Proceedings of Machine Learning Research vol 144:1–12, 2021

The benefits of sharing: a cloud-aided performance-driven framework to learn optimal feedback policies

Laura Ferrarotti LAURA.FERRAROTTI@IMTLUCCA.IT
IMT School for Advanced Studies Lucca, 55100, Lucca, Italy

Valentina Breschi VALENTINA.BRESCHI@POLIMI.IT
Politecnico di Milano, 20133 Milano, Italy

Alberto Bemporad ALBERTO.BEMPORAD@IMTLUCCA.IT
IMT School for Advanced Studies Lucca, 55100, Lucca, Italy

Abstract

Mass-produced systems are constructed and calibrated to be nominally the same, and they usually have similar goals. When several of these systems can share information with the cloud, one can exploit their similarities to improve the design of individual control policies. In this framework, we aim to exploit these similarities and the connection to the cloud to solve a sharing-based control policy optimization problem, leveraging on information provided by trustworthy agents. In this paper, we propose to combine the optimal policy search method introduced in (Ferrarotti and Bemporad, 2019) with the Alternating Direction Method of Multipliers, by relying on a weighted surrogate of the experiences of each device shared within the cloud. A preliminary example shows the effectiveness of the proposed sharing-based method, which results in improved performance with respect to the ones attained when neglecting the similarities among devices and when enforcing consensus among their policies.

Keywords: Reinforcement Learning control, Sharing, Control over networks.

1. Introduction

In a world where the complexity of processes to be controlled is always increasing, along with the availability of large datasets, data-driven techniques allow one to use data to obtain control actions directly. This enables the designer to bypass the identification step characterizing model-based control strategies, that is known to be time consuming and usually aimed at finding models whose accuracy and complexity might be excessive with respect to the control task at hand. However, many data-driven control methods, see *e.g.,* (Campi et al., 2002; Hjalmarsson, 2002; Karimi et al., 2004), rely on reference models that shape the desired closed-loop response of the system and that have to be blindly selected before-hand. This might be a rather challenging task and only few methods have been proposed in the literature to automatize such choice, see (Selvi et al., 2018; Campestrini et al., 2011; Kergus et al., 2019).

By sticking to the principles of model-based predictive control, several approaches have been recently proposed that rely on behavioral system theory to avoid an explicit identification step (Coulson et al., 2019, 2020). Alternatively, model-free *Reinforcement Learning* (RL) approaches are often used for control design, where the actions are determined so as to maximize a predefined reward (Recht, 2019; Konda and Tsitsiklis, 2003; Watkins and Dayan, 1992; Grondman, 2015). All these

methods mainly focus on the design of policies for one plant only, thus usually requiring rather long and expensive experimental campaigns to obtain informative data.

The advances in cloud computing allow designers to overcome this limitation, by leveraging on the increasing connectivity between devices to gather more information when searching for control policies for systems that share similar dynamics and related individual tasks. This is indeed quite common, particularly for mass produced devices, that are by construction and calibration nominally the same. Shared information can thus be exploited to improve exploration and, ultimately, to retrieve better individual policies. As a motivating example, consider N vehicles that aim at performing path following tasks that are similar in nature, but somehow different in their individual realizations. By sharing either their experiences or surrogate of the latter, more insights on how to handle the considered task are likely available for the design of the local policies. The sharing process is enabled among modern connected vehicles, that maintain a constant Vehicle-to-Cloud bi-directional connection with sufficient data bandwidth for different monitoring, maintenance, and updating services.

Sharing-based principles have already been exploited for policy search purposes, in scenarios where all agents are cooperating in the same environment towards the achievement of the same goal, see *e.g.,* (Dimakopoulou et al., 2018), or to steer all local policies to the same value, under the assumption that they lie in the same space, *e.g.,* as in (Nair et al., 2015; Khan et al., 2018). In (Breschi et al., 2020), as well, agents characterized by *similar*, yet unknown, dynamics and by *similar goals* are driven to "agree" on a global consensus, while working in diverse environments and having different local tasks. Differently from (Dimakopoulou et al., 2018), the agents are not required to share the same goal and to operate within the same environment. Unlike (Khan et al., 2018), the approach proposed in (Breschi et al., 2020) does not require the agents to share their private states, but surrogate of their experiences only.

Considering the same setup of (Breschi et al., 2020), in this work we present a policy search method that explicitly exploits the analogies between the plants and their connection to the cloud. Similarly to (Breschi et al., 2020), we propose to embed the policy search method described in (Ferrarotti and Bemporad, 2019) within a scheme based on the *Alternating Direction Method of Multipliers* (ADMM) (Boyd et al., 2010), that allows the agents to share surrogate of their experiences, while privately retaining their states, actions and rewards. Nonetheless, here the shared information is not used to impose hard constraints on the policy of each agent as in (Breschi et al., 2020), but we rather exploit them to softy steer the agents to follow the ones evaluated as the most "trustworthy" based on local performances or other factors of interest, leading to the formulation of a sharing problem. As in (Breschi et al., 2020), the method preliminarily presented in this paper, is suited for an ideal setting, in which there are negligible latencies in the back and forth communications between the systems and the cloud.

The paper is organized as follows. The cloud-based policy search problem over groups of similar agents is formalized in Section 2, while the proposed strategy to tackle it is described in Section 3. Section 4 is focused on the application of the presented policy search approach to the output-tracking problem, whose effectiveness is assessed via the results of a preliminary simulation example shown in Section 5. Conclusions and directions for future research are finally outlined in Section 6.

Notation Let $\mathbb{N}$ and $\mathbb{R}^n$ denote the set of natural numbers and the one of n-dimensional real vectors, respectively. Let I be the identity matrix. Given $A \in \mathbb{R}^{n\times m}$, we denote its transpose as $A' \in \mathbb{R}^{m\times n}$. Given a vector $x \in \mathbb{R}^n$, its Euclidean norm is denoted as $\|x\|_2$, while $\|x\|_Q^2 =$

$x'Qx$ with $Q \in \mathbb{R}^{n\times n}$. The uniform distribution over the interval $[a,\ b]$ is denoted as $\mathcal{U}_{[a,b]}$, while $\mathcal{N}(\mu, \sigma^2)$ indicates the Gaussian distribution with mean μ and variance σ^2.

2. Setting and goal

Consider N dynamical systems (also referred to as *agents*) connected to the cloud and characterized by the *same*, yet *unknown*, nominal dynamics. Under this assumption, let the interaction of the each agent with the environment it operates in be described via a Markovian signal $s_n(t) \in \mathbb{R}^{n_s}$, for $n = 1, \dots, N$, with $t \in \mathbb{N}$. Each $s_n(t)$ evolves over time according to the (unknown) model

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

where $u_n(t) \in \mathbb{R}^{n_u}$ is a vector of local decision variables, while $p_n(t) \in \mathbb{R}^{n_p}$ and $d_n(t) \in \mathbb{R}^{n_d}$ comprise exogenous signals and unmeasured disturbances acting on the n-th system, respectively. Note that, due to the similarities between the N agents, the function $h : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_s}$ is *shared* by all systems.

Suppose that the N agents have to accomplish different local tasks that are *similar* in nature, *e.g.,* they all seek to perfectly track their own step-like set points. In this paper, our aim is to exploit the analogies between the systems and their connection to the cloud in order to retrieve N *local parametric* and *deterministic* policies $\pi_n : \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_u}$, for $n = 1, \dots, N$, for each system to *optimally* attain its own control task. Since the agents have the same nominal dynamics and similar control objectives, we impose the local policies to share the same structure, *i.e.,*

$$u_n(t) = \pi_n(s_n(t), p_n(t)) = \pi(s_n(t), p_n(t), K_n), \qquad n = 1, \dots, N, \tag{2}$$

where each local inputs is shaped by an individual set of parameters K_n.

The optimality of the local policies is characterized, as in (Ferrarotti and Bemporad, 2019), through a task-dependent *stage cost* $\rho : \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_u} \to \mathbb{R}$, indicating the cost of applying the policies at a time instant $t \in \mathbb{N}$, assumed to be common to all agents due to the similarity of their tasks. Given an initial state $s_n(0)$ and a sequence $\{p_n(l), d_n(l)\}_{l=0}^{\infty}$ of exogenous signals and disturbances, the cost of each local policy over an infinite horizon is thus defined, for $n = 1, \dots, N$, as

$$\mathcal{J}_n^{\infty}(s_n(0), \{p_n(l), d_n(l)\}_{l=0}^{\infty}, K_n) = \sum_{l=0}^{\infty} \rho(s_n(l), p_n(l), \pi(s_n(l), p_n(l), K_n)), \tag{3}$$

with $s_n(l)$ evolving according to (1). Not to restrict ourselves to a single realization of the initial state, the exogenous signals and unmeasurable disturbances, we consider the average performance of each local policy, namely

$$J_n^{\infty}(K_n) = \mathop{\mathbb{E}}_{\substack{S_n(0) \\ \{P_n(l), D_n(l)\}_{l=0}^{\infty}}} \left[\mathcal{J}_n^{\infty}(s_n(0), \{p_n(l), d_n(l)\}_{l=0}^{\infty}, K_n)\right], \quad n = 1, \dots, N, \tag{4}$$

where $S_n(0), \{P_n(l), D_n(l)\}_{l=0}^{\infty}$ are random vectors, whose realizations are initial states, exogenous signals and disturbances acting on the n-th system, respectively.

Although accounting for the similarities between the agents, the cost in (4) does not exploit the connection of the systems to the cloud, that enables them to share his experiences or surrogates of the latter with the other agents. Based on the intuition that this shared information can improve the

local policies and that the analogies between the systems are likely to lead to similar values for the parameters $\{K_n\}_{n=1}^N$ in (2), we formulate our problem as

$$\{K_n^*\}_{n=1}^N = \underset{\{K_n\}_{n=1}^N}{\operatorname{argmin}} \sum_{n=1}^N J_n^\infty(K_n) + \frac{\lambda}{2}\left\| K_n - \sum_{m=1}^N w_m K_m \right\|_2^2, \tag{5}$$

where the local cost $J_n^\infty(\cdot)$ of each agent is augmented by an additional term, weighted by a tunable parameter $\lambda > 0$, that can in principle be different for each agent. The additional term introduced in (5) can be seen as a softened version of a constraint steering each set of local parameters to the weighted average over the policies of the N agents. The positive-definite weights $\{w_n\}_{n=1}^N$, $w_n \geq 0$, $\sum_{n=1}^N w_n = 1$ are customizable by the user and they should be chosen so as to be approximate indicators of the "level of trust" on the policy of each agent. If the policies are updated at each time step, time-varying weights can be chosen so to account for up-to-date information on the local policy. Analogously, λ can vary in time to account for either individual or common actual needs to exploit the shared information.

As in (Breschi et al., 2020), the proposed formulation allows us to explicitly account for the similarities between the N agents in the computation of the local policies. At the same time, with respect to the collaborative strategy proposed in (Breschi et al., 2020), problem (5) enables one to explicitly account for the performance of each agent through the weights $\{w_n\}_{n=1}^N$, so as to steer the fleet of systems towards the behavior of the better ones.

3. Policy search strategy: a sharing-based approach

Despite its generality, the abstract policy learning problem formulated in (5) cannot be solved in practice. As in (Ferrarotti and Bemporad, 2019; Breschi et al., 2020), by restricting our search over a finite horizon of tunable length $L > 0$, we consider the following approximation of (3)

$$J_n^L(K_n) = \underset{\substack{S_n(0) \\ \{P_n(l),D_n(l)\}_{l=0}^{L-1}}}{\mathbb{E}} \left[\mathcal{J}_n^L(s_n(0), \{p_n(l), d_n(l)\}_{l=0}^{L-1}, K_n) \right], \quad n = 1, \dots, N, \tag{6a}$$

where
$$\mathcal{J}_n^L(s_n(0), \{p_n(l), d_n(l)\}_{l=0}^{L-1}, K_n) = \sum_{l=0}^{L-1} \rho(s_n(l), p_n(l), \pi(s_n(l), p_n(l), K_n)). \tag{6b}$$

This allows us to compute the policies, with the parameters

$$\{K_n^\star\}_{n=1}^N = \underset{\{K_n\}_{n=1}^N}{\operatorname{argmin}} \sum_{n=1}^N J_n^L(K_n) + \frac{\lambda}{2}\left\| K_n - \sum_{m=1}^N w_m K_m \right\|_2^2, \tag{7}$$

being sub-optimal with respect to the ones retrieved according to (5). Even though this approximated learning problem can be tackled in practice, the cost in (7) is not separable over the agents. To bypass this limitation, we propose an ADMM-based scheme to learn the local policies, that allows us to fully exploit the computational power of the cloud and the resources available either on board of the single agents or individually allocated on the cloud for each system. Let us reformulate the problem in (7) as follows

$$\min_{\{K_n\}_{n=1}^N} \quad \sum_{n=1}^N \left[J_n^L(K_n) + \frac{\lambda}{2}\left\| K_n - \sum_{m=1}^N Z_m \right\|_2^2 \right] \quad \text{s.t.} \quad Z_m = w_m K_m, \quad m = 1, \dots, N, \tag{8}$$

where the auxiliary variables $\{Z_n\}_{n=1}^N$ are introduced to decouple the original problem. The associated Lagrangian (in scaled form) is given by:

$$\mathcal{L}\left(\{K_n, Z_n, \upsilon_n\}_{n=1}^N\right) = \sum_{n=1}^N \left[J_n^L(K_n) + \frac{\lambda}{2}\left\| K_n - \sum_{m=1}^N Z_m \right\|_2^2 + \frac{\beta}{2}\left\| w_n K_n - Z_n + \upsilon_n \right\|_2^2 \right],$$

where $\beta > 0$ is a tunable penalty parameter and $\{\upsilon_n\}_{n=1}^N$ are the normalized Lagrange multipliers associated with (8). Accordingly, the ADMM steps needed to solve the considered policy search problem are:

$$K_n^{i+1} \leftarrow \underset{K_n}{\operatorname{argmin}}\, \mathcal{L}\left(\{K_n, Z_n^i, \upsilon_n^i\}_{n=1}^N\right), \qquad n = 1, \dots, N, \tag{9a}$$

$$\left\{Z_n^{i+1}\right\} \leftarrow \underset{\{Z_n\}_{n=1}^N}{\operatorname{argmin}} \sum_{n=1}^N \left[\frac{\lambda}{2}\left\| K_n^{i+1} - \sum_{m=1}^N Z_m \right\|_2^2 + \frac{\beta}{2}\left\| w_n K_n^{i+1} - Z_n + \upsilon_n^i \right\|_2^2 \right], \tag{9b}$$

$$\upsilon_n^{i+1} \leftarrow \upsilon_n^i + \left(w_n K_n^{i+1} - Z_n^{i+1}\right), \qquad n = 1, \dots, N, \tag{9c}$$

where $i \in \mathbb{N}$ denotes the ADMM iteration. As it can be seen by looking at (9a), the local parameters can be computed separately. Instead, the auxiliary variables have all to be retrieved at once (see (9b)). Let us then focus on the second ADMM step. Problem (9b) can be recast as:

$$\min_{\{Z_n\}_{n=1}^N, \bar{Z}} \sum_{n=1}^N \left[\frac{\lambda}{2}\left\| K_n^{i+1} - \bar{Z} \right\|_2^2 + \frac{\beta}{2}\left\| w_n K_n^{i+1} - Z_n + \upsilon_n^i \right\|_2^2 \right] \quad \text{s.t.} \quad \bar{Z} = \sum_{m=1}^N Z_m. \tag{10}$$

As in (Boyd et al., 2010), by fixing $\bar{Z}$ it can be easily proven that:

$$Z_n = w_n K_n^{i+1} + \upsilon_n^i, \qquad n = 1, \dots, N. \tag{11a}$$

This further implies that

$$\bar{Z} = \sum_{n=1}^N Z_n = \bar{K}_w^{i+1} + N\bar{\upsilon}^i, \quad \text{where} \quad \bar{K}_w^{i+1} = \sum_{n=1}^N w_n K_n^{i+1}, \;\; \bar{\upsilon}^i = \frac{1}{N}\sum_{n=1}^N \upsilon_n^i. \tag{11b}$$

By exploiting (11b), the constrained problem in (10) is equivalent to the following unconstrained one on the variable $\bar{Z}$:

$$\min_{\bar{Z}} \sum_{n=1}^N \left[\frac{\lambda}{2}\left\| K_n^{i+1} - \bar{Z} \right\|_2^2 \right] + \frac{\beta}{2}\left\| \frac{1}{N}\left(\bar{K}_w^{i+1} - \bar{Z}\right) + \bar{\upsilon}^i \right\|_2^2. \tag{12}$$

We can thus reduce the number of auxiliary variables, directly searching for $\bar{Z}$. Once $\bar{Z}^{i+1}$ has been computed via (12), this shift implies that we have also to change the update of the Lagrange multipliers, using the equalities in (11), as $\bar{\upsilon}^{i+1} \leftarrow \bar{\upsilon}^i + \frac{1}{N}\left(\bar{K}_w^{i+1} - \bar{Z}^{i+1}\right)$. This allows us to solely consider $\bar{\upsilon}$ and it is easy to prove that the original ADMM-based scheme in (9) reduces to

$$K_n^{i+1} \leftarrow \underset{K_n}{\operatorname{argmin}}\; J_n^L(K_n) + \frac{\lambda}{2}\left\| K_n - \bar{Z}^i \right\|_2^2 + \frac{\beta}{2}\left\| w_n(K_n - K_n^i) + \frac{1}{N}\left(\bar{K}_w^i - \bar{Z}^i\right) + \bar{\upsilon}^i \right\|_2^2 \;\; \forall n, \tag{13a}$$

$$\bar{Z}^{i+1} \leftarrow \underset{\bar{Z}}{\operatorname{argmin}} \sum_{n=1}^N \left[\frac{\lambda}{2}\left\| K_n^{i+1} - \bar{Z} \right\|_2^2 \right] + \frac{\beta}{2}\left\| \frac{1}{N}\left(\bar{K}_w^{i+1} - \bar{Z}\right) + \bar{\upsilon}^i \right\|_2^2, \tag{13b}$$

$$\bar{\upsilon}^{i+1} \leftarrow \bar{\upsilon}^i + \frac{1}{N}\left(\bar{K}_w^{i+1} - \bar{Z}^{i+1}\right). \tag{13c}$$

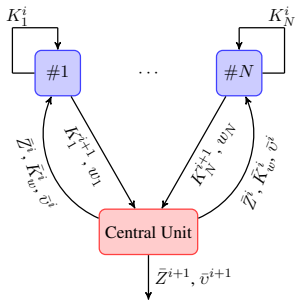

Figure 1: Transmission scheme over an ADMM iteration i.

The problem in (13b) can be explicitly solved, with the closed-form expression for $\bar{Z}$ given by:

$$\bar{Z}^{i+1} = \frac{\lambda N}{\lambda N^2 + \beta} \sum_{n=1}^{N} K_n^{i+1} + \frac{\beta}{\lambda N^2 + \beta} \left(\bar{K}_w^{i+1} + \bar{v}^i \right). \tag{14}$$

As it can be noticed, the parameters of the local policies can now be computed separately according to (13a). This operation can be either performed on board of each agent, if the computational power available locally is sufficient, or on resources allocated for each agent on the cloud. Then, the agents have to share with a central processing unit their parameters and local weights to update the auxiliary variable $\bar{Z}$ as in (13b) and the Lagrange multiplier (see (13c)), whose values are then broadcast back to the agents, along with the weighted average of the local policies. Note that this communication strategy, summarized in Figure 1, allows the systems to retain private information, such as their states, actions and rewards, while communicate to the central unit a surrogate of their experiences.

On the update of the local parameters Inspired by (Ferrarotti and Bemporad, 2019), the local policy search problem in (13a) is solved via M iterations of mini-batch *stochastic gradient descent* (SGD) (Robbins and Monro, 1951), with the dimension of the batches M being a customizable parameter. At the i-th ADMM iteration, given an initial estimate $K_n^{i,0}$, the parameters of the n-th policy are thus updated following a descent direction $\mathcal{D}_n$ with positive learning rates $\{\alpha_n^m\}_{m=1}^M$, as

$$K_n^{i,m+1} = K_n^{i,m} + \alpha_n^{m+1} \mathcal{D}_n(K_n^{i,m}), \quad m = 0, \dots, M-1,$$

Based on the characteristics of the cost to be optimized, the descent direction $\mathcal{D}_n(K_n)$ at the i-th ADMM iteration can be explicitly computed as follows:

$$\frac{\sum_{k=1}^{N_b} \nabla_{K_n} \mathcal{J}_n^L(\omega_k, K_n)}{N_b} + \lambda(K_n - \bar{Z}^{i-1}) + \beta w_n \left(w_n(K_n - K_n^{i-1}) + \frac{\left(\bar{K}_w^{i-1} - \bar{Z}^{i-1}\right)}{N} + \bar{v}^{i-1} \right), \tag{15}$$

where $\{\omega_k = (s_n^k(0), \{p_n^k(l), d_n^k(l)\}_{l=0}^{L-1})\}_{k=1}^{N_b}$ is a sampled mini-batch of admissible initial states, exogenous signals and disturbances at each SGD run. This implies the need for a tailored sampling strategy. Due to the dependence of $\mathcal{J}_n^L(\cdot)$ in (15) on the Markovian signal $s_n(t)$, the descent direction relies on the dynamics in (1) to be explicitly computed. However, the latter is assumed to be unknown. As proposed in (Ferrarotti and Bemporad, 2019), we bypass this problem by learning

simple linear models from data, approximating the behavior of the system around its current operating point. This choice allows us to keep the complexity of the recursive identification problem low, such that they are likely manageable on board of each agent, which is seldom the case if one has to retrieve a more general model for the systems. At the same time, it leads to an approximation in the computation of the descent direction.

4. A case study: the output-tracking problem

Let us focus now on the specific case in which the N agents aim at perfectly tracking the output reference signals $r_n(t) \in \mathbb{R}^{n_y}$ over time $t \in \mathbb{N}$, for $n = 1, \dots, N$. Denote as $q_n(t) \in \mathbb{R}^{n_y}$ the integral on the tracking error at time t of the n-th agent, whose evolution is dictated by the difference equation $q_n(t+1) = q_n(t) + (y_n(t) - r_n(t))$, where $y_n(t) \in \mathbb{R}^{n_y}$ is the measured output at time t. As in (Ferrarotti and Bemporad, 2019), to achieve steady-state offset free tracking we define

$$s_n(t) = \begin{bmatrix} x_n(t) \\ q_n(t) \end{bmatrix}, \qquad p_n(t) = r_n(t), \qquad t \in \mathbb{N}, \tag{16a}$$

where $x_n(t) \in \mathbb{R}^{n_x}$ denotes the state of the n-th plant. Especially in data-driven settings, this signal might not be fully measured. Nonetheless, one can find a (possibly non-minimal) realization of the plant state by collecting a finite set of input/output samples, *i.e.*,

$$x_n(t) = \begin{bmatrix} y_n(t)' & y_n(t-1)' & \cdots & y_n(t-n_a)' & u_n(t-1)' & \cdots & u_n(t-n_b)' \end{bmatrix}', \tag{16b}$$

with $n_a, n_b \in \mathbb{N}$ fixed by the user, possibly according to some priors on the order of the system. Within this setting, the stage cost in (6b) can be chosen as

$$\rho(s_n(l), r_n(l), \pi(s_n(l), r_n(l), K_n)) = \|Cx_n(l) - r_n(l)\|^2_{Q_y} + \|\Delta u_n(l)\|^2_R + \|q_n(l)\|^2_{Q_q}, \tag{17}$$

where $\Delta u_n(l) = u_n(l) - u_n(l-1)$ is the input increment, $C \in \mathbb{R}^{n_y \times n_x}$ depends on the definition of the state and is common to all agents, while Q_y, Q_q and R are *customizable* positive definite symmetric matrices weighting the tracking performance and the control effort.

As pointed out in Section 3, the computation of the descent direction (see (15)) requires one to retrieve an approximate model for the dynamics in (1). Since the dynamics of $q_n(t)$ is known, one can exploit a Kalman filter (Kalman, 1960) to recursively update the parameters Θ_n characterizing the dynamics of $x_n(t)$, such that

$$\Theta_n(t) = \Theta_n(t-1) + \xi_n(t), \qquad y_n(t+1) = \Theta_n(t) \begin{bmatrix} x_n(t) \\ u_n(t) \end{bmatrix} + d_n(t),$$

where $\xi_n(t)$ and $d_n(t)$ being zero-mean Gaussian white noises with covariance matrices Q_n^K and R_n^K, respectively. The covariances Q_n^K and R_n^K are here regarded as tuning parameters. The use of the Kalman filter allows us to approximate the evolution of the state (16a), using $\Theta_n(t)$, with parameters held constant over the horizon of length L. Note that, the simple computations needed for the update the local models can usually be performed on board of each plant. The reader is referred to (Ferrarotti and Bemporad, 2019) for further details on this scenario.

To compute the descent direction as in (15) at the i-th ADMM iteration and m-th mini-batch SGD run, one further needs a tailored sampling strategy. Within the considered framework , one can exploit the one already proposed in (Ferrarotti and Bemporad, 2019), summarized in Algorithm 1.

Algorithm 1: Sampling strategy of agent n at time t, at each SGD run

Input: States history $X_n(t)$; number of states, tracking integral, exogenous signals and disturbances samples N_x, N_q, N_p, N_d; exploration parameters r_n^{min}, r_n^{max}, σ_v^2, σ_q^2, d_n^{max}; horizon L.

1. **for** $k = 1, \dots, N_x$ **do**
   1.1. **randomly draw uniformly** $\tau_n^k \in 1, \dots, t$;
   1.2. **get** $x_n(\tau_n^k)$ **from** $X_n(t)$ **and the associated local model** $\Theta_n(\tau_n^k)$;
   1.3. **sample** $v_n^k \sim \mathcal{N}(0, \sigma_v^2)$ **and build** $x_n^k(0) = x_n(\tau_n^k) + v_n^k$;
   1.4. **for** $z = 1, \dots, N_q$ **do**
      1.4.1. **sample** $q_n^z(0) \sim \mathcal{N}(0, \sigma_q^2)$ **and build** $s_n^{k,z}(0) = \left[x_n^k(0)' \quad q_n^z(0)'\right]'$;
      1.4.2. **for** $j = 1, \dots, N_p$ **do**
         1.4.2.1. **sample** $r_n^j \sim \mathcal{U}_{[r_n^{min}, r_n^{max}]}$, **and assign** $p_n^j(l) = r_n^j \quad \forall\, l = 0, \dots, L-1$;
         1.4.2.2. **for** $h = 1, \dots, N_d$ **do**
         1.4.2.3. **for** $l = 0, \dots, L-1$ **sample** $d_n^h(l) \sim \mathcal{N}(0, \sigma_d^2)$;
         1.4.2.4. **built and collect** $\omega(k, z, j, h) = (s_n^{k,z}(0), \{p_n^j(l), d_n^h(l)\}_{l=0}^{L-1})$

Output: $\{\omega(k, z, j, h)\}_{k,z,j,h}$ sampled mini-batch of $N_b = N_x\, N_q\, N_p\, N_d$ elements.

Table 1: Numerical example: chosen parameters.

n_a	n_b	Q_n^K	R_n^K	L	N_x	N_r	N_q	N_d	σ_v	σ_q	$[r_n^{min}, r_n^{max}]$	σ_d	M	β
3	3	$10^{-3}\, I$	0.1	10	50	1	10	1	10^6	1	$[-10^3, 10^3]$	0.01	1	10^{10}

The latter relies on the use of the *states history* $X_n^t = \{x_n(0), \dots, x_n(t)\}$, comprising the realizations of the states computed with the available input/output pairs up to time t. Therefore, this sampling strategy requires the memorization of all the input/output samples collected up to time t by each agent and the corresponding approximate local models (see step 1.2). This might be challenging if the local storage space is limited. Nonetheless, when dedicated processing units are available on the cloud, it might be reasonable to exploit them and compute the descent direction on these private resources.

Remark 1 *The parameters r_n^{min}, r_n^{max}, σ_v^2, σ_q^2, σ_d^2 in Algorithm 1 dictate the degree of exploration of the state, references and disturbance spaces over sampling. While r_n^{min} and r_n^{max} can be chosen according to the ranges of set points one is interested in tracking, σ_v^2 and σ_q^2 should be selected accounting for the limited validity of the local models gathered at step 1.2.* ■

5. Numerical example

Consider a group of $N = 12$ linear *single-input single-output* systems connected to the cloud, with dynamics described by $x_n(t+1) = A\, x_n(t) + B\, u_n$, $y_n(t) = C\, x_n(t)$, with

$$A = \begin{bmatrix} -0.669 & 0.378 & 0.233 \\ -0.288 & -0.147 & -0.638 \\ -0.377 & 0.589 & 0.043 \end{bmatrix}, \; B = \begin{bmatrix} -0.295 \\ -0.325 \\ -0.026 \end{bmatrix}, \; C = \begin{bmatrix} -1.139 & 0.319 & -0.571 \end{bmatrix}. \quad (18)$$

These systems are treated as "black box" sources of inputs and measured outputs, subject to additive

Table 2: Average local results - convergence and tracking cost

	single-agent setup	consensus setup	sharing setup
average final distance from K_{opt}	3.7219	1.5014	0.4393
average local tracking cost	19606.6	9185.4	1924.6

Gaussian white noise with standard deviation $\sigma = 0.01$.

We aim at finding optimal local policies in real-time for each system to track individual piecewise constant reference signals over an horizon of $T = 500$ steps, while actively exploiting their connection to the cloud. To this end, the stage cost weights in (17) are set to $Q_y = 1$, $Q_q = 1$ and $R = 0.1$, and we consider a linear parameterization of the policies, *i.e.,*

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

As per Section 4, the optimal solution can be computed explicitly thanks to the linearity of the dynamics in (18) and it is $K_{opt} = [-0.116, 0.219, 0.653, 0.898, 0.050, 1.141, -3.337]'$. Accordingly, the optimal policy search problem also requires that the parameters of all local policies converge to K_{opt}. We remark that the optimal policy K_{opt} is not exploited for any computation, but it is just regarded as ground truth to assess the performance of the proposed approach.

We apply the proposed method to the described problem using $\lambda = 10^{10}/N$, by using the parameters reported in Table 1. In our tests, we have used time-varying weights w_n^t in (5), based on the local performance of each agent on its tracking task and on the distance of each local policy from $\bar{Z}^t$. The weights are updated at every time instant according to

$$w_n^t = \frac{1}{2(N-1)} \left(2 - \frac{CT_n^t}{\Sigma_m CT_m^t} - \frac{d_n^t}{\Sigma_m d_m^t} \right), \quad \text{with} \quad CT_n^t = \frac{c_n^t}{\Sigma_m c_m^t} / \frac{t_n^t}{\Sigma_m t_m^t},$$

where $d_n^t = \| K_n^t - \bar{Z}^t \|_2^2$, c_n^t is the sum of the last $n_w = 10$ stage costs, and t_n^t is defined as

$$t_n^t = \| r_n(t - n_w + 1) - y_n(t - n_w + 1) \|_2^2 + \sum_{j=t-n_w+2}^{t} \| r_n(j) - r_n(j-1) \|_2^2 .$$

The index c_n^t describes the quality of the local performance of the n-th agent over a time window of n_w steps, while t_n^t indicates how difficult is the local task associated to the same agent and window of time[1]. We remark that this is one possible definition of the weights, while alternative choices will be explored in future works.

We compare the local policies K_n^t obtained via the proposed sharing strategy with the ones resulting from the application of the single-agent approach presented in (Ferrarotti and Bemporad, 2019), denoted as $K_{(n)}^t$, and also with the ones synthesized by applying the multi-agent consensus method introduced in (Breschi et al., 2020), denoted as $K_{C,n}^t$. For all the methods, we initialize all the local policies as $K_n^0 = \underline{1}$ and all the initial states (16b) as $x_n(0) = \underline{0}$. We update the local policies using a faster variant of SGD, AMSGrad (Reddi et al., 2019), reinitializing it after few initial steps to avoid the spreading of negative effects due to the initial lack of information. Table 2 shows that the average convergence to the optimal policy is faster for the proposed method than the

1. We assign weight zero to agents whose local policy is unstable over the last n_w steps and to the agents that are "outliers" with respect to the majority, i.e., agents such that $d_n^t \geq \frac{5}{N} \sum_m d_m^t$ or $CT_n^t \geq \frac{5}{N} \sum_m CT_m^t$. Such agents are also disregarded in the normalization of c_n^t, t_n^t, d_n^t.

others, benefiting from the shared information provided by $\bar{Z}^t$. The improvement of $\bar{Z}^t$, compared with the global consensus policy K_G^t retrieved forcing consensus can be seen in Figure 2 (left plot). The faster convergence to K_{opt} further improves the average cost of performing the local tracking tasks (see Table 2).

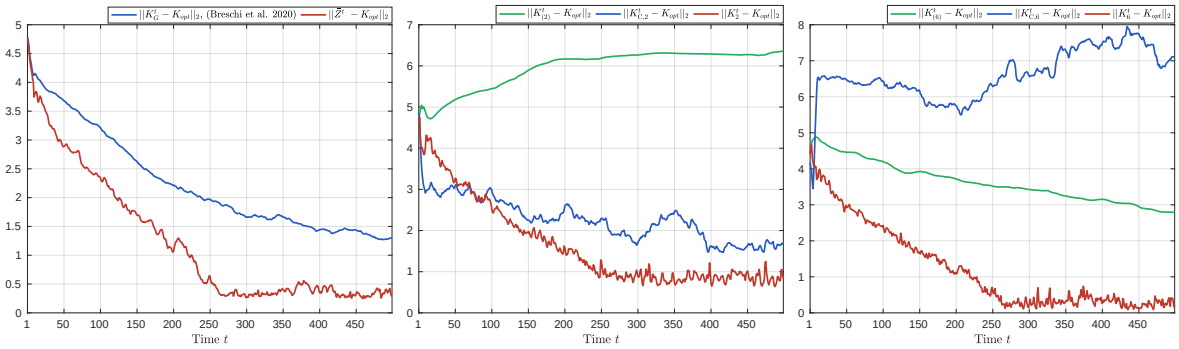

Figure 2: Convergence to the optimal policy K_{opt} of $\bar{Z}^t$ and K_G^t (left) and of the local policies $K_{(n)}^t$ (green), $K_{C,n}^t$ (blue) and K_n^t (red) of agent $n = 2$ (center) and $n = 6$ (right).

By looking at Figure 2, we also observe that the sharing-based approach is more resilient, as compared to the other frameworks, against the loss of convergence that might result from the randomness of sampling, the non-convexity of the cost function with respect to the policy parameters, or from the error in approximating $\mathcal{J}_n^\infty$ in (6b). Indeed, the center and right plots of Figure 2 show the evolution of the local parameters of agents 2 and 6 respectively, as an example of such phenomenon. In the center plot, agent 2 is subject to mistakes in the choice of the update direction when the parameters are optimized in the single-agent framework, while in the right plot the same thing happens to the evolution of the parameters in the consensus framework. In this last setup, the contribution of such agents might slow down the convergence of the global policy K_G^t to K_{opt}, and, hence, deteriorate the performance of the other agents too. This is avoided when using the proposed sharing-based approach, where low quality control laws are not propagated by assigning them low weights.

6. Conclusions

In this paper, we have extended the policy search approach proposed in (Ferrarotti and Bemporad, 2019) so as to exploit the similarities between a group of systems sharing the same nominal dynamics, that are all connected to the cloud. To this end, we augment the performance-oriented cost with a sharing-based term, that softly enforces the local policies to be as close as possible to the better-performing agents. The benefits of the chosen strategy to exploit the similarities between systems and their connection to the cloud is shown via a preliminary simulation example.

Future works will be devoted to extend the approach for it to handle more challenging scenarios, in which linear policies might not suffice to attain the local goals and the communications between the agents and the central units might be asynchronous.

Acknowledgments

This paper was partially supported by the Italian Ministry of University and Research under the PRIN'17 project "Data-driven learning of constrained control systems" , contract no. 2017J89ARP.

References

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Machine Learning*, 3(1):1–122, 2010.

V. Breschi, L. Ferrarotti, and A. Bemporad. Cloud-based collaborative learning of optimal feedback controllers. *2020 21st IFAC World Congress*, 2020.

L. Campestrini, D. Eckhard, M. Gevers, and A.S. Bazanella. Virtual reference feedback tuning for non-minimum phase plants. *Automatica*, 47(8):1778 – 1784, 2011.

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

J. Coulson, J. Lygeros, and F. Dörfler. Data-enabled predictive control: In the shallows of the deepc, 2019.

J. Coulson, J. Lygeros, and F. Dörfler. Distributionally robust chance constrained data-enabled predictive control, 2020.

M. Dimakopoulou, I. Osband, and B. Van Roy. Scalable coordinated exploration in concurrent reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 42234232, 2018.

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

I Grondman. Online model learning algorithms for actor-critic control. 2015.

H. Hjalmarsson. Iterative feedback tuningan overview. *International Journal of Adaptive Control and Signal Processing*, 16(5):373–395, 2002.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

A. Karimi, L. Miskovic, and D. Bonvin. Iterative correlation-based controller tuning. *International Journal of Adaptive Control and Signal Processing*, 18(8):645–664, 2004.

P. Kergus, M. Olivi, C. Poussot-Vassal, and F. Demourant. Data-driven reference model selection and application to l-ddc design, 2019.

Arbaaz Khan, Clark Zhang, Daniel D. Lee, Vijay Kumar, and Alejandro Ribeiro. Scalable centralized deep multi-agent reinforcement learning via policy gradients. *CoRR*, abs/1805.08776, 2018. URL http://arxiv.org/abs/1805.08776.

V. R. Konda and J.N. Tsitsiklis. On actor-critic algorithms. *SIAM J. Control Optim.*, 42(4): 11431166, 2003.

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

B. Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.

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

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

D. Selvi, D. Piga, and A. Bemporad. Towards direct data-driven model-free design of optimal controllers. In *2018 European Control Conference (ECC)*, pages 2836–2841, 2018.

C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.