Meta-Learning Online Control for Linear Dynamical Systems

Deepan Muthirayan, Dileep Kalathil, and Pramod P. Khargonekar

Abstract—In this work, we consider the problem of finding a meta-learning online control algorithm that can learn across the tasks when faced with a sequence of \( N \) (similar) control tasks. Each task involves controlling a linear dynamical system for \( T \) time steps. The cost function and system noise at each time step are adversarial and unknown to the algorithm before taking the control action. The goal of a meta-learning algorithm is to sequentially prescribe the individual online control policies for each new task by exploiting the information from previous tasks and the property of task similarity. We propose a meta-learning online control algorithm for this setting and characterize its performance using the metric of *meta-regret*, which is the average cumulative regret of the tasks. We show that, when the number of tasks are sufficiently large, the meta-regret of our proposed approach is smaller by a factor \( D/D^* \) compared to an independent-learning online control algorithm which does not perform learning across the task, where \( D \) is a problem constant and \( D^* \) is a scalar that decreases with increase in task similarity. Thus, when the sequence of tasks are similar, the regret of the proposed meta-learning online control is significantly lower than that of the naive approaches without meta-learning. We also present numerical results to demonstrate the superior performance achieved by our meta-learning algorithm.

I. INTRODUCTION

Meta-learning is a powerful paradigm in machine learning for learning-to-learn new tasks with limited data [1]. Meta-learning relies on the intuitive idea that if the new task is similar to previous tasks, it can be learned very quickly by using the data from previously encountered related tasks. Recently, there has been tremendous progress in practical algorithms for meta-learning [2]–[4], with impressive performance in many applications such as image classification [5], natural language processing [6], and robotic control [7]. These algorithms, however, are in the batch learning setting, where data sets composed of different tasks are available for offline training. A meta-model (typically a neural network) is then trained using these data sets with the objective of adaptation to a new/unseen task at test time, with only a few data samples corresponding to the new task.

Significantly different from the batch learning setting, which are offline by nature, many learning algorithms often have to operate in an online learning setting, where the data samples are obtained in a sequential manner, typically as an outcome of the action taken by the learning algorithm. Some standard online learning applications include personalized recommendation systems [8], various applications in robotics [9], demand response management in smart grid [10], and load balancing in data centers [11].

Online convex optimization (OCO) [12], [13] focuses on developing algorithms for such online learning setting where the loss functions are sequentially revealed and the learner is trained as well as tested at each round. The standard OCO objective is to minimize the regret, which is defined as the difference between the cumulative cost incurred by the online algorithm and the best policy from a certain class of policies. Even though the OCO approach offers a fundamental theoretical framework to analyze a variety of online learning scenarios, the existing works do not consider how the past experience can be used to accelerate adaptation to a new task, which is the key idea behind meta-learning.

Recently, there has been many works in the area of online control algorithms for dynamical systems with uncertain/unknown disturbances, system parameters and cost functions. This online control literature extends the OCO approach to problems with dynamics, focusing on the finite time performance guarantees using the metric of regret [14]–[17]. However, the existing works in the online control literature only consider the problem of learning within a task assuming that the task is fixed. In particular, they do not consider the possibility of learning across the tasks, when faced with a sequence of similar control tasks.

In this work, we consider the problem of finding a meta-learning online control algorithm that learns across the tasks when faced with a sequence of \( N \) (similar) control tasks. Each task involves controlling a linear dynamical system for \( T \) time steps and the cost function and system noise at each time step are adversarial and unknown to the algorithm before taking the control action. The key idea behind meta-learning is to prescribe an online control policy for any new unseen task exploiting the information from prior tasks and the similarity between the tasks. We characterize the performance of such a meta-learning online control algorithm using the metric of *meta-regret*, which is the average (averaged over the tasks) cumulative regret of the tasks. In particular, our goal is to develop a meta-learning online control algorithm that can achieve superior performance, in theory and practice, over an independent-learning online control algorithm which applies a standard online control algorithm independently to each task without performing any learning across the tasks.

Our approach is motivated by some recent works in online meta-learning [18]–[20], which combines the meta-learning idea with the OCO framework. In [18], the authors extend the model-agnostic meta-learning (MAML) approach
[2] to the online setting. Their goal is to learn a good meta-policy parameter that allows fast adaptation to all the previously seen tasks by taking only a few gradient steps from this meta-policy parameter. The work that is closest to ours is [19], which proposes the Follow-the-Meta-Regularized-Leader (FTMRL) approach. FTMRL learns a meta-initialization for a task specific OCO algorithm such that the individual task regret of these algorithms improves with the similarity of the online tasks. However, these works consider only the online optimization setting without state evolution. In particular, they do not consider the more challenging problem of online control of uncertain dynamical systems.

Our contributions: We consider the problem of developing a meta-learning online control algorithm for a sequence of similar control tasks. Each task involves controlling a linear dynamical system with adversarial cost functions and disturbances, which are unknown before taking the control action. The algorithm we propose has a two loop structure, where the outer loop performs the meta-learning update to prescribe an initialization parameter for the task specific online control algorithm used in the inner loop. We show that, when the number of tasks are sufficiently large, the meta-regret of our approach is smaller by a factor $D/D^*$ compared to an independent-learning online control algorithm that does not perform learning across the tasks, where $D$ is a problem constant and $D^*$ is a scalar that represents the task similarity ($D^*$ decreases with similarity between tasks). Therefore, when the sequence of tasks are similar, i.e., when $D^* \ll D$, we achieve a regret that is significantly lower than that of the naive approaches without meta-learning. We also present experiments results to demonstrate the superior performance of our meta-learning algorithm. Our technical contribution lies in expanding the framework and technical analysis of online control to incorporate meta-learning. To the best of our knowledge, ours is the first work that combines the ideas of meta-learning and online control to develop a learning algorithm with provable guarantees for its performance.

Related Works:

Online Control: Recently, a substantial number of works have been published in the area of online control [14]–[17], [21]. Majority of these works focus on developing online control algorithms for linear dynamical systems with provable guarantees for the regret. In our work, we make use of the task specific online control algorithm proposed in [16], which considers the control of a known linear dynamic system with adversarial disturbance and (convex) cost functions and show that $O(\sqrt{T})$ regret is achievable for a given task. Our meta-learning online control algorithm is developed by extending the task specific online control algorithm proposed in [16], with an additional outer loop for performing the meta-learning update and with a slight modification to the task specific (inner loop) update.

Adaptive and Robust Control: Classical adaptive and robust control literature addresses the problem of control of systems with parametric, structural, modeling and disturbance uncertainties [22]–[24]. Typically, these classical approaches are concerned with stability and asymptotic performance guarantees. On the other hand, online control literature is typically focussed on the finite time performance guarantee such as the regret. This is a key difference and challenge compared to the conventional adaptive and robust control literature, and it requires combining techniques from statistical learning, online optimization and control. In this work, we focus on the online control approach for developing our meta-learning algorithm.

Notations: Unless otherwise specified $\|\cdot\|$ denotes the Euclidean norm and the Frobenious norm for vectors and matrices, respectively. We use $O(\cdot)$ for the the standard order notation while $\tilde{O}(\cdot)$ denotes the order neglecting the poly-log terms. We denote the sequence $(x_{m_1}, x_{m_1+1}, \ldots, x_{m_2})$ compactly by $x_{m_1:m_2}$.

II. PROBLEM SETTING

We consider the problem of finding a meta-learning online control (M-OC) algorithm that learns across the tasks when faced with a sequence of (similar) control tasks. The sequence of tasks are denoted as $\tau_1, \tau_2, \ldots, \tau_N$. Each control task $\tau_i$ involves controlling a linear dynamical system for $T$ time steps, whose system dynamics is given by the equation

\[ x_{i,t+1} = A_i x_{i,t} + B_i u_{i,t} + w_{i,t}, \quad 1 \leq t \leq T, \]

where $A_i \in \mathbb{R}^{n \times n}$ and $B_i \in \mathbb{R}^{n \times m}$ are the matrices that parameterize the system, and $x_{i,t} \in \mathbb{R}^n$ is the state, $u_{i,t} \in \mathbb{R}^m$ is the action, $w_{i,t} \in \mathbb{R}^n$ is the system noise at time $t$, respectively. For conciseness, we represent the system parameters in task $\tau_i$ collectively as $\theta_i = [A_i, B_i]$. We assume that the system noise is adversarial, and therefore we make no stochasticity assumption.

A control policy $\pi$ for a task $\tau_i$ selects a control action $u^\pi_{i,t}$ at each time $t$ depending on the available information, resulting in a sequence of actions $u^\pi_{1:T}$ and a state trajectory $x^\pi_{1:T}$. Then, the cumulative cost of a policy $\pi$ under the system dynamics (1) can be written as

\[ J_i(\pi) = \sum_{t=1}^{T} c_{i,t}(x^\pi_{i,t}, u^\pi_{i,t}), \]

where $c_{i,t}$ is the cost function in task $\tau_i$ at time $t$. We assume that $c_{i,t}$s are arbitrary convex functions. The typical goal is to find the optimal policy $\pi^*_i$ such that $\pi^*_i = \arg \min_{\pi} J_i(\pi)$. Clearly, computing $\pi^*_i$ requires the knowledge of the system parameter $\theta_i$ and the entire sequence of cost functions $c_{i,1:T}$.

The online control framework considers a more realistic setting, where the future cost functions are unavailable for deciding the control action $u_{i,t}$ at time $t$. More precisely, the policy $\pi_t$ for a task $\tau_i$ has only the following information at each time $t$: (i) past and current state observations $x_{i,1:t}$, (ii) past control actions $u_{i,1:t-1}$, and (iii) past cost functions $c_{i,1:t-1}$. We also assume that the system parameter $\theta_i$ is known to the control policy.

A standard metric that is used to evaluate the performance of a online control policy is the regret. The task regret of
a control policy $\pi_i$ for a task $\tau_i$ is given by

$$R_T^i(\pi_i) = J_i(\pi_i) - \min_{\pi \in \Pi} J_i(\pi), \quad (3)$$

where $\Pi$ is a certain class of control policies. The goal in online control is to find a policy that minimizes the task regret. However, the existing online control algorithms do not consider learning across tasks to exploit the similarity between tasks when faced with a sequence of tasks.

In contrast, the goal in online meta-learning is to design a meta-policy $\pi^m$ that can learn across the tasks when faced with a sequence of (similar) control tasks $\tau_1, \tau_2, \ldots, \tau_N$ and thereby exploit the similarity between tasks. The function of a meta-policy $\pi^m$ is to produce a sequence of task specific policies $\pi^m_i, 1 \leq i \leq N$, by learning across the tasks. For deciding the task specific policy $\pi^m_i$ for task $\tau_i$, the meta-policy $\pi^m$ may make use of the observation available from the previous tasks: the state observations, cost functions, and task specific policies for all previous tasks $j < i - 1$. Since the objective of the meta-policy is to generate task specific policies which can do well on individual tasks, the performance of the meta-policy is characterized using the metric meta-regret, defined as

$$R_N^\text{meta}(\pi^m) = \frac{1}{N} \sum_{i=1}^N R_T^i(\pi^m_i). \quad (4)$$

Our objective is to find a meta-policy that performs better than an independent-learning online control algorithm that applies a standard online control algorithm independently to each task without performing any learning across the tasks.

We make the following assumptions. Please note that the assumptions made below are standard in the (task specific) online control literature [16]. We emphasize that we make no additional assumptions other than these standard assumptions.

**Assumption 1** (System Model). (i) The system matrices for each task are bounded, $|A_i| \leq \kappa_A$, and $|B_i| \leq \kappa_B$, where $\kappa_A$ and $\kappa_B$ are constants. (ii) The disturbance at time $t$ of any task is bounded, $|w_{i,t}| \leq \kappa_w$, where $\kappa_w$ is a constant.

**Assumption 2** (Cost Functions). For all tasks $i, 1 \leq i \leq N$ and all time steps $t, 1 \leq t \leq T$, (i) the costs functions $c_{i,t}$s are convex, (ii) for any $x$ and $u$ with $|x| \leq S, |u| \leq S$,

$$\|c_{i,t}(x,u)\| \leq \beta S^2, \|\nabla_x c_{i,t}(x,u)\|, \|\nabla_u c_{i,t}(x,u)\| \leq GS,$$

III. REVIEW: ONLINE CONTROL ALGORITHM

In this section, we give a brief description of the task specific online control (OC) algorithm proposed in [16]. We drop the task subscript $i$ because the discussion here focuses on the algorithm for a single task. Our meta-learning online control algorithm has been developed by extending the task specific OC algorithm with an additional outer loop for performing the meta-learning update and appropriately modifying the task specific (inner loop) update.

The OC algorithm proposed in [16] uses a control policy parameterized by two matrices, a fixed matrix $K$ and a time varying matrix $M_t = (M_t^{[1]}, M_t^{[2]}, \ldots, M_t^{[H]})$. The control action $u_t$ computed by this policy is given by

$$u_t = -Kx_t + \sum_{k=1}^H M_t^{[k]}w_{t-k}. \quad (5)$$

Thus, the control action is a linear map of the current state and the past disturbances up to a certain history. This property is convenient in that it permits efficient optimization over the parameters $M_t$ of the policy. We note that, since the state is fully observable, the past disturbances can be precisely estimated using the information at time $t$ and thereby can be used to compute the control action as per the policy.

The parameter $K$ is a $(\kappa, \gamma)$-strongly stable linear feedback control matrix for the underlying system. A linear feedback control policy specified by the gain $K$ is $(\kappa, \gamma)$-strongly stable if there exists matrices $L, H$ satisfying $A - BK = HLH^{-1}$, such that following two conditions are met: (i) $\|L\| \leq 1 - \gamma$, and (ii) $\|K\| \leq \kappa, \|H\|, \|H^{-1}\| \leq \kappa$. The OC algorithm considers the class $\Pi$ of all $(\kappa, \gamma)$-strongly stable linear feedback controllers for characterizing its regret performance according to (3).

The OC algorithm uses the framework of Online Convex Optimization (OCO) to update the parameters $M_t$ at each time step. The key idea behind the algorithm is a sequence of “idealized” cost functions $f_{1:T}$ in terms of the parameters $M_{1:T}$ that correctly represents the actual cost incurred by the true cost functions $c_{1:T}$. To define these functions we introduce an idealized state $s_t$ and an idealized control input $a_t$. The idealized state $s_t$ is defined as the state the system would reach if the controller executes the policy with parameters $(M_{t-H}, \ldots, M_{t-1})$ from time step $t-H$ till time step $t-1$ under the condition that the state at $t-H$ is $0$. The idealized action $a_t$ is the action that would be executed at time $t$ if the state observed at time $t$ is $s_t$. We can then define the idealized cost as $f_t(s_t, a_t) = c_t(s_t, a_t)$.

The complete OC algorithm proposed in [16] is shown in Algorithm 1. The algorithm uses an Online Gradient Descent (OGD) approach to update the parameters $M_t$ at each time step. The key idea behind the algorithm is a sequence of “idealized” cost functions $f_{1:T}$ in terms of the parameters $M_{1:T}$ that correctly represents the actual cost incurred by the true cost functions $c_{1:T}$. To define these functions we introduce an idealized state $s_t$ and an idealized control input $a_t$. The idealized state $s_t$ is defined as the state the system would reach if the controller executes the policy with parameters $(M_{t-H}, \ldots, M_{t-1})$ from time step $t-H$ till time step $t-1$ under the condition that the state at $t-H$ is $0$. The idealized action $a_t$ is the action that would be executed at time $t$ if the state observed at time $t$ is $s_t$. We can then define the idealized cost as $f_t(s_t, a_t) = c_t(s_t, a_t)$.

Theorem 1 (Theorem 5.1, [16]). Let Assumptions 1 and 2 hold and let $\eta = \frac{D}{\sqrt{G_f(G_f/2 + LH^2)^T}}$, $D = \frac{\kappa_B \kappa^3 \sqrt{\gamma}}{\gamma}$. Then, under Algorithm 1,

$$R_T \leq \frac{3D\sqrt{G_f(G_f/2 + LH^2)^T}}{2} + \bar{O}(1), \quad \text{where}$$

$$L = 2G\tilde{D}\kappa_w \kappa_B \kappa^3, \quad G_f = G\tilde{D}\kappa_w H \left(\frac{2\kappa_B \kappa^3}{\gamma} + H\right),$$

$$\tilde{D} = \frac{\kappa_B \kappa^3 \kappa_w}{\gamma(1 - \kappa^2(1 - \gamma)^{H+1})},$$

1437
Algorithm 1 Online Control (OC) Algorithm

Input: Step size $\eta$, parameters $\kappa_B, \kappa, \gamma, T$, $(\kappa, \gamma)$-strongly stable control matrix $K$

Define $H = \log T / (\log (1/1 - \gamma))$

Define $\mathcal{M} = \{M = (M^1, \ldots, M^H) : \|M[k]\| \leq \kappa^3 \kappa_B (1 - \gamma)^k\}$

Define $g_i(M) = f_i(M, \ldots, M)$

Initialize $M_1 \in \mathcal{M}$

for $t = 1, \ldots, T$

Choose the action $u_t = -Kx_t + \sum_{k=1}^{H} M^{[k]}_t w_{t-k}$

Observe the new state $x_{t+1}$, and $w_t = x_{t+1} - Ax_t - Bu_t$

Update $M_{t+1} = \text{Proj}_\mathcal{M} (M_t - \eta \nabla g_i(M_t))$

end

Remark 1 (Diameter of the domain). It can be shown that the multiplicative constant $D$ appearing in the regret bound is the diameter of the domain $\mathcal{M}$ of the control policy parameters, i.e., $D = \max_{M_1, M_2 \in \mathcal{M}}\|M_1 - M_2\|$ (16, Theorem 5.1). In the next section, we show that our meta-learning approach can significantly reduce this multiplicative constant by learning across the tasks.

IV. META LEARNING ONLINE CONTROL ALGORITHM

Our meta-learning online control (M-OC) algorithm builds on the simple yet powerful idea of meta-initialization. In the standard OC algorithms, the initialization parameter for the control policy is selected arbitrarily from the domain of possible parameters. So, the regret guarantee for such algorithms inevitably includes a multiplicative constant corresponding to the radius of the domain (see Remark 1), which can be very large in many problems. When an independent-learning OC algorithm is applied to a sequence of tasks, the parameters of the control policy are initialized arbitrarily each time, without considering the benefits of learning across tasks. However, when the tasks are similar, the optimal parameters for the individual tasks can be closer to each other. This scenario is illustrated in Fig. 1, where the diameter $D$ of the original domain $\mathcal{M}$ is significantly larger than $D^*$, which is the diameter of the smaller set $\mathcal{M}^*$ that contains the optimal parameters corresponding to the similar tasks. Here the diameter $D^*$ can be interpreted as the similarity of the sequence of tasks. Our M-OC algorithm leverages this intuitive observation to provide a clever initialization for the control policy for each task by learning from the previous tasks. This results in a multiplicative constant (in the regret) that is proportional to the diameter $D^*$ of a much smaller subset that contains the parameters of the optimal control policies of the individual tasks, instead of the diameter of the generic domain.

The architecture of our M-OC algorithm is given in Fig. 2. Our architecture has a two loop structure. The outer loop is the meta-learner, which provides the meta-initialization for the task specific OC algorithm, which constitutes the inner loop. The task specific OC algorithm in our case is of the same form as the independent learning OC algorithm (5). At the beginning of any task $\tau_i$, a $(\kappa, \gamma)$ stabilizing feedback gain matrix $K_i$ for the task $\tau_i$ is computed. During the task, the algorithm updates the task specific parameter matrices $M_{i,t}$ exactly as in Algorithm 1. The control action $u_{i,t}$ is computed using the parameters $M_{i,t}$ and the feedback gain matrix $K_i$ with the same form as the independent learning OC algorithm (5). The difference between the M-OC algorithm and Algorithm 1 lies in the initialization of the parameter $M_{i,1}$. In particular, Algorithm 1 selects $M_{i,1}$ arbitrarily from the domain $\mathcal{M}$, whereas the outer loop of meta-learner provides the initialization $M^m_i$ for each task $\tau_i$.

Specifically, the inner loop updates the control policy parameter $M_{i,t}$ within each task $\tau_i$ by

$$M_{i,t+1} = \text{Proj}_\mathcal{M} (M_{i,t} - \nabla g_i(M_{i,t})) , M_{i,1} = M^m_i. \quad (6)$$

In the outer loop, the meta learner computes the initialization parameter $M^m_i$ for the inner loop as follows. Let $M^*_i$ be the optimal parameter in hindsight for task $\tau_i$, i.e.,

$$M^*_i = \arg\min_{M \in \mathcal{M}} \sum_{t=1}^{T} g_i(M) . \quad (7)$$

We note that $M^*_i$ is computable at the end of task $\tau_i$. Given that $g_i$s are convex functions, finding $M^*_i$ is a convex optimization problem and therefore can be solved efficiently. We define the meta-learner’s loss for task $i$ as

$$\mathcal{L}(M^m) = \frac{1}{2} \|M^*_i - M^m\|^2 . \quad (8)$$

The meta-learner performs an online gradient descent step to find the initialization $M^m_{i+1}$ for task $\tau_{i+1}$:

$$M^m_{i+1} = \text{Proj}_\mathcal{M} \left( M^m_i - \frac{1}{i} \nabla \mathcal{L}(M^m) \right) . \quad (9)$$

We note that performing the naive initialization $M^m_{i+1} = M^*_i$ does not improve the regret because this effectively throws away the information from all the previous tasks. Instead, the meta-learner solves an online convex optimization problem with $N$ steps, with the cost function at each step $i$ given by $\mathcal{L}^i$. Since online gradient descent efficiently solves this problem with provable guarantees for the regret, we adapt this approach as our meta-learning algorithm in the outer loop. The complete M-OC algorithm is given in Algorithm 2.

We now present our main result which characterizes the performance of our M-OC algorithm.
Remark 4 (Achievability by meta learning). We note that the meta-regret scaling with respect to the duration $T$ of a control task is $O(\sqrt{T})$, which is the same scaling achieved by the independent learning OC algorithm. This aspect is consistent with the existing theoretical results in online meta-learning [18]–[20]. This is expected, as the meta learner will never be able to learn a initialization that does not require further adaptation, especially, since the cost functions and the disturbances are arbitrary. Furthermore, as pointed in [19, Theorem 2.2], even in the simpler OCO setting, reductions in the multiplicative constant are the best that can be achieved.

V. NUMERICAL EXPERIMENTS

In this section, we present numerical experiments to demonstrate the benefits of our proposed meta-learning online control algorithm. In our experiments, each task $\tau_i$ is equivalent to the problem of controlling a linear dynamical system given in 1 with dimensions $n = 2, m = 1$. For each task $\tau_i$, the system model $A_i$ is selected as a random matrix that is a perturbation around a nominal matrix. In particular, we select $A_i = \frac{1}{\sqrt{m}} I + \frac{1}{m} W_i$, where $W_i$ is a random matrix with the value of each element generated uniformly from the interval $[0, 1]$. This structure implicitly incorporates the idea of task similarity. The cost functions $c_i$’s are selected as quadratic cost functions, $c_i(x, u) = x^T Q_i x + u^T R_i u$, where $Q_i$ and $R_i$ are randomly chosen diagonal matrices with each diagonal element lying in the range $[0.375, 0.625]$. The other relevant parameters are selected as $\kappa_a = \kappa_b = \kappa_w = 1, \gamma = \sqrt{\frac{1}{m}}, \gamma = 0.5$.

In our experiments, we compare the performance of our M-OC algorithm with the following benchmarks.

(i) Non-adaptive control algorithm which employs the control policy $u_{i,t} = -K_i x_{i,t}$, where $K_i$ is a stabilizing controller for task $\tau_i$ with system parameter $\theta_i = [A_i, B_i]$. We select $K_i$ by solving a standard linear matrix inequality (LMI) for finding a stabilizing controller. We call this non-adaptive control because the control policy is not changing over the duration of the control tasks. Moreover, there is no learning across the tasks.

(ii) Independent-learning online control algorithm which employs the task specific OC algorithm (Algorithm 1) independently for each control task. While this approach is capable of learning within a task, it does not perform any meta-learning across the tasks.

Different from these benchmarks, our M-OC algorithm can learn within and across the tasks.

Figure 3 shows the meta-regret $R^\text{meta}_N$ as a function of the number of tasks $N$, with $T = 25$ for all tasks. Note that meta-regret is equivalent to the average (averaged over the tasks) cumulative regret of the tasks, see (4). Since the non-adaptive control algorithm and the independent-learning OC algorithm are not performing any learning across the tasks, their meta-regret saturates to a constant. In stark difference to this, the meta-regret of our M-OC algorithm decreases as the number
of tasks increases (see Remark 2 also). This is because our M-OC algorithm is performing meta-learning across the tasks. This clearly demonstrates the superior performance of the M-OC algorithm over the benchmarks without meta-learning.

Figure 4 shows the variation of the meta-regret with $N = 15$ tasks as a function of the duration $T$ of each control task. We see that, when the task duration is small, the M-OC outperforms independent learning OC by a notable margin. This indeed is the very purpose meta-learning, i.e., to improve adaptation when the data or experience available to learn is limited.

VI. CONCLUSION

In this paper, we develop a meta-learning online control algorithm for a sequence of similar control tasks. We specifically study the setting where each task is that of controlling a linear dynamical system with arbitrary disturbance sequence and arbitrary cost function sequence. We propose a meta-learning online control algorithm for this setting that provably achieves superior performance over a standard online control algorithm that does not use meta-learning. We also numerically demonstrate the superior performance of our algorithm.