Configurable Mirror Descent: Towards a Unification of Decision Making

Pengdeng Li[1] Shuxin Li[1*] Chang Yang[2*] Xinrun Wang[3] Shuyue Hu[4] Xiao Huang[2] Hau Chan[5] Bo An[1 6]

Abstract

Decision-making problems, categorized as single-agent, e.g., Atari, cooperative multi-agent, e.g., Hanabi, competitive multi-agent, e.g., Hold'em poker, and mixed cooperative and competitive, e.g., football, are ubiquitous in the real world. Although various methods have been proposed to address the specific decision-making categories, these methods typically evolve independently and cannot generalize to other categories. Therefore, a fundamental question for decision-making is: *Can we develop **a single algorithm** to tackle **ALL** categories of decision-making problems?* There are several main challenges to address this question: i) different categories involve different numbers of agents and different relationships between agents, ii) different categories have different solution concepts and evaluation measures, and iii) there lacks a comprehensive benchmark covering all the categories. This work presents a preliminary attempt to address the question with three main contributions. i) We propose the generalized mirror descent (GMD), a generalization of MD variants, which considers multiple historical policies and works with a broader class of Bregman divergences. ii) We propose the configurable mirror descent (CMD) where a meta-controller is introduced to dynamically adjust the hyper-parameters in GMD conditional on the evaluation measures. iii) We construct the GameBench with 15 academic-friendly games across different decision-making categories. Extensive experiments demonstrate that CMD achieves empirically competitive or better outcomes compared to baselines while providing the capability of exploring diverse dimensions of decision making.

[*]Equal contribution [1]Nanyang Technological University [2]The Hong Kong Polytechnic University [3]Singapore Management University (work done while at NTU) [4]Shanghai Artifcial Intelligence Laboratory [5]University of Nebraska-Lincoln [6]Skywork AI. Correspondence to: Xinrun Wang <xrwang@smu.edu.sg>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Decision-making problems are pervasive in the real world (Sutton & Barto, 2018; Shoham & Leyton-Brown, 2008), which can be generally categorized into single-agent, e.g., Atari (Mnih et al., 2015), cooperative multi-agent, e.g., Hanabi game (Bard et al., 2020), competitive multi-agent, e.g., Hold'em poker (Brown & Sandholm, 2018; 2019), and mixed cooperative and competitive (MCC), e.g., football (Kurach et al., 2020; Liu et al., 2022a). To solve these problems, various methods are proposed where notable examples include PPO (Schulman et al., 2017) for single-agent category, QMIX (Rashid et al., 2018) for cooperative multi-agent category and PSRO (Lanctot et al., 2017) for competitive category. Despite the successes in specific categories, these methods are developed almost independently and cannot generalize to other categories. Therefore, a fundamental question for decision making to answer is:

*Can we develop **a single algorithm** to tackle **ALL** categories of decision-making problems?*

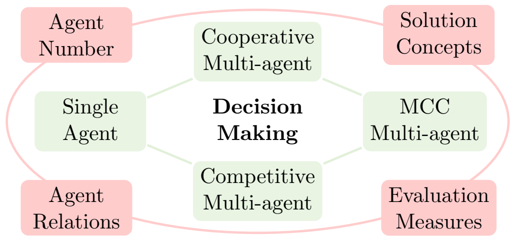


Figure 1. Overview of the categories of decision making and the four desiderata for the required method to satisfy.

There are several critical challenges to address this fundamental question. First, the different categories of decision-making problems include different numbers of agents and different relationships between agents. There is one agent for the single-agent category, while multiple agents for the other three categories, therefore, the reinforcement learning methods, e.g., PPO, mainly developed for single-agent decision-making problems, cannot be directly applied to multi-agent categories. Furthermore, even for multi-agent categories, QMIX (Rashid et al., 2018) is developed to handle the cooperative multi-agent category and cannot be applied to the competitive category. Second, different decision-making categories have different solution concepts, where the optimal (joint) policy is considered in the single-agent and cooperative multi-agent categories, while for the competitive and MCC multi-agent categories, Nash equilibrium (NE) (Nash, 1951) is the canonical solution concept and other solution concepts, e.g., correlated equilibrium (Aumann, 1987) are also considered. Furthermore, even for one solution concept, e.g., NE, there are different evaluation measures, e.g., NashConv or NashConv with social welfare and fairness[1]. To summarize the challenges, we propose the four desiderata that the methods should satisfy:

- **D1**: Applicable to single- and multi-agent categories
- **D2**: Applicable to coop., comp., & MCC categories
- **D3**: Applicable to different solution concepts
- **D4**: Applicable to different evaluation measures

An overall illustration of the categories of the decision making and the desiderata is displayed in Figure 1. Third, existing benchmarks are typically specialized for specific decision-making categories, while a comprehensive benchmark that satisfies the following two desiderata is lacking.

- **D5**: (Comprehensive) It covers all categories
- **D6**: (Academic-friendly) It is less resource-intensive

In this work, we make a preliminary attempt to address these challenges and provide three main contributions. i) We propose the generalized mirror descent (GMD), a generalization of existing MD algorithms (Nemirovskij & Yudin, 1983; Beck & Teboulle, 2003), which incorporates multiple historical policies into the policy updating and is able to explore a broader class of Bregman divergence by addressing the Karush–Kuhn–Tucker (KKT) conditions at each iteration. As GMD is adopted by each agent independently, it can be applied to different decision-making categories involving different numbers of agents and different relationships between agents (**D1** and **D2**). ii) We propose the configurable mirror descent (CMD) by introducing a meta-controller to dynamically adjust the hyper-parameters in GMD conditional on the evaluation measures, allowing us to study different solution concepts as well as evaluation measures (**D3** and **D4**), with minimal modifications. iii) We construct the GameBench consisting of 15 games which cover all the decision-making categories (**D5**) and are deliberately constructed with the principle that running algorithms on these games does not require much computational resource (**D6**), and hence, forming a comprehensive and academic-friendly testbed for researchers to efficiently develop and test novel algorithms. Extensive experiments on the GameBench show that CMD achieves empirically competitive or better outcomes compared to baselines while offering the ability to investigate diverse dimensions of decision making.

[1]This is related to the equilibrium selection problem (Harsanyi et al., 1988) and different measures lead to different equilibria.

2. A Real-World Motivating Scenario

We provide an illustrative example to highlight the importance and real-world implications of a unified algorithm framework. Consider that a robotic company is developing and selling *generalist domestic robots* to users. The user may ask the robot to learn to complete different novel tasks, including single-agent, cooperative, competitive, and MCC categories, by specifying the objective. Therefore, if we can deploy a unified algorithm into the robot, the robot can learn to complete different novel tasks with a single algorithm.

Developing and deploying such a unified algorithm would benefit both the development and users. *For the development side*, as only a single policy learning rule is required, the deployment and user interface design could be largely simplified, which would be more cost-efficient than deploying different specialized algorithms such as MAPPO and PSRO as they may complicate the development pipeline and user interface design. *For the user side*, the user only needs to configure one set of parameters for different novel tasks, e.g., only needs to specify the optimization objective of the meta-controller in our proposed CMD algorithm.

3. Related Work

The related literature is too vast to cover in its entirety. We present an overview below to emphasize our contributions while more related works can be found in Appendix B.

Decision Making. Substantial progress has been achieved in developing algorithms to address different categories of decision-making problems, e.g., DQN (Mnih et al., 2015) and PPO (Schulman et al., 2017) for single-agent category, QMIX (Rashid et al., 2018) and MAPPO (Yu et al., 2022) for cooperative multi-agent category, self-play (Tesauro et al., 1995) and PSRO (Lanctot et al., 2017) for competitive and MCC categories, to name just a few. Despite the successes in specific categories, these methods often cannot directly generalize to different categories. In this work, we make a preliminary attempt to develop *a single algorithm* capable of *tackling all categories of decision-making problems* which typically involve different numbers of agents, different relationships between agents, different solution concepts as well as different evaluation measures.

Mirror Descent. Mirror descent (MD) (Nemirovskij & Yudin, 1983; Beck & Teboulle, 2003; Vural et al., 2022) has shown effectiveness in learning optimal policies in single-agent RL (Tomar et al., 2022) and proved the last-iterate convergence in learning approximate equilibrium in zero-sum games (Bailey & Piliouras, 2018; Kangarshahi et al., 2018; Wibisono et al., 2022; Kozuno et al., 2021; Lee et al., 2021; Jain et al., 2022; Ao et al., 2023; Liu et al., 2023; Cen et al., 2023; Sokota et al., 2023) and some classes of general-sum games, e.g., polymatrix and potential games (Anagnostides et al., 2022b). Despite the progress, existing works typically focus on some specific Bregman divergence such as the KL divergence. We relax this premise by addressing the KKT conditions at each iteration, enabling us to *explore a broader class of Bregman divergence*. Moreover, by introducing a meta-controller to dynamically adjust the hyper-parameters, our CMD can be applied to *different solution concepts and evaluation measures* with minimal modifications.

Hyper-Parameter Tuning. Existing works typically determine the hyper-parameter values of the MD algorithms depending on domain knowledge (Sokota et al., 2023; Anagnostides et al., 2022b; Hsieh et al., 2021), which may not be easy to generalize to different games. On the other hand, gradient-based hyper-parameter tuning methods such as STAC (Zahavy et al., 2020) are less applicable as the evaluation measures, e.g., NashConv, could be non-differentiable with respect to the hyper-parameters. To address the issue, we propose a simple yet effective zero-order optimization method where the performance difference between two candidates is used to only determine the *update direction* of the hyper-parameters rather than the update magnitude, which is more effective than existing methods (Wang et al., 2022) when the value of the performance is extremely small.

4. Preliminaries

In this section, we first introduce the model of decision making and the solution concepts and evaluation measures considered in our work. Then, we present the classic mirror descent algorithm (Beck & Teboulle, 2003).

4.1. Decision Making

POSG. A decision-making problem, either single-agent, cooperative, competitive, or mixed cooperative and competitive category, can be described as a partially observable stochastic game (POSG) (Oliehoek & Amato, 2016) denoted as $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \Omega, P, R, \gamma, \nu \rangle$. $\mathcal{N} = \{1, \cdots, N\}$ is the set of agents. $\mathcal{S}$ is the *finite* set of the states. $\mathcal{A} = \times_{i\in\mathcal{N}}\mathcal{A}_i$ and $\mathcal{O} = \times_{i\in\mathcal{N}}\mathcal{O}_i$ where $\mathcal{A}_i$ and $\mathcal{O}_i$ are the *finite* set of actions and observations of agent i, respectively. Let $\boldsymbol{a} \in \mathcal{A}$ denote the joint action of agents where $a_i \in \mathcal{A}_i$ is agent i's action. $\Omega = \times_{i\in\mathcal{N}}\Omega_i$ where $\Omega_i : \mathcal{S} \times \mathcal{A} \to \mathcal{O}_i$ is the observation function specifying agent i's observation $o_i \in \mathcal{O}_i$ when all agents take $\boldsymbol{a} \in \mathcal{A}$ at state $s \in \mathcal{S}$. $P : \mathcal{S}\times\mathcal{A} \to \Delta(\mathcal{S})$ is the transition function which specifies the probability of transiting to $s' \in \mathcal{S}$ when agents take $\boldsymbol{a} \in \mathcal{A}$ at state $s \in \mathcal{S}$. $\Delta(\cdot)$ denotes the simplex. $R = \{r_i\}_{i\in\mathcal{N}}$ where $r_i : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is the reward function of agent i and $\gamma \in [0, 1)$ is the discount factor. $\nu \in \Delta(\mathcal{S})$ denotes the distribution over initial states. At time step $t \geq 0$, each agent has an action-observation history (i.e., a decision point) $\tau_i^t \in \mathcal{T}_i^t$ where $\mathcal{T}_i^t = (\mathcal{O}_i \times \mathcal{A}_i)^t$ and constructs its policy $\pi_i : \mathcal{T}_i^t \to \Delta(\mathcal{A}_i)$ to maximize its own return. Let Π_i denote the policy space of agent i, that is, we have $\pi_i \in \Pi_i$. The joint policy of all agents is denoted as $\boldsymbol{\pi} = \pi_1 \odot \cdots \odot \pi_N$ and $\boldsymbol{\pi} \in \Pi$ where Π denotes the joint policy space of all agents. A special case of joint policy is the *product policy* denoted as $\boldsymbol{\pi} = \pi_1 \times \cdots \times \pi_N$. Also, let $\boldsymbol{\pi}_{-i} = \pi_1 \odot \cdots \pi_{i-1} \odot \pi_{i+1} \cdots \odot \pi_N$ denote the joint policy of all agents except i. Given the initial state $s_0 = s$, the value function of agent i is $V_i(s, \boldsymbol{\pi}) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_i^t | s, \boldsymbol{\pi}]$ where r_i^t is the agent i's reward at time $t \geq 0$. Furthermore, we have $V_i(\nu, \boldsymbol{\pi}) := \mathbb{E}_{s\sim\nu}[V_i(s, \boldsymbol{\pi})]$.

Solution Concepts. The policy of an agent is said to be optimal if it is optimal in every decision point belonging to the agent. In single-agent and cooperative categories, this *optimal policy* maximizes the expected return for the agent or the team. In multi-agent competitive and mixed cooperative and competitive categories, we consider two common equilibrium concepts: *Nash equilibrium (NE)* (Nash, 1951) and *coarse correlated equilibrium (CCE)* (Moulin & Vial, 1978). Let $\pi_i \times \boldsymbol{\pi}_{-i}$ denote the *product policy* and $\pi_i \odot \boldsymbol{\pi}_{-i}$ denote the joint policy. Then, $\boldsymbol{\pi}^*$ is called an NE if for each agent i it satisfies: $\forall \pi_i' \in \Pi_i$, $V_i(\nu, \boldsymbol{\pi}^*) \geq V_i(\nu, \pi_i' \times \boldsymbol{\pi}_{-i}^*)$. Similarly, $\boldsymbol{\pi}^*$ is called a CCE if for each agent i it satisfies: $\forall \pi_i' \in \Pi_i$, $V_i(\nu, \boldsymbol{\pi}^*) \geq V_i(\nu, \pi_i' \odot \boldsymbol{\pi}_{-i}^*)$.

Evaluation Measures. Let $\mathcal{L}(\boldsymbol{\pi})$ denote measures used to evaluate a (joint) policy $\boldsymbol{\pi}$. In single-agent and cooperative categories, the measure is the distance of the (joint) policy to the optimal (joint) policy $\boldsymbol{\pi}^*$, which is defined as $\mathcal{L}(\boldsymbol{\pi}) = \text{OptGap}(\boldsymbol{\pi}) = V(\nu, \boldsymbol{\pi}^*) - V(\nu, \boldsymbol{\pi})$. In other categories, we consider multiple evaluation measures. The first one is the distance of the joint policy to the equilibrium (NE or CCE). For NE, we refer to this distance as NashConv, and for CCE, we refer to it as CCEGap, as is convention in previous works (Lanctot et al., 2017; Marris et al., 2021). More specifically, we have $\text{NashConv}(\boldsymbol{\pi}) = \sum_{i\in\mathcal{N}}[V_i(\nu, \pi_i^{\text{BR}} \times \boldsymbol{\pi}_{-i}) - V_i(\nu, \boldsymbol{\pi})]$ and $\text{CCEGap}(\boldsymbol{\pi}) = \sum_{i\in\mathcal{N}}[V_i(\nu, \pi_i^{\text{BR}} \odot \boldsymbol{\pi}_{-i}) - V_i(\nu, \boldsymbol{\pi})]$, where π_i^{BR} is the best response (BR) policy of agent i against all other agents. The second evaluation measure we consider is the social welfare (SW) (Davis & Whinston, 1962), denoted as $\mathcal{L}(\boldsymbol{\pi}) = \sum_{i\in\mathcal{N}} V_i(\nu, \boldsymbol{\pi})$.

4.2. Mirror Descent

From a single agent's perspective, the condition for the optimal or equilibrium policy can be expressed by the following optimization problem at each decision point of the agent (Tomar et al., 2022; Sokota et al., 2023): $\forall \tau_i^t \in \mathcal{T}_i^t$,

$$\pi_i^*(\tau_i^t) = \arg\max_{\pi_i\in\Pi_i} \mathbb{E}_{a\sim\pi_i(\tau_i^t)} Q(\tau_i^t, a, \pi_i \odot \boldsymbol{\pi}_{-i}^*), \quad (1)$$

where $Q(\tau_i^t, a, \boldsymbol{\pi}) = \mathbb{E}[\sum_{h=t+1}^{\infty} \gamma^h r_i^h | \tau_i^t, a_i^t = a, \boldsymbol{\pi}]$ is the action-value function of the action $a \in \mathcal{A}_i$ at the decision point τ_i^t. Without loss of generality, we will only focus on the policy learning of a single agent i in a single decision point $\tau_i^t \in \mathcal{T}_i^t$ and henceforth, the index i and τ_i^t are ignored as they are clear from the context, and with a slight abuse of

Table 1. Comparison of different methods. *Note that MMD can be regarded as the method that can consider multiple previous policies by setting the magnet policy to the initial policy (typically a uniform policy).

Method	Multiple previous policies	Working on any Bregman divergence	Working on any solution concept	Configurable for any measure
MD (Nemirovskij & Yudin, 1983)	✗	✗	✗	✗
MMD* (Sokota et al., 2023)	✓	✗	✗	✗
GMD (This work)	✓	✓	✗	✗
CMD (This work)	✓	✓	✓	✓

notation, we denote $\mathcal{A}$ the action set $\mathcal{A}_i$ of agent i, $\pi \in \Pi$ the agent's policy, and $Q(a)$ (or $Q(a, \pi)$) the action-value of the action $a \in \mathcal{A}$. Then, to learn the optimal or equilibrium policy, we aim to solve the optimization problem: $\pi^* = \arg\max_{\pi\in\Pi} \mathbb{E}_{a\sim\pi} Q(a)$. A feasible method to solve this problem is the mirror descent (MD), which takes the form (Beck & Teboulle, 2003; Tomar et al., 2022):

$$\pi_{k+1} = \arg\max\nolimits_{\pi\in\Pi} \langle Q(\pi_k), \pi\rangle - \alpha\mathcal{B}_\phi(\pi, \pi_k), \tag{2}$$

where $1 \leq k \leq K$ is the iteration, $Q(\pi_k)$ is the action-value vector induced by π_k (for simplicity, we let $Q(\pi_k) = (Q(a, \pi_k))_{a\in\mathcal{A}}$), α is the regularization intensity, $\mathcal{B}_\phi$ is the Bregman divergence with respect to the mirror map ϕ, defined as $\mathcal{B}_\phi(x; y) = \phi(x) - \phi(y) - \langle\nabla\phi(y), x - y\rangle$ with $\langle\cdot\rangle$ being the standard inner product and $x, y \in \Delta(\mathcal{A})$.

5. Configurable Mirror Descent

In this section, we propose a novel algorithm which satisfies the four desiderata (**D1–D4**) presented in the Introduction. First, we propose the generalized mirror descent (GMD), a generalization of existing MD algorithms, which when independently executed by each agent, can effectively tackle different decision-making categories involving different numbers of agents and different relationships between agents (**D1** and **D2**). Second, we propose the configurable mirror descent (CMD) where a meta-controller is introduced to dynamically adjust the hyper-parameters of GMD conditional on the evaluation measures, which can be configured to account for different solution concepts as well as evaluation measures (**D3** and **D4**), with minimal modifications. CMD shares similarities with the centralized training and decentralized execution (CTDE) (Lowe et al., 2017) since the meta-controller considers all agents to optimize the targeted evaluation measures ("centralized" training from the controller's perspective) while GMD is executed by each agent independently ("decentralized" execution from each agent's perspective). The overview of CMD is shown in Figure 2 and Table 1 presents a comparison to more clearly position our methods in the context of related literature.

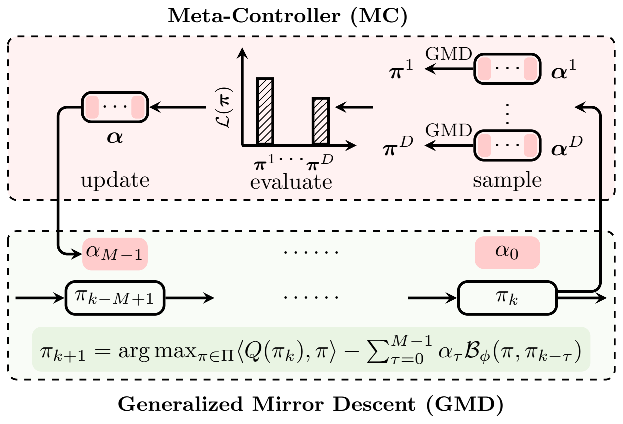


Figure 2. Overview of CMD.

5.1. Generalized Mirror Descent

Though existing MD algorithms, e.g., Eq. (2), can be also executed by each agent independently, they could not generalize well to satisfy the desiderata **D1** and **D2**. The main reasons are two-fold. First, classic MD algorithms (Beck & Teboulle, 2003; Tomar et al., 2022) typically only consider the current policy when deriving the new policy at each iteration. However, it has been shown that incorporating multiple previous policies (e.g., the initial and current policies) could be powerful in solving two-player zero-sum games (Sokota et al., 2023; Liu et al., 2023). Second, even though multiple previous policies are considered, existing MD algorithms typically focus on some specific Bregman divergences by restricting ϕ to certain convex functions which may not be the optimal choices across different decision-making categories. To address these challenges, we propose a more general MD method satisfying the desiderata **D1** and **D2**.

A General MD Method. We propose a more general MD approach which takes multiple historical policies into account when deriving the new policy, as given below:

$$\pi_{k+1} = \arg\max_{\pi\in\Pi}\langle Q(\pi_k), \pi\rangle - \sum\nolimits_{\tau=0}^{M-1} \alpha_\tau \mathcal{B}_\phi(\pi, \pi_{k-\tau}),$$
$$\textit{s.t.} \sum\nolimits_{a\in\mathcal{A}} \pi_k(a) = 1 \text{ and } \pi_k(a) \geq 0, \forall a \in \mathcal{A}, \tag{3}$$

where $M \geq 1$ is the number of historical policies, $\alpha_\tau \in (0, 1]$ is the regularization intensity of $\pi_{k-\tau}$, $0 \leq \tau \leq M-1$, and let $\boldsymbol{\alpha} = (\alpha_\tau)_{0 \leq \tau \leq M-1}$. Note that solving the problem (3) to derive the policy updating rule could be challenging. In practice, the problem could have a closed-form solution only in certain settings such as the convex function ϕ is the negative entropy and the constraints are removed (i.e., the unconstrained domains (Sokota et al., 2023)). To address this issue, we propose a novel method to solve the problem (3), which does not rely on the availability of the closed-form solution and hence, can consider more possible options of ϕ. To this end, first, we have the following result:

Proposition 5.1. *Assume that i) $\pi(a) \geq \epsilon$, $\forall a \in \mathcal{A}$, where ϵ is a small positive value and ii) the $\phi(\pi)$ defined on Π can be decomposed to[2] $\phi(\pi) = \sum_{a \in \mathcal{A}} \psi(\pi(a))$ where ψ is some convex function defined on $[0, 1]$. Then, solving the problem (3) can be converted to solve the following equation:*

$$\sum\nolimits_{a \in \mathcal{A}} \pi(a) = \sum\nolimits_{a \in \mathcal{A}} \psi'^{-1}\left(A(a) - \lambda)/B\right) = 1, \quad (4)$$

where $A = Q(\pi_k) + \sum_{\tau=0}^{M-1} \alpha_\tau \phi'(\pi_{k-\tau})$, $B = \sum_{\tau=0}^{M-1} \alpha_\tau$, λ is the dual variable, ψ'^{-1} is the inverse function of ψ' (the derivative of ψ), and $\pi(a) = \psi'^{-1}\left(A(a) - \lambda)/B\right)$.

This result is obtained via the Karush–Kuhn–Tucker (KKT) conditions of the Lagrange function obtained by applying the Lagrange multiplier λ (i.e., the dual variable) to the problem (3). The full derivation can be found in Appendix D.1.

Numerical Method for Computing λ. Now we need to solve Eq. (4) to obtain the value of λ. Unfortunately, this typically cannot be solved *analytically*, rendering it less possible to derive the policy updating rule without the availability of the closed-form solution. To address this issue, we use the Newton method (Ypma, 1995) to compute the value of λ: repeatedly executing $\lambda = \lambda - g(\lambda)/g'(\lambda)$ for $C > 0$ iterations, where $g(\lambda) = \left[\sum_{a \in \mathcal{A}} \psi'^{-1}(A(a) - \lambda)/B\right)\right] - 1$, $g'(\lambda) = \sum_{a \in \mathcal{A}} -\frac{1}{B}[\psi'^{-1}]'(A(a) - \lambda)/B)$, and $[\psi'^{-1}]'$ is the derivative of ψ'^{-1}. The pseudo-code can be found in Algorithm 3 in Appendix D.1.

Projection Operation. After computing the value of λ, we can get the policy $\pi(a)$ by substituting it into the expression of $\pi(a)$ as presented in Eq. (4). Furthermore, we employ a projection operation to ensure that $\pi(a) \geq \epsilon$. Specifically, we have: $\forall a \in \mathcal{A}$, $\pi_{k+1}(a) = \frac{\max\{\epsilon, \pi(a)\}}{\sum_{a' \in \mathcal{A}} \max\{\epsilon, \pi(a')\}}$.

Different Bregman Divergences. In addition to taking multiple historical policies into consideration, GMD[3] further generalizes existing MD algorithms with the capability of exploring a broader class of Bregman divergence as, via the numerical method to compute the value of λ, it is capable of taking more possible convex functions into account. Table 2 presents the functions considered in our work. See (Boyd & Vandenberghe, 2004) for more examples. x^2 (i.e., $n = 2$) and $x \ln x$ respectively correspond to the Euclidean norm and entropy. More details can be found in Appendix D.1.

Table 2. List of convex functions and related functions, $x \in (0, 1]$.

$\psi(x)$	$\psi'(x)$	$\psi'^{-1}(x)$	$[\psi'^{-1}]'(x)$
x^n, $n > 1$	nx^{n-1}	$(\frac{x}{n})^{\frac{1}{n-1}}$	$\frac{1}{n-1}(\frac{x}{n})^{\frac{2-n}{n-1}}$
$x \ln x$	$\ln x + 1$	e^{x-1}	e^{x-1}
$-x^n$, $0 < n < 1$	$-nx^{n-1}$	$(\frac{-x}{n})^{\frac{1}{n-1}}$	$\frac{1}{1-n}(\frac{-x}{n})^{\frac{2-n}{n-1}}$
e^{kx}, $k > 0$	ke^{kx}	$\ln(x/k)/k$	$\frac{1}{x}$

GMD Summary. The pseudo-code of GMD is provided in Algorithm 1. Compared to existing MD algorithms, GMD could satisfy well the desiderata **D1** and **D2** as it not only takes multiple previous policies into account but is also capable of leveraging more possible Bregman divergences that may be better than existing choices such as KL divergence across different decision-making categories.

Algorithm 1 Generalized Mirror Descent (GMD)

```
1: Given $\psi$, initial policy $\pi_1$, $M$, $\boldsymbol{\alpha}$, $\epsilon$
2: for $k = 1, \cdots, K$ do
3:     Compute $A$ and $B$ with $\pi_k$ and $\boldsymbol{\alpha}$
4:     Compute $\lambda$ via Newton method (Algorithm 3)
5:     Compute $\pi(a) = \psi'^{-1}(A(a) - \lambda)/B)$, $\forall a \in \mathcal{A}$
6:     Compute $\pi_{k+1}(a)$ via projection operation, $\forall a \in \mathcal{A}$
7: end for
```

5.2. Meta-Controller for Different Measures

While GMD can satisfy the desiderata **D1** and **D2**, it cannot satisfy well the last two desiderata **D3** and **D4**. The primary reason is that when each agent independently executes GMD, it is not immediately feasible to investigate different solution concepts and evaluation measures as no explicit objective regarding the different measures arises in such a "decentralized" execution process (different measures could lead to different solution concepts and henceforth, we will only focus on the different measures). To address this problem, we propose the configurable mirror descent (CMD) by introducing a meta-controller (MC) to adjust the hyper-parameters in GMD *conditional on the evaluation measures*, which is a "centralized" process from the meta-controller's perspective as it considers all the agents (*joint policy*) to optimize the targeted evaluation measure (and hence, the targeted solution concept), i.e., **D3** and **D4**.

[2]The sum of convex functions is still a convex function. Furthermore, the negative entropy and squared Euclidean norm are two special variants that have been extensively adopted in literature.

[3]The term GMD is also used in (Radhakrishnan et al., 2020), which differs from our method.

Zero-Order Meta-Controller. As shown in Eq. (3), given the number of historical policies $M \geq 1$, the only controllable variable is the hyper-parameters $\boldsymbol{\alpha} = (\alpha_\tau)_{0 \leq \tau \leq M-1}$. At the iteration k, let $\boldsymbol{\pi} = \text{GMD}(\boldsymbol{\alpha})$[4] denote the *joint policy* derived from the previous *joint policy* $\boldsymbol{\pi}_k$ by using GMD with the given $\boldsymbol{\alpha}$, and the performance of this joint policy is denoted as $\mathcal{L}(\boldsymbol{\pi})$. Notably, optimizing $\boldsymbol{\alpha}$, unfortunately, is non-trivial as the evaluation measure $\mathcal{L}$ is non-differentiable with respect to $\boldsymbol{\alpha}$. To address this issue, we construct an efficient zero-order MC by leveraging a zero-order method to optimize $\boldsymbol{\alpha}$. As shown in Figure 2, MC updates $\boldsymbol{\alpha}$ through three steps: i) it *samples* D candidates $\{\boldsymbol{\alpha}^j\}_{j=1}^D$ by perturbing the current $\boldsymbol{\alpha}$ and then derives D new joint policies $\{\boldsymbol{\pi}^j = \text{GMD}(\boldsymbol{\alpha}^j)\}_{j=1}^D$ by employing GMD, ii) it *evaluates* these new joint policies $\{\mathcal{L}(\boldsymbol{\pi}^j)\}_{j=1}^D$, and iii) it *updates* $\boldsymbol{\alpha}$ based on the performance of these new joint policies.

Direction-Guided Update. Let $\boldsymbol{\alpha}^1$ and $\boldsymbol{\alpha}^2$ denote the two candidates sampled by perturbing the current $\boldsymbol{\alpha}$ and the corresponding joint policies $\boldsymbol{\pi}^1$ and $\boldsymbol{\pi}^2$ are obtained via GMD. Existing zero-order methods such as the random search (RS) (Liu et al., 2020) typically update the $\boldsymbol{\alpha}$ directly based on the performance difference between the two candidates $\delta = \mathcal{L}(\boldsymbol{\pi}^1) - \mathcal{L}(\boldsymbol{\pi}^2)$, which could be ineffective as the value of $\mathcal{L}$ could be too small (as shown in our experiments) to derive an effective update. To address this problem, we propose to update $\boldsymbol{\alpha}$ based on the *sign* of the performance difference. Precisely, δ only determines the *update direction*, not the update magnitude, which is more effective when the value of $\mathcal{L}$ is too small. We call this simple yet effective technique the *direction-guided update*. In our experiments, we construct an MC – *direction-guided random search (DRS)* – by applying this method to the existing RS (Wang et al., 2022). More details on different MCs can be found in Appendix D.4.

Algorithm 2 Configurable Mirror Descent (CMD)

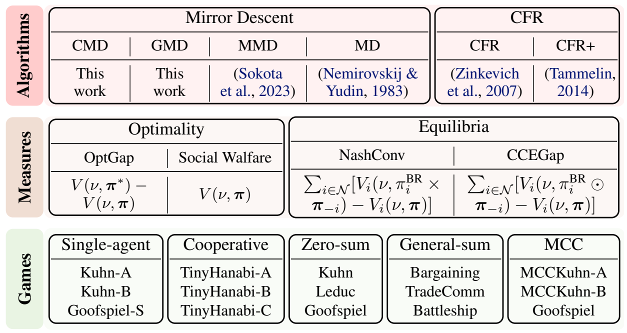
```
1: Given $\mathcal{L}$, $\psi$, initial (joint) policy $\boldsymbol{\pi}_1$, $M$, $D$, $\epsilon$
2: for $k = 1, \cdots, K$ do
3:     Sample $D$ candidates $\{\boldsymbol{\alpha}^j\}_{j=1}^D$
4:     Derive new joint policies $\{\boldsymbol{\pi}^j = \text{GMD}(\boldsymbol{\alpha}^j)\}_{j=1}^D$
5:     Evaluate new joint policies $\{\mathcal{L}(\boldsymbol{\pi}^j)\}_{j=1}^D$
6:     Update $\boldsymbol{\alpha}$ based on $\{\mathcal{L}(\boldsymbol{\pi}^j)\}_{j=1}^D$
7:     Compute $\boldsymbol{\pi}_{k+1}$ via GMD with the updated $\boldsymbol{\alpha}$
8: end for
```

CMD Summary. By incorporating the MC into GMD, we establish the CMD. Intuitively, CMD can be configured to apply to different evaluation measures and hence, can satisfy the desiderata **D3** and **D4** while only minimal modifications are required: specifying the MC's optimization objective $\mathcal{L}$. The pseudo-code of CMD is shown in Algorithm 2.

[4] $\boldsymbol{\alpha}$ is applied to all the agents in multi-agent categories.

6. GameBench

In this section, we present the GameBench, a novel benchmark which consists of 15 games covering all categories of decision making and includes different evaluation measures and different algorithms, which is shown in Figure 3.



Algorithms	Mirror Descent				CFR	
	CMD	GMD	MMD	MD	CFR	CFR+
	This work	This work	(Sokota et al., 2023)	(Nemirovskij & Yudin, 1983)	(Zinkevich et al., 2007)	(Tammelin, 2014)

Measures	Optimality		Equilibria	
	OptGap	Social Walfare	NashConv	CCEGap
	$V(\nu, \boldsymbol{\pi}^*) - V(\nu, \boldsymbol{\pi})$	$V(\nu, \boldsymbol{\pi})$	$\sum_{i \in \mathcal{N}}[V_i(\nu, \pi_i^{\text{BR}} \times \boldsymbol{\pi}_{-i}) - V_i(\nu, \boldsymbol{\pi})]$	$\sum_{i \in \mathcal{N}}[V_i(\nu, \pi_i^{\text{BR}} \odot \boldsymbol{\pi}_{-i}) - V_i(\nu, \boldsymbol{\pi})]$

Games	Single-agent	Cooperative	Zero-sum	General-sum	MCC
	Kuhn-A	TinyHanabi-A	Kuhn	Bargaining	MCCKuhn-A
	Kuhn-B	TinyHanabi-B	Leduc	TradeComm	MCCKuhn-B
	Goofspiel-S	TinyHanabi-C	Goofspiel	Battleship	Goofspiel

Figure 3. Overview of GameBench.

Motivation. Although various benchmarks have been suggested in literature, they are typically specialized for specific decision-making categories, e.g., Atari (Bellemare et al., 2013) for single-agent category, Hanabi (Bard et al., 2020) for cooperative category, Hold'em poker (Brown & Sandholm, 2018; 2019) for competitive category, and football (Kurach et al., 2020) for mixed cooperative and competitive category. On the other hand, as MD algorithms require to execute the policy updating at each decision point at each iteration, running them on the existing benchmarks could be resource-intensive as the number of decision points in the environments could be extremely large (e.g., it is impractical to enumerate the observations in Atari as they are images).

Desiderata. Motivated by the above facts, we construct a new benchmark – GameBench. It satisfies the two desiderata **D5** and **D6** presented in the Introduction. That is, it covers all categories of decision-making problems (comprehensive), and running MD algorithms (or other algorithms such as CFR (Zinkevich et al., 2007)) on all the games does not require much computational resource (academic-friendly). The components of GameBench are given below.

Games. We curate the GameBench on top of the OpenSpiel (Lanctot et al., 2019). There are 15 games which are divided into 5 categories: single-agent, cooperative multi-agent, competitive multi-agent zero-sum, competitive multi-agent general-sum, and mixed cooperative and competitive (MCC) categories. In our GameBench, the original competitive category is further divided into two subcategories – zero-sum and general-sum – as they can involve different solution concepts and evaluation measures (given below). We construct all 15 games under two primary principles: i) these games involve as many aspects of decision making as possible, e.g., the number of agents (single or multiple) and the relationship between agents (cooperative, competitive,

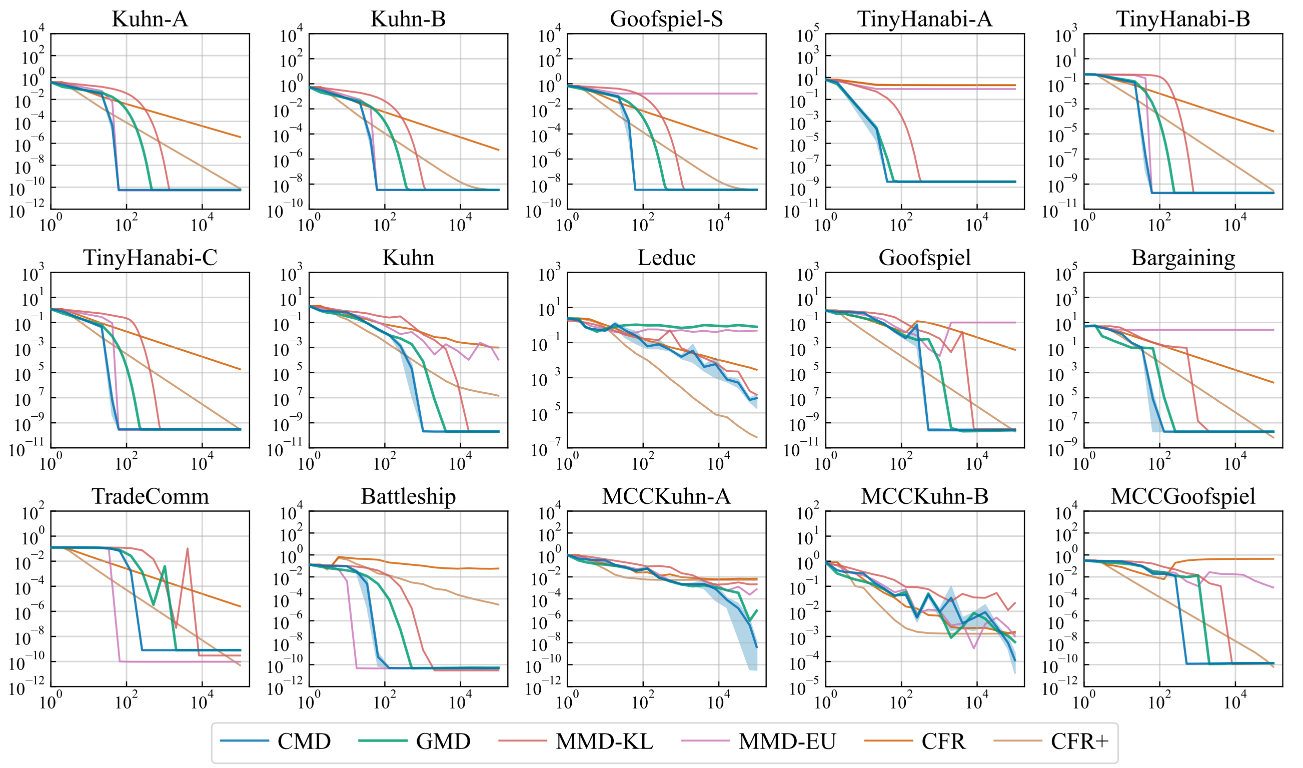


Figure 4. Summary of results. The first 6 figures correspond to single-agent and cooperative categories where the y-axis is *OptGap*. The rest figures correspond to other categories where the y-axis is *NashConv*. For all the figures, the x-axis is the number of iterations.

or mixed), and ii) these games are relatively simple, have a low barrier to entry, and yet complex enough, and hence, running algorithms on these games is less resource-intensive. The details of the constructions and the statistics of the 15 games can be found in Appendix E.2.

Measures. As GameBench includes different categories of decision-making problems, it is indispensable to consider multiple evaluation measures. Roughly speaking, there are two types of measures: i) the notion of *optimality*, including OptGap and social welfare, and ii) the notion of *equilibrium*, including NashConv and CCEGap. Note that computing the equilibrium-type measures for the MCC category requires a new method to compute the team's best response (BR) (not a single agent's). The details can be found in Appendix E.3.

Algorithms. We incorporate different MD algorithms into the GameBench, including the state-of-the-art baselines. The comparison between these MD algorithms can be found in Table 1. In addition, we also include CFR-type algorithms as the baselines, including the CFR (Zinkevich et al., 2007) and CFR+ (Tammelin, 2014). Although these algorithms need to update the policy at each decision point, since the numbers of decision points of the games in GameBench are not too large, running these algorithms on these games is relatively easy (academic-friendly).

7. Experiments

In this section, we evaluate our method on GameBench. We first describe the experimental setup. Then, we present the results by answering several research questions (***RQs***)

Setup. We compare the following methods: i) CMD: our method where the hyper-parameters are determined by the meta-controller introduced in Section 5.2, ii) GMD: the hyper-parameters are fixed to $\alpha_\tau = 1/M, 0 \leq \tau \leq M-1$ (a uniform distribution), iii) MMD-KL: the state-of-the-art method called magnetic mirror descent (Sokota et al., 2023) where the policy updating rule is induced with KL divergence, iv) MMD-EU: similar to MMD-KL but the policy updating rule is induced with squared Euclidean norm (see Appendix D.3 for the derivation), v) CFR: the policy is updated based on regret (Zinkevich et al., 2007), and vi) CFR+: an advanced version of CFR (Tammelin, 2014). In CMD and GMD, we also include a magnet policy, which has been argued desirable (Sokota et al., 2023; Liu et al., 2023). Nevertheless, we note that this does not cause inconsistency with our method as we can equivalently obtain them by setting M and α_τ (see Appendix D.2). Moreover, without explicitly specifying, the results are obtained under $\psi(x) = x\ln x, x \in (0, 1]$. In ***RQ4***, we study more possible Bregman divergences by setting different ψ in Table 2.

7.1. Results and Analysis

***RQ1. (Desiderata D1 and D2)** Can CMD effectively tackle all categories of decision-making problems?* In Figure 4, we show the learning performance of different methods across 15 games. From the results, we can see that, CMD can effectively solve all categories of decision-making problems: in single-agent and cooperative categories, CMD can find the approximate optimal (joint) policy (the OptGap converges to an extremely small value), and in other categories, CMD can find the approximate Nash equilibrium (the NashConv converges to an extremely small value).

***RQ2. (Comparison with Baselines)** How does CMD perform compared with baselines?* As shown in Figure 4, by comparison, we can obtain the following takeaways.

- Incorporating multiple historical policies and dynamically adjusting the hyper-parameters could *accelerate policy learning in terms of the number of iterations*. This can be verified by comparing CMD with MMD-KL where CMD can converge to a similar OptGap or NashConv value with MMD-KL using fewer iterations. This advantage holds even without tuning the hyper-parameters: in most of the games, GMD (the hyper-parameters are fixed) can also converge to a similar OptGap or NashConv value with MMD-KL using fewer iterations. For GMD with different heuristic hyper-parameter adjustment strategies such as linear decay can be found in Appendix F.4.
- Introducing the meta-controller is important as it not only accelerates policy learning but also could *achieve competitive or better outcomes*. Specifically, in terms of NashConv, CMD outperforms the baselines in MCCKuhn-A and MCCKuhn-B and performs on par with the baselines in all other games. However, in the most difficult Leduc poker, GMD cannot effectively decrease the NashConv, showcasing the indispensability of the MC.
- Regarding the CFR-type algorithms, the results similar to previous works (Sokota et al., 2023) are also observed: i) The vanilla CFR is typically inferior to CFR+ and CMD in all the games, and ii) in most of the games, CFR+ outperforms CMD over short iteration horizons but is quickly caught by CMD for longer horizons.

***RQ3. (Different MCs)** Is the direction-guided update in the meta-controller important?* In Figure 5, we compare DRS proposed in Section 5.2 with DGLDS, RS, GLDS, and GLD (see Appendix D.4 for details on these MCs). As we can see, our proposed MC significantly outperforms others either in convergence rate or final performance. In Appendix F.5, we visualize the evolution of the hyper-parameter values during the policy learning, verifying the intuition that our DRS can more efficiently adjust the hyper-parameters.

***RQ4. (Different Bregman Divergences)** How does CMD perform under different Bregman divergences?* In Figure 6,

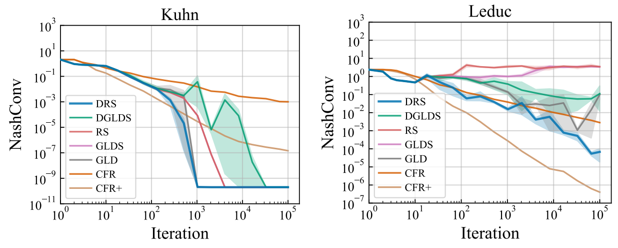


Figure 5. Results for different types of MC.

we show the learning curves of different instances of CMD instantiated with different convex functions. From the results, we can see that the KL divergence ($\psi(x) = x \ln x$), though has been widely adopted, could be not the optimal choice across all the decision-making categories. To our knowledge, CMD is the first algorithm that is endowed with the capability of (empirically) exploring a broader class of Bregman divergence, a prominent feature compared with existing MD methods. See Appendix F.6 for more results.

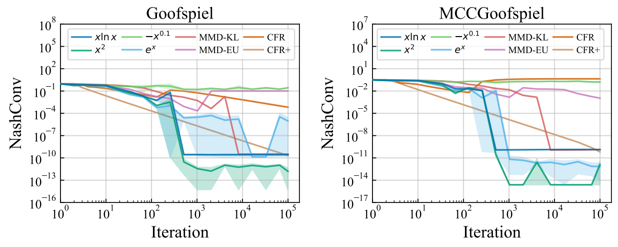


Figure 6. Results for different convex functions.

***RQ5. (Desiderata D3 and D4)** Can CMD generalize to consider different solution concepts and evaluation measures?* In Figure 7, we apply CMD to two different measures: CCE-Gap (top line) and social welfare (bottom line). The results verify the effectiveness of incorporating multiple historical policies and dynamically adjusting the hyper-parameter values conditional on the evaluation measures when considering other evaluation measures beyond NashConv. More results and analysis can be found in Appendix F.8. Notably, our CMD can be easily applied to different measures with minimal modifications: changing the MC's objective $\mathcal{L}$.

***RQ6. (Desiderata D5 and D6)** Is running the different algorithms computationally difficult?* We found that, although extra operations may be required, running the MD and CFR algorithms on GAMEBENCH does not cause much burden on the computational resource. The analysis of the computational complexity can be found in Appendix F.9.

8. Limitations, Future Works, and Conclusions

In this section, we discuss the limitations of the current version of our approach and present the future directions, followed by conclusions of this work.

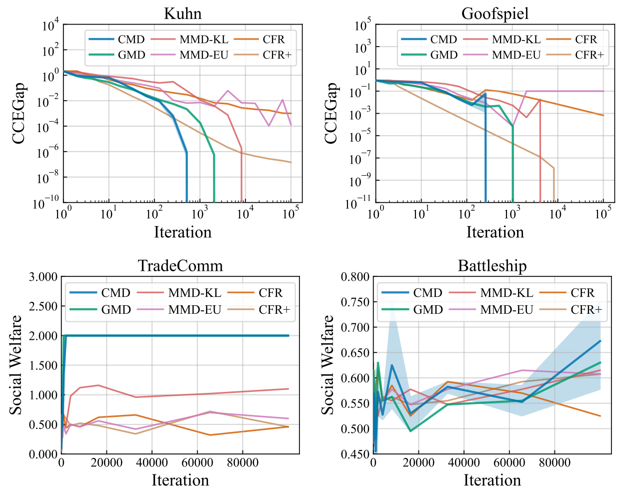


Figure 7. Results for the measures CCEGap and social welfare.

8.1. Limitations and Future Works

This work aims to develop a single algorithm to effectively tackle all categories of decision-making problems: single-agent, cooperative multi-agent, competitive multi-agent, and mixed cooperative and competitive multi-agent categories. As a preliminary attempt, there are still some limitations that are worth investigating in future works.

Firstly, as the meta-controller determines the values of the hyper-parameter by sampling multiple candidates, extra computational cost is needed to evaluate the performance of these candidates. In Appendix F.9, we present the running time of an iteration of different methods. While requiring extra computational cost, we view this as one of the future directions: developing more computationally efficient hyper-parameter value updating methods without sacrificing performance. For example, in contrast to the current sampling method where the historical samples are entirely ignored after each update, these historical samples could be used to guide the selection of the new hyper-parameter values, e.g., via Bayesian optimization (Lindauer et al., 2022) or offline learning approaches (Chen et al., 2022). As a result, we may not need to sample multiple candidates, which could further reduce the extra computational cost.

Secondly, we evaluated CMD primarily from the empirical perspective, and the results demonstrate its promise in solving all categories of decision-making problems. Theoretical analysis of the behavior (e.g., the convergence rate) of CMD could be an interesting problem and may require novel tools that may be different from existing works since the policy updating rule of CMD is established with a numerical method, rather than depends on the closed-form solution to the regularized optimization problem in each decision point under some specific Bregman divergence (Sokota et al., 2023; Liu et al., 2023; Lee et al., 2021).

Thirdly, though our methods are capable of considering a broader class of Bregman divergence, they still require the mathematical formulation of the convex function ψ as computing the value of the dual variable by using the Newton method requires a set of related functions derived from ψ (as shown in Table 2). In other words, the number of Bregman divergences considered in our method is still limited. An interesting future direction is to develop a novel method to more effectively explore the entire space of the convex function, such as using a neural network to represent the convex function ψ, which leads to the neural Bregman divergence (Lu et al., 2023; Siahkamari et al., 2020; Cilingir et al., 2020; Amos et al., 2017). Moreover, using neural Bregman divergence could also be a possible solution for automatically choosing the Bregman divergences for different decision-making categories. Nevertheless, it could be non-trivial to integrate the neural Bregman divergence as training the neural network to well approximate the optimal convex function for the given category may not be easy and thus may require new treatment.

Finally, in its current version, GAMEBENCH consists of 15 academic-friendly games covering all categories of decision-making problems, different evaluation measures, and several baseline algorithms. We believe further extensions could be valuable. i) We could include more games with varying complexity (e.g., different numbers of decision points) (Lanctot et al., 2023). ii) We could include more evaluation measures such as fairness (Rabin, 1993). iii) We could include more algorithms. In particular, we may include deep learning-based algorithms (Schulman et al., 2017; Yu et al., 2022; Lanctot et al., 2017) and investigate whether there exists a single deep learning-based algorithm that can effectively solve all categories of decision-making problems. iv) Recently, much attention has been drawn to studying the ability of large language models (LLMs) to solve various decision-making problems (Hong et al., 2023; Bakhtin et al., 2022; Mao et al., 2023). Therefore, an interesting extension is to include LLMs as the baselines, and if necessary, develop new LLMs to more effectively solve different categories of decision-making problems.

8.2. Conclusions

In this work, we make the preliminary attempt to develop ***a single algorithm*** *to tackle* ***ALL*** *categories of decision-making problems* and provide three contributions: i) the GMD, a generalization of exiting MD algorithms, which can be applied to different decision-making categories involving different numbers of agents and different relationships between agents (**D1** and **D2**), ii) the CMD which can be configured to apply to different solution concepts and evaluation measures (**D3** and **D4**), and iii) the comprehensive (**D5**) and academic-friendly (**D6**) benchmark – GAMEBENCH. Extensive experiments demonstrate the effectiveness of CMD.

Acknowledgements

This work is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. Hau Chan is supported by the National Institute of General Medical Sciences of the National Institutes of Health [P20GM130461], the Rural Drug Addiction Research Center at the University of Nebraska-Lincoln, and the National Science Foundation under grant IIS:RI #2302999. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies. Shuyue Hu is supported by the Shanghai Artificial Intelligence Laboratory.

Impact Statement

This paper presents work whose goal is to advance the field of decision making. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Albrecht, J., Fetterman, A., Fogelman, B., Kitanidis, E., Wróblewski, B., Seo, N., Rosenthal, M., Knutins, M., Polizzi, Z., Simon, J., and Qiu, K. Avalon: A benchmark for RL generalization using procedurally generated worlds. In *NeurIPS Datasets and Benchmarks Track*, pp. 12813–12825, 2022.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *ICML*, pp. 146–155, 2017.

Anagnostides, I., Farina, G., Panageas, I., and Sandholm, T. Optimistic mirror descent either converges to Nash or to strong coarse correlated equilibria in bimatrix games. In *NeurIPS*, pp. 16439–16454, 2022a.

Anagnostides, I., Panageas, I., Farina, G., and Sandholm, T. On last-iterate convergence beyond zero-sum games. In *ICML*, pp. 536–581, 2022b.

Ao, R., Cen, S., and Chi, Y. Asynchronous gradient play in zero-sum multi-agent games. In *ICLR*, 2023.

Aumann, R. J. Correlated equilibrium as an expression of Bayesian rationality. *Econometrica: Journal of the Econometric Society*, 55(1):1–18, 1987.

Bailey, J. P. and Piliouras, G. Multiplicative weights update in zero-sum games. In *EC*, pp. 321–338, 2018.

Bailey, J. P. and Piliouras, G. Fast and furious learning in zero-sum games: Vanishing regret with non-vanishing step sizes. In *NeurIPS*, pp. 12997–13007, 2019.

Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., Jacob, A. P., Komeili, M., Konath, K., Kwon, M., Lerer, A., Lewis, M., Miller, A. H., Mitts, S., Renduchintala, A., Roller, S., Rowe, D., Shi, W., Spisak, J., Wei, A., Wu, D., Zhang, H., and Zijlstra, M. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E., Dunning, I., Mourad, S., Larochelle, H., Bellemare, M. G., and Bowling, M. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, 280:103216, 2020.

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H. P. d. O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Boyd, S. P. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Brown, N. and Sandholm, T. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Brown, N. and Sandholm, T. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019.

Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. Deep Blue. *Artificial Intelligence*, 134(1-2):57–83, 2002.

Carminati, L., Zhang, B. H., Farina, G., Gatti, N., and Sandholm, T. Hidden-role games: Equilibrium concepts and computation. *arXiv preprint arXiv:2308.16017*, 2023.

Cen, S., Chi, Y., Du, S. S., and Xiao, L. Faster last-iterate convergence of policy optimization in zero-sum Markov games. In *ICLR*, 2023.

Chen, Y., Song, X., Lee, C., Wang, Z., Zhang, R., Dohan, D., Kawakami, K., Kochanski, G., Doucet, A., Ranzato, M., et al. Towards learning universal hyperparameter optimizers with transformers. In *NeurIPS*, pp. 32053–32068, 2022.

Cilingir, H. K., Manzelli, R., and Kulis, B. Deep divergence learning. In *ICML*, pp. 2027–2037, 2020.

Davis, O. A. and Whinston, A. Externalities, welfare, and the theory of games. *Journal of Political Economy*, 70(3): 241–262, 1962.

de Witt, C. S., Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P. H., Sun, M., and Whiteson, S. Is independent learning all you need in the StarCraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Fang, Y., Tang, Z., Ren, K., Liu, W., Zhao, L., Bian, J., Li, D., Zhang, W., Yu, Y., and Liu, T.-Y. Learning multi-agent intention-aware communication for optimal multi-order execution in finance. In *KDD*, pp. 4003–4012, 2023.

Farina, G., Bianchi, T., and Sandholm, T. Coarse correlation in extensive-form games. In *AAAI*, pp. 1934–1941, 2020.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *AAAI*, pp. 2974–2982, 2018.

Foerster, J., Song, F., Hughes, E., Burch, N., Dunning, I., Whiteson, S., Botvinick, M., and Bowling, M. Bayesian action decoder for deep multi-agent reinforcement learning. In *ICML*, pp. 1942–1951, 2019.

Golowich, N., Pattathil, S., and Daskalakis, C. Tight last-iterate convergence rates for no-regret learning in multi-player games. In *NeurIPS*, pp. 20766–20778, 2020.

Harsanyi, J. C., Selten, R., et al. *A General Theory of Equilibrium Selection in Games*. The MIT Press, 1988.

Hong, S., Zheng, X., Chen, J., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and Schmidhuber, J. MetaGPT: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Hsieh, Y.-G., Antonakopoulos, K., and Mertikopoulos, P. Adaptive learning in continuous games: Optimal regret bounds and convergence to Nash equilibrium. In *COLT*, pp. 2388–2422, 2021.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. In *ICML*, pp. 2137–2146, 2018.

Jain, R., Piliouras, G., and Sim, R. Matrix multiplicative weights updates in quantum zero-sum games: Conservation laws & recurrence. In *NeurIPS*, pp. 4123–4135, 2022.

Kangarshahi, E. A., Hsieh, Y.-P., Sahin, M. F., and Cevher, V. Let's be honest: An optimal no-regret framework for zero-sum games. In *ICML*, pp. 2488–2496, 2018.

Kozuno, T., Ménard, P., Munos, R., and Valko, M. Model-free learning for two-player zero-sum partially observable Markov games with perfect recall. In *NeurIPS*, pp. 11987–11998, 2021.

Kuhn, H. W. A simplified two-person poker. *Contributions to the Theory of Games*, 1:97–103, 1950.

Kurach, K., Raichuk, A., Stańczyk, P., Zając, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., and Gelly, S. Google research football: A novel reinforcement learning environment. In *AAAI*, pp. 4501–4510, 2020.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. In *NeurIPS*, pp. 4190–4203, 2017.

Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., Hennes, D., Morrill, D., Muller, P., Ewalds, T., Faulkner, R., Kramár, J., De Vylder, B., Saeta, B., Bradbury, J., Ding, D., Borgeaud, S., Lai, M., Schrittwieser, J., Anthony, T., Hughes, E., Danihelka, I., and Ryan-Davis, J. OpenSpiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

Lanctot, M., Schultz, J., Burch, N., Smith, M. O., Hennes, D., Anthony, T., and Perolat, J. Population-based evaluation in repeated rock-paper-scissors as a benchmark for multiagent reinforcement learning. *TMLR*, 2023. ISSN 2835-8856.

Lee, C.-W., Kroer, C., and Luo, H. Last-iterate convergence in extensive-form games. In *NeurIPS*, pp. 14293–14305, 2021.

Lewis, M., Yarats, D., Dauphin, Y., Parikh, D., and Batra, D. Deal or no deal? End-to-end learning of negotiation dialogues. In *EMNLP*, pp. 2443–2453, 2017.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *The Journal of Machine Learning Research*, 23(1):2475–2483, 2022.

Liu, M., Ozdaglar, A. E., Yu, T., and Zhang, K. The power of regularization in solving extensive-form games. In *ICLR*, 2023.

Liu, S., Chen, P.-Y., Kailkhura, B., Zhang, G., Hero III, A. O., and Varshney, P. K. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020.

Liu, S., Lever, G., Wang, Z., Merel, J., Eslami, S. A., Hennes, D., Czarnecki, W. M., Tassa, Y., Omidshafiei, S., Abdolmaleki, A., et al. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69): eabo0235, 2022a.

Liu, W., Jiang, H., Li, B., and Li, H. Equivalence analysis between counterfactual regret minimization and online mirror descent. In *ICML*, pp. 13717–13745, 2022b.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*, pp. 6382–6393, 2017.

Lu, F., Raff, E., and Ferraro, F. Neural Bregman divergences for distance learning. In *ICLR*, 2023.

Mao, S., Cai, Y., Xia, Y., Wu, W., Wang, X., Wang, F., Ge, T., and Wei, F. Alympics: Language agents meet game theory – exploring strategic decision-making with AI agents. *arXiv preprint arXiv:2311.03220*, 2023.

Marris, L., Muller, P., Lanctot, M., Tuyls, K., and Graepel, T. Multi-agent training beyond zero-sum with correlated equilibrium meta-solvers. In *ICML*, pp. 7480–7491, 2021.

Mertikopoulos, P., Lecouat, B., Zenati, H., Foo, C.-S., Chandrasekhar, V., and Piliouras, G. Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. In *ICLR*, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Moulin, H. and Vial, J. P. Strategically zero-sum games: The class of games whose completely mixed equilibria cannot be improved upon. *International Journal of Game Theory*, 7:201–221, 1978.

Nash, J. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

Nemirovskij, A. S. and Yudin, D. B. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience, 1983.

Oliehoek, F. A. and Amato, C. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

Perolat, J., De Vylder, B., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., McAleer, S., Elie, R., Cen, S. H., Wang, Z., Gruslys, A., Malysheva, A., Khan, M., Ozair, S., Timbers, F., Pohlen, T., Eccles, T., Rowland, M., Lanctot, M., Lespiau, J.-B., Piot, B., Omidshafiei, S., Lockhart, E., Sifre, L., Beauguerlange, N., Munos, R., Silver, D., Singh, S., Hassabis, D., and Tuyls, K. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

Rabin, M. Incorporating fairness into game theory and economics. *The American Economic Review*, 83(5):1281–1302, 1993.

Radhakrishnan, A., Belkin, M., and Uhler, C. Linear convergence of generalized mirror descent with time-dependent mirrors. *arXiv preprint arXiv:2009.08574*, 2020.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, pp. 4295–4304, 2018.

Rashid, T., Farquhar, G., Peng, B., and Whiteson, S. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *NeurIPS*, pp. 10199–10210, 2020.

Rizk, Y., Awad, M., and Tunstel, E. W. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys*, 52(2):1–31, 2019.

Ross, S. M. Goofspiel–the game of pure strategy. *Journal of Applied Probability*, 8(3):621–625, 1971.

Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The StarCraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Schmid, M., Moravcik, M., Burch, N., Kadlec, R., Davidson, J., Waugh, K., Bard, N., Timbers, F., Lanctot, M., Zacharias Holland, G., Davoodi, E., Christianson, A., and Bowling, M. Student of games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256, 2023.

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

Serrino, J., Kleiman-Weiner, M., Parkes, D. C., and Tenenbaum, J. Finding friend and foe in multi-agent games. In *NeurIPS*, pp. 1251–1261, 2019.

Shoham, Y. and Leyton-Brown, K. *Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations*. Cambridge University Press, 2008.

Siahkamari, A., XIA, X., Saligrama, V., Castañón, D., and Kulis, B. Learning to approximate a Bregman divergence. In *NeurIPS*, pp. 3603–3612, 2020.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Singh, B., Kumar, R., and Singh, V. P. Reinforcement learning in robotic applications: A comprehensive survey. *Artificial Intelligence Review*, 55(2):945–990, 2022.

Sokota, S., Lockhart, E., Timbers, F., Davoodi, E., D'Orazio, R., Burch, N., Schmid, M., Bowling, M., and Lanctot, M. Solving common-payoff games with approximate policy iteration. In *AAAI*, pp. 9695–9703, 2021.

Sokota, S., D'Orazio, R., Kolter, J. Z., Loizou, N., Lanctot, M., Mitliagkas, I., Brown, N., and Kroer, C. A unified approach to reinforcement learning, quantal response equilibria, and two-player zero-sum games. In *ICLR*, 2023.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*, pp. 5887–5896, 2019.

Song, X., Gao, W., Yang, Y., Choromanski, K., Pacchiano, A., and Tang, Y. ES-MAML: Simple Hessian-free meta learning. In *ICLR*, 2020.

Su, H., Zhong, Y. D., Dey, B., and Chakraborty, A. Emvlight: A decentralized reinforcement learning framework for efficient passage of emergency vehicles. In *AAAI*, pp. 4593–4601, 2022.

Sun, M., Devlin, S., Beck, J., Hofmann, K., and Whiteson, S. Trust region bounds for decentralized PPO under non-stationarity. In *AAMAS*, pp. 5–13, 2023a.

Sun, S., Wang, R., and An, B. Reinforcement learning for quantitative trading. *ACM Transactions on Intelligent Systems and Technology*, 14(3):1–29, 2023b.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Tammelin, O. Solving large imperfect information games using CFR+. *arXiv preprint arXiv:1407.5042*, 2014.

Tesauro, G. et al. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Tomar, M., Shani, L., Efroni, Y., and Ghavamzadeh, M. Mirror descent policy optimization. In *ICLR*, 2022.

Tsai, Y.-Y., Chen, P.-Y., and Ho, T.-Y. Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. In *ICML*, pp. 9614–9624, 2020.

v. Neumann, J. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

Vural, N. M., Yu, L., Balasubramanian, K., Volgushev, S., and Erdogdu, M. A. Mirror descent strikes again: Optimal stochastic convex optimization under infinite noise variance. In *COLT*, pp. 65–102, 2022.

Wang, J., Ren, Z., Liu, T., Yu, Y., and Zhang, C. QPLEX: Duplex dueling multi-agent Q-learning. In *ICLR*, 2021a.

Wang, J., Xu, W., Gu, Y., Song, W., and Green, T. C. Multi-agent reinforcement learning for active voltage control on power distribution networks. In *NeurIPS*, pp. 3271–3284, 2021b.

Wang, T. and Kaneko, T. Application of deep reinforcement learning in werewolf game agents. In *TAAI*, pp. 28–33, 2018.

Wang, X., Guo, W., Su, J., Yang, X., and Yan, J. ZARTS: On zero-order optimization for neural architecture search. In *NeurIPS*, pp. 12868–12880, 2022.

Wibisono, A., Tao, M., and Piliouras, G. Alternating mirror descent for constrained min-max games. In *NeurIPS*, pp. 35201–35212, 2022.

Xu, B., Wang, Y., Wang, Z., Jia, H., and Lu, Z. Hierarchically and cooperatively learning traffic signal control. In *AAAI*, pp. 669–677, 2021.

Xu, Z., Liang, Y., Yu, C., Wang, Y., and Wu, Y. Fictitious cross-play: Learning global Nash equilibrium in mixed cooperative-competitive games. In *AAMAS*, pp. 1053–1061, 2023.

Ypma, T. J. Historical development of the Newton-Raphson method. *SIAM Review*, 37(4):531–551, 1995.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of PPO in cooperative multi-agent games. In *NeurIPS Datasets and Benchmarks Track*, pp. 24611–24624, 2022.

Zahavy, T., Xu, Z., Veeriah, V., Hessel, M., Oh, J., van Hasselt, H., Silver, D., and Singh, S. A self-tuning actor-critic algorithm. In *NeurIPS*, pp. 20913–20924, 2020.

Zhou, Z., Mertikopoulos, P., Athey, S., Bambos, N., Glynn, P. W., and Ye, Y. Learning in games with lossy feedback. In *NeurIPS*, pp. 5134–5144, 2018.

Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. Regret minimization in games with incomplete information. In *NeurIPS*, pp. 1729–1736, 2007.

A. Code Repository

Code for experiments is available at `https://github.com/IpadLi/CMD`.

B. More Related Works

Single-Agent Category. In the single-agent category, reinforcement learning (RL) (Sutton & Barto, 2018) has proved successful in many real-world applications. The power of RL is further amplified with the integration of deep neural networks, leading to various deep RL algorithms that have been successfully applied to various application domains such as video games (Mnih et al., 2015), robot navigation (Singh et al., 2022), and financial technology (Sun et al., 2023b). Among these algorithms, PPO (Schulman et al., 2017) is one of the most commonly used methods to solve single-agent RL problems. Recent works have shown that independent PPO (de Witt et al., 2020; Sun et al., 2023a) can effectively solve single-agent and cooperative multi-agent RL problems. In addition, a variant of PPO is also shown to be effective in solving two-player zero-sum games when both players adopt this algorithm (Sokota et al., 2023). Nevertheless, it remains elusive whether these single-agent algorithms can be applied to solve other categories of decision-making problems which may involve different properties including different numbers of agents, different relationships between agents, different solution concepts, and different evaluation measures. In this work, we aim to develop a single algorithm that, when executed by each agent, provides an effective approach to address different categories of decision-making problems.

Cooperative Multi-Agent Category. Cooperative multi-agent RL (MARL) has been demonstrated successful in solving many real-world cooperative tasks such as traffic signal control (Xu et al., 2021; Su et al., 2022), power management (Wang et al., 2021b), finance (Fang et al., 2023), and multi-robot cooperation (Rizk et al., 2019). In the past decade, a variety of MARL algorithms, e.g., QMIX (Rashid et al., 2018) and its variants (Son et al., 2019; Rashid et al., 2020; Wang et al., 2021a), MADDPG (Lowe et al., 2017), COMA (Foerster et al., 2018), and MAPPO (Yu et al., 2022), to name just a few, have been proposed and achieved significant performance in various multi-agent benchmarks, e.g., SMAC (Samvelyan et al., 2019) and Dota II (Berner et al., 2019). These algorithms typically follow the principle of centralized training and decentralized execution (CTDE) where global information is only available during training. Despite their success, they cannot be directly applied to competitive and mixed cooperative and competitive categories. In this work, our proposed CMD can be applied to different decision-making categories and share similarities with the CTDE paradigm: the meta-controller takes all the agents (i.e., the joint policy) into account to optimize the hyper-parameters conditional on the targeted evaluation measure (a "centralized" process) while each agent in the environment independently execute the GMD with the given hyper-parameters to update the policy (a "decentralized" process).

Competitive Multi-Agent Category. There has long been a history of researchers pursuing artificial intelligence (AI) agents that can achieve human-level or super-human-level performance in solving various competitive multi-agent games such as chess (Campbell et al., 2002), Go (Silver et al., 2017), poker (Brown & Sandholm, 2019), and Stratego (Perolat et al., 2022). Due to the competitive nature, the development of learning algorithms for solving these games is typically largely different from single-agent and cooperative MARL. Among others, counterfactual regret minimization (CFR) (Zinkevich et al., 2007) and policy-space response oracles (PSRO) (Lanctot et al., 2017) are two representative algorithms that have been widely used to solve complex games (Schmid et al., 2023). Another category of algorithm that has drawn increasing attention recently is the mirror descent (MD) (Nemirovskij & Yudin, 1983; Beck & Teboulle, 2003). In contrast to CFR and PSRO which are "average-iterate" algorithms, MD has proved the "last-iterate" convergence property in solving two-player zero-sum games (Bailey & Piliouras, 2018; Kangarshahi et al., 2018; Wibisono et al., 2022; Kozuno et al., 2021; Lee et al., 2021; Jain et al., 2022; Ao et al., 2023; Liu et al., 2023; Cen et al., 2023; Sokota et al., 2023) and some classes of general-sum games (Anagnostides et al., 2022b). Moreover, MD has also been demonstrated effective in solving single-agent RL problems (Tomar et al., 2022). Despite their success, existing MD algorithms typically focus on some specific Bregman divergences which may not be the optimal choices across different decision-making categories. Our proposed CMD generalizes existing MD algorithms to consider a broader class of Bregman divergence, which could achieve better learning performance in addressing different categories of decision-making problems.

Mixed Cooperative and Competitive Category. In some real-world scenarios, the relationship between agents could be neither purely cooperative nor purely competitive. For example, in a football game, the agents belonging to the same team need to cooperate while also competing with the other team (Kurach et al., 2020). In hidden-role games (Carminati et al., 2023), each agent tries to identify their (unknown) teammates and compete with other (unknown) adversaries (Wang & Kaneko, 2018; Serrino et al., 2019; Albrecht et al., 2022). However, in contrast to the other three categories (single-agent, purely cooperative, and purely competitive), mixed cooperative and competitive (MCC) games are largely unstudied (Xu et al., 2023). Furthermore, as MD algorithms typically require updating the policy at each decision point, running them on the current benchmark games such as football (Kurach et al., 2020) could be computationally prohibited. In the present work, we construct 3 MCC games that are academic-friendly – their numbers of decision points are not too large and hence, running MD algorithms (and other algorithms such as CFR-type (Zinkevich et al., 2007; Tammelin, 2014)) on these games does not require much computational resource (e.g., running time and memory usage).

Hyper-Parameter Tuning. Existing works typically determine the hyper-parameter values of the MD algorithms depending on the domain knowledge (Sokota et al., 2023; Anagnostides et al., 2022b; Hsieh et al., 2021; Zhou et al., 2018; Mertikopoulos et al., 2019; Bailey & Piliouras, 2019; Golowich et al., 2020), which, though convenient for theoretical analysis, may not be easy to generalize to different games. On the other hand, as the evaluation measures, e.g., NashConv, could be non-differentiable with respect to the hyper-parameters, the gradient-based methods such as STAC (Zahavy et al., 2020) could also be less applicable. In this sense, a more feasible method is the zero-order hyper-parameter optimization which can update the parameters of interest without access to the true gradient, which has been extensively adopted in the adversarial robustness of deep neural networks (Ilyas et al., 2018), meta-learning (Song et al., 2020), transfer learning (Tsai et al., 2020), and neural architecture search (NSA) (Wang et al., 2022). Nevertheless, we found that directly applying existing zero-order methods could be ineffective as when the value of the evaluation measure is too small, they may not be able to derive an effective update for the hyper-parameter. To address this issue, we propose a simple yet effective technique – *direction-guided update* – where the performance difference between two candidates is used to only determine the *update direction* of the hyper-parameters rather than the update magnitude, which is more effective than existing methods (Wang et al., 2022) when the value of the performance is extremely small.

C. Notation Table

Table 3. Notation Table.

$\mathcal{N}$	$\mathcal{N} = \{1, \cdots, N\}$, the set of N agents.
$\mathcal{S}$	the finite set of states.
$\mathcal{A}$	$\mathcal{A} = \times_{i\in\mathcal{N}}\mathcal{A}_i$ where $\mathcal{A}_i$ is the finite set of actions of agent i.
$\mathcal{O}$	$\mathcal{O} = \times_{i\in\mathcal{N}}\mathcal{O}_i$ where $\mathcal{O}_i$ is the finite set of observations of agent i.
Ω	$\Omega = \times_{i\in\mathcal{N}}\Omega_i$ where $\Omega_i : \mathcal{S} \times \mathcal{A} \to \mathcal{O}_i$ is the observation function of agent i.
P	$P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, the state transition function.
R	$R = \{r_i\}_{i\in\mathcal{N}}$ where $r_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function of agent i.
γ	$\gamma \in [0, 1)$, the discount factor.
ν	$\nu \in \Delta(\mathcal{S})$, the initial state distribution.
τ_i^t	the decision point (action-observation history) of agent i at time t, $\tau_i^t \in \mathcal{T}_i^t$.
$\mathcal{T}_i^t$	$\mathcal{T}_i^t = (\mathcal{O}_i \times \mathcal{A}_i)^t$, the space of decision points of agent i at time step t.
Π	$\Pi = \times_{i\in\mathcal{N}}\Pi_i$ where Π_i is the policy space of agent i.
$\boldsymbol{\pi}$	$\boldsymbol{\pi} = \pi_1 \odot \cdots \odot \pi_N$, the joint policy, $\boldsymbol{\pi} = \pi_1 \times \cdots \times \pi_N$, the product policy.
$V_i(s, \boldsymbol{\pi})$, $V_i(\nu, \boldsymbol{\pi})$	the value functions of agent i, $V_i(\nu, \boldsymbol{\pi}) := \mathbb{E}_{s\sim\nu}[V_i(s, \boldsymbol{\pi})]$.
$\mathcal{L}(\boldsymbol{\pi})$	the evaluation measure of the joint policy $\boldsymbol{\pi}$.
π_k	the single agent's policy at the iteration k of an algorithm.
$\boldsymbol{\pi}_k$	the joint policy at the iteration k of an algorithm.
$Q(\pi_k)$	$Q(\pi_k) = (Q(a, \pi_k))_{a\in\mathcal{A}}$, the action-value vector of a single agent induced by π_k.
$\mathcal{B}_\phi(x; y)$	$\mathcal{B}_\phi(x; y) = \phi(x) - \phi(y) - \langle\nabla\phi(y), x - y\rangle$, the Bregman divergence with respect to ϕ.
K	the number of iterations of an algorithm.
M	$M \geq 1$, the number of historical policies.
$\boldsymbol{\alpha}$	$\boldsymbol{\alpha} = (\alpha_\tau)_{0\leq\tau\leq M-1}$, α_τ is the regularization intensity of $\pi_{k-\tau}$.
ϵ	$\epsilon > 0$, the smallest probability of an action.
$\phi(\pi)$	$\phi(\pi) = \sum_{a\in\mathcal{A}} \psi(\pi(a))$, ψ is some convex function defined on $[0, 1]$.
λ, $\boldsymbol{\beta}$	$\boldsymbol{\beta} = (\beta_a)_{a\in\mathcal{A}}$, the dual variables.
A, B	$A = Q(\pi_k) + \sum_{\tau=0}^{M-1} \alpha_\tau\phi'(\pi_{k-\tau})$, $B = \sum_{\tau=0}^{M-1} \alpha_\tau$, where ϕ' is the derivative of ϕ.
ψ'^{-1}	the inverse function of ψ' (the derivative of ψ).
C	$C > 0$, the number of iterations for the Newton method.
$g(\lambda)$, $g'(\lambda)$	$g(\lambda) = \left[\sum_{a\in\mathcal{A}} \psi'^{-1}(A(a) - \lambda)/B)\right] - 1$, $g'(\lambda) = \sum_{a\in\mathcal{A}} -\frac{1}{B}[\psi'^{-1}]'(A(a) - \lambda)/B)$.
$[\psi'^{-1}]'$	the derivative of ψ'^{-1}.
D	the number of sampled candidate $\boldsymbol{\alpha}$'s.
$\{\boldsymbol{\alpha}^j\}_{j=1}^D$	D candidates by perturbing the current $\boldsymbol{\alpha}$.
$\{\boldsymbol{\pi}^j\}_{j=1}^D$	$\{\boldsymbol{\pi}^j = \text{GMD}(\boldsymbol{\alpha}^j)\}_{j=1}^D$, D new joint policies derived via GMD.
μ	the smoothing parameter in DRS and RS.
$[r_L, r_H]$	the interval of the radiuses of the spheres in DGLDS, GLDS, and GLD.
$\{\boldsymbol{u}^j\}_{j=1}^D$	D candidate updates sampled from a spherically symmetric distribution $\boldsymbol{u}^j \sim q$.
$\boldsymbol{\alpha}_+^j$, $\boldsymbol{\alpha}_-^j$	$\boldsymbol{\alpha}_+^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \mu\boldsymbol{u}^j)$, $\boldsymbol{\alpha}_-^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} - \mu\boldsymbol{u}^j)$, the candidates by perturbing the current $\boldsymbol{\alpha}$.
$\boldsymbol{\pi}_+^j$, $\boldsymbol{\pi}_-^j$	$\boldsymbol{\pi}_+^j = \text{GMD}(\boldsymbol{\alpha}_+^j)$, $\boldsymbol{\pi}_-^j = \text{GMD}(\boldsymbol{\alpha}_-^j)$, the new joint policies obtained via GMD.
δ^j	$\delta^j = \mathcal{L}(\boldsymbol{\pi}_+^j) - \mathcal{L}(\boldsymbol{\pi}_-^j)$, the performance difference between $\boldsymbol{\pi}_+^j$ and $\boldsymbol{\pi}_-^j$.
$\boldsymbol{u}^*$	the final update.
Sgn	$\text{Sgn}(z) = 1$ if $z > 0$, $\text{Sgn}(z) = -1$ if $z < 0$, otherwise, $\text{Sgn}(z) = 0$.
CLIP_ι^1	$\text{CLIP}_\iota^1(z) = \iota$ if $z < \iota$, $\text{CLIP}_\iota^1(z) = 1$ if $z > 1$, otherwise, $\text{CLIP}_\iota^1(z) = z$, where $0 < \iota < 1$.
κ	$\kappa \geq 1$, update the $\boldsymbol{\alpha}$ every κ iterations.
ρ	the magnet policy in MMD.
ξ	$\xi > 0$, the regularization intensity of the magnet policy.
η	$\eta > 0$, the step size in MMD.
$\tilde{\eta}$	$\tilde{\eta} > 0$, the step size of the magnet policy in MMD.

D. Configurable Mirror Descent

In this section, we present all the details of our methods. In Section D.1, we present the details of GMD. In Section D.2, we establish some connections between GMD and existing MD algorithms. In Section D.3, we restrict the convex function ϕ to the squared Euclidean norm and derive the closed-form solution under the MMD policy updating rule. Finally, in Section D.4, we present the details of different meta-controllers.

D.1. Generalized Mirror Descent

In this section, we present the proof of Proposition 5.1, the pseudo-code of Newton's method for computing the value of the dual variable, and the convex functions considered in our work.

Proof of Proposition 5.1. Consider the optimization problem (3). By the definition of Bregman divergence, we have:

$$\pi_{k+1} = \arg\max_{\pi\in\Pi}\langle Q(\pi_k), \pi\rangle - \sum\nolimits_{\tau=0}^{M-1} \alpha_\tau \mathcal{B}_\phi(\pi, \pi_{k-\tau}), \tag{5}$$

$$\Rightarrow \pi_{k+1} = \arg\max_{\pi\in\Pi}\langle Q(\pi_k), \pi\rangle - \sum\nolimits_{\tau=0}^{M-1} \alpha_\tau [\phi(\pi) - \phi(\pi_{k-\tau}) - \langle \phi'(\pi_{k-\tau}), \pi - \pi_{k-\tau}\rangle], \tag{6}$$

$$\Rightarrow \pi_{k+1} = \arg\max_{\pi\in\Pi}\langle Q(\pi_k) + \sum\nolimits_{\tau=0}^{M-1} \alpha_\tau \phi'(\pi_{k-\tau}), \pi\rangle - \phi(\pi)\sum\nolimits_{\tau=0}^{M-1} \alpha_\tau + \text{const.}, \tag{7}$$

where "const." summarizes all terms that are irrelevant to π. Let $A = Q(\pi_k) + \sum_{\tau=0}^{M-1} \alpha_\tau \phi'(\pi_{k-\tau})$ and $B = \sum_{\tau=0}^{M-1} \alpha_\tau$ which are fixed at the current iteration k. Then, we can convert Eq. (3) to the following optimization problem:

$$\begin{aligned} &\pi_{k+1} = \arg\max_{\pi\in\Pi}\langle A, \pi\rangle - B\phi(\pi) + \text{const.}, \\ &s.t. \sum\nolimits_{a\in\mathcal{A}} \pi_k(a) = 1 \text{ and } \pi_k(a) \geq 0, \forall a \in \mathcal{A}. \end{aligned} \tag{8}$$

To solve the constrained optimization problem (8), we can apply the Lagrange multiplier, which gives us:

$$L(\pi, \lambda, \boldsymbol{\beta}) = \langle A, \pi\rangle - B\phi(\pi) + \text{const.} - \lambda\left(\sum\nolimits_{a\in\mathcal{A}} \pi(a) - 1\right) + \sum\nolimits_{a\in\mathcal{A}} \beta_a \pi(a), \tag{9}$$

where λ and $\boldsymbol{\beta} = (\beta_a)_{a\in\mathcal{A}}$ are the dual variables. For such problems, we can get the Karush–Kuhn–Tucker (KKT) conditions for each component (action) $a \in \mathcal{A}$ as follows:

$$A(a) + B\phi'(\pi)(a) - \lambda + \beta_a = 0, \tag{10a}$$

$$\sum\nolimits_{a\in\mathcal{A}} \pi(a) = 1, \tag{10b}$$

$$\beta_a \pi(a) = 0, \tag{10c}$$

$$\pi(a) \geq 0, \beta_a \geq 0. \tag{10d}$$

Then the problem is to find a policy π such that it satisfies all the above conditions, which could be difficult owing to two reasons: i) it simultaneously involves the two dual variables λ and β_a, and ii) in Eq. (10a), computing the value $\phi'(\pi)(a)$ involves all the components (actions) as ϕ is defined on the policy π, not the individual component (action) $a \in \mathcal{A}$.

To address the challenges, we apply the two conditions: $\pi(a) \geq \epsilon$ and $\phi(\pi) = \sum_{a\in\mathcal{A}} \psi(\pi(a))$. Then, we have $\phi'(\pi)(a) = \psi'(\pi(a))$. As a result, the problem (10a–10d) is simplified to the following problem:

$$A(a) + B\psi'(\pi(a)) - \lambda = 0, \tag{11a}$$

$$\sum\nolimits_{a\in\mathcal{A}} \pi(a) = 1, \tag{11b}$$

$$\pi(a) \geq \epsilon. \tag{11c}$$

From Eq. (11a), we can get that: $\forall a \in \mathcal{A}$,

$$\pi(a) = \psi'^{-1}\left(\frac{A(a) - \lambda}{B}\right), \tag{12}$$

where ψ'^{-1} is the inverse function of ψ'. Substituting the above expression for $\pi(a)$ into the constraint (10b), we have:

$$\sum\nolimits_{a\in\mathcal{A}} \pi(a) = \sum\nolimits_{a\in\mathcal{A}} \psi'^{-1}\left(\frac{A(a)-\lambda}{B}\right) = 1, \tag{13}$$

which completes the proof. □

Numerical Method for Computing λ. Notably, Eq. (13) typically cannot be solved *analytically*. To address this problem, we propose to use a numerical method to compute the value of λ, offering the possibility of exploring a broader class of Bregman divergence. Specifically, for any convex function ψ, we employ the Newton method (Ypma, 1995) to compute the value of λ, which is shown in Algorithm 3, where C is the number of iterations.

Algorithm 3 Newton method for computing the value of λ

```
1: Given $\psi$, $A$, and $B$. Randomly initialize the value of $\lambda$
2: for $C$ iterations do
3:     $\lambda = \lambda - \frac{g(\lambda)}{g'(\lambda)}$
4: end for
```

Different Bregman Divergences. In Table 2, we list the convex functions considered in our work. To be more intuitive, we plot these convex functions in Figure 8.

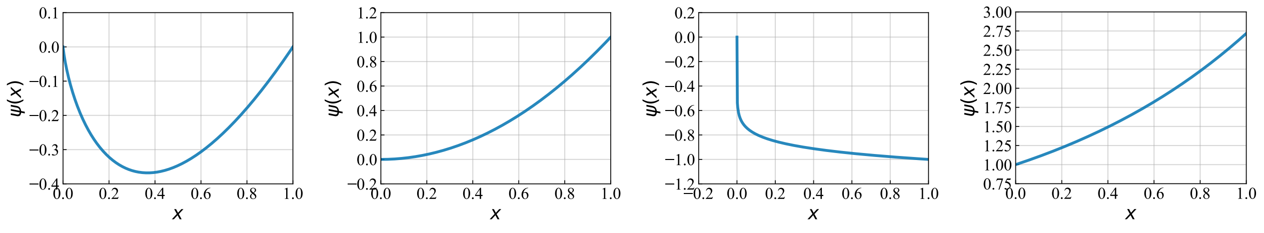

(a) $x \ln x$ (b) x^2 (c) $-x^{0.1}$ (d) e^x

Figure 8. Plots for different convex functions ψ.

D.2. Connection Between GMD and Existing MD Algorithms

In this section, we present some discussion on the connection between GMD and existing MD algorithms. In Table 4, we present the conditions for converting GMD to different MD algorithms and their formulations.

Table 4. Connection between GMD and existing MD algorithms.

Method	Conditions	Formulation
MD	$M = 1, \alpha_0 \in (0, 1]$	$\pi_{k+1} = \arg\max_{\pi\in\Pi}\langle Q(\pi_k), \pi\rangle - \alpha_0\mathcal{B}_\phi(\pi, \pi_k)$
MMD	$M = k$, $\alpha_{k-1}, \alpha_0 \in (0, 1]$, $\alpha_\tau = 0, 0 < \tau < k-1$	$\pi_{k+1} = \arg\max_{\pi\in\Pi}\langle Q(\pi_k), \pi\rangle - \alpha_{k-1}\mathcal{B}_\phi(\pi, \pi_1) - \alpha_0\mathcal{B}_\phi(\pi, \pi_k)$

GMD $\rightarrow$ MD. It is trivial to get the MD algorithm (Nemirovskij & Yudin, 1983; Beck & Teboulle, 2003) by setting $M = 1$ and $\alpha_0 > 0$, that is, MD only considers the current policy π_k when deriving the new policy π_{k+1}.

GMD $\rightarrow$ MMD. To obtain MMD (Sokota et al., 2023), we can set $M = k$ and then only let α_{k-1} and α_0 to be positive, while all other terms are 0. That is, MMD considers two previous policies – the initial policy π_1 and the current policy π_k – when deriving the new policy. In the MMD's terminology, the initial policy π_1 serves as the magnet policy.

In practice, we can get more variants by setting the M and $\boldsymbol{\alpha}$, which shows that GMD is a general method. For example, we can consider both $M < k$ previous policies and the initial policy π_1 (i.e., adding a magnet policy) when deriving the new policy π_{k+1}, which is taken as the default choice for instantiating the GMD in our experiments, that is,

$$\pi_{k+1} = \arg\max_{\pi\in\Pi} \langle Q(\pi_k), \pi\rangle - \alpha_{k-1}\mathcal{B}_\phi(\pi, \pi_1) - \sum_{\tau=0}^{M-1} \alpha_\tau \mathcal{B}_\phi(\pi, \pi_{k-\tau}). \tag{14}$$

In Appendix F.7, we perform an ablation study to show the effectiveness of adding such a magnet policy. Nevertheless, it is worth noting that this particular choice should not be confused with the original MMD even when $M = 1$ as the policy updating rule is derived via a numerical method, instead of relying on the closed-form solution (Sokota et al., 2023).

D.3. Derivation of MMD-EU

In this section, we present the details of the baseline, MMD-EU, used in our experiments. This baseline follows the spirit of MMD-KL (Sokota et al., 2023). Consider the following problem:

$$\pi_{k+1} = \arg\max_{\pi\in\Pi} \langle Q(\pi_k), \pi\rangle - \xi\mathcal{B}_\phi(\pi, \rho) - \frac{1}{\eta}\mathcal{B}_\phi(\pi, \pi_k), \tag{15}$$

where $\xi > 0$ is the regularization intensity, $\eta > 0$ is the step size, and ρ is the magnet policy. Let $\phi(\pi) = \sum_{a\in\mathcal{A}} \frac{1}{2}\|\pi(a)\|^2$, i.e., the squared Euclidean norm. Then, we need to optimize the following objective:

$$\langle Q(\pi_k), \pi\rangle - \frac{\xi}{2}||\pi - \rho||_2^2 - \frac{1}{2\eta}||\pi - \pi_k||_2^2, \tag{16}$$

with the constraint $\sum_{a\in\mathcal{A}} \pi(a) = 1$ and $\pi(a) > 0$. We can use the Lagrange multiplier to get the following objective:

$$\langle Q(\pi_k), \pi\rangle - \frac{\xi}{2}||\pi - \rho||_2^2 - \frac{1}{2\eta}||\pi - \pi_k||_2^2 + \lambda\left(1 - \sum_{a\in\mathcal{A}} \pi(a)\right). \tag{17}$$

Taking the derivative of both π and λ, we have:

$$Q(a, \pi_k) - \xi(\pi(a) - \rho(a)) - \frac{1}{\eta}(\pi(a) - \pi_k(a)) - \lambda = 0, \forall a \in \mathcal{A}, \tag{18}$$

$$\sum_{a\in\mathcal{A}} \pi(a) = 1. \tag{19}$$

Therefore from Eq. (18), we have:

$$\pi(a) = \frac{\xi\rho(a) + \frac{1}{\eta}\pi_k(a) + Q(a, \pi_k) - \lambda}{(\xi + \frac{1}{\eta})}. \tag{20}$$

Substituting the above equation to Eq. (19), we have:

$$\sum_{a\in\mathcal{A}} \frac{\xi\rho(a) + \frac{1}{\eta}\pi_k(a) + Q(a, \pi_k) - \lambda}{(\xi + \frac{1}{\eta})} = 1, \tag{21}$$

$$\Rightarrow \sum_{a\in\mathcal{A}} \left[\xi\rho(a) + \frac{1}{\eta}\pi_k(a) + Q(a, \pi_k)\right] = (\xi + \frac{1}{\eta}) + \sum_{a\in\mathcal{A}} \lambda. \tag{22}$$

Note that $\sum_{a\in\mathcal{A}} \left[\xi\rho(a) + \frac{1}{\eta}\pi_k(a)\right] = \xi + \frac{1}{\eta}$, we have:

$$\lambda = \frac{\sum_{a\in\mathcal{A}} Q(a, \pi_k)}{|\mathcal{A}|}. \tag{23}$$

Then we can compute the new policy as follows:

$$\pi(a) = \frac{\xi\rho(a) + \frac{1}{\eta}\pi_k(a) + Q(a, \pi_k) - \frac{1}{|\mathcal{A}|}\sum_{a'\in\mathcal{A}} Q(a', \pi_k)}{(\xi + \frac{1}{\eta})}. \tag{24}$$

Theoretically, we note that by choosing the suitable values for ξ and η, we can always ensure that π is well-defined, i.e., $\pi(a) \geq 0, \forall a \in \mathcal{A}$. In experiments, we can use a projection operation to ensure this condition ($\zeta = 1e-10$ is used to avoid division by zero):

$$\pi_{k+1}(a) = \frac{\max\{0, \pi(a)\} + \zeta}{\sum_{a' \in \mathcal{A}} \max\{0, \pi(a')\} + \zeta}. \tag{25}$$

In addition, similar to MMD-KL, the magnet policy is updated as (i.e., moving magnet):

$$\rho_{k+1}(a) = (1 - \tilde{\eta})\rho_k(a) + \tilde{\eta}\pi_{k+1}(a), \tag{26}$$

where $\tilde{\eta} > 0$ is the learning rate for the magnet policy ρ. In practice, the initial magnet policy ρ_1 can be set to the initial policy π_1 which is typically a uniform policy.

D.4. Meta-Controller for Different Measures

GMD generalizes exiting MD algorithms in two aspects: i) it takes multiple previous policies into account and can recover some of the existing MD algorithms by setting the M and $\boldsymbol{\alpha}$, and ii) it can consider a broader class of Bregman divergence by setting ϕ to more possible convex functions (Table 2). As a consequence, we argue that when GMD is executed by each agent independently, it could satisfy the first two desiderata **D1** and **D2** presented in the Introduction. However, as mentioned in Section 5.2, since there is no explicit objective regarding different evaluation measures (and different solution concepts) arises in this "decentralized" execution process, GMD itself cannot satisfy well the last two desiderata **D3** and **D4**. To address the challenges, our solution is the zero-order meta-controller (MC) which dynamically adjusts the hyper-parameters conditional on the evaluation measures (Section 5.2). In this section, we present the details of different MCs.

Direction-Guided Random Search (DRS). Our DRS method is obtained by applying the *direction-guided update* (Section 5.2) to the existing RS method presented in (Wang et al., 2022). Specifically, at the iteration k, we first sample D candidate updates $\{\boldsymbol{u}^j\}_{j=1}^{D}$ from a spherically symmetric distribution $\boldsymbol{u}^j \sim q$. Then, we update $\boldsymbol{\alpha}$ as follows:

$$\begin{aligned}
&\boldsymbol{\alpha}_+^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \mu\boldsymbol{u}^j),\ \boldsymbol{\alpha}_-^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} - \mu\boldsymbol{u}^j),\ 1 \leq j \leq D,\\
&\boldsymbol{\pi}_+^j = \text{GMD}(\boldsymbol{\alpha}_+^j),\ \boldsymbol{\pi}_-^j = \text{GMD}(\boldsymbol{\alpha}_-^j),\ 1 \leq j \leq D,\\
&\delta^j = \mathcal{L}(\boldsymbol{\pi}_+^j) - \mathcal{L}(\boldsymbol{\pi}_-^j),\ 1 \leq j \leq D,\\
&\boldsymbol{u}^* = -\sum\nolimits_{j=1}^{D} \text{Sgn}(\delta^j)\boldsymbol{u}^j,\\
&\boldsymbol{\alpha} \leftarrow \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \boldsymbol{u}^*).
\end{aligned} \tag{\textbf{DRS}}$$

$\text{Sgn}(z)$ is defined as: $\text{Sgn}(z) = 1$ if $z > 0$, $\text{Sgn}(z) = -1$ if $z < 0$, otherwise, $\text{Sgn}(z) = 0$. μ is the smoothing parameter determining the radius of the sphere. CLIP_ι^1 is the element-wise clipping operation defined as: $\text{CLIP}_\iota^1(z) = \iota$ if $z < \iota$, $\text{CLIP}_\iota^1(z) = 1$ if $z > 1$, otherwise, $\text{CLIP}_\iota^1(z) = z$, where $0 < \iota < 1$. Note that the clipping operation which bounds $\boldsymbol{\alpha}$ above $\iota > 0$ is necessary as the term B is used as the denominator in Eq. (12). In addition, the operation $\text{Sgn}(\cdot)$ plays an important role and differentiates our DRS from the vanilla RS (Wang et al., 2022). Intuitively, in the games where the value of the evaluation measure $\mathcal{L}$ is extremely small and converges quickly, the magnitude of δ^j would be too small to derive an effective update. In contrast, by using the operation $\text{Sgn}(\cdot)$, the difference between the performance of $\boldsymbol{\alpha}_+^j$ and $\boldsymbol{\alpha}_-^j$ will only determine the *update direction*, not the update magnitude, which could be more effective.

Random Search (RS). The vanilla RS which is adapted from (Wang et al., 2022). The only difference from DRS is it updates $\boldsymbol{\alpha}$ directly based on the performance difference δ^j. Precisely, we have:

$$\begin{aligned}
&\boldsymbol{\alpha}_+^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \mu\boldsymbol{u}^j),\ \boldsymbol{\alpha}_-^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} - \mu\boldsymbol{u}^j),\ 1 \leq j \leq D,\\
&\boldsymbol{\pi}_+^j = \text{GMD}(\boldsymbol{\alpha}_+^j),\ \boldsymbol{\pi}_-^j = \text{GMD}(\boldsymbol{\alpha}_-^j),\ 1 \leq j \leq D,\\
&\delta^j = \mathcal{L}(\boldsymbol{\pi}_+^j) - \mathcal{L}(\boldsymbol{\pi}_-^j),\ 1 \leq j \leq D,\\
&\boldsymbol{u}^* = -\sum\nolimits_{j=1}^{D} \delta^j\boldsymbol{u}^j,\\
&\boldsymbol{\alpha} \leftarrow \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \boldsymbol{u}^*).
\end{aligned} \tag{\textbf{RS}}$$

GradientLess Descent (GLD). This method is adapted from (Wang et al., 2022). At the iteration k, we first sample D candidate updates $\{\boldsymbol{u}^j\}_{j=1}^{D}$. Different from RS which samples the candidates from a fixed radius (the smoothing parameter

μ in DRS and RS), we independently sample the candidates on spheres with various radiuses uniformly sampled from the interval $[r_L, r_H]$. Then, we update $\boldsymbol{\alpha}$ as follows:

$$\begin{aligned}
&\boldsymbol{\alpha}_+^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \boldsymbol{u}^j),\ \boldsymbol{\pi}_+^j = \text{GMD}(\boldsymbol{\alpha}_+^j),\ 1 \le j \le D,\\
&j^* = \arg\min_j \{\mathcal{L}(\boldsymbol{\pi}_+^j)\}_{j=1}^D,\\
&\boldsymbol{u}^* = \boldsymbol{u}^{j^*},\\
&\boldsymbol{\alpha} \leftarrow \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \boldsymbol{u}^*).
\end{aligned} \tag{GLD}$$

Intuitively, by comparing the performance of D candidates, $\boldsymbol{\alpha}$ is updated by the candidate with the smallest value of $\mathcal{L}$.

GradientLess Descent with Summation (GLDS). Different from GLD which uses only one of the D samples to update the $\boldsymbol{\alpha}$, we can follow the idea of RS/DRS to take all the candidates into account by summation. Specifically, let $\mathcal{L}(\pi_k)$ denote the performance of the current policy π_k, then we have:

$$\begin{aligned}
&\boldsymbol{\alpha}_+^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \boldsymbol{u}^j),\ \boldsymbol{\pi}_+^j = \text{GMD}(\boldsymbol{\alpha}_+^j),\ 1 \le j \le D,\\
&\delta^j = \mathcal{L}(\boldsymbol{\pi}_+^j) - \mathcal{L}(\boldsymbol{\pi}_k),\ 1 \le j \le D,\\
&\boldsymbol{u}^* = -\sum\nolimits_{j=1}^D \delta^j \boldsymbol{u}^j,\\
&\boldsymbol{\alpha} \leftarrow \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \boldsymbol{u}^*).
\end{aligned} \tag{GLDS}$$

Direction-Guided GLDS (DGLDS). Applying the direction-guided update to the GLDS, we can get this method. Precisely, let $\mathcal{L}(\pi_k)$ denote the performance of the current policy π_k, then we have:

$$\begin{aligned}
&\boldsymbol{\alpha}_+^j = \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \boldsymbol{u}^j),\ \boldsymbol{\pi}_+^j = \text{GMD}(\boldsymbol{\alpha}_+^j),\ 1 \le j \le D,\\
&\delta^j = \mathcal{L}(\boldsymbol{\pi}_+^j) - \mathcal{L}(\boldsymbol{\pi}_k),\ 1 \le j \le D,\\
&\boldsymbol{u}^* = -\sum\nolimits_{j=1}^D \text{Sgn}(\delta^j) \boldsymbol{u}^j,\\
&\boldsymbol{\alpha} \leftarrow \text{CLIP}_\iota^1(\boldsymbol{\alpha} + \boldsymbol{u}^*).
\end{aligned} \tag{DGLDS}$$

As the meta-controller needs to evaluate the performance of the candidates, extra computational cost is required. In our experiments, to trade-off between the learning performance and running time, we update $\boldsymbol{\alpha}$ every $\kappa \ge 1$ iteration. In addition, during the first $M-1$ iterations, i.e., $k < M$, as there are only $k < M$ historical policies, we set $\alpha_\tau = \frac{1}{k}$ for $0 \le \tau \le k-1$. In other words, MC will start to update $\boldsymbol{\alpha}$ only after M iterations. Algorithm 2 in the main text is the simplified version which shows the primary principle of CMD. In Algorithm 4, we present the full details of CMD.

Algorithm 4 Configurable Mirror Descent (CMD)

```
1: Given $\mathcal{L}$, $\psi$, initial (joint) policy $\boldsymbol{\pi}_1$, $M$, $D$, $\epsilon$, $\iota$,
2: for $k = 1, \cdots, K$ do
3:     if $k \le M$ then
4:         $\alpha_\tau = \frac{1}{k}, \forall 0 \le \tau \le k-1$
5:     else
6:         if $k \% \kappa = 0$ then
7:             Sample $D$ candidates $\{\boldsymbol{\alpha}^j\}_{j=1}^D$
8:             Derive new joint policies $\{\boldsymbol{\pi}^j = \text{GMD}(\boldsymbol{\alpha}^j)\}_{j=1}^D$
9:             Evaluate new joint policies $\{\mathcal{L}(\boldsymbol{\pi}^j)\}_{j=1}^D$
10:            Update $\boldsymbol{\alpha}$ based on $\{\mathcal{L}(\boldsymbol{\pi}^j)\}_{j=1}^D$
11:        end if
12:    end if
13:    Compute $\boldsymbol{\pi}_{k+1}$ via GMD with the updated $\boldsymbol{\alpha}$
14: end for
```

E. GameBench

In this section, we present the details of GameBench (see Figure 3 for an overview). In Section E.1, we discuss the motivation and desiderata by briefly reviewing the games that have been employed to test existing MD algorithms. In Section E.2, we present the details of the construction of all 15 games. Finally, in Section E.3, we present the evaluation measures considered in this work.

E.1. Motivation and Desiderata

As mentioned in Section 6, existing benchmarks for decision making are typically specialized for some specific categories. Furthermore, running MD algorithms on these benchmarks could be computationally prohibitive as the number of decision points in the environments could be extremely large. On the other hand, though MD algorithms have been demonstrated powerful in single-agent RL (Tomar et al., 2022) and two-player zero-sum games (Wibisono et al., 2022; Kozuno et al., 2021; Lee et al., 2021; Liu et al., 2022b; Jain et al., 2022; Ao et al., 2023; Liu et al., 2023; Cen et al., 2023; Sokota et al., 2023) in recent works, their experiments are typically conducted on a handful of games. It remains elusive how will these MD algorithms perform when applied to other categories of decision-making problems. In Table 5, we briefly review the games that have been used in some recent works.

Table 5. The games that have been used in recent works on MD algorithms. Note that this list does not include the games that are used to benchmark deep learning-based algorithms in these references. [1]This game is made to be a general-sum game via a tie-breaking mechanism in this reference. [2]This game is made to be a zero-sum game in this reference.

Reference	Game	Category
(Sokota et al., 2023)	Kuhn Poker	Two-Player Zero-Sum
	Leduc Poker	Two-Player Zero-Sum
	2x2 Abrupt Dark Hex	Two-Player Zero-Sum
	4-Sided Liar's Dice	Two-Player Zero-Sum
(Liu et al., 2023)	Kuhn Poker	Two-Player Zero-Sum
	Leduc Poker	Two-Player Zero-Sum
(Anagnostides et al., 2022b)	Kuhn Poker	Two-Player Zero-Sum
	Leduc Poker	Two-Player Zero-Sum
(Anagnostides et al., 2022a)	Sheriff	Two-Player General-Sum
	Battleship	Two-Player General-Sum
	Goofspiel[1]	Two-Player General-Sum
	Liar's Dice	Two-Player Zero-Sum
(Lee et al., 2021)	Kuhn Poker	Two-Player Zero-Sum
	Leduc Poker	Two-Player Zero-Sum
	Pursuit-Evasion	Two-Player Zero-Sum
(Liu et al., 2022b)	Leduc Poker	Two-Player Zero-Sum
	Goofspiel	Two-Player Zero-Sum
	Liar's Dice	Two-Player Zero-Sum
	Battleship[2]	Two-Player Zero-Sum

In view of the above facts, we aim to construct a novel benchmark which should satisfy two desiderata (**D5** and **D6** presented in the Introduction): i) it should cover all categories of decision making (comprehensive), and ii) the games are relatively simple and running MD algorithms on these games does not require much computational resource (academic-friendly).

E.2. Games

In this section, we present the details of the construction of all 15 games in our GameBench. All the games are divided into 5 categories: single-agent, cooperative multi-agent, competitive multi-agent zero-sum (zero-sum), competitive multi-agent general-sum (general-sum), and mixed cooperative and competitive (MCC) categories. In Table 6, we give an overview of all the games. We curate the GAMEBENCH on top of OpenSpiel (Lanctot et al., 2019). For cooperative, zero-sum, and general-sum categories, we construct the game by passing the configurations to the games implemented in OpenSpiel. The configurations for these games are deliberately selected such that the instances of these games are academic-friendly (i.e., their numbers of decision points are not too large). For single-agent and MCC categories, we obtain the games by modifying the original games in OpenSpiel. In the following, we present the details of each category.

Table 6. The games and their statistics in GAMEBENCH. N is the number of players and "#DP" stands for the number of decision points.

Category	Name of Game w/ Config.	Shorthand	N	#DP	Evaluation Measure
Single-Agent	single_agent_kuhn_a	Kuhn-A	1	6	OptGap
	single_agent_kuhn_b	Kuhn-B	1	6	OptGap
	single_agent_goofspiel	Goofspiel-S	1	8	OptGap
Cooperative	tiny_hanabi_game_a	TinyHanabi-A	2	8	OptGap
	tiny_hanabi_game_b	TinyHanabi-B	2	6	OptGap
	tiny_hanabi_game_c	TinyHanabi-C	2	6	OptGap
Zero-Sum	kuhn_poker(players=3)	Kuhn	3	48	NashConv, CCEGap
	leduc_poker(players=2)	Leduc	2	936	NashConv
	goofspiel(players=3)	Goofspiel	3	30	NashConv, CCEGap
General-Sum	bargaining(max_turns=2)	Bargaining	2	178	NashConv, SW
	trade_comm(num_items=2)	TradeComm	2	22	NashConv, SW
	battleship	Battleship	2	210	NashConv, SW
MCC	mix_kuhn_3p_game_a	MCCKuhn-A	3	48	NashConv
	mix_kuhn_3p_game_b	MCCKuhn-B	3	48	NashConv
	mix_goofspiel_3p	MCCGoofspiel	3	30	NashConv

Single-Agent. We construct three single-agent games: Kuhn-A, Kuhn-B, and Goofspiel-S, from the original two-player Kuhn poker and Goofspiel in OpenSpiel. Consider a two-player Kuhn poker game. To obtain a single-agent counterpart, we fix one player's policy as the uniform policy (called the background player) while only updating the other player's policy (called the focal player) at each iteration. In Kuhn-A, player 1 is selected as the focal player while in Kuhn-B, player 2 is chosen as the focal player, as the two players are asymmetric (Kuhn, 1950). Similarly, we can get Goofspiel-S. As the two players are symmetric in Goofspiel (Ross, 1971), we choose player 1 as the focal player without loss of generality.

Cooperative. For cooperative games, we consider the following three *two-player* tiny Hanabi games (Foerster et al., 2019; Sokota et al., 2021): TinyHanabi-A, TinyHanabi-B, and TinyHanabi-C. The payoff matrices along with the optimal values of these games are given in Figure 9. These games are easy to obtain in OpenSpiel by setting the three parameters: `num_chance`, `num_actions`, and `payoff`. For `num_chance`, they are 2, 2, and 2, respectively. For `num_actions`, they are 3, 2, and 2, respectively.

Competitive Zero-Sum and General-Sum. We consider the following three zero-sum games: *three-player* Kuhn, *two-player* Leduc, and *three-player* Goofspiel, and the following three general-sum games: *two-player* Battleship (Farina et al., 2020), *two-player* TradeComm (Sokota et al., 2021), and *two-player* Bargaining (Lewis et al., 2017), which are implemented in OpenSpiel. The configurations of these games are given in the second column in Table 6. Note that in contrast to most of the existing works which only focus on two-player games, we set the number of players to more than two players in some of the games: Kuhn and Goofspiel are three-player games.

Mixed Cooperative and Competitive (MCC). We construct the following *three-player* MCC games: MCCKuhn-A, MCCKuhn-B, and MCCGoofspiel, from the original three-player Kuhn poker and three-player Goofspiel in OpenSpiel. Consider a three-player Kuhn poker game. To obtain an MCC counterpart, we partition the three players into two teams: Team 1 includes two players while Team 2 only consists of one player (i.e., two *vs.* one). When computing the rewards of the players, in Team 1, each player will get the average reward of the team. Precisely, let $r^{\text{team}} = r^1 + r^2$ denote the team reward which is the sum of the original rewards of the two team members. Then, the true rewards of the two players are $\tilde{r}^1 = \tilde{r}^2 = r^{\text{team}}/2$. In MCCKuhn-A, Team 1 includes players 1 and 2 (i.e., $\{1, 2\}$ *vs.* 3), while in MCCKuhn-B, Team 1 includes players 1 and 3 (i.e., $\{1, 3\}$ *vs.* 2). Similarly, we can get MCCGoofspiel in the same manner.

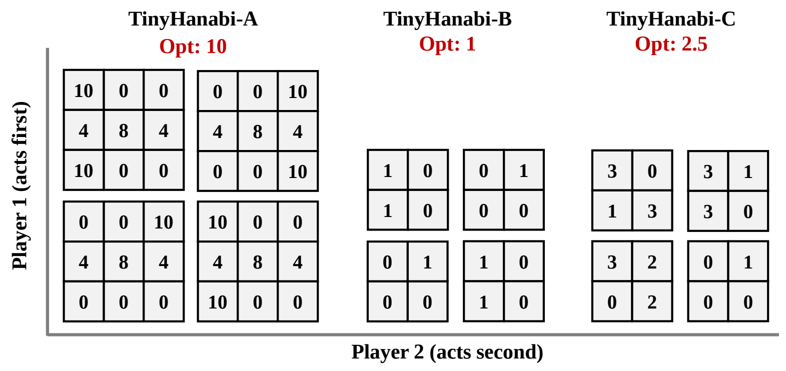


Figure 9. Payoff matrices and optimal values of the three tiny Hanabi games.

As shown in Table 6, the number of decision points (#DP) varies across different categories, which shows that GAMEBENCH includes diverse enough environments as, to some extent, the number of decision points reflects the difficulty of the game.

E.3. Evaluation Measures

As shown in Figure 3, we consider multiple evaluation measures in GAMEBENCH. There are two types of measures: i) the notion of *optimality*, including OptGap and social welfare, and ii) the notion of *equilibrium*, including NashConv and CCEGap. In the last column of Table 6, we present the measures employed in each game. In single-agent and cooperative categories, we use OptGap as the measure which captures the distance of the current (joint) policy to the optimal (joint) policy. In the other three categories, the primary measure is NashConv which captures the distance of the current joint policy to the *Nash equilibrium*. In addition, we also consider other solution concepts and evaluation measures in some of the games. For zero-sum Kuhn and Goofspiel, as there are three players, we also consider the measure CCEGap which captures the distance of the current joint policy to the coarse correlated equilibrium (CCE). For general-sum games, we also consider the social welfare (SW) of all the agents.

Except for the MCC category, all the measures can be easily computed by using the built-in implementation functions in OpenSpiel. However, to compute the NashConv in the MCC games, we need to compute the best response policy of the team, i.e., a joint policy of the team members, rather than the policy of a single agent. This is incompatible with the built-in implementation in OpenSpiel, which only computes the best response policy of a single agent. In other words, if we directly adopt the built-in implementation, the NashConv will correspond to the original three-player game, not the modified game. Unfortunately, computing the exact joint policy of the team members is not easy in practice. Nevertheless, it is worth noting that from our experiments, we found that MMD-KL can effectively solve cooperative decision-making problems. As a result, we can apply MMD-KL to compute the approximate best response of the team as it is a purely cooperative environment from the team's perspective (the other team's policy is fixed when computing the best response of the team). For a team that only has a single player, we use the built-in implementation in OpenSpiel to compute the exact best response policy of the player. In summary, during the policy learning process, when the evaluation of the current joint policy is needed, we use MMD-KL as a subroutine to compute a team's approximate best response while using built-in implementation to compute a single player's exact best response. In the MMD-KL subroutine, the starting point of the best response is set to the current joint policy of the team members. In experiments, to balance the accuracy of the approximate best response and running time, the number of updates in the MMD-KL subroutine is set to 100 (the returned joint policy can be also called a better response).

For example, in MCCKuhn-A, suppose the current joint policy is $\boldsymbol{\pi} = \boldsymbol{\pi}_{\text{team}} \times \pi_3$ where $\boldsymbol{\pi}_{\text{team}} = \pi_1 \odot \pi_2$ is the team's joint policy. The built-in implementation in OpenSpiel can only compute the best response policy for every single agent and hence, the resulting $\text{NashConv}(\boldsymbol{\pi}) = \sum_{i=1}^{3}[V_i(\nu, \pi_i^{\text{BR}} \times \boldsymbol{\pi}_{-i}) - V_i(\nu, \boldsymbol{\pi})]$ corresponds to the original three-player game. In contrast, in our method, we use MMD-KL to compute the team's best response rather than the single agent's. Therefore, the NashConv of $\boldsymbol{\pi}$ is:

$$\begin{aligned}\text{NashConv}(\boldsymbol{\pi}) = V_{\text{team}}(\nu, \boldsymbol{\pi}_{\text{team}}^{\text{BR}} \times \pi_3) - V_{\text{team}}(\nu, \boldsymbol{\pi}) \\ + V_3(\nu, \boldsymbol{\pi}_{\text{team}} \times \pi_3^{\text{BR}}) - V_3(\nu, \boldsymbol{\pi}),\end{aligned} \tag{27}$$

where $\boldsymbol{\pi}_{\text{team}}^{\text{BR}}$ is the team's BR policy computed via MMD-KL given that player 3 is fixed to π_3 (that is, player 3 is a part of the environment from the team's perspective). As players 1 and 2 are fully cooperative, they share the same value V_{team}.

F. More Experimental Results

In this section, we provide more experimental details, results, and analysis. We briefly summarize each section below.

- Section F.1. More details on the experimental setup, including the hyper-parameter settings for different methods.
- Section F.2. Searching of M (the number of previous policies) and μ (the smoothing parameter in DRS) (**D1** and **D2**).
- Section F.3. Investigation of performance w.r.t. the number of joint actions (**D1** and **D2**).
- Section F.4. Investigation of GMD with different heuristic strategies for adjusting $\boldsymbol{\alpha}$ (**D1** and **D2**).
- Section F.5. Investigation of different meta-controllers (**D1** and **D2**).
- Section F.6. Investigation of different Bregman divergences (**D1** and **D2**).
- Section F.7. Investigation of the effectiveness of adding the magnet policy (**D1** and **D2**).
- Section F.8. Investigation of different evaluation measures and different solution concepts (**D3** and **D4**).
- Section F.9. Analysis of the computational complexity for running different algorithms on GAMEBENCH (**D5** and **D6**).

F.1. Experimental Setup

Hyper-parameters. Table 7 provides the default values of hyper-parameters used in different methods. In the RS-type meta-controllers (RS and DRS), the spherically symmetric distribution q is a standard multivariate normal distribution $\mathbb{N}(\mathbf{0}, \mathbf{I})$. For CMD/GMD, there are two critical hyper-parameters: the number of previous policies $M \geq 1$ and the smoothing parameter μ in DRS. In Section F.2, we perform an ablation study to determine their default values (given in Table 7), which will be fixed in other experiments. The specific setups for each experiment will be given in each of the following sections.

Baselines. We consider the MMD-type (MMD-KL and MMD-EU) and CFR-type (CFR and CFR+) algorithms as the baselines. It is worth noting that CFR-type algorithms can be also applied to single-agent and cooperative categories.

Computational Resources. Experiments are performed on a machine with a 24-core i9 and NVIDIA A4000. For CMD, the results are obtained with 3 random seeds. For other methods, as there is no randomness, no multiple runs are needed.

Table 7. Default values of the hyper-parameters in different methods. All the hyper-parameters in GMD – C, ι, and M – are also used in CMD. For CMD, its hyper-parameters also include i) D (the number of samples) and κ (update interval), which are shared for different MCs, ii) μ in the DRS and RS, and iii) r_L and r_H in the GLD, GLDS, and DGLDS.

			CMD								MMD-KL/-EU		
			GMD			Shared		(D)RS	(D)GLD(S)				
Game	K	ϵ	C	ι	M	D	κ	μ	r_L	r_H	ξ	η	$\tilde{\eta}$
Kuhn-A	100000	1e-10	50	1e-6	1	5	10	0.05	0.01	0.05	1	0.1	0.05
Kuhn-B	100000	1e-10	50	1e-6	1	5	10	0.05	0.01	0.05	1	0.1	0.05
Goofspiel-S	100000	1e-10	50	1e-6	1	5	10	0.05	0.01	0.05	1	0.1	0.05
TinyHanabi-A	100000	1e-10	50	1e-6	3	5	10	0.05	0.01	0.05	1	0.1	0.05
TinyHanabi-B	100000	1e-10	50	1e-6	1	5	10	0.05	0.01	0.05	1	0.1	0.05
TinyHanabi-C	100000	1e-10	50	1e-6	1	5	10	0.05	0.01	0.05	1	0.1	0.05
Kuhn	100000	1e-10	50	1e-6	5	5	10	0.01	0.01	0.05	1	0.1	0.05
Leduc	100000	1e-10	50	1e-6	3	5	10	0.05	0.01	0.05	1	0.1	0.05
Goofspiel	100000	1e-10	50	1e-6	3	5	10	0.01	0.01	0.05	1	0.1	0.05
Bargaining	100000	1e-10	50	1e-6	5	5	10	0.05	0.01	0.05	1	0.1	0.05
TradeComm	100000	1e-10	50	1e-6	1	5	10	0.01	0.01	0.05	1	0.1	0.05
Battleship	100000	1e-10	50	1e-6	1	5	10	0.05	0.01	0.05	1	0.1	0.05
MCCKuhn-A	100000	1e-10	50	1e-6	1	5	10	0.01	0.01	0.05	1	0.1	0.05
MCCKuhn-B	100000	1e-10	50	1e-6	1	5	10	0.01	0.01	0.05	1	0.1	0.05
MCCGoofspiel	100000	1e-10	50	1e-6	1	5	10	0.01	0.01	0.05	1	0.1	0.05

F.2. Number of Historical Policies and Smoothing Parameter

In this section, we explore the influence of the number of previous policies M and the smoothing parameter μ in DRS on the learning performance. We consider $M \in \{1, 3, 5\}$ and $\mu \in \{0.01, 0.05\}$ and thus, there are 6 combinations of (M, μ). Note that it would be impractical to enumerate all the combinations as M can be any integer greater than 0 and μ can be any real number greater than 0.

The experimental results are shown in Figure 10. From the results, we can see that different decision-making problems may require different M and μ. Notably, $M = 1$, i.e., only considering the current policy when deriving the new policy which is common in existing MD algorithms, is not always the optimal choice across different decision-making problems. For example, in the most difficult Leduc poker game, when $M = 1$, CMD cannot decrease the NashConv, meaning that only considering the current policy is ineffective in solving this game. By comparison, we determine the default values of M and μ for different games, which are given in Table 7 and will be fixed in other experiments.

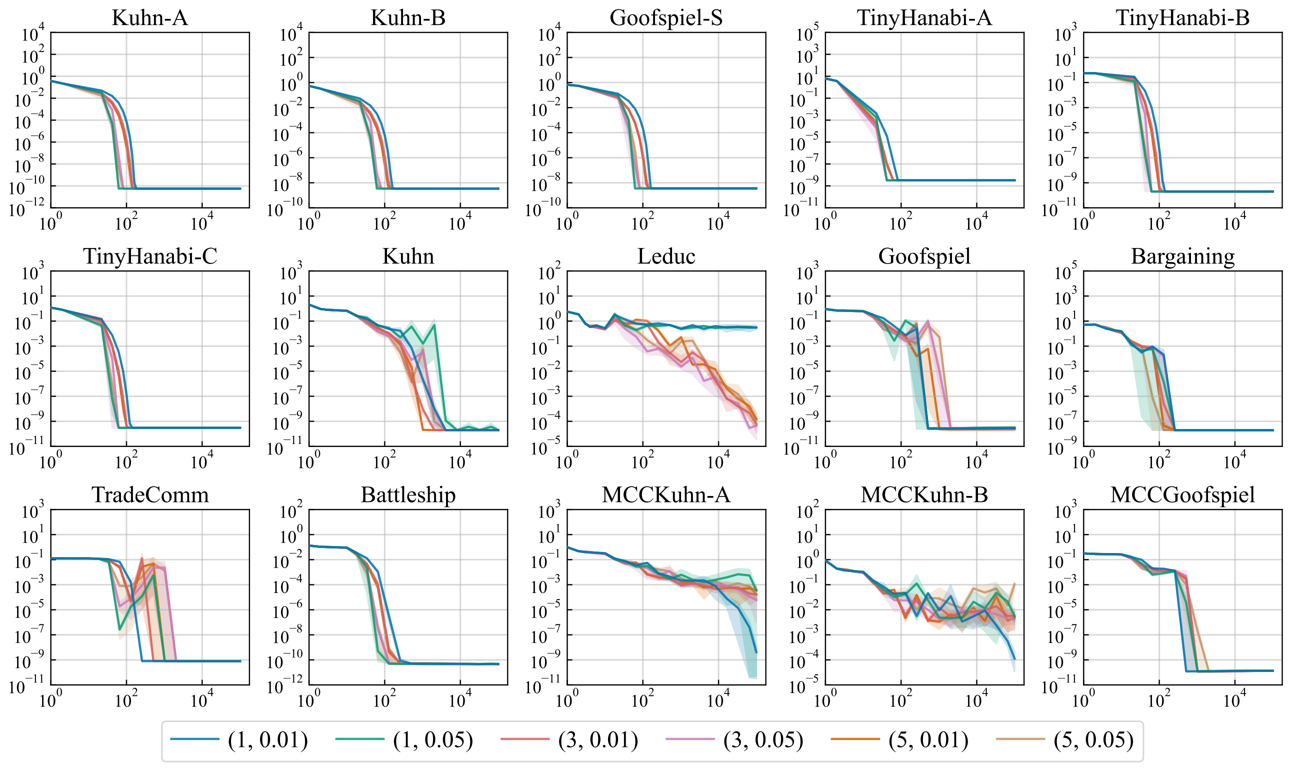


Figure 10. Experimental results for the combinations of (M, μ). The first 6 figures correspond to single-agent and cooperative categories where the y-axis is *OptGap*. The rest figures correspond to other categories where the y-axis is *NashConv*. For all the figures, the x-axis is the number of iterations.

F.3. Performance w.r.t. the Number of Joint Actions

In Figure 11, we plot the performance of different methods with respect to the number of joint actions involved in each iteration. As both MD-type and CFR-type algorithms will traverse the whole game tree, the number of joint actions for a given joint policy is the same. Therefore, in the figure, we only need to change the scale of the x-axis for the CMD by multiplying the constant D (the number of joint policies evaluated at each iteration), while keeping the scales of other methods unchanged. We note that as D is small in our experiments ($D = 5$, i.e., sample 5 candidate joint policies), the conclusions in terms of the number of iterations presented in the main text still hold in terms of the number of joint actions.

As discussed in Section 8, one of the future directions of our work would be the development of a more efficient method for updating the $\boldsymbol{\alpha}$, e.g., a method that only needs to sample one candidate (in this case, the number of joint actions will be the same for both CMD and other baselines). Nevertheless, compared to baseline MD and CFR-type algorithms, CMD provides a feasible way to study different solution concepts and evaluation measures, though, in the current version, it requires evaluating multiple candidates at each iteration.

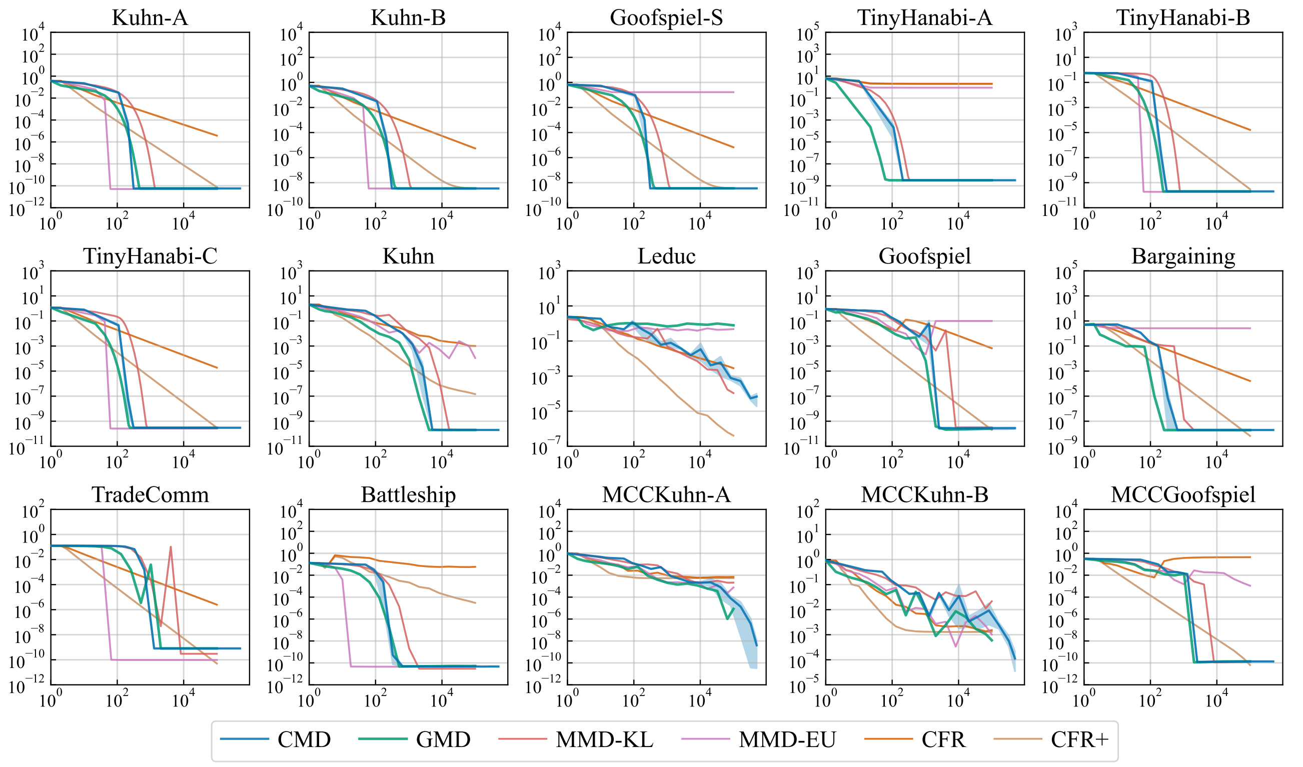


Figure 11. Performance of different methods w.r.t. the number of joint actions. The first 6 figures correspond to single-agent and cooperative categories where the y-axis is *OptGap*. The rest figures correspond to other categories where the y-axis is *NashConv*. For all the figures, the x-axis is the number of iterations.

F.4. Different Heuristic Strategies for Adjusting α in GMD

In the main text, the baseline method GMD employs a fixed strategy – a uniform distribution – to determine the value of $\boldsymbol{\alpha}$. In this section, we consider two more heuristic strategies: i) "GMD(LD)" denotes that the $\boldsymbol{\alpha}$ is linearly decayed with the iteration, and ii) "GMD(ISR)" denotes that the $\boldsymbol{\alpha}$ is decayed with the iteration in the form of inverse square root function $\alpha_\tau = \frac{1}{\sqrt{k}}$, where k is the k-th iteration. The results are shown in Figure 12. From the results, we can see that different heuristic strategies can perform differently in different decision-making scenarios; one can beat others in some scenarios while it can also be beaten by others in other scenarios.

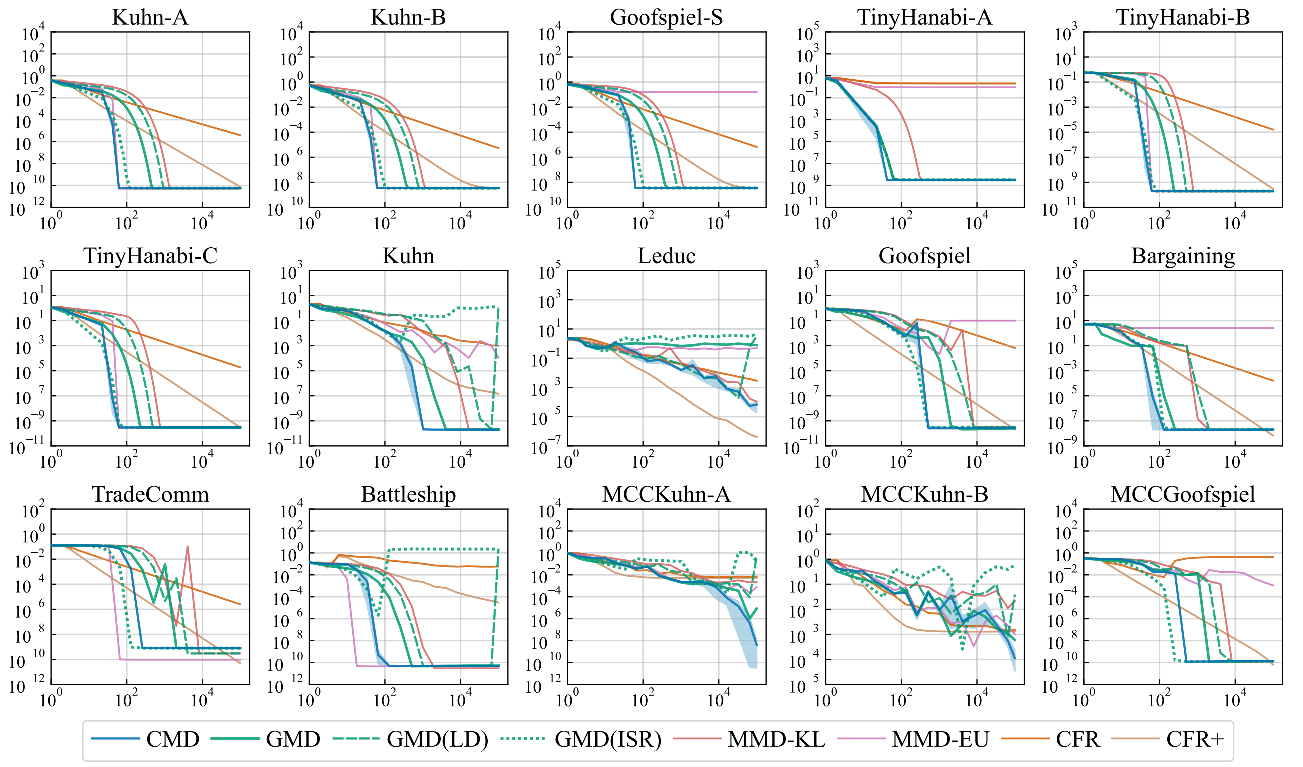


Figure 12. Experimental results for GMD with different heuristic strategies for adjusting $\boldsymbol{\alpha}$. The first 6 figures correspond to single-agent and cooperative categories where the y-axis is *OptGap*. The rest figures correspond to other categories where the y-axis is *NashConv*. For all the figures, the x-axis is the number of iterations.

F.5. Different Meta-Controllers

In this section, we investigate the effectiveness of different MCs, and the results are shown in Figure 13. From the results, we can see that DRS can consistently outperform all the other baseline MCs across almost all of the decision-making problems. Particularly, in Leduc and MCCKuhn-B, DRS achieves a significantly better convergent performance than other baseline MCs. Although in MCCKuhn-A, GLD finally converges to a lower NashConv than DRS, it can perform much worse in other games, e.g., in Battleship, GLD cannot decrease the NashConv, in Leduc and MCCKuhn-B, it only converges to a high value of NashConv. In other words, GLD cannot consistently work well across all the decision-making categories. In addition, in most of the games, the RS-type MCs typically perform better than the GLD-type MCs. We hypothesize that the RS-type MCs are more efficient in exploring the parameter space as they use more samples ($\boldsymbol{\alpha}_+^j$ and $\boldsymbol{\alpha}_-^j$ for each $\boldsymbol{u}^j$) to obtain the final update for the hyper-parameters.

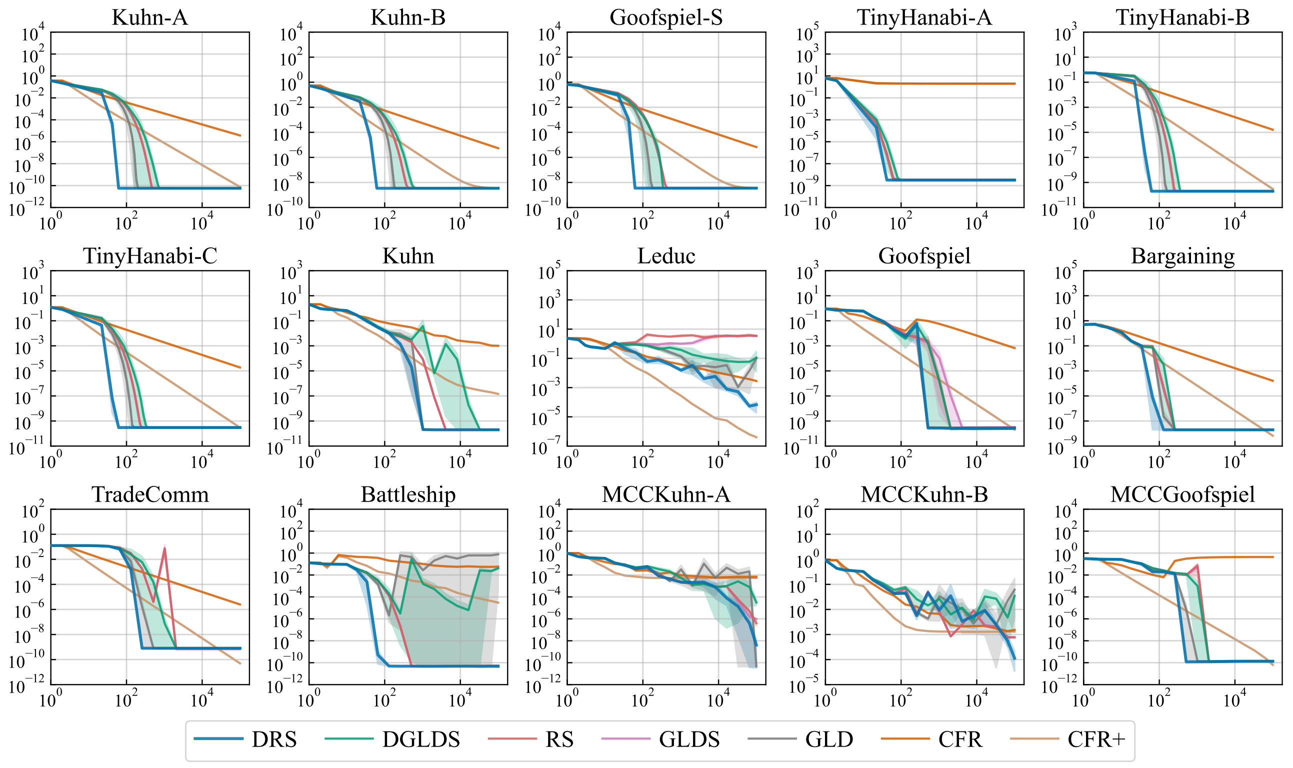


Figure 13. Experimental results for different MCs. The first 6 figures correspond to single-agent and cooperative categories where the y-axis is *OptGap*. The rest figures correspond to other categories where the y-axis is *NashConv*. For all the figures, the x-axis is the number of iterations.

Evolution of Hyper-Parameters. The critical observation that supports our proposed DRS is that the value of the evaluation measure $\mathcal{L}$ is extremely small and converges relatively quickly, which makes the original zero-order methods struggle in adjusting the hyper-parameters as they typically adjust the hyper-parameters directly based on the performance. To further verify this intuition, we visualize the evolution of the hyper-parameter $\boldsymbol{\alpha}$ over the learning process, which is shown in Figure 14–Figure 18 (respectively corresponds to single-agent, cooperative, zero-sum, general-sum, and MCC categories). We use index 0 to represent the magnet policy and the recent M historical policies are indexed by $\{1, \cdots, M\}$. From the results, we can see that in all the games except Leduc, the value of $\boldsymbol{\alpha}$ determined by RS almost does not change over the learning process (the same phenomenon is observed for GLDS as it follows the same idea of RS). In Leduc, this value tends to decrease to 0 over the learning process. In other words, the regularization is vanishing, which explains why RS and GLDS cannot decrease the NashConv in this game as adding regularization has been proven important to solve two-player zero-sum games (Sokota et al., 2023; Liu et al., 2023). In all the games, DRS and DGLDS share some similarities in determining the value of $\boldsymbol{\alpha}$ and differ from GLD. Nevertheless, the convergence results in Figure 13 show that DRS is the best choice among them as it can consistently work well across all categories of decision-making problems.

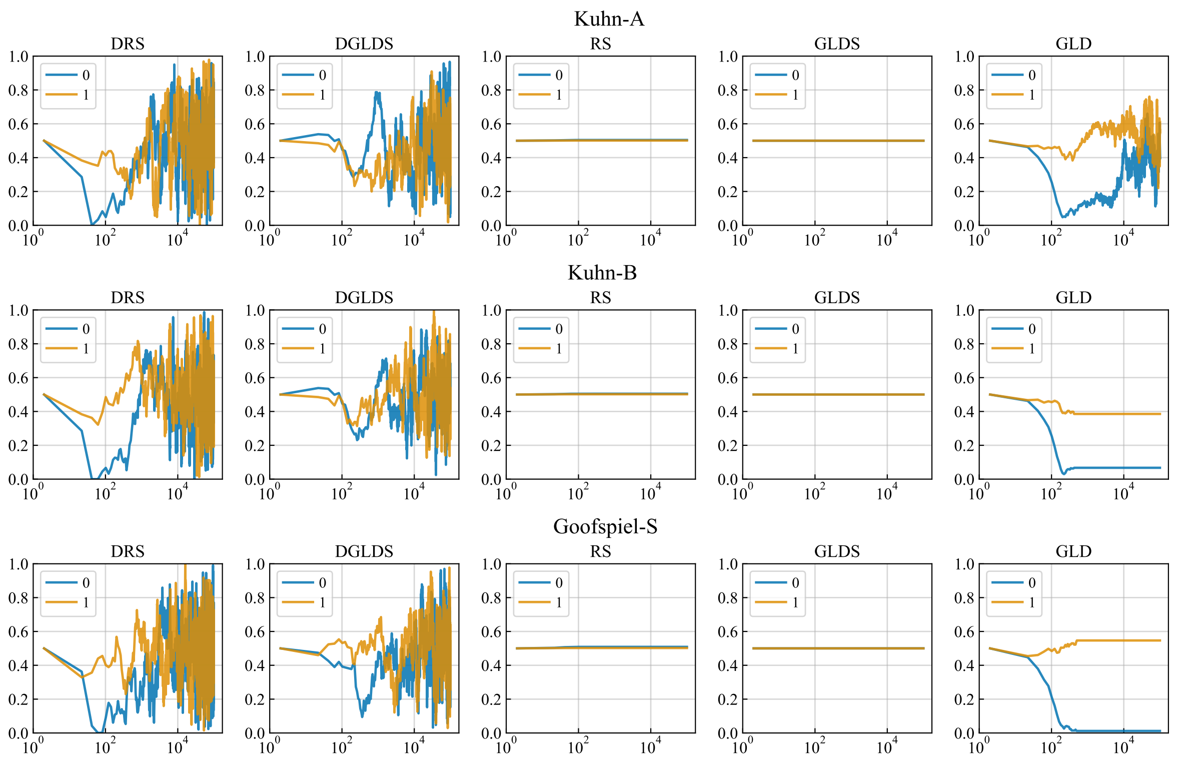


Figure 14. The evolution of the hyper-parameter values of different MCs in the **Single-Agent** category. The y-axis is the value of $\boldsymbol{\alpha}$. The x-axis is the number of iterations.

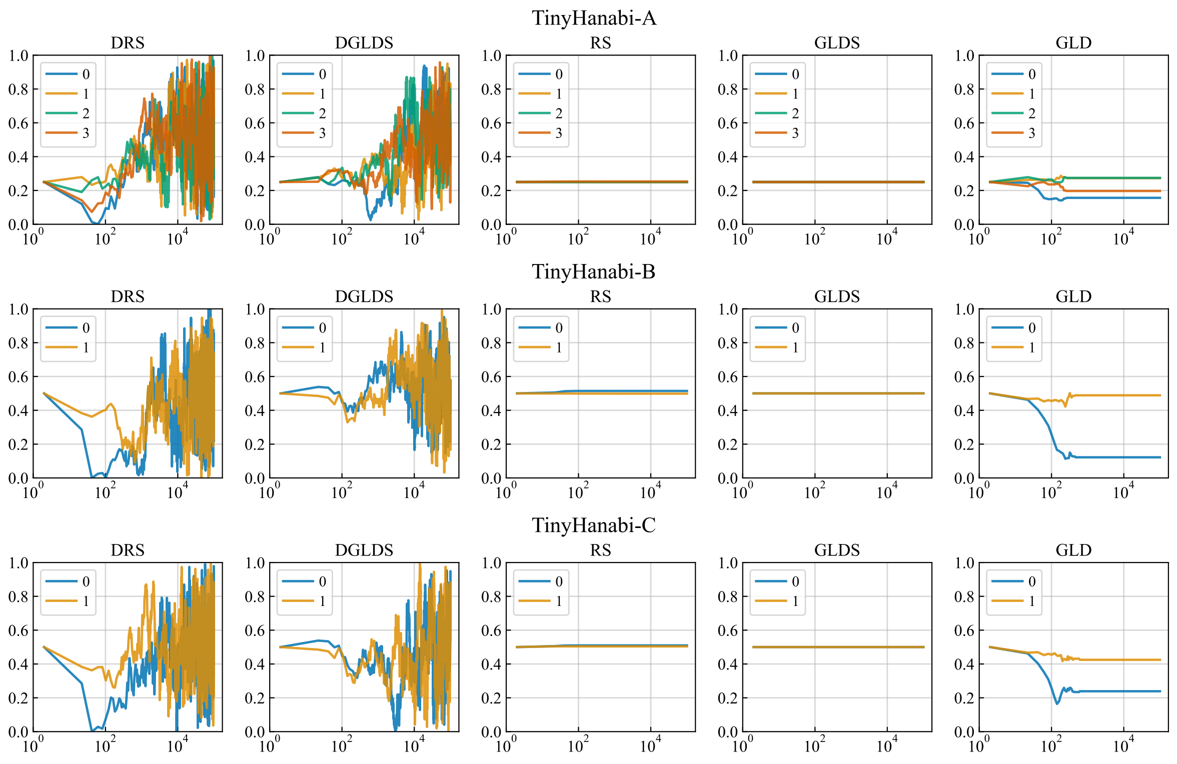


Figure 15. The evolution of the hyper-parameter values of different MCs in the **Cooperative** category. The y-axis is the value of $\boldsymbol{\alpha}$. The x-axis is the number of iterations.

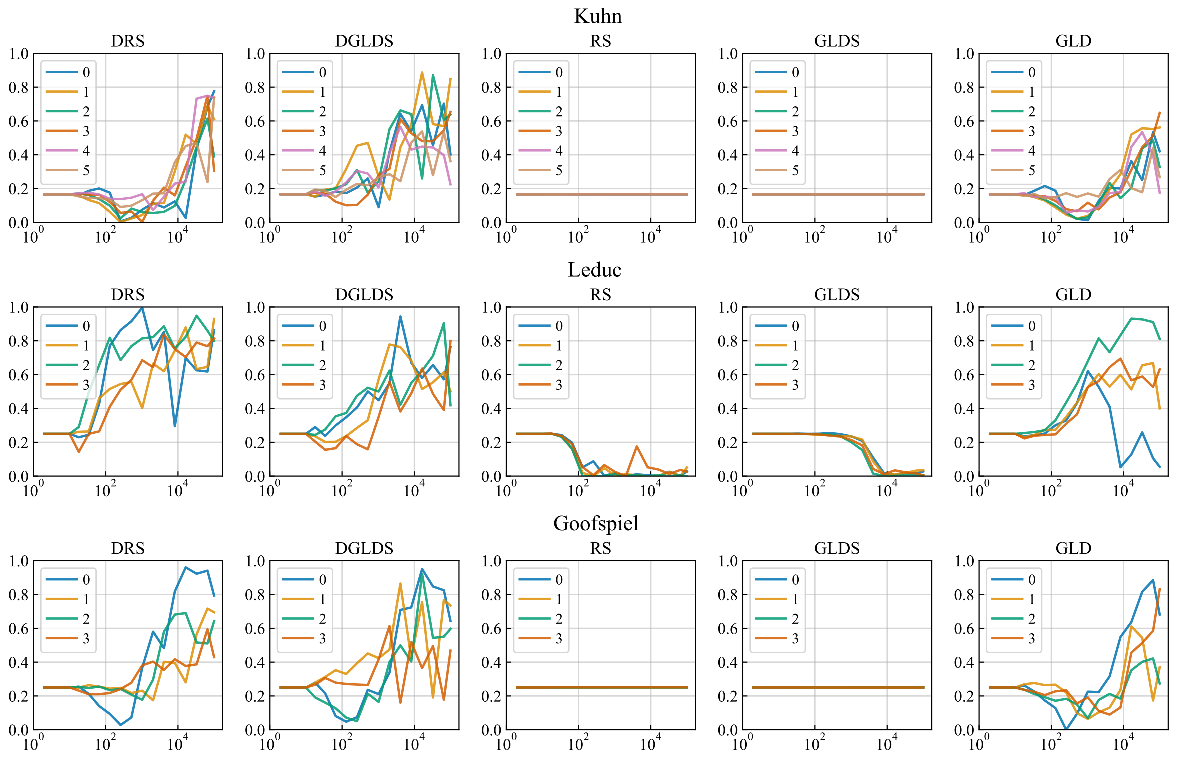


Figure 16. The evolution of the hyper-parameter values of different MCs in the **Zero-Sum** category. The y-axis is the value of $\boldsymbol{\alpha}$. The x-axis is the number of iterations.

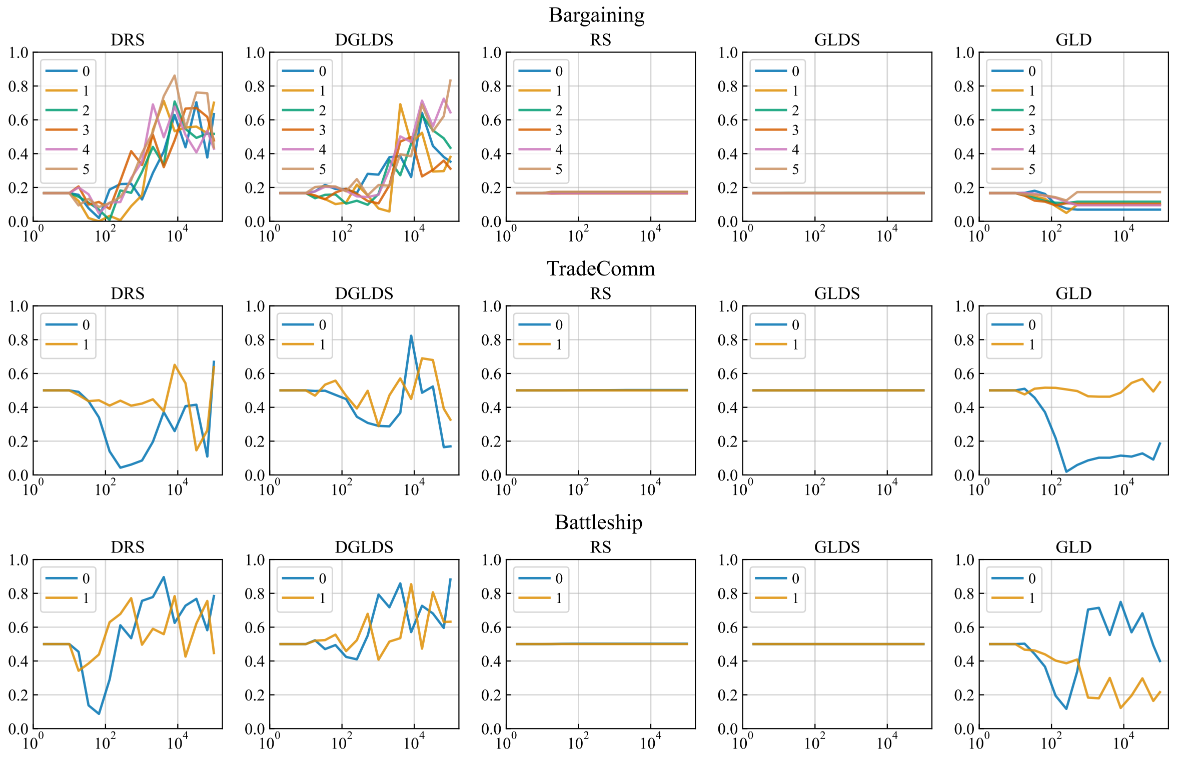


Figure 17. The evolution of the hyper-parameter values of different MCs in the **General-Sum** category. The y-axis is the value of $\boldsymbol{\alpha}$. The x-axis is the number of iterations.

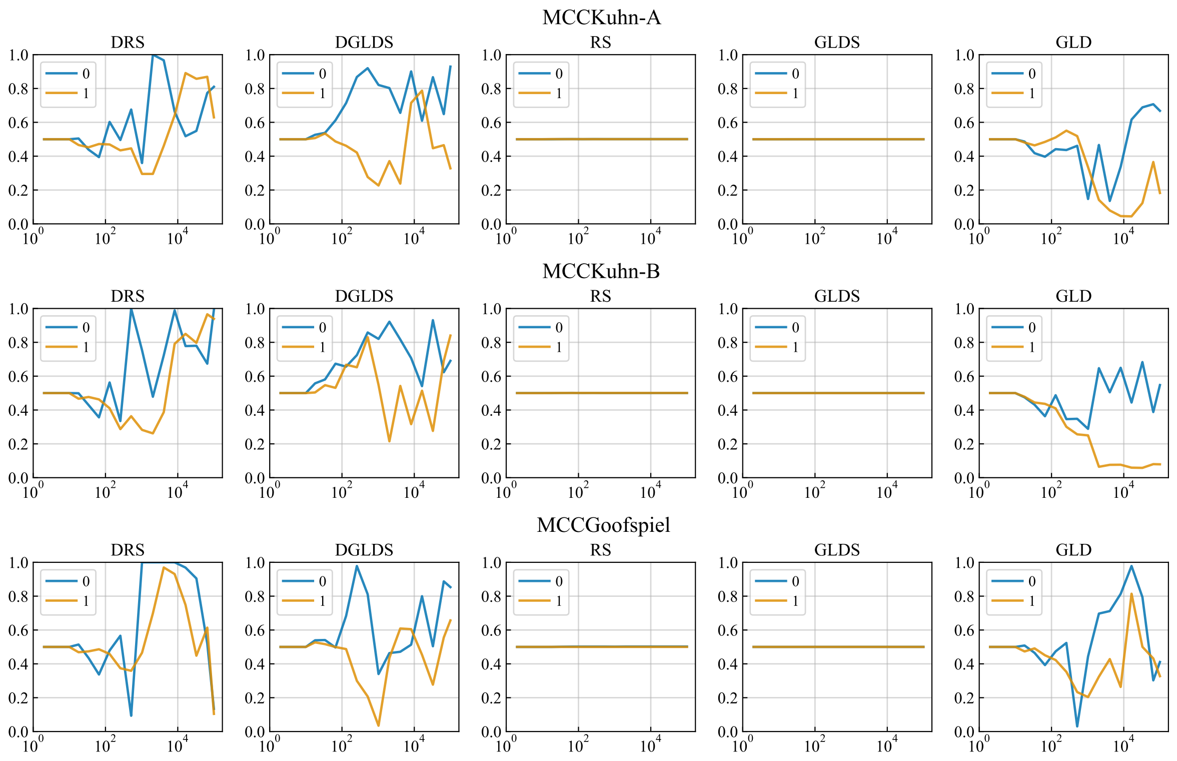


Figure 18. The evolution of the hyper-parameter values of different MCs in the **MCC** category. The y-axis is the value of $\boldsymbol{\alpha}$. The x-axis is the number of iterations.

F.6. Different Bregman Divergences

One of the prominent features of our CMD (GMD) is that it is capable of exploring more possible Bregman divergences. In this section, we investigate how CMD performs under the different Bregman divergences induced by different convex functions in Table 2 (the plots for these convex functions are shown in Figure 8).

The experimental results are shown in Figure 19. From the results, we can see that the entropy function $x \ln x$ is still a good choice across all the games. Nevertheless, in some games, there exist other convex functions that are better choices. For example, in Kuhn-A, TinyHanabi-B, TinyHanabi-C, Goofspiel, TradeComm, and MCCGoofspiel, x^2 is better than $x \ln x$. Furthermore, in MCCGoofspiel, e^x is also better than $x \ln x$, which verifies that the KL divergence ($x \ln x$) or squared Euclidean norm (x^2) could be not always the best choice across different games. On the other hand, even under the entropy function $x \ln x$, our CMD could also outperform MMD-KL in some games such as MCCKuhn-A and MCCKuhn-B.

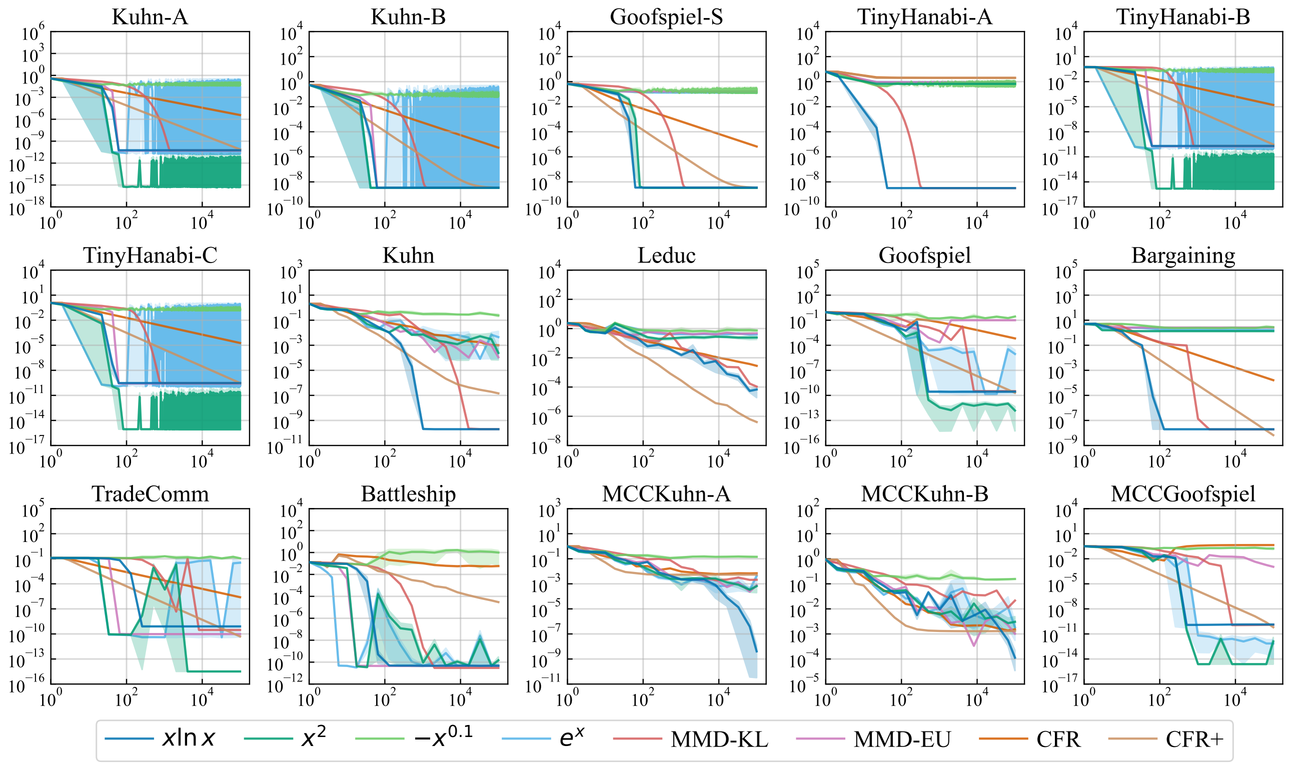


Figure 19. Experimental results for different Bregman divergences. The first 6 figures correspond to the single-agent and cooperative categories where the y-axis is *OptGap*. The rest figures correspond to other categories where the y-axis is *NashConv*. For all the figures, the x-axis is the number of iterations.

F.7. Effectiveness of Magnet

As mentioned in Section D.2, in our experiments, when instantiating CMD, we by default add a magnet policy (the initial policy) into the policy updating as it has been demonstrated that adding a magnet policy is powerful in solving two-player zero-sum games (Sokota et al., 2023; Liu et al., 2023). To verify this, we conduct an ablation study where "CMD w/o Mag" denotes the method that only considers the most recent M historical policies without adding the initial policy.

The experimental results are shown in Figure 20. From the results, we can see that i) For single-agent and cooperative categories, adding the magnet policy could result in a slightly slower convergence rate; we hypothesize that this may be due to the fact that the single-agent and cooperative games are relatively simpler than the other games (as shown in Table 6, the numbers of decision points of single-agent and cooperative games are smaller than the other games). ii) For the other three categories, adding the magnet policy is necessary for CMD to work consistently well across all the games; though in MCCGoofspiel, CMD finally converges to a lower NashConv without the magnet, it could perform worse in some other games, e.g., in Battleship, it finally diverges without the magnet, and in Leduc, MCCKuhn-A, and MCCKuhn-B, it converges to a high NashConv without the magnet. Nevertheless, as pointed out in Section D.2, this default instance of CMD (GMD) should not be confused with the original MMD even when $M = 1$ as the policy updating rule is derived via a numerical method, instead of relying on the closed-form solution (Sokota et al., 2023).

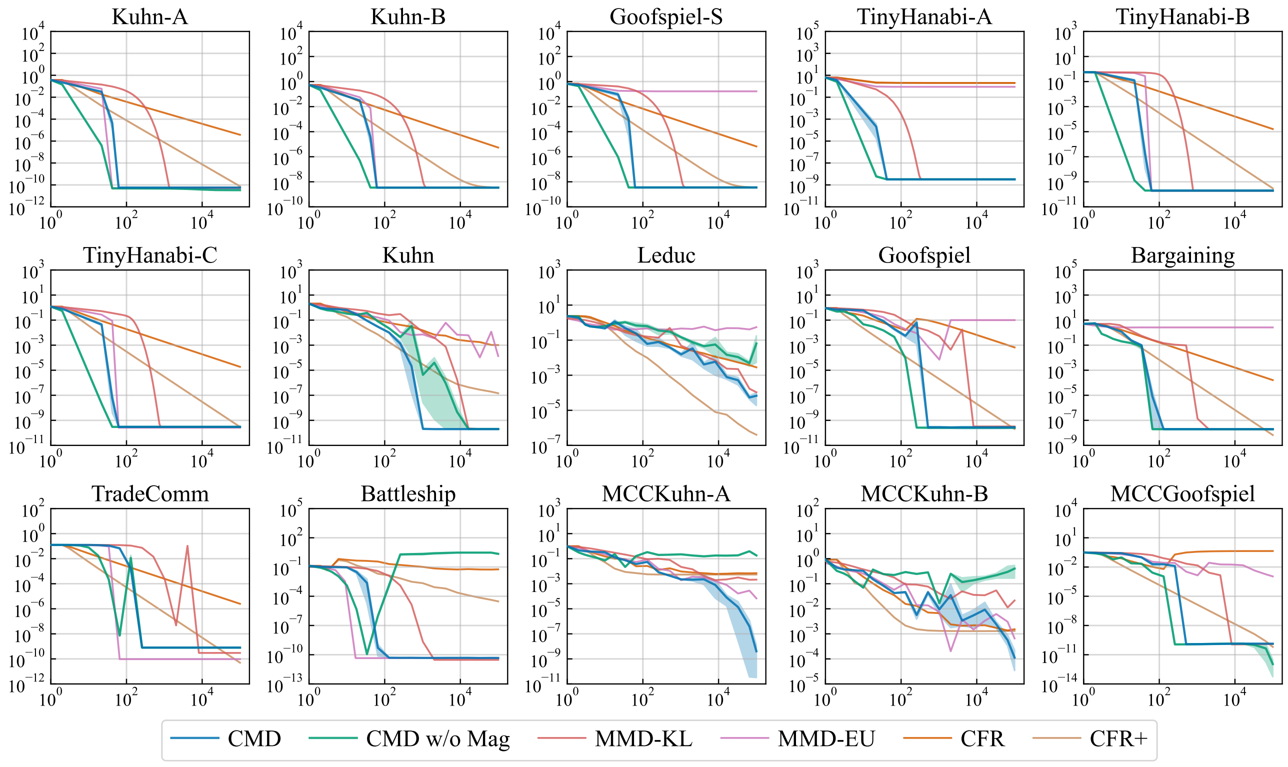


Figure 20. Experimental results for the effectiveness of adding the magnet policy. The first 6 figures correspond to single-agent and cooperative categories where the y-axis is *OptGap*. The rest figures correspond to other categories where the y-axis is *NashConv*. For all the figures, the x-axis is the number of iterations.

F.8. Different Measures

In this section, we apply our CMD to different solution concepts and evaluation measures (i.e., the desiderata **D3** and **D4** presented in the Introduction). Note that when running CMD for different evaluation measures, only minimal modifications are required: changing the MC's optimization objective $\mathcal{L}$. We first investigate the CCEGap (Moulin & Vial, 1978; Marris et al., 2021) in Section F.8.1 and then the social welfare (Davis & Whinston, 1962) in Section F.8.2.

F.8.1. CCEGap

Note that in two-player zero-sum games, NE and CCE can be shown to be payoff equivalent (v. Neumann, 1928). Therefore, we conduct experiments on the *three-player* Kuhn and Goofspiel. We follow the same experimental pipeline of OptGap and NashConv: i) investigating the combination of M and μ, ii) investigating different MCs, iii) investigating different Bregman divergences, and iv) investigating the effectiveness of magnet.

Number of Historical Policies and Smoothing Parameter. We first investigate the influence of the number of previous policies M and the smoothing parameter μ in DRS on the learning performance. Similar to OptGap/NashConv, we consider $M \in \{1,3,5\}$ and $\mu \in \{0.01, 0.05\}$, and the results are shown in Figure 21. We can get the same conclusion: different games may require different M and μ. By comparison, we determine their default values which will be fixed in the other experiments: $(M, \mu) = (3, 0.01)$ for both Kuhn and Goofspiel. All the other hyper-parameter settings are the same as OptGap/NashConv given in Table 7.

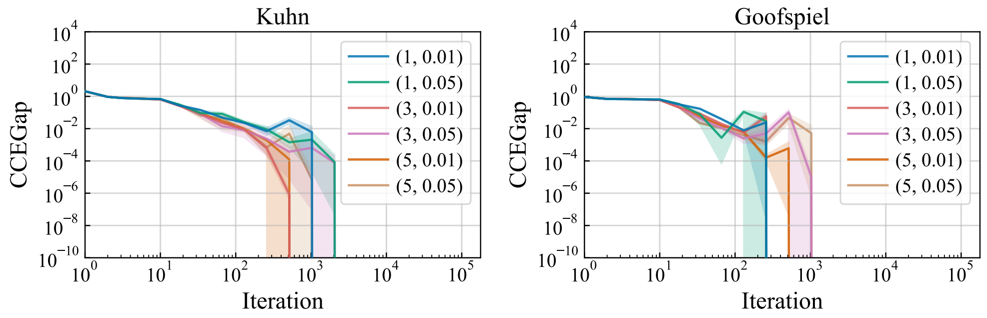


Figure 21. Experimental results for the combinations of (M, μ) under the measure CCEGap.

Different Meta-Controllers. Then, we investigate the effectiveness of different MCs. In Figure 22, we present the learning curves of the performance of different MCs, and in Figure 23, we present the evolution of the value of $\boldsymbol{\alpha}$ determined by different MCs over the learning process. We can get the same conclusion as OptGap/NashConv: DRS is the best choice among the 5 MCs. From the evolution of $\boldsymbol{\alpha}$ we observe that DRS and DGLDS follow two different patterns to determine the value of $\boldsymbol{\alpha}$, which is not the case for OptGap/NashConv (see Figure 14–Figure 18). In contrast, we found that since GLD follows a similar pattern to DRS, it performs on par with or better than DGLDS.

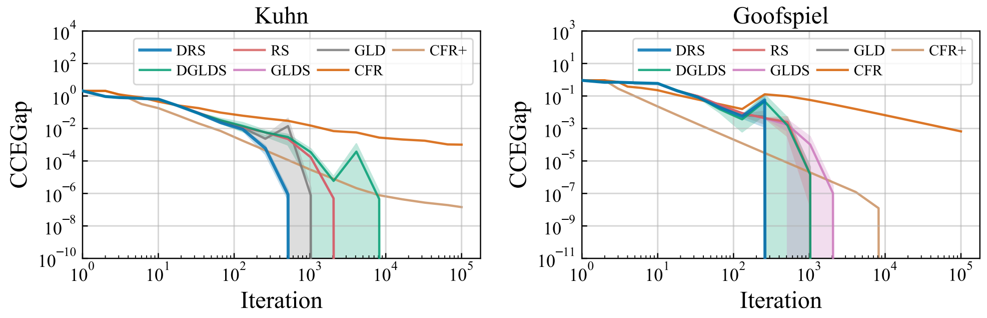


Figure 22. Experimental results for different meta-controllers under the measure CCEGap.

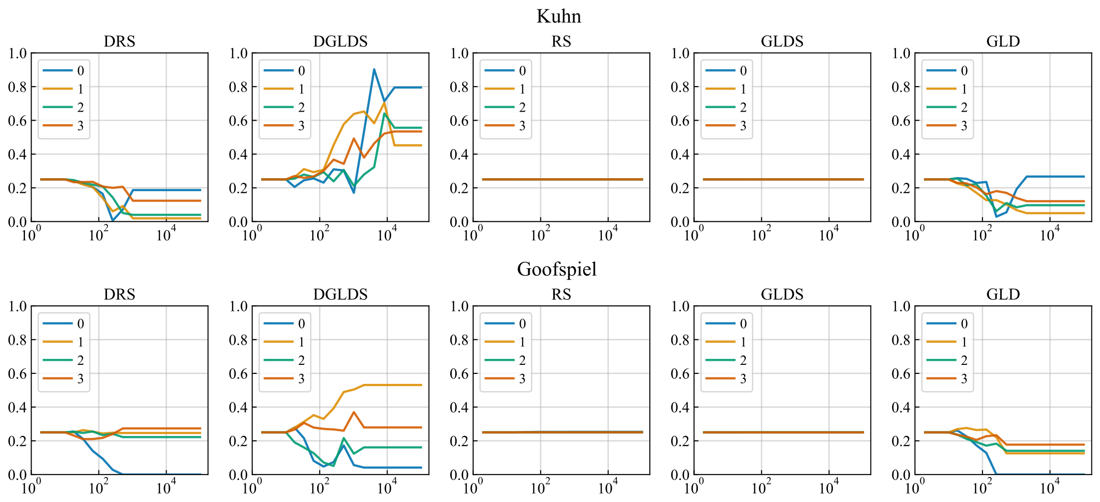


Figure 23. Evolution of the hyper-parameters of different MCs. The y-axis is the value of $\boldsymbol{\alpha}$. The x-axis is the number of iterations.

Different Bregman Divergences. Next, we investigate how CMD performs under different Bregman divergences induced by different convex functions in Table 2, and the results are given in Figure 24. We can get the same conclusions as for OptGap/NashConv: i) the entropy function $\psi(x) = x \ln x$ is still a good choice in different games, ii) there could exist other convex functions that are better than the entropy function, e.g., x^2 and e^x in Goofspiel, iii) even under the entropy function $\psi(x) = x \ln x$, our CMD can converge faster than the SOTA MMD-KL in terms of the number of iterations.

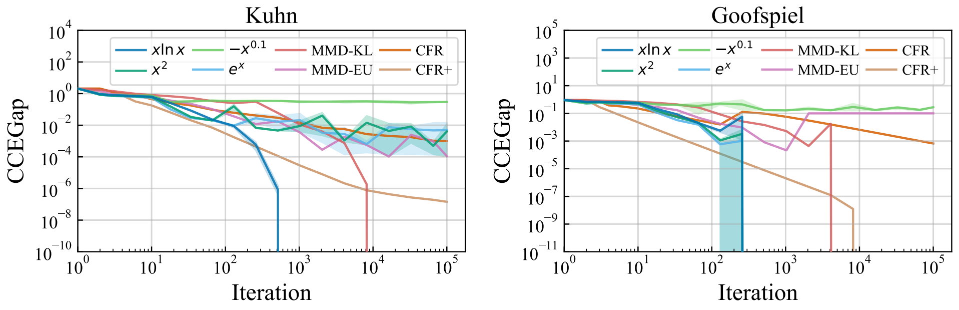


Figure 24. Experimental results for different Bregman divergences under the measure CCEGap.

Effectiveness of Magnet. Finally, we investigate the effectiveness of adding the magnet policy to the policy updating, and the results are presented in Figure 25. We can observe a similar phenomenon to NashConv: in Kuhn, CMD converges remarkably faster (in terms of the number of iterations) than all the other methods, and in Goofspiel, it converges remarkably faster (in terms of the number of iterations) than all the other methods except the “CMD w/o Mag”. Nevertheless, we can still conclude that adding the magnet policy is necessary for our CMD.

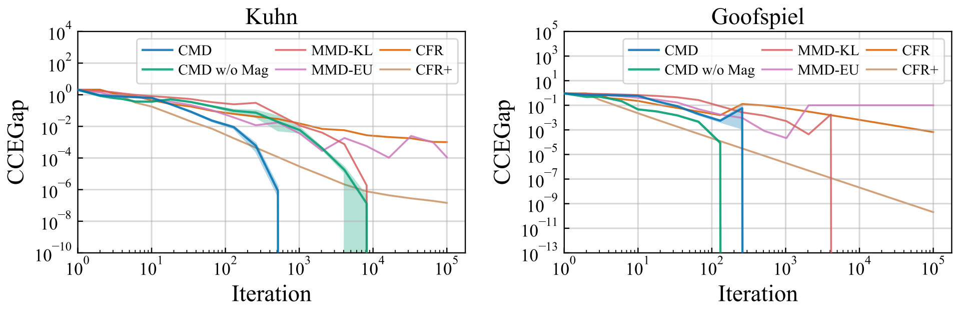


Figure 25. Experimental results for the effectiveness of the magnet policy under the measure CCEGap.

F.8.2. Social Welfare

In this section, we apply our methods to the evaluation measure – social welfare (Davis & Whinston, 1962). We conduct experiments on the general-sum games, and the results are shown in Figure 26. In this experiment, we use the default values in Table 7 for the hyper-parameters.

From the top line of the figure, we can see that CMD/GMD can empirically achieve competitive or better social welfare compared to other baselines, demonstrating the effectiveness of our method when considering different measures.

In the median line of the figure, we plot the NashConv when the MC's objective $\mathcal{L}$ is social welfare. We can see that in Bargaining and TradeComm, the learning still can converge to the approximate NE even though the MC's objective is social welfare, not the NashConv. In Battleship, while CMD can get a higher (average) social welfare, the final joint policy is not an NE as its NashConv cannot converge. In other words, an efficient (in terms of social welfare) joint policy could not necessarily be an NE. This suggests one of the future directions: how to efficiently learn the *NE with maximum social welfare*, which involves the equilibrium selection problem (Harsanyi et al., 1988).

Another intuitive consequence of setting the MC's optimization objective to social welfare is that the learning could not converge to the NE or converge slower than the case where the MC's objective is directly the NashConv. As shown in the bottom line of the figure, we plot the NashConv of the two cases. In Bargaining and TradeComm, CMD with NashConv as the MC's objective can converge to the NE faster than that with SW as the MC's objective. In Battleship, CMD with SW as the MC's objective, though could achieve a higher social welfare, cannot converge the Nash equilibrium.

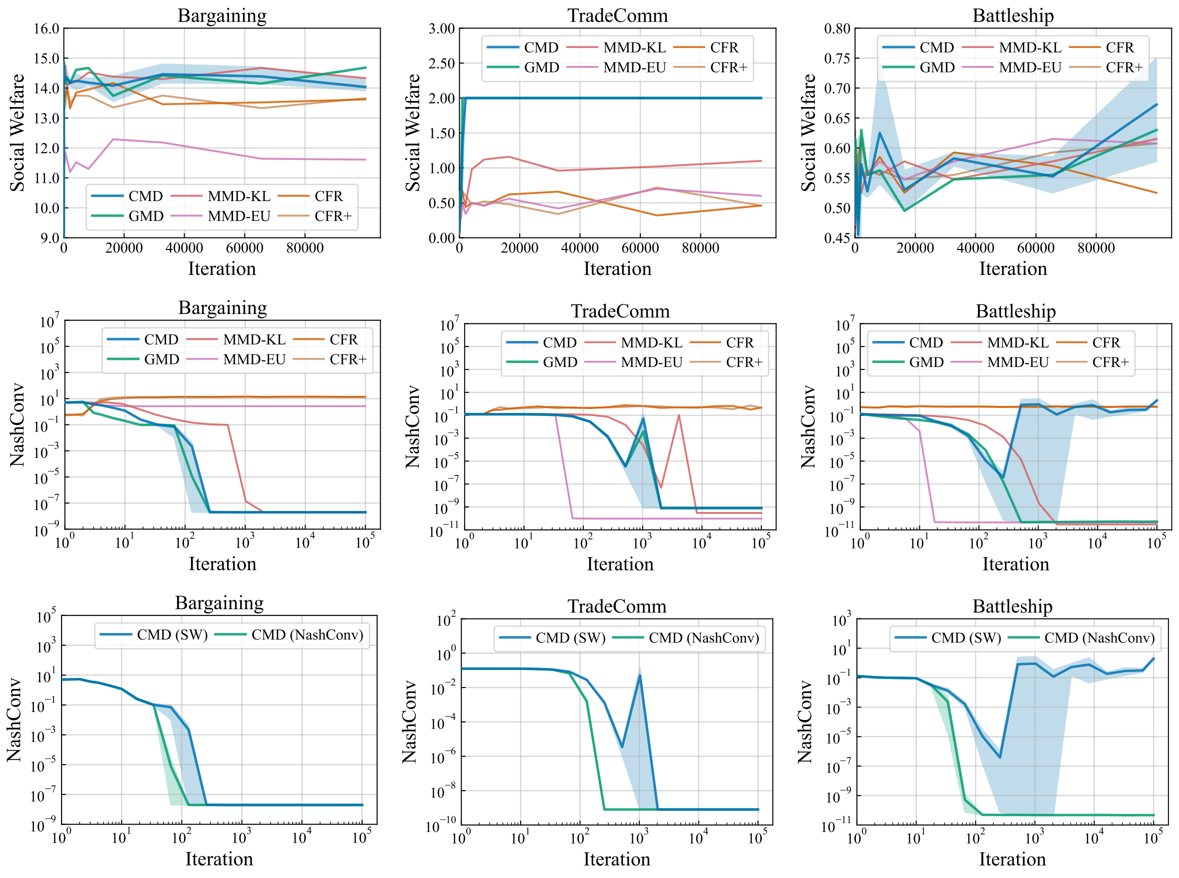


Figure 26. Experimental results for the evaluation measure–social welfare.

F.9. Computational Complexity

In this section, we give some remarks on the computational complexity of different methods. In Section F.9.1, we focus on the running time of different methods, and in Section F.9.2 we present the memory consumption of different methods. Note that i) these numbers are obtained under the default values of hyper-parameters (Table 7), and ii) these numbers are not absolute and depend on the property of the game (see Table 6) and the computational resources used to run the experiments (see Section F.1); they only provide intuition on the computational complexity of different methods to show that our GAMEBENCH can satisfy the desiderata **D5** and **D6** mentioned in the Introduction.

F.9.1. Running Time

The running time of different methods in different games is shown in Table 8. From the results, we can see that in most of the games, running all the algorithms does not cause a very long running time (the desiderata **D5** and **D6** presented in the Introduction), even for our methods where a set of extra operations is required: GMD requires computing the value of the dual variable via a numerical method and CMD further requires to evaluate multiple candidates of the hyper-parameters. Notably, we emphasize that: i) Our methods (CMD/GMD) provide the capability of exploring more dimensions of decision making, though they require extra computational cost (the major limitation of the current version of our methods); ii) Comparing CMD and GMD, we can see that the major cost comes from evaluating multiple samples. Therefore, as pointed out in Section 8, we view this as a future direction: developing more computationally efficient hyper-parameter value updating methods without sacrificing performance. In this regard, other techniques such as Bayesian optimization (Lindauer et al., 2022) or offline hyper-parameter optimization approaches (Chen et al., 2022) may be required.

Table 8. The running time of **one iteration** of different methods in different games (second).

Game	CFR	CFR+	MMD -KL	MMD -EU	GMD	CMD				
						DRS	RS	DGLDS	GLDS	GLD
Kuhn-A	0.0004	0.0004	0.0003	0.0003	0.0034	0.0372	0.0372	0.0204	0.0204	0.0203
Kuhn-B	0.0004	0.0004	0.0003	0.0003	0.0033	0.0370	0.0365	0.0202	0.0201	0.0202
Goofspiel-S	0.0007	0.0006	0.0004	0.0004	0.0046	0.0491	0.0489	0.0269	0.0275	0.0270
TinyHanabi-A	0.0006	0.0006	0.0004	0.0004	0.0045	0.0474	0.0472	0.0262	0.0268	0.0260
TinyHanabi-B	0.0004	0.0004	0.0003	0.0003	0.0033	0.0366	0.0370	0.0198	0.0192	0.0197
TinyHanabi-C	0.0004	0.0004	0.0003	0.0003	0.0032	0.0364	0.0359	0.0195	0.0195	0.0199
Kuhn	0.0084	0.0082	0.0022	0.0021	0.0267	0.4102	0.4058	0.2273	0.2282	0.2288
Leduc	0.0942	0.0961	0.0422	0.0412	0.5146	6.8897	6.9749	3.8188	3.8116	3.8744
Goofspiel	0.0072	0.0073	0.0014	0.0014	0.0167	0.2879	0.2899	0.1636	0.1610	0.1625
Bargaining	0.0279	0.0273	0.0130	0.0116	0.1093	1.5311	1.5308	0.8428	0.8579	0.8516
TradeComm	0.0028	0.0029	0.0011	0.0010	0.0121	0.1704	0.1699	0.0957	0.0942	0.0939
Battleship	0.0245	0.0248	0.0097	0.0094	0.1125	1.5570	1.5771	0.8833	0.8774	0.8835
MCCKuhn-A	0.0083	0.0084	0.0021	0.0021	0.0264	6.9054	6.8377	4.0453	4.1461	4.1034
MCCKuhn-B	0.0080	0.0083	0.0021	0.0021	0.0260	6.8100	6.7847	4.1047	4.0944	4.1104
MCCGoofspiel	0.0070	0.0073	0.0014	0.0014	0.0167	5.3541	5.3852	3.1817	3.1945	3.1836

F.9.2. Memory Usage

The memory usage of different methods in different games is provided in Table 9. From the results, we can see that running all the algorithms does not cause much memory consumption, which shows that our GameBench is academic-friendly.

Table 9. The memory usage of different methods in different games (MB).

Game	CFR	CFR+	MMD -KL	MMD -EU	GMD	CMD				
						DRS	RS	DGLDS	GLDS	GLD
Kuhn-A	0.8750	0.9805	0.3711	0.3750	0.9492	1.0352	1.0859	0.3750	0.3750	0.3750
Kuhn-B	0.8672	0.8672	0.4297	0.3750	0.7461	1.1289	1.0352	0.3750	0.3750	0.3164
Goofspiel-S	1.1875	1.2969	1.2617	1.2539	1.2578	1.2578	1.2578	1.6992	1.6953	1.6992
TinyHanabi-A	1.0352	1.0156	0.4805	0.4258	0.4883	1.0312	1.1836	0.4453	0.4297	0.4883
TinyHanabi-B	0.9922	1.0898	0.4922	0.4844	0.4336	0.4922	0.5430	0.4922	0.4297	0.4922
TinyHanabi-C	0.9805	0.9922	0.4336	0.4297	0.4336	0.4922	0.4336	0.4336	0.4336	0.4336
Kuhn	1.9961	2.0078	3.0352	3.1367	3.5586	3.7734	3.7227	3.5156	3.5352	3.5273
Leduc	26.262	26.344	51.664	52.465	58.273	59.063	58.555	52.688	51.949	51.359
Goofspiel	2.4844	2.4297	3.5039	3.4961	4.3555	4.3359	4.3047	4.0391	4.0430	4.0898
Bargaining	10.633	10.578	24.816	24.852	35.129	35.422	35.020	31.941	32.344	31.781
TradeComm	1.4961	1.5430	2.0156	2.0703	2.0664	2.0078	2.0117	2.2461	2.2461	2.3633
Battleship	6.9102	6.9023	13.539	13.422	13.543	13.543	13.484	12.098	12.148	12.332
MCCKuhn-A	2.5742	2.5195	2.0312	2.0273	2.5586	2.6719	2.7266	2.3047	2.2930	2.2930
MCCKuhn-B	2.4688	2.5703	2.0273	1.9648	1.9727	2.6562	2.7617	2.2383	2.2305	2.1797
MCCGoofspiel	2.7500	2.7500	3.0078	3.0820	3.0781	3.0195	3.1289	2.8203	2.8047	2.8672