteson, S.: Maven: Multi-agent variational exploration. arXiv preprint arXiv:1910.07483 (2019)
18. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
19. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature (2015)
20. Nguyen, D.T., Yeoh, W., Lau, H.C., Zilberstein, S., Zhang, C.: Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs. In: AAAI (2014)
21. Oliehoek, F.A., Amato, C., et al.: A concise introduction to decentralized POMDPs, vol. 1. Springer (2016)
22. Prashanth, L., Ghavamzadeh, M.: Variance-constrained actor-critic algorithms for discounted and average reward MDPs. Springer Machine Learning (2016)
23. Rashid, T., Farquhar, G., Peng, B., Whiteson, S.: Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. NeurIPS (2020)
24. Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., Whiteson, S.: QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: ICML (2018)
25. Saleet, H., Langar, R., Naik, K., Boutaba, R., Nayak, A., Goel, N.: Intersection-based geographical routing protocol for VANETs: A proposal and analysis. IEEE TVT (2011)
26. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V.F., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., et al.: Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Springer AAMAS (2018)
27. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: ICML (1993)
28. Wang, F., Wang, F., Liu, J., Shea, R., Sun, L.: Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach. In: IEEE INFOCOM (2020)
29. Yang, Y., Wen, Y., Wang, J., Chen, L., Shao, K., Mguni, D., Zhang, W.: Multi-agent determinantal Q-learning. In: ICML (2020)