A REGRET MINIMIZATION APPROACH TO MULTI-AGENT CONTROL

Udaya Ghai*
Google AI Princeton & Princeton University

Udari Madhushani
Princeton University

Naomi Leonard
Princeton University

Elad Hazan
Google AI Princeton & Princeton University

ABSTRACT

We study the problem of multi-agent control of a dynamical system with known dynamics and adversarial disturbances. Our study focuses on optimal control without centralized precomputed policies, but rather with adaptive control policies for the different agents that are only equipped with a stabilizing controller. We give a reduction from any (standard) regret minimizing control method to a distributed algorithm. The reduction guarantees that the resulting distributed algorithm has low regret relative to the optimal precomputed joint policy. Our methodology involves generalizing online convex optimization to a multi-agent setting and applying recent tools from nonstochastic control derived for a single agent. We empirically evaluate our method on a model of an overactuated aircraft. We show that the distributed method is robust to failure and to adversarial perturbations in the dynamics.

1 INTRODUCTION

In many real-world scenarios it is desirable to apply multi-agent control algorithms, as opposed to a centralized designed policy, for a number of practical reasons. First is increased robustness: agents, serving as control actuators, may be added or removed from an existing set, without change to the distributed policies. This also allows for fault-tolerance; agents can adapt on the fly to faulty actuators. Second, limited computational resources and/or system information may limit applicability of a sophisticated centralized policy, whereas simple distributed policy agents are feasible and can adapt to the underlying dynamics. Third, multi-agent control allows robustness to noisy state and control observations by the individual agents. These advantages motivate the study of distributed control, and indeed a vast literature exists on the design and analysis of such methods.

In this paper we propose a novel game-theoretic performance metric for multi-agent control (MAC), which we call **multi-agent regret**. This metric measures the difference between the total loss of the joint policy of individual agents vs. the loss of the best joint policy in hindsight from a certain reference class of policies. Vanishing multi-agent regret thus implies competitive performance with respect to that of the optimal policy in a certain policy class.

We study a mechanism for MAC that takes individual controllers and merge them into a MAC method with vanishing regret. For this purpose, we make use of recent advances in online learning for control and specifically the framework of *nonstochastic control*. This methodology was recently proposed for the study of robust adaptive control algorithms through the lens of online convex optimization. As opposed to optimal and robust control, nonstochastic control studies adaptive control algorithms that minimize *regret*, or the difference in loss/reward of the controller vs. the best policy in hindsight from a reference class. This is a game-theoretic performance metric, which is naturally applicable to agents interacting in a multi-agent environment.

*Corresponding author; ughai@cs.princeton.edu

1.1 Our result and techniques

Our main result is a reduction from single agent to multi-agent control with provable regret guarantees. More precisely, we assume that we have access to single-agent controllers with the following guarantees:

1. Each agent has sublinear regret vs. a class of policies under adversarial perturbations/costs.
2. Each agent should be able to evaluate their policy given the controls of the other agents.
3. The policy that each agent uses is independent of the controls that the other agents play.
4. The joint cost function over all agents needs to be jointly convex.

This set of assumptions is well motivated: we review several well-studied control settings in the literature which provide agents with such guarantees. Under these assumptions we give a reduction that takes regret minimizing control agents and guarantees low multi-agent regret of the joint control policy, w.r.t. the optimal joint policy in hindsight.

Let $\mathcal{A}$ be a single agent control algorithm[1] that, given a state x_t, produces a control u_t, and suffers cost $c_t(x_t, u_t)$. The average (policy) regret of $\mathcal{A}$ w.r.t. policy class Π is defined to be the difference between its total cost and that of the best policy in hindsight, namely

$$\mathcal{R}_T(\mathcal{A}) = \frac{1}{T}\sum_t c_t(x_t, u_t) - \min_{\pi\in\Pi} \frac{1}{T}\sum_t c_t(x_t^\pi, u_t^\pi) \ .$$

Notice that the comparator are the states x_t^π and controls u_t^π that would have been played by policy π. Then algorithm $\mathcal{C}$, described in Algorithm 3, takes as an input k such controllers, and under the aforementioned assumptions, controls a multi-agent system with the guarantee that

$$\mathcal{R}_T(\mathcal{C}) \leq \sum_{i=1}^{k} \mathcal{R}_T(\mathcal{A}_i) + \tilde{O}\left(\frac{1}{T} + \varepsilon\right) \ ,$$

where $\mathcal{A}$ is the control algorithm for agent i, which is not necessarily the same for every agent, and ε is an upper bound on the error of each agent's policy evaluation. Here $\mathcal{R}_T(\mathcal{C})$ is the multi-agent regret of the joint policy, namely the difference between the cost of $\mathcal{C}$, and that of the best joint-policy in hindsight. The reference policy class we use for comparison is the Cartesian product of the individual agent policy classes. This is defined precisely in Section 4.

In order to obtain this result we study a generalization of online convex optimization to the multi-agent setting. Specifically, we propose a reduction from online convex optimization (OCO) to multi-agent OCO, such that the overall multi-agent regret is guaranteed to be the sum of the individual agents' regret on an induced loss function.

1.2 Related work

Multi-agent RL. Multi-agent RL (Sutton & Barto, 2018) considers the problem of finding globally optimal joint policies through optimizing local policies. Common approaches for finding optimal policies in RL includes dynamic programming, Monte-Carlo methods (Williams, 1992), temporal difference learning (Watkins & Dayan, 1992), combining temporal difference and Monte-Carlo learning, evaluation strategies (Salimans et al., 2017), and policy gradient methods (Williams, 1992; Mnih et al., 2016). Policy gradient methods, which are closest to our algorithm, compute stochastic gradients from trajectories (Sutton et al., 2000; Sutton & Barto, 2018). More recently, Jin et al. (2021) provide a decentralized value-based method with game-theoretic equilibrium guarantees. Performance in multi-agent coordination significantly improves with agents' ability to predict the behaviour of other agents. In opponent modeling methods, wherein agents learn to model the behaviours of their partners, (Foerster et al., 2018) agents use counterfactual policies of other agents to update their policy parameters. In contrast, our approach decouples the agents' policies, forgoing the need to deal with counterfactuals of other agents.

Decentralized and Distributed control. Communication-free decentralized control is the earliest multi-agent control model considered in the literature. In this model, each agent observes a different partial observation of the state and each agent provides its own control input to the system. This classic setting is well understood with exact characterizations of stabilizability (Wang & Davison,

[1]More formally, it is an online convex optimization with memory algorithm, which we precisely define in coming sections.

1973) and controllability (Lefebvre et al., 1980). More recently, decentralized control, also called distributed control, has been generalized to include a network layer of communication among the controllers, modeled by a network graph. Coordinated and cooperative control over a communication network has been studied using a wide range of approaches, such as distributed optimization (Nedić et al., 2018), model predictive control (Christofides et al., 2013; Stürz et al., 2020), and for a wide variety of problems, including stabilization of formations, coverage, and search (Olfati-Saber et al., 2007; Bullo et al., 2009; Cao et al., 2013; Knorn et al., 2016).

One particular direction in this setting is on designing distributed observers, using information from neighbors to provide local estimates of the global state (Olfati-Saber, 2005; Carli et al., 2008; Matei & Baras, 2010; Das & Moura, 2017; Park & Martins, 2017). Such distributed observers can be applied to produce end-to-end control laws. One such approach stabilizes systems when the neighbor graph is strongly connected and the joint system is stabilizable (Wang et al., 2020). In contrast to the present work, approaches to this problem are decentralized from an *information* perspective, but not a *game theoretic* perspective as controllers are designed centrally.

Exploiting input redundancy in control. Input redundancy in overactuated systems enables efficient responses (Sobel & Shapiro, 1985) and robustness to failures (Oppenheimer et al., 2006; Tohidi et al., 2017). A number of optimization methods have been developed for redistributing desired control input under different faulty conditions. Our approach does not tackle this directly, but the action-centric rather than policy-centric philosophy of the algorithm leads to some robustness in such settings.

Online learning and online convex optimization. The framework of learning in games has been extensively studied as a model for learning in adversarial and nonstochastic environments (Cesa-Bianchi & Lugosi, 2006). Online learning was infused with algorithmic techniques from mathematical optimization into the setting of online convex optimization , see (Hazan, 2019) for a comprehensive introduction. We particularly make use of the property that no-regret algorithms converge to equilibrium, a phenomenon originally proposed by (Hart & Mas-Colell, 2000).

Online and nonstochastic control. The starting point of our study are algorithms which enjoy sublinear regret for online control of dynamical systems; that is, whose performance tracks a given benchmark of policies up to a term which is vanishing relative to the problem horizon. (Abbasi-Yadkori & Szepesvári, 2011) initiated the study of online control under the regret benchmark for Linear Time-invariant (LTI) dynamical systems. Our work instead adopts the *nonstochastic control setting* (Agarwal et al., 2019), that allows for adversarially chosen (i.e. non-Gaussian and potentially adaptively chosen (Ghai et al., 2021)) noise and costs that may vary with time. The nonstochastic control model was extended to consider nonlinear and time-varying dynamics in (Gradu et al., 2020; Minasyan et al., 2021; Chen et al., 2021). See (Hazan & Singh, 2021) for a comprehensive survey of results and advances in online and nonstochastic control.

2 Problem Setting

2.1 Notation

The norm $\|\cdot\|$ refers to the ℓ_2 norm for vectors and spectral norm for matrices. For any natural number n, the set $[n]$ refers to the set $\{1, 2 \dots n\}$. We use the window notation $z_{a:b} := [z_a, \dots, z_b]$ to represent a sequence of vectors. We use the convention where indices i, j refer to agents, with superscripts (typically) denoting an agent-specific quantity, while lack of a superscript denotes something global. We use the index $-i$ to represent information for all agents except i, for example $u_t^{-i} = (u_t^1 \dots u_t^{i-1}, u_t^{i+1} \dots u_t^k)$.

2.2 Multi-Agent Nonstochastic Control Problem

We consider the following multi-agent control problem. Let $f_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \to \mathbb{R}^{d_x}$ and $g_i : \mathbb{R}^{d_x} \to \mathbb{R}^{d_{y_i}}$ define known time varying dynamics. Now, starting from state x_0, the system follows

$$y_t^i = g_i(x_t) + e_t^i, \quad x_{t+1} = f_t(x_t, u_t) + w_t, \tag{1}$$

where x_t is the state, u_t is the joint control consisting of $u_t^i \in \mathbb{R}^{d_{u_i}}$ for each $i \in [k]$, y_t^i is a partial observation, and e_t^i, w_t are adversarial chosen disturbance terms. After committing to a control u_t^i, agent i observes the controls of all other players u_t^{-i} along with a convex cost $c_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \to \mathbb{R}$.

Together, the k agents suffer cost $c_t(x_t, u_t)$. It should be noted that while each agent shares its action with other agents, it does not directly share its policy or its observation.

A control policy for agent i, $\pi^i : \bigcup_{s\in\mathbb{N}} \mathbb{R}^{sd_{y_i}} \times \mathbb{R}^{(s-1)d_u} \to \mathbb{R}^{d_u}$ is a mapping from all available information to a control $u_t^i = \pi^i(y_{1:t}, u_{1:t-1})$. A joint policy for the multi-agent setting is a collection of policies for each agent $\pi = \pi^{1:k}$. Policy classes are defined analogously with Π^i being a set of agent i policies and Π being a set of joint policies. We often consider policies parameterized by vectors $\theta^i \subseteq \Theta_i$, where Θ_i is some bounded convex domain and use the parameterizations and policies interchangeably.

Performance of joint policy π is measured by the average cost along the trajectory $(x_1^\pi, u_1^\pi \dots)$,

$$J(\pi|w_{1:T}, e_{1:T}) = \frac{1}{T}\sum_{t=1}^{T} c_t(x_t^\pi, u_t^\pi) \ .$$

For a set of online control algorithms for each agent $\mathcal{C} = \mathcal{C}^{1:k}$, we are interested in the average (policy) *regret* relative to some joint policy class Π, defined as

$$\mathcal{R}_T(\mathcal{C}) = J(\mathcal{C}|w_{1:T}, e_{1:T}) - \inf_{\pi\in\Pi} J(\pi|w_{1:T}, e_{1:T}) \ .$$

2.3 Assumptions

Converging towards optimal play for an online control algorithm may not happen immediately, so our system cannot explode too quickly. In order to bound the cost during this stage, x_t must not explode in the absence of control. The corresponding outputs in the absence of any controller is a concept called *Nature's y's* in (Simchowitz et al., 2020), which are an integral part of DAC policies, defined below. We analogously define *Nature's x's* here.

Definition 2.1 (Nature's x's and y's)**.** Given a sequence of disturbances $(w_t, e_t)_{t>1}$ we define Nature's x's and Nature's y's respectively as the sequences $x_t^{\text{nat},i}, y_t^{\text{nat},i}$ where

$$y_t^{\text{nat},i} = g_i(x_t) + e_t^i, \quad x_{t+1}^{\text{nat}} = f_t(x_t, 0) + w_t \ .$$

Assumption 2.2 (Bounded Nature's x's)**.** We assume that w_t and e_t are chosen by an oblivious adversary, and that $\|x_t^{\text{nat}}\| \le R_{\text{nat}}$ for all t.

We also need well behaved cost functions for learning:

Assumption 2.3. The costs $c_t(x, u)$ are convex and if $\|x\|, \|u\| \le D$ then there exists constant C such that $0 \le c_t(x, u)) \le CD^2$.

In control, the most ubiquitous cost functions are quadratics, which satisfy this assumption. Finally, we must restrict our joint policy class Π in order to obtain global regret guarantees. We make the following assumption about each agent's policy class:

Assumption 2.4 (**Decoupling**)**.** For any policy $\pi^i \in \Pi^i$, the policy can always be represented as a function of the disturbances e_t^i, w_t. In particular, there exists a function $\bar{\pi} : \bigcup_{s\in\mathbb{N}} \mathbb{R}^{(s-1)d_x} \times \mathbb{R}^{sd_{y_i}} \to \mathbb{R}^{d_u}$ for any set of controls $u_{1:t-1}$ and the resultant set of observations $y_{1:t}$, such that $\pi(y_{1:t}, u_{1:t-1}) = \bar{\pi}(w_{1:t-1}, e_{1:t}^i)$.

Note, that if Assumption 2.4 holds for each agent's policy class, then the joint policy class Π will depend implicitly on only $w_{1:t-1}, e_{1:t}$. Policies that satisfy this assumption are special in that the controls played by one agent's policy are completely independent of another agent's policy, given the disturbances are chosen obliviously.

2.4 Policy Classes

We now introduce a few important policy classes that satisfy Assumption 2.4.

Disturbance-action Control. For fully observed linear systems, disturbances w_t can be derived from states and joint controls. This allows us to use Disturbance Action Control, a policy parameterization that is linear in the disturbance history.

Definition 2.5. A Disturbance-action Controller (DAC) (Agarwal et al., 2019), $\pi(M)$, is specified by parameters $M \in \mathbb{R}^{d_{u_i}\times md_x}$ where the control $u_t^i = Mw_{t-m:t-1}$.

A comparator policy class Π^i can be defined as a bounded convex set $\mathcal{M} \subset \mathbb{R}^{d_u \times md_x}$ of policy matrices. It has been shown that DACs can approximate stable Linear Dynamic Controllers (LDCs), a powerful generalization of linear controllers that encompasses $\mathcal{H}_2$ and $\mathcal{H}_\infty$ optimal controllers under partial observation, and is a mainstay in control applications (see Definition A.1 in Appendix A.2). Furthermore, when costs are quadratic, it can be shown that a DAC also approximates the offline optimal policy (Goel & Hassibi, 2021).

Disturbance-response Control. Another policy class that suits our needs, especially if the system is partial observed, is that of Disturbance-Response Control (DRC). Unlike in the fully observed setting, with partial observations it is generally impossible for an agent to calculate w_t for use in a policy. Instead, DRC controllers are a linear function of a window of previous Nature's y's defined as follows:

Definition 2.6. A Disturbance-response Controller (DRC) (Simchowitz et al., 2020), $\pi(M)$, is specified by parameters $M \in \mathbb{R}^{d_u \times md_y}$ where the control $u_t^i = My_{t-m:t-1}^{\text{nat}}$.

Like DAC, DRC is a powerful control law which also can approximate stable LDCs. Furthermore, Nature's y's are just a function of w's and e's so DRC policy classes satisfy Assumption 2.4.

Neural Network Policies We also consider neural network policies. In particular, fully-connected neural-networks with ReLU activations that act on normalized windows of w's or Nature's y' were explored in (Chen et al., 2021). While there are more restrictions for this class, neural-networks are a richer policy class than the linear counterparts DAC and DRC.

3 Multiagent Online Convex Optimization

Our derivation henceforth for the multi-agent control problem hinges upon a generalization of OCO to the multi-agent setting. Now we describe this generalization, explain the challenge, and give a reduction from vanilla OCO, as well as OCO with Memory (OCO-M), to the multi-agent settings.

In multiplayer OCO we have k players and the ith learner iteratively chooses a vector from a convex set $\mathcal{K}_i \subseteq \mathbb{R}^d$. We denote the total number of rounds as T. In each round, the ith learner commits to a decision $x_t^i \in \mathcal{K}_i$. We define the joint decision $x_t = x_t^{1:k} \in \mathcal{K} = \mathcal{K}^{1:k}$. Afterwards, the learner incurs a loss $\ell_t(x_t)$ where ℓ_t is a convex loss function $\ell_t : \mathcal{K} \to \mathbb{R}$. The learner observers $\ell_t^i : \mathcal{K}_i \to \mathbb{R}$ defined by $\ell_t^i(x) = \ell_t(x, x_t^{-i})$, where we use the notation (x, x_t^{-i}) to represent $(x_t^1, \dots x_t^{i-1}, x, x_t^{i+1}, \dots x_t^k)$.

Regret is defined to be the total loss incurred by the algorithm with respect to the loss of the best fixed single prediction found in hindsight for each player. Formally, the average **multi-agent regret** of a set of learning algorithms $\mathcal{A}_{1:k} = \mathcal{A}_1, \dots \mathcal{A}_k$ is defined as

$$\mathcal{R}_T(\mathcal{A}_{1:k}) \stackrel{\text{def}}{=} \sup_{\ell_1 \dots \ell_t} \left\{ \frac{1}{T} \sum_{t=1}^{T} \ell_t(x_t) - \min_{x^* \in \mathcal{K}} \frac{1}{T} \sum_{t=1}^{T} \ell_t(x^*) \right\}.$$

Our goal is to create a black-box mechanism that takes as input regret minimizing strategies, and allows for joint-regret minimization with minimal or no central coordination.

3.1 Challenges of multi-agent OCO

A natural first attempt for joint regret minimization in multi-agent OCO is to allow each regret minimizing agent to operate on its own on the given loss function.

In this subsection we show that this naive strategy fails. As an example, consider the simple scalar two player game. Let $\ell_t(x^1, x^2) = (x^1 - x^2)^2 + 0.1\|(x^1, x^2)\|_2^2$ for all t. Now we consider $\mathcal{A}_1$ and $\mathcal{A}_2$ to be algorithms that each alternate between playing 1 on odd t and -1 on even t. In this setup, the online learners see losses:

$$\ell_t^1(z) = \ell_t^2(z) = \begin{cases} (z-1)^2 + 0.1z^2 + 0.1 & t \text{ is odd} \\ (z+1)^2 + 0.1z^2 + 0.1 & t \text{ is even} \end{cases}$$

By design of $\mathcal{A}_1$ and $\mathcal{A}_2$, predictions alternate between $(1, 1)$ and $(-1, -1)$ and $\ell_t^1(x_t^1) = \ell_t^2(x_t^2) = 0.2$. In contrast, the best fixed decision in hindsight is 0 for both players, suffering loss $\ell_t^1(0) = \ell_t^2(0) = 1.1$ for all t, so each player has *negative* regret. However, for the multiplayer game, the best loss in hindsight is 0 by playing $x_t^1 = x_t^2 = 0$ for all t, so the multi-agent regret is $0.2T$. We

note that this phenomenon is well known in the game theory literature, where failure of no-regret algorithms to converge to Nash equilibrium has been studied (Blum et al., 2008). Our study extends this observation to the dynamic games setting.

3.2 Reduction from OCO to multi-agent OCO

To overcome the lower bound of the previous section, we choose to apply the reduction from multi-agent to single agent OCO via the instantaneous linearization. This simple change makes all the difference, and allows for provable sublinear multi-agent regret (See Appendix. A.1 for proof).

Algorithm 1 Multiplayer OCO

Input: OCO algorithm $\mathcal{A}_i$, convex domain $\mathcal{K}_i \subseteq \mathbb{R}^d$
for $t = 1$ **to** T **do**
  Learner $\mathcal{A}_i$ predicts x_t^i
  Calculate $g_t^i = \nabla \ell_t^i(x_t^i)$
  Feed $\mathcal{A}_i$ linear loss function $\langle g_t^i, \cdot \rangle$
end for

Algorithm 2 Multiplayer OCO with Memory

Input: OCO-M algorithm $\mathcal{A}_i$, convex domain $\mathcal{K}_i \subseteq \mathbb{R}^d$ and memory length h
for $t = 1$ **to** T **do**
  Learner $\mathcal{A}_i$ predicts x_t^i
  Calculate $g_t^i = \nabla \ell_t^i(x_{t-h:t}^i)$
  Feed $\mathcal{A}_i$ linear loss function $\langle g_t^i, \cdot \rangle$
end for

Theorem 3.1. *Suppose for each $i \in [k]$, OCO algorithm $\mathcal{A}_i$ has regret $\mathcal{R}_T(\mathcal{A}_i)$ over $\mathcal{K}_i \subseteq \mathbb{R}^d$ with linear loss functions provided by Alg. 1, then the multi-agent regret of Alg. 1 is upper bounded by $\mathcal{R}_T(\mathcal{A}_{1:k}) \leq \sum_{i=1}^k \mathcal{R}_T(\mathcal{A}_i)$.*

3.3 Multiplayer OCO with Memory

The control setting which motivates our whole study is inherently state-based. Therefore we need to extend the multi-agent OCO to allow for memory.

This is similar to the previous section but the loss functions now have *memory* of the h previous controls. After each of the k online learners commits to the joint decision x_t as defined in the previous section. The learners incur a loss $\ell_t(x_{t-h:t})$ where ℓ_t is a convex loss function $\ell_t : \mathcal{K}^{h+1} \to \mathbb{R}$. The learner observers $\ell_t^i : \mathcal{K}_i^{h+1} \to \mathbb{R}$ by $\ell_t^i(z) = \ell_t(z, x_{t-h:t}^{-i})$.

Regret is defined to be the total loss incurred by the algorithm with respect to the loss of the best fixed single prediction found in hindsight for each player. Formally, the average regret of a set of learning algorithms $\mathcal{A}_{1:k}$ is

$$\mathcal{R}_T(\mathcal{A}_{1:k}) \stackrel{\text{def}}{=} \sup_{\ell_1 \ldots \ell_t} \left\{ \frac{1}{T} \big(\sum_{t=1}^T \ell_t(x_{t-h:t}) - \min_{x^* \in \mathcal{K}} \sum_{t=1}^T \bar{\ell}_t(x^*) \big) \right\}$$

where $\bar{\ell}_t(x) = \ell_t(x, \ldots x)$.

The main result of this section is given by the following theorem, which is formally proved in the appendix. The proof (See Appendix A.1) follows the memoryless setting.

Theorem 3.2. *Suppose for each $i \in [k]$ OCO-M algorithm with memory $\mathcal{A}_i$ has regret $\mathcal{R}_T(\mathcal{A}_i)$ over $\mathcal{K}_i \subseteq \mathbb{R}^d$ with linear loss functions provided by Alg. 2, then the regret of Alg. 2 has regret upper bounded by $\mathcal{R}_T(\mathcal{A}_{1:k}) \leq \sum_{i=1}^k \mathcal{R}_T(\mathcal{A}_i)$.*

4 The Meta-Algorithm and its Analysis

This section contains our main result: a meta-algorithm for multi-agent nonstochastic control. We start by defining the precise requirements from the single agent controllers that are given as input, and assumptions on them. We then proceed to describe the meta-algorithm and its main performance guarantee on the multi-agent regret. We describe a few special cases of interest in Appendix B.

4.1 Assumptions and definitions

The single agent control algorithms that are the basis for the reduction need to satisfy following requirements:

- Each agent should be able to evaluate their policy given the controls of the other agents.
- The joint cost function over all agents is jointly convex.

- Agent policies need to be decoupled in the sense of Assumption 2.4.

We now specify these requirements more formally.

We introduce two related notions of a (ε, h)-policy evaluation oracle (PEO) for a multi-agent control problem. A joint PEO allows us to counterfactually evaluate a complete multi-agent policy, while a local PEO allows us to counterfactually evaluate the policy of a single agent, fixing the remainder of the controls. ε is the accuracy required of this oracle, while h is something like a mixing time such that the cost of the tth state can be well approximated regardless of the history before $t - h$.

Given the available control information $u_{1:t-1}$, observations $y^i_{1:t}$, and cost function c_t, a loss function ℓ_t is a joint (ε, H)-PEO, if $\ell_t(\theta^{1:k}_{1:H})$ provides an ε approximate estimate of $c_t(\tilde{x}_t, \tilde{u}_t)$ where $\tilde{x}_t$ is the counterfactual state had $\tilde{u}_{t-r+1}$ been played according to joint policy $\pi_{\theta^{1:k}_r}$. Likewise, ℓ^i_t is the local PEO of the ith agent, if $\ell^i_t(\theta_{1:h})$ provides an ε approximate estimate of $c_t(\tilde{x}_t, \tilde{u}_t)$ where $\tilde{u}_t$ and $\tilde{x}_t$ are the counterfactual state and observation had u^i_{t-r+1} been played according to policy π_{θ_r} while all other controls $u^{-i}_{1:t-1}$ remain the same. These oracles are defined formally in Appendix A.3.

Remark 4.1. In order for a local agent to be able to provide an accurate estimate of the cost using only partial observations and controls, intuitively the cost may need to be restricted in in its dependence on the fully observed state. For example, if all agents' partial observations contain certain features from the state (e.g. the x-y-z position of the center of gravity of drone but no other velocity or angular features), the cost can depend on these and an agent PEO may exist.

A key observation is that for policies satisfying Assumption 2.4, if $u^{-i}_{t-h+1:t}$ are generated by policies $\theta^{-i}_{t-h+1:t}$ respectively, then a local PEO and a joint PEO can be related via $\ell^i_t = \ell_t(\cdot, \theta^{-i}_{t:t-h+1})$. Because all controls are determined directly from the disturbances, changes in agent i's controls have no impact on the controls of other agents.

We now assume that beyond an initial burn-in time T_b, we have the information to construct such an (ε, h)-local PEO. We also must guarantee that the PEOs we use are convex in order to use OCO.

Assumption 4.2 (Information)**.** For $t \geq T_b > h$, for each agent i, there exists a functional $\mathcal{L}^i$ such that $\ell^i_t = \mathcal{L}^i(c_t, u_{1:t}, y^i_{1:t})$ is a (ε, h)-local PEO for policy class Θ_i.

Assumption 4.3 (Convexity)**.** There exists a (ε, h) joint PEO ℓ_t that is convex such that $\ell^i_t = \mathcal{L}^i(c_t, u_{1:t}, y^i_{1:t})$ satisfies $\ell^i_t = \ell_t(\cdot, \theta^{-i}_{t-h+1:t})$.

4.2 Multi-Agent Control Meta-Algorithm and Analysis

Our reduction is described in Algorithm 3. It accepts as input OCO-M algorithms $\mathcal{A}_i$ and applies them sequentially using the linearization technique of the previous section.

Algorithm 3 Multi-Agent Nonstochastic Control

1: **Input:** Horizon length T, burn in time T_b, policy evaluation horizon h, and for each $i \in [k]$, OCO-M algorithms $\mathcal{A}_i$ over policy parameterization class Θ^i, policy evaluation functional $\mathcal{L}^i$.
2: Initialize $\theta^i_0 = \cdots = \theta^i_{T_b}$ as a policy that always plays control 0.
3: **for** $t = 1$ **to** T **do**
4: **for** $i = 1$ **to** k **do**
5: Observe y^i_t and play control $u^i_t = \pi_{\theta^i_t}(u_{1:t-1}, y^i_{1:t})$
6: Observe controls from other agents to form joint control $u_t = u^{1:k}_t$ and cost c_t
7: **if** $t > T_b$ **then**
8: Construct policy evaluation oracle $\ell^i_t = \mathcal{L}^i(c_t, u_{1:t}, y^i_{1:t})$ and compute $g^i_t = \nabla \ell^i_t(\theta^i_{t-h,t})$
9: Feed $\mathcal{A}_i$ linear loss function $\langle g^i_t, \cdot \rangle$ and receive policy parameterization θ^i_{t+1}.
10: **end if**
11: **end for**Pay $c_t(x_t, u_t)$
12: **end for**

The inner loop of the algorithm starting at Line 5 is what is performed by a local agent. The information available in this scope is the local observation and the controls of the other agents. We note that Alg. 3 only uses ℓ^i_t instead of ℓ_t, which is only used for analysis. This is fundamental as each agent is unaware of the policies of other agents.

The performance guarantee for this meta-algorithm is given by:

Theorem 4.4. *Suppose Assumptions 2.2, 2.3, 2.4 hold and Assumptions 4.2, 4.3 holds with burn-in T_b, horizon h and error ε. If $\mathcal{A}_i$ are OCO-M algorithms with regret $\mathcal{R}_T(\mathcal{A}_i)$ on linear loss functions provided by Algorithm 3 over policy parameterization class Θ^i, then the multi-agent regret of Algorithm 3 is bounded by*

$$\mathcal{R}_T(\mathcal{C}) \leq \sum_{i=1}^{k} \mathcal{R}_T(\mathcal{A}_i) + \frac{T_b C R_{nat}^2}{T} + 2\varepsilon \, .$$

See Section A.4 for proof. Algorithm 3 leaves most of the technical challenge to a policy evaluation oracle. It's not immediately obvious that such a function exists and is efficient to compute. Fortunately, recent work from nonstochastic control provides exactly what we need for Linear Dynamical Systems with a variety of policy classes. Appendix B contains a thorough discussion of applications for Linear Dynamical Systems.

5 Experiments

5.1 Robustness to System Failures

Our algorithm guarantee can also be extended to handle situations where certain agents no longer are locally regret minimizing (e.g. an actuator breaks). Because Algorithm 3 is agnostic to the policy played by other players, if we want to bound the regret for a subset $I \subset [k]$ of controllers, we can just view the policy class of the remaining agents as singleton open-loop policies that always play the same controls and so Theorem 4.4 will be a regret guarantee with respect to just the agents I with the remaining controls obliviously fixed.

Such robust guarantees are not achieved with existing approaches in nonstochastic control. Typically, as in the Gradient Perturbation Controller (GPC) (Agarwal et al., 2019), a PEO $\ell_t(\theta_{1:h})$ is passed to an online learning algorithm. If the system breaks down and certain controls are not played according to the chosen policy, there will be a mismatch between the true cost and the PEO. In contrast, because ℓ_t^i uses the actual controls played by the algorithm, for an agent i that is not experiencing failure, the oracle accuracy remains unchanged and provides the correct gradient signal.

Algorithm 3 may be useful to provide robustness even for a single agent system with some failures.

5.2 ADMIRE experiments

In Fig. 1 we run GPC, LQR control, and a linear $\mathcal{H}_{\text{inf}}$ control policy, along with Multi-Agent GPC (MAGPC), which is Algorithm 3 using gradient descent as the online learners $\mathcal{A}^i$. These are used to stabilize the ADMIRE overactuated aircraft model (Härkegård & Glad, 2005) in the presence of 3 different disturbance profiles: Gaussian noise, a random walk, and a sinusoidal disturbance pattern. We run the algorithms on the complete system, and with the 4th controller forced inactive by zeroing out the control to demonstrate the robustness of our approach. Figure 1 demonstrates the GPC failure described in Section 5.1. In the random walk experiment, GPC explodes, but MAGPC stabilizes the system. See Appendix C for full details. We note that, when a control failure does not occur and learning rates are sufficiently small, GPC and MAGPC are essentially the same algorithm. This is because the policies change slowly so $\ell_t^i(\theta_{t-h:t}) \approx \ell_t^i(\theta_t \dots \theta_t)$, which is GPC's proxy loss.

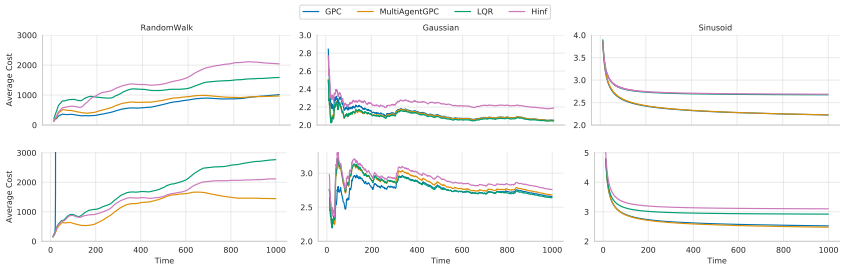

Figure 1: Cost of GPC, MultiAgentGPC, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control before and after the 4th controller becomes inactive in the presence of gaussian disturbances, disturbances generated by a random walk, and sinusoidal disturbances.

6 Discussion and Conclusions

Conclusions We have described a new approach for regret minimization in multi-agent control that is based on a new proposed performance metric: multi-agent regret. This measures the difference in cost between that of a distributed control system and that of the best joint policy from a reference class. We give a reduction that takes any regret minimizing controller and converts it into a regret minimizing multi-agent distributed controller.

Future directions The most interesting question is whether similar metrics of performance and ideas can be applied to general multi-agent reinforcement learning (MARL), since here we exploit full knowledge of the dynamics. It will be interesting to see if our information model can be relaxed and still allow sublinear multi-agent regret. It also remains to explore the extent of robustness obtained from a multi-agent regret minimizing algorithm.

REFERENCES

Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 1–26, 2011.

Naman Agarwal, Brian Bullins, Elad Hazan, Sham Kakade, and Karan Singh. Online control with adversarial disturbances. In *International Conference on Machine Learning*, pp. 111–119. PMLR, 2019.

Avrim Blum, MohammadTaghi Hajiaghayi, Katrina Ligett, and Aaron Roth. Regret minimization and the price of total anarchy. pp. 373–382, 05 2008. doi: 10.1145/1374376.1374430.

F. Bullo, J. Cortes, and S. Martinez. *Distributed control of robotic networks*. Princeton University Press, 2009.

Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9(1): 427–438, 2013. doi: 10.1109/TII.2012.2219061.

Ruggero Carli, Alessandro Chiuso, Luca Schenato, and Sandro Zampieri. Distributed kalman filtering based on consensus strategies. *IEEE Journal on Selected Areas in Communications*, 26(4): 622–633, 2008. doi: 10.1109/JSAC.2008.080505.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Xinyi Chen, Edgar Minasyan, Jason D. Lee, and Elad Hazan. Provable regret bounds for deep online learning and control, 2021.

Panagiotis D Christofides, Riccardo Scattolini, David Munoz de la Pena, and Jinfeng Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41, 2013.

Subhro Das and José M. F. Moura. Consensus+innovations distributed kalman filter with optimized gains. *IEEE Transactions on Signal Processing*, 65(2):467–481, 2017. doi: 10.1109/TSP.2016.2617827.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130, 2018.

Udaya Ghai, David Snyder, Anirudha Majumdar, and Elad Hazan. Generating adversarial disturbances for controller verification. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pp. 1192–1204. PMLR, 07 – 08 June 2021.

Gautam Goel and Babak Hassibi. Competitive control. *arXiv preprint arXiv:2107.13657*, 2021.

Paula Gradu, Elad Hazan, and Edgar Minasyan. Adaptive regret for control of time-varying dynamics. *arXiv preprint arXiv:2007.04393*, 2020.

Ola Härkegård and S Torkel Glad. Resolving actuator redundancy—optimal control vs. control allocation. *Automatica*, 41(1):137–144, 2005.

Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

Elad Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019.

Elad Hazan and Karan Singh. Tutorial: online and non-stochastic control, July 2021.

Chi Jin, Qinghua Liu, Yuanhao Wang, and Tiancheng Yu. V-learning–a simple, efficient, decentralized algorithm for multiagent rl. *arXiv preprint arXiv:2110.14555*, 2021.

Steffi Knorn, Zhiyong Chen, and Richard H. Middleton. Overview: Collective control of multi-agent systems. *IEEE Transactions on Control of Network Systems*, 3(4):334–347, 2016. doi: 10.1109/TCNS.2015.2468991.

Serge Lefebvre, Stephen Richter, and Raymond DeCarlo. Decentralized control of linear interconnected multivariable systems. In *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, pp. 525–529, 1980. doi: 10.1109/CDC.1980.271852.

Ion Matei and John S. Baras. Consensus-based distributed linear filtering. In *49th IEEE Conference on Decision and Control (CDC)*, pp. 7009–7014, 2010. doi: 10.1109/CDC.2010.5718072.

Edgar Minasyan, Paula Gradu, Max Simchowitz, and Elad Hazan. Online control of unknown time-varying dynamical systems. *Advances in Neural Information Processing Systems*, 34, 2021.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Angelia Nedić, Alex Olshevsky, and Michael G. Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018. doi: 10.1109/JPROC.2018.2817461.

R. Olfati-Saber. Distributed kalman filter with embedded consensus filters. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 8179–8184, 2005. doi: 10.1109/CDC.2005.1583486.

Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007. doi: 10.1109/JPROC.2006.887293.

Michael W. Oppenheimer, David B. Doman, and Michael A. Bolender. Control allocation for over-actuated systems. In *2006 14th Mediterranean Conference on Control and Automation*, pp. 1–6, 2006. doi: 10.1109/MED.2006.328750.

Shinkyu Park and Nuno C. Martins. Design of distributed lti observers for state omniscience. *IEEE Transactions on Automatic Control*, 62(2):561–576, 2017. doi: 10.1109/TAC.2016.2560766.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Max Simchowitz, Karan Singh, and Elad Hazan. Improper learning for non-stochastic control. In *Conference on Learning Theory*, pp. 3320–3436. PMLR, 2020.

K. Sobel and E. Shapiro. Eigenstructure assignment for design of multimode flight control systems. *IEEE Control Systems Magazine*, 5(2):9–15, 1985. doi: 10.1109/MCS.1985.1104941.

Yvonne R Stürz, Edward L Zhu, Ugo Rosolia, Karl H Johansson, and Francesco Borrelli. Distributed learning model predictive control for linear systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 4366–4373. IEEE, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Seyed Shahabaldin Tohidi, Yildiray Yildiz, and Ilya V. Kolmanovsky. Adaptive control allocation for over-actuated systems with actuator saturation. *IFAC-PapersOnLine*, 50:5492–5497, 2017.

L. Wang, D. Fullmer, F. Liu, and A. S. Morse. Distributed control of linear multi-channel systems: Summary of results. In *2020 American Control Conference (ACC)*, pp. 4576–4581, 2020. doi: 10.23919/ACC45564.2020.9148043.

Shih-Ho Wang and E. Davison. On the stabilization of decentralized control systems. *IEEE Transactions on Automatic Control*, 18(5):473–478, 1973. doi: 10.1109/TAC.1973.1100362.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

A Deferred Technical Material

A.1 Proof of Theorems 3.1 and 3.2

We provide a proof of Theorem 3.1, for the multi-agent regret of an OCO algorithm.

Proof. Let $\bar{x} \in \arg\min_{x \in \mathcal{K}} \sum_{t=1}^{T} \ell_t(x)$ with $\bar{x}^i$ the component corresponding to player i. By the convexity of ℓ_t,

$$\mathcal{R}_T(\mathcal{A}_{1:k}) = \sum_{t=1}^{T} (\ell_t(x_t) - \ell_t(\bar{x})) \leq \sum_{t=1}^{T} \nabla \ell_t(x_t)^\top (x_t - \bar{x}) \ .$$

Now we note that we can decompose $\nabla \ell_t$ by player, so

$$\nabla \ell_t(x_t)^\top = [\nabla \ell_t^1(x_t^1)^\top \dots \nabla \ell_t^k(x_t^k)^\top] \Rightarrow \nabla \ell_t(x_t)^\top (x_t - \bar{x}) = \sum_{i=1}^{k} \nabla \ell_t^i(x_t^i)^\top (x_t^i - \bar{x}^i) \ .$$

Now we note that, $\mathcal{R}_T(\mathcal{A}_i) = \sum_{t=1}^{T} \nabla \ell_t^i(x_t^i)^\top (x_t^i - \bar{x}^i)$ and the result follows. □

Following the same ideas, we provide a proof of Theorem 3.2, for the multi-agent regret of an OCO-M algorithm.

Proof. Let $\bar{x} \in \arg\min_{x \in \mathcal{K}} \sum_{t=1}^{T} \ell_t(x)$ with $\bar{x}^i$ the component corresponding to player i. Now by the convexity of ℓ_t,

$$\mathcal{R}_T(\mathcal{A}_1, \dots \mathcal{A}_k) = \sum_{t=1}^{T} \ell_t(x_{t-h:t}) - \sum_{t=1}^{T} \ell_t(\bar{x}, \dots \bar{x}) \leq \sum_{t=1}^{T} \nabla \ell_t(x_{t-h:t})^\top (x_{t-h:t} - (\bar{x}, \dots, \bar{x})) \ .$$

Now we note that we can decompose $\nabla \ell_t$ by player (after reorganizing coordinates), so

$$\nabla \ell_t(x_{t-h:t}) = [\nabla \ell_t^1(x_{t-h:t}^1)^\top, \nabla \ell_t^2(x_{t-h:t}^2)^\top, \dots \nabla \ell_t^k(x_{t-h:t}^k)^\top]$$

$$\Rightarrow \nabla \ell_t(x_{t-h:t})^\top (x_{t-h:t} - (\bar{x}, \dots, \bar{x})) = \sum_{i=1}^{k} \nabla \ell_t(x_{t-h:t}^i)^\top (x_{t-h:t}^i - (\bar{x}^i, \dots, \bar{x}^i)) \ .$$

Now we note that, $\mathcal{R}_T(\mathcal{A}_i) = \sum_{t=1}^{T} \nabla \ell_t(x_{t-h:t}^i)^\top (x_{t-h:t}^i - (\bar{x}^i, \dots, \bar{x}^i))$ and the result follows. □

A.2 Linear Dynamical Controller

Definition A.1 (Linear Dynamic Controller)**.** A *linear dynamic controller* π is a linear dynamical system $(A_\pi, B_\pi, C_\pi, D_\pi)$ with internal state $s_t \in \mathbb{R}^{d_\pi}$, input $x_t \in \mathbb{R}^{d_x}$, and output $u_t \in \mathbb{R}^{d_u}$ that satisfies

$$s_{t+1} = A_\pi s_t + B_\pi x_t, \ u_t = C_\pi s_t + D_\pi x_t \ .$$

Linear dynamical controllers involve an internal linear dynamical system to produce a control. The combination of a Kalman Filter with Optimal control on the state estimates is a classic example.

A.3 Technical Details from Section 4

Below are the formal definition of joint and local policy evaluation oracles:

Definition A.2. $\ell_t : \Theta^h \to \mathbb{R}$ is an (ε, h)-joint PEO if $|\ell_t(\theta_{1:h}^{1:k}) - c_t(\tilde{x}_t, \tilde{u}_t)| \leq \varepsilon$ where $(\tilde{x}_{1:t-h}, \tilde{y}_{1:t-h}, \tilde{u}_{1:t-h-1}) = (x_{1:t-h}, y_{1:t-h}, u_{1:t-h-1})$ and for $s > t - h$

$$\tilde{u}_s = \pi_{\theta_{t-s+1}^{1:k}}(\tilde{y}_{1:s}, \tilde{u}_{1:s-1}), \tag{2}$$

$$\tilde{x}_{s+1} = f_s(\tilde{x}_t, \tilde{u}_s) + w_s, \quad \forall i \in [k], \tilde{y}_{s+1}^i = g_i(\tilde{x}_{s+1}) + e_{s+1}^i \ .$$

$\ell_t^i : \Theta_i^h \to \mathbb{R}$ is an (ε, h)-local PEO for agent i if $|\ell_t^i(\theta_{1:h}) - c_t(\tilde{x}_t, \tilde{u}_t)| \leq \varepsilon$ where $(\tilde{x}_{1:t-h}, \tilde{y}_{1:t-h}^i, \tilde{u}_{1:t-h-1}) = (x_{1:t-h}, y_{1:t-h}^i, u_{1:t-h-1})$, $\tilde{u}_{t-h+1:t}^{-i} = u_{t-h+1:t}^{-i}$ and for $s > t - h$

$$\tilde{u}_s^i = \pi_{\theta_{t-s+1}^i}(\tilde{y}_{1:s}^i, \tilde{u}_{1:s-1}), \tag{3}$$

$$\tilde{x}_{s+1} = f_s(\tilde{x}_t, \tilde{u}_s) + w_s, \quad \tilde{y}_{s+1}^i = g_i(\tilde{x}_{s+1}) + e_{s+1}^i \ .$$

A.4 Proof of Main Theorem

We provide a complete proof of Theorem 4.4:

Proof. In order to analyze the regret we begin with the following regret decomposition. We first let $\theta^* = \arg\min_{\theta\in\Theta} \sum_{t=1}^{T} c_t(y_t^{\pi_\theta}, u_t^{\pi_\theta})$ and let $\pi^* = \pi_{\theta^*}$ denote offline optimal joint policy from our comparator policy class.

$$\mathcal{R}_T(\mathcal{C}) = \frac{1}{T}\sum_{t=1}^{T} c_t(x_t^{\mathcal{C}}, u_t^{\mathcal{C}}) - \frac{1}{T}\sum_{t=1}^{T} c_t(x_t^{\pi^*}, u_t^{\pi^*}) \leq \underbrace{\left(\frac{1}{T}\sum_{t=1}^{T_b} c_t(x_t^{\text{nat}}, 0) - \sum_{t=1}^{T_b} c_t(x_t^{\pi_{\theta^*}}, u_t^{\pi_{\theta^*}})\right)}_{\text{burn-in cost}}$$

$$+ \underbrace{\left(\frac{1}{T}\sum_{t=T_b+1}^{T} c_t(x_t^{\mathcal{C}}, u_t^{\mathcal{C}}) - \frac{1}{T}\sum_{t=T_b+1}^{T} \ell_t(\theta_{t:t-h})\right)}_{\text{algorithm truncation error}} + \underbrace{\left(\frac{1}{T}\sum_{t=T_b+1}^{T} \ell_t(\theta_{t:t-h}) - \frac{1}{T}\sum_{t=T_b+1}^{T} \ell_t(\theta^*, \ldots, \theta^*)\right)}_{\text{multi-agent } \ell \text{ policy regret}}$$

$$+ \underbrace{\left(\frac{1}{T}\sum_{t=T_b+1}^{T} \ell_t(\theta^*, \ldots, \theta^*) - \frac{1}{T}\sum_{t=T_b+1}^{T} c_t(x_t^{\pi_{\theta^*}}, u_t^{\pi_{\theta^*}})\right)}_{\text{comparator truncation error}}$$

The first term of the decomposition is exactly the cost of the first T_b iterations because the Alg. 3 starts with T_b zero controls so by definition $x_t^{\mathcal{C}}$ are Nature's x's. By Assumption 2.3, losses are nonnegative so we have $-\sum_{t=1}^{T_b} c_t(x_t^{\pi_{\theta^*}}, u_t^{\pi_\theta}) \leq 0$ so we drop these terms. By combining Assumptions 2.3 and 2.2 we bound the burn-in cost as

$$\frac{1}{T}\sum_{t=1}^{T_b} c_t(x_t^{\text{nat}}, 0) \leq \frac{C}{T}\sum_{t=1}^{T_b} \|x_t\|^2 \leq \frac{CT_b R_{\text{nat}}^2}{T} .$$

The comparator truncation error is bounded by ε because for $t > T_b$, $|\ell_t(\theta^*, \ldots, \theta^*) - c_t(x_t^{\pi_{\theta^*}}, u_t^{\pi_{\theta^*}})| \leq \varepsilon$ by definition of an (ε, h) joint PEO. Similarly, since $\mathcal{C}$ produces $x_t^{\mathcal{C}}$ with controls u_s from joint policy θ_s for the preceding h controls, the algorithm truncation error is also bounded by ε.

To bound the ℓ_t multi-agent policy regret term, we note that Algorithm 3 chooses θ_t exactly by the mechanism in Algorithm 2 with loss functions ℓ_t^i. By Assumption 4.2, $\ell_t^i = \ell_t(\cdot, \theta_{t-h:t-1}^{-i})$ and by Assumption 4.3, ℓ_t is convex as desired. As such, we bound the ℓ_t multi-agent policy regret term using Theorem 3.2 as

$$\sum_{t=T_b}^{T} \ell_t(\theta_{t:t-h}) - \sum_{t=T_b+1}^{T} \ell_t(\theta^*, \ldots, \theta^*) \leq T\sum_{i=1}^{k} \mathcal{R}_T(\mathcal{A}_i) .$$

The result follows after combining the above bounds. □

B Applications to Linear Dynamical Systems

For simplicity of presentation, we consider only stable systems. However, we note that in settings with a single shared observation y_t, these algorithms can be extended to work for unstable systems. In order to do this, all agents must be aware of baseline stabilizing controllers employed by each agent, and Algorithm 3 can then be applied on the stable closed-loop system including the stabilizing component. For a thorough treatment of this approach, we refer the reader to Appendix G of (Simchowitz et al., 2020).

We assume strong stability, which guarantees spectral radius of A strictly less than 1. Strong stability assures us that impact of disturbances and controls from the far history have a negligible impact on the current state. This geometric decay enables the construction of PEO's with horizon h just logarithmic in T while providing $\varepsilon = \frac{1}{T}$ approximation error.

Fully Observed LDS with Disturbance Action Control We consider the case of a fully observed LDS defined as follows:

$$y_{t+1} = x_{t+1} = f_t(x_t, u_t) + w_t = Ax_t + Bu_t + w_t \ ,$$

with bounded adverserial disturbances w_t, known strongly stable dynamics, and well behaved convex costs c_t satisfying Assumption 2.3[2].

We now show that a linear system with a DAC policy class $\mathcal{M}$ can satisfy assumptions required for Theorem 4.4. We already saw that Assumption 2.4 holds for DAC policies. Next we show that Assumption 2.2 holds. By fully unrolling dynamics, Nature's x's can be written as a linear combination of disturbances $x_t^{\text{nat}} = \sum_{s=0}^{t-1} A^s w_{t-s}$.

Now because w_t are bounded, and A is strongly stable, $\|x_t^{\text{nat}}\|$ can be bounded by a convergent geometric series. The last piece is constructing a PEO for a DAC policy. Unrolling dynamics, we express states in terms of controls:

$$x_t \approx x_t^{\text{nat}} + [B, AB, \dots A^{h-1}B]u_{t-1:t-h} = \hat{G}u_{t-1:t-h} \ .$$

The truncation follows from strong stability of the dynamics. Decomposing $\hat{G}$, our Markov operator by agent, we can write $x_t \approx x_t^{\text{nat}} + \sum_{i=1}^{k} \hat{G}_i u_{t-1:t-h}^i$. Now to compute $\ell_t^i(M_{1:h}) = c_t(\tilde{x}(M_{1:h}), Mw_{t-m:t-1})$ we hold u_s^{-i} fixed while changing the policies for agent i. Because the counterfactual state $\tilde{x}(M_{1:h})$ is affine in $u_{t-1:t-h}^i$ and by the DAC policy parameterization agent i's controls are linear, the counterfactual state can be approximated as linear in $M_{1:h}$. Composing with a convex cost c_t gives a convex PEO ℓ_t^i. A complete analysis in the single agent setting can be found in (Agarwal et al., 2019). In order to compute $\ell_t^i(M_{1:h})$, the disturbances w_t are needed. Because agent i has access to the joint control u_t, disturbances can be computed as $w_t = x_{t+1} - Ax_t - Bu_t$. Once $T_b = m + h$ disturbances are observed, ℓ_t^i can be computed.

Similar ideas can be applied using Nature's y's and DRC to handle partial observations. Furthermore, Theorem 4.4 can be applied with richer neural network policy classes as well. For a more complete discussion, see Appendix B.

Partially Observed LDS with Disturbance Response Control Another setting that we can capture with out meta-algorithm is partially observed linear dynamical systems, where all agents share the same partial observation. In this case, the dynamics are as follows,

$$\begin{aligned} y_t &= g(x_t) + e_t = Cx_t + e_t, \\ x_{t+1} &= f_t(x_t, u_t) + w_t = Ax_t + Bu_t + w_t \ , \end{aligned}$$

with bounded adversarial disturbances e_t, w_t and known strongly stable dynamics. For a partially observed system with adversarial disturbances, it is known that many states can be consistent with the observations and so costs must be restricted to be well defined for the agents. This is solved by using convex costs of the *observation* $c_t : \mathbb{R}^{d_y} \times \mathbb{R}^{d_u}$, so at time t the cost incurred is $c_t(y_t, u_t)$. Because y_t is an affine function of x_t, this cost is still convex in x to fit the general setting.

A policy class, that suits our needs is that of Disturbance-Response Control (DRC). The satisfaction of the assumptions of Algorithm 3 follows from similar arguments to DAC in the fully observable case. In particular, to compute PEO ℓ_t^i, the following approximate Markov operator is used

$$\begin{aligned} y_t &\approx y_t^{\text{nat}} + \sum_{i=1}^{k} \hat{G}u_{t-1:t-h}^i \\ \hat{G} &= [CB, CAB, \dots CA^{h-1}B] \ . \end{aligned} \tag{4}$$

In the above equation $\hat{G}_i$ is the block of the Markov operator $\hat{G}$ corresponding to the ith agents controls. Beyond this, the core ideas are the same, just replacing disturbances with Nature's y's. The last piece necessary is computing Nature's y's. This is not challenging given we have access to all agent's controls, so we can let $y_t^{\text{nat}} \approx y_t - \hat{G}_i u_{t-1:t-h}^i$. For a complete analysis of this approach in the single agent setting, refer to (Simchowitz et al., 2020).

[2]In general, we may actually need more assumptions on the costs, and the size of our comparator policy classes as $\|g_t^i\|$ in Line 11 of Algorithm 3 depends on these pieces, and hence so does $\mathcal{R}_T(\mathcal{A}_i)$.

Locally Partially Observed LDS with Disturbance Response Control An extension of the prior setting of interest is for each agent to have it's own local partial observation without needing to share it in the protocol. In particular, consider the multi-agent system

$$x_{t+1} = f_t(x_t, u_t, w_t) = Ax_t + Bu_t + w_t,$$
$$y^i_{t+1} = g_i(x_{t+1}, e_t) = C^i x_{t+1} + e^i_{t+1} \ ,$$

with bounded adversarial disturbances e^i_t, w_t and known strongly stable dynamics. As noted in Remark 4.1, in order for each agent to have a local PEO, the cost $c_t(x_t, u_t)$ needs to be computable with only local information at each agent. Therefore, we assume there are local cost functions $c^i_t : \mathbb{R}^{d_{y_i}} \times \mathbb{R}^{d_u}$ for each agent such that for all $i \in [k], c_t(x, u) = c^i_t(C^i x + e^i_t, u_t)$.

With access to the global loss in each agent, agent i can use DRC with $y^{\text{nat},i}_t$ to satisfy all assumptions required for our meta-algorithm. Each observation matrix C^i produces a different markov operator $G^i = [C^i B, \ldots, C^i A^{h-1} B]$, such that $y^i_t = y^{\text{nat},i}_t + G^i u_{t-1:t-h}$. Using G^i agent i can compute $y^{\text{nat},i}_t$ and build a PEO in the same way as the single partial observation setting.

Linear Dynamics with Neural Network Policies All the previous policy parameterizations were linear in some feature representation. Recent work shows that nonstochastic control can extend beyond linear policies using analysis from deep learning theory (Chen et al., 2021). The authors consider the fully observed setting with fully-connected ReLU neural network policies that act on normalized windows of disturbances. A regret bound for online episodic control is shown by proving that under some technical conditions, a PEO is ε-*nearly-convex*, which means

$$p(x) - p(y) \leq \langle \nabla p(x), x - y \rangle - \varepsilon \ .$$

It can be readily shown that Theorem 3.2 can be extended to nearly convex functions using near-convexity in place of convexity, so we can get a multi-agent control regret guarantee for two-layer neural nets that act on normalized windows of disturbances. In addition, the proof that a PEO is nearly convex can be extended to neural net policies that act on normalized windows of Nature's y's, so neural net policies can also be used in the partial observation settings described above.

C Robustness to System Failures: Overactuated Aircraft

In this section we provide the mathematical model of an overactuated aircraft based on ADMIRE (Härkegård & Glad, 2005), and demonstrate the robustness of our controller to system failures.

Let α, β, p, q, r be angle of attack, slideslip angle, roll rate, pitch rate and yaw rate respectively. State space can be given as $x = [\alpha \quad \beta \quad p \quad q \quad r]^T$ and input space consists of 4 one-dimensional controllers, $\begin{bmatrix} u^1 & u^2 & u^3 & u^4 \end{bmatrix}$. Discrete linear dynamics of the system can be given as $x(t+1) = Ax(t) + \sum_{i=1}^{4} B_i u^i(t) + W(t)$ where,

$$A = \begin{bmatrix} 1.5109 & 0.0084 & 0.0009 & 0.8598 & -0.0043 \\ 0 & -0.0295 & 0.0903 & 0 & -0.4500 \\ 0 & -3.1070 & -0.1427 & 0 & 2.7006 \\ 2.3057 & 0.0097 & 0.0006 & 1.5439 & -0.0029 \\ 0 & 0.5000 & 0.0125 & 0 & 0.4878 \end{bmatrix}$$

$$B = [B_1 \quad B_2 \quad B_3 \quad B_4] = \begin{bmatrix} 0.6981 & -0.5388 & -0.5367 & 0.0029 \\ 0 & -0.2031 & 0.2031 & 0.3912 \\ 0 & -2.0768 & 2.0768 & -0.4667 \\ 1.8415 & -1.4190 & -1.4190 & 0.0035 \\ 0 & -0.1854 & 0.1854 & -0.7047 \end{bmatrix}$$

and $W(t)$ is the disturbance.

C.1 Experimental Setup

Controller details:

1. GPC has a decaying learning rate of $\frac{0.001}{t}$, a rollout length h of 5 and uses DACs with windows of size 5.
2. MAGPC is split into 4 1-d controllers, each using decaying learning rate of $\frac{0.001}{t}$, a rollout length 5 and a DAC with window length 5.

3. We use a linear $\mathcal{H}_\infty$ controller and standard infinite horizon LQR linear controller.

Disturbance details:

1. Random walk chooses $w_t = w_{t-1} + X_t$ where X_t is a standard Gaussian random variable.
2. Gaussian noise is iid. and has variance 1.
3. Sinusoidal noise is chosen via $w_i = \sin(2t + \phi_i)$ where $\phi = [12.0, 21.0, 3.0, 42.0, 1.0]$, picked arbitrarily once.

Fourth control is set to 0 in one experiment and otherwise controls are left unchanged.