Cloud-based collaborative learning of optimal feedback controllers

Valentina Breschi * **Laura Ferrarotti** ** **Alberto Bemporad** **

* *Dipartimento di Elettronica e Informazione, Politecnico di Milano, 20133 Milan, Italy (e-mail: valentina.breschi@polimi.it).*
** *IMT School for Advanced Studies Lucca, 55100 Lucca, Italy (e-mail: {laura.ferrarotti,alberto.bemporad}@imtlucca.it)*

Abstract: Industrial systems deployed in mass production, such as automobiles, can greatly benefit from sharing selected data among them through the cloud to self-adapt their control laws. The reason is that in mass production systems are clones of each other, designed, constructed, and calibrated by the manufacturer in the same way, and thus they share the same nominal dynamics. Hence, sharing information during closed-loop operations can dramatically help each system to adapt its local control laws so to attain its own goals, in particular when optimal performance is sought. This paper proposes an approach to learn optimal feedback control laws for reference tracking via a policy search technique that exploits the similarities between systems. By using resources available locally and on the cloud, global and local control laws are concurrently synthesized through the combined use of the alternating direction method of multipliers (ADMM) and stochastic gradient descent (SGD). The enhancement of learning performance due to sharing knowledge on the cloud is shown in a simple numerical example.

Keywords: Consensus and Reinforcement learning control, Control over networks.

1. INTRODUCTION

Research on data-driven control is now gaining renewed interest within the control community, given the limitations of model-based design in coping with uncertain real-world systems and varying operating conditions. Data-driven control strategies, such as virtual reference feedback tuning (Campi et al., 2002), aim at synthesizing controllers from batches of input/output data, bypassing the model identification phase. These approaches rely on the choice of a reference model, representing the desired behavior in closed loop. Although this decision is of paramount importance for closed-loop performance, the selection of the reference model is generally performed a priori, without any guarantee that the desired behavior is actually attainable given the selected class of controllers. By sharing the same philosophy of data-driven control strategies without requiring the selection of a reference model, *model-free reinforcement learning* (RL) exploits information collected from interactions between a process and the environment to determine optimal *policies*, *i.e.,* control laws that maximize a given *reward*. RL methods are generally classified as *actor-critic*, *critic-only*, and *actor-only* (or *policy search*) approaches (Konda and Tsitsiklis, 2003), with the latter exploiting a specific parameterization of the policy instead of resorting to successive approximations of the cost to be optimized. Although investigated within both the control and the machine learning communities, methods for data-driven design of optimal controllers generally leverage information gathered from a single plant. As collecting informative data from a single plant might require running it for a long time, exploration throughout the learning phase is generally quite limited, often requiring additional efforts to satisfactorily search the state and action spaces (*e.g.,* add exploration noise to the best decision).

Thanks to recent advances in cloud computing, it is now technically and economically feasible to collect and store information gathered from different plants with the same nominal behavior, for example in a large volume production setting. Since it is likely that these plants (or *agents*) share similar objectives while operating within different environments, the design of local control policies might be improved by using the additional information within experiences shared by other agents. In fact, each plant may explore different regions of the state and action spaces than others, so that the union of such explorations can provide a wide coverage of the operating space. In case of agents having limited embedded computing capabilities but access to resources on the cloud, we can even assume that each agent locally performs simple operations only.

In this paper, we present a *policy search* approach to design feedback optimal controllers for systems whose dynamics are only known to be similar, that are allowed to share their experiences through the cloud. Differently from *multi-agent* RL approaches, such as the one in (Dimakopoulou et al., 2018), the proposed method is not aimed at finding local policies for systems interacting within the same (non-stationary) environment and cooperating to perform a common task. Instead, here the advantage of a cloud-aided framework is exploited by introducing a *global* policy, that is related to the local control laws by (known) constraints, reflecting the similarity among the agents. We will handle such constraints via the *Alternating Direction Method of Multipliers* (ADMM) (Boyd et al., 2011). Differently from what is generally

done in *parallel* RL, see, *e.g.,* (Nair et al., 2015), the use of ADMM allows systems to share a surrogate of their experiences, *e.g.,* their policies, while their states, actions and rewards are retained locally, which might be useful when the agents cannot share information other than their policies for privacy or security reasons. Although the rationale behind the method is fairly general, we focus on the combination of ADMM with the *policy gradient* strategy proposed in (Ferrarotti and Bemporad, 2019) and we consider output tracking problems over groups of similar agents, that are not forced to track exactly the same reference, but are supposed to pursue similar goals. Given the similarities between the agents and their goals, we search for an optimal *global* policy for the whole group. Within a multi-agent cooperative setup, the problem of searching for a unique policy is also tackled in (Khan et al., 2018) under the assumption that the local policies of the agents lie in the same space. Differently from our approach, in (Khan et al., 2018) the quest for a global policy is introduced to reduce the computational complexity due to the interactions of multiple agents, that work together within the same environment to attain a common goal.
The paper is organized as follows. The considered policy search problem is presented in Section 2, along with the description of the specific setting considered throughout the paper. The proposed approach is presented in Section 3, and its performance is assessed on a simple simulation example in Section 4. Conclusions and directions for future work are finally reported in Section 5.

Notation: Let $\mathbb{N}$ and $\mathbb{R}^n$ be the set of natural numbers and of real vectors of dimension n, respectively. Let the identity matrix be denoted by I. Given a $(n \times m)$-dimensional matrix $A \in \mathbb{R}^{n\times m}$, A' indicates its transpose. Given a vector $x \in \mathbb{R}^n$, $\|x\|_2$ denotes its Euclidean norm, while $\|x\|_Q^2 = x'Qx$, where $Q \in \mathbb{R}^{n\times n}$.

2. PROBLEM FORMULATION

Consider N dynamical systems (also denoted as *agents*) characterized by similar (but unknown) dynamics and interaction with the environment. Let $s_n(t) \in \mathbb{R}^{n_s}$ be a Markovian signal specifying the behavior of the n-th system, $n = 1, \dots, N$, evolving over time according to a shared (unknown) model

$$s_n(t+1) = h(s_n(t), p_n(t), u_n(t), d_n(t)), \tag{1}$$

where $p_n(t) \in \mathbb{R}^{n_s}$ is a vector of measurable exogenous signals, $d_n(t) \in \mathbb{R}^{n_d}$ is a vector of unmeasured disturbances and $h : \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_s}$ is common to all agents. The signal $s_n(t)$ might include the state of the n-th system, whose dynamic evolution is described by the shared (unknown) state-space model

$$\begin{aligned} x_n(t+1) &= f(x_n(t), u_n(t), d_n(t)), \\ y_n(t) &= g(x_n(t), d_n(t)), \end{aligned} \tag{2}$$

where $x_n(t) \in \mathbb{R}^{n_x}$, $u_n(t) \in \mathbb{R}^{n_u}$ and $y_n(t) \in \mathbb{R}^{n_y}$ are the state, deterministic input and measured output of the n-th agent at time t, and the (unknown) nonlinear functions $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_y}$ are common to all agents.

In this paper we aim at finding N *optimal deterministic control policies* $\pi_n : \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_u}$, providing the input to the n-th system at each time t

$$u_n(t) = \pi_n(s_n(t), p_n(t)). \tag{3}$$

Similarly to (Ferrarotti and Bemporad, 2019), the concept of *optimal policy* is introduced via the definition of a *local stage cost* $\rho_n : \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_u} \to \mathbb{R}$, which returns the cost of applying the policy π_n at time t. In this work, we assume individual stage costs to be similar, *e.g.,* all the agents aim at optimally tracking their own output reference. Based on ρ_n, and given an initial condition $s_n(0)$ and a sequence $\{p_n(l), d_n(l)\}_{l=0}^{\infty}$, the local cost of the deterministic policy π_n over an infinite control horizon is defined as

$$\begin{aligned} J_n^{\infty}(\pi_n, &s_n(0), \{p_n(l), d_n(l)\}_{l=0}^{\infty}) = \\ &= \sum_{l=0}^{\infty} \rho_n(s_n(l+1), p_n(l), \pi(s_n(l), p_n(l))), \end{aligned} \tag{4}$$

with $s_n(l)$ evolving according to (1). The *overall* performance of the deterministic policy π_n can thus be defined as

$$J_n(\pi_n) = \mathop{\mathbb{E}}_{\substack{S_n(0) \\ \{P_n(l), D_n(l)\}_{l=0}^{\infty}}} [J_n^{\infty}(\pi_n, S_n(0), \{P_n(l), D_n(l)\}_{l=1}^{\infty})], \tag{5}$$

where $S_n(0)$ is a random variable representing the initial value of the trajectory of the n-th agent, which evolves according to (1), and $\{P_n(l), D_n(l)\}_{l=0}^{\infty}$ are random variables whose realizations are the signals $\{p_n(l), d_n(l)\}_{l=0}^{\infty}$.
By accounting for the similarities between the N systems and relying on (4)-(5), our aim is to find the policies $\{\pi_n^\star\}_{n=1}^N$ solving the following optimization problem:

$$\begin{aligned} \underset{\{\pi_n\}_{n=1}^N, \pi}{\text{minimize}} \ & \sum_{n=1}^{N} J_n(\pi_n) \\ \text{s.t. } & \phi(\pi_n) = \pi, \quad n \in \{1, \dots, N\}, \end{aligned} \tag{6}$$

where the (known) function ϕ describes the relation between the local policies and a *global* control law $\pi : \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_u}$, thus embedding the similarities between the local policies.

Since the cost of each policy π_n in (5) is defined over an infinite control horizon, problem (6) cannot be solved in practice. For computing the solution, we approximately evaluate the cost over trajectories of prefixed finite length L, with the *performance index* (4) recast as:

$$\begin{aligned} J_n^{L}(\pi_n, &S_n(0), \{P_n(l), D_n(l)\}_{l=0}^{L-1}) = \\ &= \sum_{l=0}^{L-1} \rho_n(s_n(l+1), p_n(l), \pi_n(s_n(l), p_n(l))). \end{aligned} \tag{7}$$

and the cost of applying the local policies given by the expectation

$$J_n(\pi_n) = \mathop{\mathbb{E}}_{\substack{S_n(0) \\ \{P_n(l), D_n(l)\}_{l=0}^{L-1}}} \left[J_n^{L}(\pi_n, S_n(0), \{P_n(l), D_n(l)\}_{l=0}^{L-1})\right]. \tag{8}$$

Because of the above approximations, the optimal policies computed through (6) with the costs in (7)-(8) only approximate the actual solutions $\{\pi_n^\star\}_{n=1}^N$ and $\pi^\star$ of the original problem.

2.1 Policy structure

Suppose that the N agents have to track given set points $\{r_n(t) \in \mathbb{R}^{n_y}\}_{n=1}^N$, which might differ from one plant to the other depending on their individual tasks. Assume that we restrict our search within the class of linear policies, namely

$$\pi_n(s_n(t), p_n(t)) = -K_n \begin{bmatrix} s_n(t) \\ p_n(t) \end{bmatrix}. \tag{9}$$

Since the state of a dynamical system is seldom directly measured in practice (especially if the model of the system is not known), we solve the control problem in an *input/output setting*, as commonly done in data-driven control (*e.g.,* see (Campi et al., 2002)) and consider the following state

$$x_n(t)=\big[y_n(t)'\cdots y_n(t-n_a+1)'\ u_n(t-1)'\cdots u_n(t-n_b)'\big]', \tag{10}$$

with n_a and n_b fixed by the user.

Let $q_n(t)$ be the integral of the tracking error that system n encounters,

$$q_n(t+1) = q_n(t) + (y_n(t+1) - r_n(t)). \tag{11}$$

To attain offset-free steady-state tracking of constant set-points, we define $s_n(t)$ and $p_n(t)$ as

$$s_n(t) = \begin{bmatrix} x_n(t) \\ q_n(t) \end{bmatrix}, \quad p_n(t) = r_n(t), \tag{12}$$

respectively and, based on these definitions, we consider the quadratic stage costs

$$\begin{aligned}&\rho_n(s_n(l+1), r_n(t), \Delta u_n(t)) = \\ &= \|Cx_n(t+1) - r_n(t)\|^2_{Q_y} + \|\Delta u_n(t)\|^2_R + \|q_n(t+1)\|^2_{Q_q},\end{aligned} \tag{13}$$

with $\Delta u_n(t) = u_n(t) - u_n(t-1)$. The constant matrices $Q_y = Q_y' \succ 0$, $Q_q = Q_q' \succ 0$, and $R = R' \succ 0$ weighting the tracking error and the control effort, respectively, and C are common to all systems.

Given the similarities between the plants and their goal, it is reasonable to search for a *global* linear policy, *i.e.,*

$$\pi(s_n(t), p_n(t)) = -K \begin{bmatrix} s_n(t) \\ p_n(t) \end{bmatrix} = -K^s s_n(t) - K^r p_n(t), \tag{14}$$

common to all systems, irrespectively of the different local set points. Problem (6) can thus be recast as follows,

$$\begin{aligned} \text{minimize} \ & \sum_{n=1}^{N} J_n(K_n) \\ \text{s.t.} \ & K_n - K = 0, \quad n = 1, \dots, N, \end{aligned} \tag{15}$$

with $J_n(K_n)$ defined as in (8), and $K \in \mathbb{R}^{n_s+n_p}$ and $\{K_n \in \mathbb{R}^{n_s+n_p}\}_{n=1}^N$ being the global and the local gains to be designed. Note that the cost in (15) depends on the unknown models (1).

3. POLICY SEARCH STRATEGY

To solve problem (15), we search for the global policy K over a learning horizon T via an iterative ADMM-based scheme, that combines updates of the local policies K_n via mini-batch *Stochastic Gradient Descent* (SGD) (Robbins and Monro, 1951), and the computation of the global gain through the local information shared by the agents.
For every $t \in \{0, \dots, T-1\}$, let the augmented Lagrangian associated to problem (15) be defined as

$$\mathcal{L} = \sum_{n=1}^{N} \mathcal{L}_n(K_n, \delta_n, K), \tag{16a}$$

$$\mathcal{L}_n(K_n, \delta_n, K) = J_n(K_n) + \delta_n'(K_n - K) + \frac{\beta}{2}\|K_n - K\|_2^2, \tag{16b}$$

with $\beta > 0$ being a tuning parameter and $\{\delta_n\}_{n=1}^N$ being the Lagrange multipliers associated with the constraints of problem (15). By running a new instance of ADMM (Boyd et al., 2011) at each time t, the local and global policies are computed via the following steps:

$$\begin{aligned} K_n^{i+1} = \underset{\substack{K_n \\ \{P_n(l), D_n(l)\}_{l=0}^{L-1}}}{\operatorname{argmin}} & \underset{S_n(0)}{\mathbb{E}}\big[J_n^L(K_n, S_n(0), \{P_n(l), D_n(l)\}_{l=0}^{L-1})\big] + \\ & + (\delta_n^i)'(K_n - K^i) + \frac{\beta}{2}\|K_n - K^i\|_2^2, \quad n = 1, \dots, N \end{aligned} \tag{17a}$$

$$K^{i+1} = \frac{1}{N}\sum_{n=1}^{N}\left[K_n^{i+1} + \frac{1}{\beta}\delta_n^i\right], \tag{17b}$$

$$\delta_n^{i+1} = \delta_n^i + \beta(K_n^{i+1} - K^{i+1}), \quad n = 1, \dots, N, \tag{17c}$$

where $i \in \mathbb{N}$ is a counter that indicates the number of ADMM iterations [1]. According to (17b), the global estimate is the average of the local polices and Lagrange multipliers, thus relying on information collected from all agents. By looking at the steps in (17), it is also clear that at each ADMM iteration the global policy can be updated on a central processing unit, provided the updated local estimates $\{K_n^{i+1}\}_{n=1}^N$, while the local policies and the local Lagrange multipliers can be updated either locally or by using parallel dedicated resources on the cloud. The iterations of the local policy search method to be carried out at a given time instant are summarized in Algorithm 1, which indicates a possible allocation of the different operations involved in (17). In particular, at each time step t, the n-th local policy K_n is updated by solving (17a) via M iterations of mini-batch SGD, with the local policies *warm-started* with the estimates obtained at the previous time step.

Remark 1. The steps in (17) can be performed either *offline* or *online*. However, a *real-time* implementation requires synchronous back and forth communications between the agents and the central processing unit, which might be unfeasible, especially for fast sampling systems. Since in the considered setting we search for the optimal constant gain in (14), after some initial steps we expect the lags in communication not to substantially deteriorate the performance of the method. ■

3.1 Local linear model update

The use of mini-batch SGD involves the computation of the stochastic gradient $\nabla_{K_n} J_n^L$, which requires the knowledge of the (unknown) dynamic relation (1). This limitation is overcome by locally estimating a linear approximation of (1). Specifically, at each time instant t N linear models, one per agent, are estimated recursively. The dynamics of the n-th system in (2) can be locally approximated as

$$y_n(t) = \Theta_n(t)\chi_n(t) + d_n(t), \tag{18a}$$

where $d_n(t)$ is a zero-mean Gaussian white noise with covariance Q_n^k and $\chi_n(t) = [x_n'(t-1), u_n(t-1)']'$, with $x_n(t)$ defined as in (10). The parameter matrix $\Theta_n(t) \in \mathbb{R}^{n_y \times (n_a n_y + n_b n_u)}$ is assumed to satisfy

$$\Theta_n(t+1) = \Theta_n(t) + \xi_n(t), \tag{18b}$$

with $\xi_n(t)$ being a zero-mean Gaussian white noise with covariance matrix R_n^k. As in (Ferrarotti and Bemporad, 2019), the parameter vector $\Theta_n(t)$ is recursively updated by Kalman filtering (Kalman, 1960) and holds constant over the whole horizon L over which the gradient is computed. Given the dependence of the stage cost (13) on

[1] The dependence on time is dropped to simplify the notation.

Algorithm 1 ADMM-based policy search at time t

Inputs: State histories $\{X_n(t)\}_{n=1}^N$; current references $\{r_n(t)\}_{n=1}^N$; local models $\{\Theta_n(\tau)\}_{n=1}^N$, $\tau = 0,\dots,t-1$; previous local $\{K_n(t-1)\}_{n=1}^N$ and global $K(t-1)$ polices; initial Lagrange multipliers $\{\delta_n^0\}_{n=1}^N$; $\beta > 0$.

for $n = 1$ to N **do**
  update the linear model $\Theta_n(t)$; (agent)
  transmit the model to the fusion center; (agent)
  set $K_n^0(t) = K_n(t-1)$ and $K^0 = K(t-1)$; (cloud)
end for
while $i < i_{max}$ or another criterion is verified **do**
  for $n = 1$ to N **do**
    set $K_n^m(t) = K_n^i(t)$, for $m = 0$; (cloud)
    for $m = 1$ to M **do**
      run Algorithm 2;
      compute $\mathcal{D}_n(K_n^{m-1}(t))$ as in (21); (cloud)
      update the local policy as in (26); (cloud)
    end for
    set $K_n^{i+1}(t) = K_n^M$; (cloud)
  end for
  compute K^{i+1} as in (17b); (cloud)
  for $n = 1$ to N **do**
    compute δ_n^{i+1} as in (17c); (cloud)
  end for
end while

Outputs: Updated policies $\{K_n(t)\}_{n=1}^N$ and $K(t)$.

both the state of the n-th agent and the input increment $\Delta u_n(t)$, it is quite straightforward to recast (18a) as

$$y_n(t) = \Theta_n^x x_n(t-1) + \Theta_n^u(t)\Delta u_n(t-1) + d_n(t), \quad (18c)$$

where $\Theta_n^x(t)$ and $\Theta_n^u(t)$ can be easily derived from $\Theta_n(t)$. From the definitions of $s_n(t)$ and $p_n(t)$ in (12), the linear approximation for the local dynamics in (1) is obtained by introducing also the integral action, which is known to evolve according to (11). Let

$$A_n(t) = \begin{bmatrix} \Theta_n^x(t) \\ \bar{A} \end{bmatrix}, \quad B_n(t) = \begin{bmatrix} \Theta_n^u(t) \\ \bar{B} \end{bmatrix}, \quad (19)$$

where $\bar{A}$ and $\bar{B}$ are fixed binary matrices. By combining (11) and (18c), starting from t the behavior of the n-th agent over an horizon of length L is thus given by the following *linear* model:

$$\begin{aligned} s_n(l+1) &= \mathcal{A}s_n(l) + \mathcal{B}\Delta u_n(l) + Ep_n(l) + Hd_n(l), \\ y_n(l) &= Cs_n(l), \end{aligned} \quad (20a)$$

with

$$\mathcal{A} = \begin{bmatrix} A_n(t) & 0 \\ CA_n(t) & I \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} B_n(t) \\ CB_n(t) \end{bmatrix}, \quad C = [I\ 0], \quad E = \begin{bmatrix} 0 \\ -I \end{bmatrix}, \quad H = \begin{bmatrix} I \\ 0 \end{bmatrix}, \quad (20b)$$

for $l = 0,\dots,L-1$.

3.2 Sampling strategy and local policy update

For each agent and each time step t, every iteration $m \in \{1,\dots,M\}$ of *local* mini-batch SGD involves the computation of the descent direction

$$\mathcal{D}_n(K_n) = \delta_n^i + \frac{\beta}{2}(K_n - K^i) + \quad (21)$$
$$+\frac{1}{N_t}\sum_{k=1}^{N_s}\sum_{j=1}^{N_r}\sum_{h=1}^{N_d}\nabla_{K_n}\hat{J}_n^L(K_n, s_n^{k,m}(0), \{r_n^{j,m}(l), d_n^{h,m}(l)\}_{l=0}^{L-1}),$$

Algorithm 2 m-th SGD iteration (on the cloud) at time t

Inputs: State histories $\{X_n(t)\}_{n=1}^N$; current references $\{r_n(t)\}_{n=1}^N$; local models $\{\Theta_n(\tau)\}_{\tau=0}^t$, $n = 1,\dots,n$; previous local policies $\{K_n^{m-1}(t)\}_{n=1}^N$; $\beta > 0$.

for $n = 1$ to N **do**
  for $w = 1$ to N_x **do**
    sample $x_n(\gamma_{n,t}^w)$ from $X_n(t)$;
    retrieve the local model $\Theta_n(\gamma_{n,t}^w)$;
    for $z = 1$ to N_q **do**
      sample $q_n^z(0)$;
      build $s_n^{w,z}(0)$ as in (24);
      for $j = 1$ to N_r **do**
        get a random reference r_n^j for (25);
        for $h = 1$ to N_d **do**
          get a trajectory $\{d_n^h(l)\}_{l=0}^{L-1}$;
          compute $\nabla_{K_n}\hat{J}_n^L(K_n^{m-1}(t), s_n^{w,z}(0), \eta_n^L)$;
        end for
      end for
    end for
  end for
end for

Outputs: local analytic gradients $\nabla_{K_n}\hat{J}_n^L$.

where $N_t = N_s N_r N_d$, and i denotes the ADMM iteration. For each sample $s_n^{k,m}(0)$ the approximated cost $\hat{J}_n^L$ is obtained by exploiting the model in (20a) over the whole horizon L from the initial state $s_n^{k,m}(0)$.

The computation of $\mathcal{D}_n(K_n)$ requires sampling the spaces of initial trajectories $\{s_n^{k,m}(0)\}_{k=1}^{N_s}$, exogenous signals $\{r_n^{j,m}(l)\}_{j=1}^{N_r}$, and disturbances $\{d_n^{h,m}(l)\}_{h=1}^{N_d}$, with $l = 0,\dots,L-1$. Since the signal $s_n(t)$ is defined as in (12), initial values for both the state of (2) and the integral of the tracking error have to be sampled to retrieve N_s values of $s_n(0)$. To this end, consider the following dataset:

$$X_n(t) = \{x_n(0), x_n(1), \dots, x_n(t)\}, \quad (22)$$

which stores all the states reconstructed according to (10) up to time t for the n-th agent. As the linear model in (20a) is locally updated at each time step and it roughly describes the behavior of the system in a neighborhood of the initial state itself, we approximately know how the n-th system behaves in the neighborhood of the points in $X_n(t)$ and, thus, we are able to compute the analytic gradient $\nabla_{K_n}\hat{J}_n^L$. As in (Ferrarotti and Bemporad, 2019), we sample N_x values for the initial local state from $X_n(t)$ as

$$x_n^{w,m}(0) = x_n(\gamma_{n,t}^{w,m}) + v_n^{w,m}, \quad w = 1,\dots,N_x, \quad (23)$$

where $\gamma_{n,t}^{w,m}$ is a randomly selected integer between 0 and t, $v_n^{w,m}$ is a perturbation sampled from a zero-mean Gaussian distribution with variance σ_v^2, introduced to explore the neighborhood of the trajectories generated by the plants. The initial states in (23) are combined with N_q samples $q_n^{z,m}(0)$, $z = 1,\dots,N_q$ drawn from a normal distribution with zero mean and variance σ_q^2, so to obtain the $N_s = N_x N_q$ samples $s_n^{k,m}(0) = s_n^{w,z,m}(0)$ of the initial trajectories as

$$s_n^{k,m}(0) = \begin{bmatrix} x_n^{w,m}(0) \\ q_n^{z,m}(0) \end{bmatrix}, w \in \{1,\dots,N_x\}, z \in \{1,\dots,N_q\}. \quad (24)$$

Along the horizon of length L, the reference of each agent is assumed to be constant, namely

$$r_n^{j,m}(l) = r_n^{j,m},\ l = 0, \dots, L,\ j = 1, \dots, N_r, \tag{25}$$

with $r_n^{j,m}$ randomly chosen in a fixed interval $[r_n^{min},\ r_n^{max}]$, which contains all the references of interest for the n-th system. Instead, $\{d_n^{h,m}(l)\}_{k=1}^{N_d}$ are sampled randomly from the box $[-d_n^{max},\ d_n^{max}]$, for $l = 0, \dots, L-1$. The proposed sampling strategy is summarized in Algorithm 2, where the dependence on the current SGD iteration m is maintained only on the local policy to simplify the notation.

Let $\eta_n^L = \{r_n^{j,m}(l), d_n^{h,m}(l)\}_{l=0}^{L-1}$. Given $(s_n^{k,m}(0), \eta_n^L)$, the n-th local policy is updated using the descent direction $\mathcal{D}_n(K_n^{m-1})$ computed at the previous SGD iteration K_n^{m-1} as in (21). In particular, at the m-th SGD step, the local policy is given by

$$K_n^m(t) = K_n^{m-1}(t) - \alpha_n^m \mathcal{D}_n(K_n^{m-1}(t)), \tag{26}$$

with $\{\alpha_n^m\}_{m=1}^M$ being a sequence of positive decreasing learning rates. Once the SGD iterations are terminated, the updated local policy is given by $K_n^{i+1}(t) = K_n^M(t)$.

Remark 2. The descent direction in (21) is clearly composed by a term depending on the local Lagrange multiplier and the global policy, while the other depends on the samples. In turn, the proposed sampling strategy requires all past histories and local models to be available. It is thus reasonable to put dedicated processing units in charge of updating the policies, while local processors can be exploited to find the local models only. ■

4. NUMERICAL RESULTS

Consider N agents described by the (unknown) *single-input single-output* (SISO) *linear* model

$$x_n(t+1) = \begin{bmatrix} -0.669 & 0.378 & 0.233 \\ -0.288 & -0.147 & -0.638 \\ -0.377 & 0.589 & 0.043 \end{bmatrix} x_n(t) + \begin{bmatrix} -0.295 \\ -0.325 \\ -0.258 \end{bmatrix} u_n(t),$$

$$y_n(t) = [-1.139\ \ 0.319\ \ -0.571]\, x_n(t),$$

and let the matrices in (13) be $Q_y = Q_q = 1$ and $R = 0.1$.

We want to find an optimal global policy K over $T = 500$ steps by means of the presented ADMM-based policy search approach within an ideal setting, with no latencies in communications and the availability of both local and global computational resources. Algorithm 1 is run online with the parameters reported in Table 1, and the local policies are updated by using a faster variant of SGD, AMSGrad (Reddi et al., 2019) [2] . AMSGrad is reinitialized every time ADMM iterations are terminated, when either $i_{max} = 100$ is attained or

$$\sum_{n=1}^{N} \|K_n^{i+1} - K^{i+1}\|_2^2 \leq \varepsilon,\quad i = 9, \dots, i_{max} - 1, \tag{27}$$

with $\varepsilon = 5 \cdot 10^{-3}$. At the first instant, global and local policies are all initialized by vectors of ones, $\{\Theta_n(0)\}_{n=1}^N$ are selected as zero vectors, the Lagrange multipliers are chosen as $\{\delta_n^0 = 10^{-3}\}_{n=1}^N$ (and then initialized with their values at the previous time step). The reference signals $\{r_n\}_{n=1}^N$ are piecewise constant and differ from one agent to the other. This implies that, despite the common optimality criterion, the agents aim at attaining different local goals. For comparison, we consider the optimal policy computed based on the real system, that is $K_{opt} = [-1.257\ \ 0.219\ \ 0.653\ \ 0.898\ \ 0.050\ \ 1.141\ \ -2.196]'$.

[2] We use the same parameters of (Ferrarotti and Bemporad, 2019), *i.e.,* $\alpha = 0.1$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The convergence of the global policy iterate $K(t)$ to K_{opt} is assessed by computing the first iteration such that the global policy is in the ϵ-neighborhood of K_{opt}, *i.e.,*

$$T_\epsilon \doteq \min\{t \in [1, T] \mid \|K(t) - K_{opt}\|_2 < \epsilon\}, \tag{28a}$$

while the quality of the estimate retrieved once the ϵ-neighborhood has been reached is evaluated via the following global indexes:

$$\text{avg}_\epsilon \doteq \frac{1}{T - T_\epsilon + 1} \sum_{t=T_\epsilon}^{T} \|K(t) - K_{opt}\|_2, \tag{28b}$$

$$\text{var}_\epsilon \doteq \frac{1}{T - T_\epsilon + 1} \sum_{t=T_\epsilon}^{T} (\|K(t) - K_{opt}\|_2 - \text{avg}_\epsilon)^2. \tag{28c}$$

We also assess the benefits of exploiting shared experiences by comparing our results to the ones obtained for N=1 with the baseline approach presented in (Ferrarotti and Bemporad, 2019). For our comparison to be fair, the ADMM-based policy search strategy is implemented online, generating the local inputs according to the current local policies.

4.1 Noiseless case

For $N = 2$, we initially consider a noise-free setting and we take $d_n = 0$ in (18a). Table 2 compares the presented strategy and the baseline (Ferrarotti and Bemporad, 2019) by means of the global performance indexes in (28). These results highlight a considerable difference in convergence speed between the single and the multi-agent case, leading to a lower average error avg_ϵ. This is further confirmed by the 2-norm of the global errors reported in Fig. 2, showing that convergence to the optimal policy drastically improves when considering two similar systems sharing information. By comparing the values of var_3 in Table 2, it is also clear that the estimate obtained with a single-agent is smoother than the retrieved global policy. As shown in Fig. 2, this is due to our choice of reinitializing AMSGrad at each ADMM iteration which, at the same time, allows us to considerably speed up convergence.

Sensitivity analysis

The performance attained by the proposed policy search strategy can be affected by different decisions that are made before the learning phase, such as the dimension of the batch N_t, and the number M of AMSGrad iterations performed at each ADMM step. The global performance indexes obtained for batches of increasing dimension by varying either N_q or N_x are shown in Table 3, imposing both agents to track the same reference. When N_q is modified, there is a slight change in the global performance indexes, which seems linked to the random nature of the batch construction process. Instead, a slight improvement in the overall performance is observed when increasing N_x. Indeed, the larger the batch is, the smaller the indexes are. For different iterations M of AMSGrad, we obtain the indexes reported in Table 4. Both $\text{avg}_{0.5}$ and $\text{var}_{0.5}$ decrease when M increases, leading to smoother variations during the learning phase. Finally, we study how the global performance change when N is increased. Table 5 reports the results obtained for groups of agents of different dimensions, showing the benefits linked to the experiences provided by the additional agents.

Table 1. Parameters for ADMM-based policy search

n_a	n_b	β	Q_n^k	R_n^k	L	N_x	N_r	N_q	N_d	σ_v	σ_q	r_n^{min}	r_n^{max}	M
3	2	1	$10^{-3}I$	0.1	20	50	1	10	0	10^6	1	-10^3	10^3	10

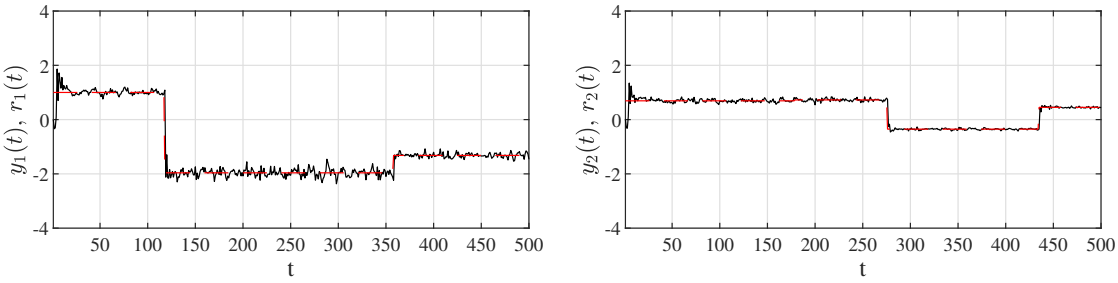

Fig. 1. Output of each plant (black) vs local references to be tracked (dashed red).

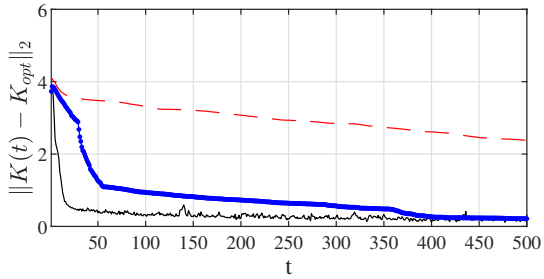

Fig. 2. $\|K(t)-K_{opt}\|_2$ with warm-started (black) and not reinitialized (dot-dashed blue) policies . N=2 (black) and $N = 1$ (dashed red).

Table 2. Global indexes: $N = 2$ vs $N = 1$.

N	T_3 [steps]	avg$_3$	var$_3$
1	224	2.695	0.035
2	5	0.310	0.051

Table 3. Global indexes vs batch dimension N_t.

N_t	N_x	N_q	$T_{0.5}$	avg$_{0.5}$	var$_{0.5}$
500	50	10	31	0.263	0.006
50	50	1	23	0.271	0.007
500	50	10	31	0.263	0.006
100	10	10	34	0.286	0.011

Table 4. Global indexes vs AMSGrad steps M.

M	$T_{0.5}$	avg$_{0.5}$	var$_{0.5}$
1	33	3.949	11.348
5	28	0.385	0.031
40	27	0.251	0.005

Table 5. Global indexes vs N.

N	$T_{0.5}$	avg$_{0.5}$	var$_{0.5}$
2	25	0.275	0.007
3	23	0.267	0.006
4	30	0.264	0.005

Table 6. Global and local indexes (noisy case).

$T_{0.5}$ [steps]	avg$_{0.5}$	var$_{0.5}$
24	0.255	0.004

4.2 Noisy case

Suppose now that the measurements are affected by noise, modeled as a zero-mean Gaussian process with variance 10^{-4}. As shown in Table 6, the performance attained by maintaining $d_n = 0$ in (18a) are comparable to the ones achieved for the noiseless case. In Fig. 1, we show the tracking performance attained during learning by each agent when controlled by its consensus-shaped local policy.

5. CONCLUSION

We have extended the optimal policy search method proposed in (Ferrarotti and Bemporad, 2019) to considerably speed up convergence by exploiting information shared by a group of similar systems operating in different conditions. The consensus on policy parameters is achieved by embedding SGD iterations within ADMM.

Future research will be devoted to relax the synchronization requirements and to investigate the effect of long term differences that might arise between agents, for example due to different aging.

REFERENCES

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1), 1–122.

Campi, M., Lecchini, A., and Savaresi, S. (2002). Virtual reference feedback tuning: a direct method for the design of feedback controllers. *Automatica*, 38(8), 1337 – 1346.

Dimakopoulou, M., Osband, I., and Van Roy, B. (2018). Scalable coordinated exploration in concurrent reinforcement learning. In *Advances in Neural Information Processing Systems 31*, 4219–4227.

Ferrarotti, L. and Bemporad, A. (2019). Synthesis of optimal feedback controllers from data via stochastic gradient descent. In *2019 18th European Control Conference (ECC)*, 2486–2491.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D), 35–45.

Khan, A., Zhang, C., Lee, D., Kumar, V., and Ribeiro, A. (2018). Scalable centralized deep multi-agent reinforcement learning via policy gradients. *arXiv preprint arXiv:1805.08776*.

Konda, V. and Tsitsiklis, J. (2003). On actor-critic algorithms. *SIAM J. Control Optim.*, 42(4), 1143–1166.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., Legg, S., Mnih, V., Kavukcuoglu, K., and Silver, D. (2015). Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

Reddi, S., Kale, S., and Kumar, S. (2019). On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3), 400–407.