Constrained Online Learning in Networks with Sublinear Regret and Fit

Santiago Paternain, Soomin Lee, Michael M. Zavlanos and Alejandro Ribeiro

***Abstract*— In this paper, we consider a network of agents that select actions in order to minimize an objective function of which they only have local observations and subject to a set of constraints. The objective function and the constraints can vary arbitrarily over time. The selection of actions, also called a strategy, is causal and decentralized, i.e., the dynamical system that determines the actions of a given agent depends only on the constraints at the current time and on its own actions and those of its neighbors. We propose a decentralized saddle point algorithm to select the actions and we show that the network fit and regret are sublinear with respect to the time horizon of the problem. Specifically, we define the global fit of a strategy as a vector that integrates over time the global constraint violations as seen by a given node. The fit is a performance loss associated with online operation as opposed to offline clairvoyant operation which can always select an action, if one exists, that satisfies the constraints at all times. If this fit grows sublinearly with the time horizon it suggests that the strategy approaches the feasible set of actions. Likewise, we define the regret of a strategy as the difference between its accumulated cost and that of the best fixed action that one could select knowing beforehand the time evolution of the objective function. Numerical examples support the theoretical conclusions.**

I. Introduction

Distributed optimization has been used to address a wide range of problems, from resource allocation in communication [1], to multi-robot teams [2]–[4] and the internet of things [5], [6]. In these problems, the agents try to optimize a common objective function of which they only have a partial observation that depends typically on the data they collect individually while satisfying constraints. Several methods, exist to solve these optimization problems, notably the saddle point algorithm by Arrow and Hurwicz [7], which has the advantage of admitting a distributed implementation [8]–[11]. These algorithms consider the particular case where the objective that the agents aim to minimize and the constraints are static over time. Methods to address problems where the objectives and constraints change over time according to a stationary probability have also been established, see eg. [12]. In this case, the goal is to minimize the expectation of the objective while satisfying the constraints in average. When the problem is unconstrained, centralized and decentralized implementations of stochastic gradient descent converge to such solutions.

Work supported by ARL DCIST CRA W911NF-17-2-0181 and AFOSR under award FA9550-19-1-0169.

Santiago Paternain and Alejandro Ribeiro are with the Dept. of Electrical and System Engineering, Univ. of Pennsylvania. Email: {spater, aribeiro} @seas.upenn.edu.

Soomin Lee is with Yahoo! Research. Email: soominl@yahoo-inc.com

Michael M.Zavlanos is with the Dept. of Mechanical Engeeniring and Material Science, Duke University. Email:michael.zavlanos@duke.edu

In this paper, we consider problems in which the costs and constraints can vary arbitrarily over time. This is the problem considered in online optimization, where the optimality is formulated in terms of regret [13], [14] whereby agents select online actions that result in a cost chosen by nature. The cost functions are revealed to the agents after the actions are selected and these values are used to adapt the future strategy. Regret for a network is defined as the difference between the total cost across the network incurred by each agent as compared to the cost of the optimal centralized clairvoyant solution. Likewise, the fit of a strategy [15], [16] is a vector that integrates over time the constraint violation incurred by each agent. In that sense the fit is a performance loss associated with online operation as opposed to offline clairvoyant operation which can always select an action that satisfies the constraints at all times, if such action exists. The network fit is a vector that contains the time integrals of the constraints of all the agents evaluated across the trajectory of a specific agent. Having small global fit means that the agent is able to satisfy the constraints of every other agent, and thus if it is placed in a different position in the network its performance is maintained.

The problem of distributed online constrained convex optimization has been studied in [17]–[19]. Continuous time approaches have advantages in the context of distributed control systems whenever signals are inherently continuous. Moreover, discrete-time approaches can be thought of as the result of discretizing continuous-time dynamics. The work in [17], [18] considers unconstrained settings where agents minimize a time-varying convex loss, whereas here we consider constrained problems. Constrained problems have been considered in [19] using a Saddle Point algorithm establishing sublinear bounds on the network disagreement and on the regret achieved by the strategy. The main difference with [19] is that instead of imposing exact consensus among agents, here we allow for small disagreement. This idea has been used for unconstrained problems in [20]. With this modification, we can establish that the trajectories that arise from the distributed online saddle point dynamics are such that the disagreement of the agents, the regret and the fit are bounded by sublinear functions of the time horizon (Section IV). The latter is the main contribution of the paper since in [19] there are no guarantees related to the constraint violation. This suggests that our proposed algorithm achieves consensus, feasibility and optimality as the time goes to infinity. This result generalizes the result in [15] where a centralized saddle point algorithm achieves fit and regret that are bounded by sublinear functions of the time horizon.

We also illustrate our algorithm on a problem involving

a team of robots driving through an urban environment to perform real-time texture classification for the purpose of mapping and object recognition. We show that the team of robots succeds in training a common classifier that allows them to distinguish between grass and pavement images even when some of the agents have only observed one of the classes.

II. Constrained Online Learning in Networks

We consider a group of N agents linked by an undirected connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V} = \{1, \ldots, N\}$ is a set of nodes and $\mathcal{E}$ is a set of edges so that $(i,j) \in \mathcal{E}$ means that i and j are connected to each other. The set $\mathcal{N}_i := \{j : (i,j) \in \mathcal{E}\}$ contains all nodes that are connected to i and is called the neighborhood of i. Note that since the graph is undirected, node j is in the neighborhood of i if and only if node i is in the neighborhood of j.

We are interested in situations where the agents in $\mathcal{G}$ have access to arbitrarily time varying local constraints and local objective functions and continuously select actions that are good not only for their local constraints and costs but for the local constraints and costs of other agents. To explain this formally, let $t \in \mathbb{R}^+$ be a continuous time index, $f_i(t,\cdot) : \mathbb{R}^n \to \mathbb{R}^{m_i}$ be a set of m_i convex constraints at agent i and $f_{0i}(t,\cdot) : \mathbb{R}^n \to \mathbb{R}$ be a local convex cost incurred at node i. A *local* goal of node i is to select an action $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n$ that satisfies local constraints $f_i(t,\mathbf{x}_i) \preceq 0$ across a time interval $[0,T]$ while minimizing the cost $f_{0i}(t,\mathbf{x}_i)$ integrated over the same interval. The latter corresponds to situations in which each of the agents is acting independently since the optimal action of i depends on its local cost and constraints, and not on those of other agents. Instead, here we are interested in situations where the actions of agents are coordinated, so that an action $\mathbf{x}_i$ of agent i can affect the costs and constraints of other agents. This results in a *global* formulation in which the optimal action of each agent $\mathbf{x}_i^*$ is defined as the one that satisfies the constraints of all agents and minimizes the integral of the sum cost,

$$\begin{aligned} \mathbf{x}_i^* \;:=\; \underset{\mathbf{x}_i \in \mathcal{X}}{\operatorname{argmin}} \; & \int_0^T = \sum_{j=1}^N f_{0j}(t,\mathbf{x}_i)\, dt, \\ \text{s.t.} \qquad & f_j(t,\mathbf{x}_i) \preceq 0, \quad \forall\ j \text{ and } t \in [0,T]. \end{aligned} \tag{1}$$

We say that the problem in (1) is global because the action $\mathbf{x}_i$ is evaluated at the constraint and costs of all nodes. This readily implies that $\mathbf{x}_i^* = \mathbf{x}_j^*$ and that there is a single global action that is optimal for all nodes. These problems arise when local functions are related to a common variable. E.g., the costs and constraints can represent local observations of a parameter to be estimated [21] or local observations and costs of a plant to be controlled [22]. Problems having the form of (1) also arise in large scale optimization where costs and constraints are *not* acquired locally but are distributed over several servers to reduce computation and storage [23].

If the functions $f_{0i}(t,\cdot)$ and $f_i(t,\cdot)$ are available for all times $t \in [0,T]$, solving (1) reduces to solving a distributed convex optimization problem for which a number of standard algorithms are applicable; see e.g., [24], [25]. In this paper we consider problems in which the constraints $f_i(t,\cdot)$ and costs $f_{0i}(t,\cdot)$ are arbitrary and observed causally and locally by node i. In this setting it makes sense to consider time varying strategies $\mathbf{x}_i(t)$ that adapt the action of agent i to the information that is revealed at time t. In this context the optimal argument in (1) is a centralized clairvoyant action that would be chosen when agents have knowledge of the future evolution of the system at time $t = 0$. The appropriate figures of merit in this case are the notions of regret [13], [26], [27] and fit [15] that we generalize to network settings in the following section. Before proceeding with definitions of network regret and fit, note that for the definition in (1) to be valid the functions $f_{0i}(t,\mathbf{x})$ have to be integrable with respect to the time variable t. In subsequent definitions and analysis we further require the constraints f_i for all $i \in \mathcal{V}$, to be integrable, convex and Lipschitz continuous with respect to $\mathbf{x}$ for all times t. We formally state these assumptions next.

Assumption 1. Let $\mathcal{X}$ be a compact convex set and the functions $f_{0i}(t,\mathbf{x})$ and $f_i(t,\mathbf{x})$ be integrable with respect to t and convex with respect to $\mathbf{x} \in \mathcal{X}$ for all $t \in [0,T]$. We further assume that the cost and constraints are Lipschitz continuous over $\mathcal{X}$ with respective constants $L_0 > 0$ and $L_f > 0$. I.e., for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and all $t \in [0,T]$ the cost functions satisfy

$$\big|f_{0i}(t,\mathbf{x}) - f_{0i}(t,\mathbf{y})\big| \leq L_0 \big\|\mathbf{x} - \mathbf{y}\big\|, \tag{2}$$

and the constraint functions satisfy

$$\big|f_{kj}(t,\mathbf{x}) - f_{kj}(t,\mathbf{y})\big| \leq L_f \big\|\mathbf{x} - \mathbf{y}\big\|, \tag{3}$$

where $f_{kj}(t,\cdot)$ denotes the jth component of the vector valued constraint function $f_k(t,\cdot)$.

We remark that integrability with respect to t is a weak condition. We do not require differentiability, not even continuity. This entails a fundamental difference with time varying optimization problems that strive to track a time varying optimal argument under the assumption of smooth time varying costs and constraints [21], [28], [29]. The goal here is to design an algorithm that can adapt to unexpected changes in the system, including, indeed, most importantly, to those that arise because of discontinuities in the cost and constraint functions. Another requirement for $\mathbf{x}_i^*$ to be well defined is existence of an action $\mathbf{x}^\dagger \in \mathcal{X}$ that satisfies the constraints at all times and all nodes as we formally state next.

Assumption 2. There exists an action $\mathbf{x}^\dagger \in \mathcal{X}$ that satisfies the constraints of all agents for all times $t \in [0,T]$,

$$f_i(t,\mathbf{x}^\dagger) \preceq 0, \qquad \forall\ i \text{ and } t \in [0,T]. \tag{4}$$

We say that $\mathcal{X}^\dagger := \big\{\mathbf{x}^\dagger \in \mathcal{X} : f_i(t,\mathbf{x}^\dagger) \preceq 0, \forall i \text{ and } t \in [0,T].\big\}$ is the set of feasible actions.

We require as well that minimum of the objective function does not become progressively smaller with time so that a uniform bound K holds for all times $t \in [0,T]$.

Assumption 3. There exists $K > 0$ independent of the time horizon T such that for all $t \in [0, T]$ it holds that

$$f_0(t, \mathbf{x}^*) - \min_{x \in \mathcal{X}^N} f_0(t, \mathbf{x}) \leq K, \tag{5}$$

where $\mathbf{x}^*$ is the solution to the problem (1).

The existence of the bound in (5) is a mild requierement. Since the function $f_0(t, \mathbf{x})$ is convex, for any time t it is lower bounded for compact set of actions $\mathcal{X}$. The only restriction imposed is that $\min_{\mathbf{x} \in \mathcal{X}^N} f_0(t, \mathbf{x})$ does not become progressively smaller with time so that a uniform bound K holds for all times $t \in [0, T]$.

A. Network Regret and Network Fit

To evaluate the cost performance of such trajectories we define the notions of network regret and network fit. Begin then by considering a trajectory $\mathbf{x}_i(t)$ chosen by agent i and the total accumulated cost $\int_0^T f_{0j}(t, \mathbf{x}_i(t))dt$ that this trajectory incurs for a possibly different agent j. We define the regret $\mathcal{R}^i_{Tj}$ as the difference of this accumulated cost relative to the corresponding cost that would be incurred by the optimal trajectory $\mathbf{x}^*_i$ of (1),

$$\mathcal{R}^i_{Tj} := \int_0^T f_{0j}(t, \mathbf{x}_i(t))\, dt - \int_0^T f_{0j}(t, \mathbf{x}^*_i)\, dt. \tag{6}$$

Likewise, we consider the accumulation $\int_0^T f_j(t, \mathbf{x}_i(t))dt$ of constraint of agent j incurred by the trajectory of agent i. The fit $\mathcal{F}^i_{Tj}$ is defined as the comparison of this constraint accumulation relative to the corresponding constraint accumulation of the optimal trajectory $\mathbf{x}^*_i$

$$\mathcal{F}^i_{Tj} := \int_0^T f_j(t, \mathbf{x}_i(t))\, dt - \int_0^T f_j(t, \mathbf{x}^*_i)\, dt. \tag{7}$$

The action $\mathbf{x}^*_i$ can be considered as an offline reference that would be chosen by an entity that is clairvoyant, because it observes the future, and omniscient, because it observes the costs and constraints of all nodes. Our objective is to consider trajectories that are chosen online by agents that are causal, because they observe the past, and local, because they observe their local costs and exchange information with neighboring nodes only. In this context regret and fit can be interpreted as performance losses associated with online causal and local operation as opposed to offline clairvoyant and omniscient operation. If $\mathcal{F}^i_{Tj}$ is positive we are in a situation in which, had the constraints of all agents be known beforehand, we could have selected an action $\mathbf{x}^\dagger$ to satisfy all constraints.

The fit measures how far the trajectory $\mathbf{x}(t)$ is from achieving that goal. Analogously, if the regret $\mathcal{R}^i_{Tj}$ is large we are in a situation in which prior knowledge of the objective functions and constraints would had resulted in the selection of a strategy $\mathbf{x}^*$ that achieves much better performance than the one achieved by $\mathbf{x}_i(t)$. In that sense $\mathcal{R}^i_{Tj}$ indicates how much we regret not having had that information a priori.

A good learning strategy is one that achieves small regret and fit as that would be an indication that the trajectory $\mathbf{x}(t)$ approaches $\mathbf{x}^*$. Notice however that since the objective function and the constraints are integrated over a time horizon T, it is natural to expect the cost and constraints to grow linearly with T. Thus, having regret and fit that grow at a sublinear rate is sufficient indication of a good learning strategy. This intuition motivates the following definitions of feasible and optimal trajectories.

Definition 1. *We define an environment as a set of constraints $f_j : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^{m_j}$ and costs $f_{0j} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ for all $j \in \mathcal{V}$. For a trajectory $\mathbf{x}_i(t)$ we consider the regret and fit definitions in* (6) *and* (7) *and further define the sum regret $\mathcal{R}^i_T := \sum_{j \in \mathcal{V}} \mathcal{R}^i_{Tj}$ and the network wide fit $\mathcal{F}^j_T = [\mathcal{F}^{j\top}_{T1}, \dots, \mathcal{F}^{j\top}_{TN}]^\top$. We say that:*

Feasibility. *The trajectories are feasible in the environment if all the local fits $\mathcal{F}^i_T$ with $i \in \mathcal{V}$ grow sublinearly with T. I.e., there exist a function $h(T)$ with $\limsup_{T\to\infty} h(T)/T = 0$ and a constant vector C_f such that for all times T it holds,*

$$\mathcal{F}^i_T := \int_0^T f(t, \mathbf{x}_i(t))\, dt \leq C_f h(T). \tag{8}$$

Optimality. *The trajectories are optimal in the environment if all regrets $\mathcal{R}^i_T$ grow sublinearlly for all $i \in \mathcal{V}$ and T. I.e. there exist a function $h(T)$ with $\limsup_{T\to\infty} h(T)/T = 0$ and a constant C such that for all times T it holds,*

$$\mathcal{R}^i_T := \int_0^T f_0(t, \mathbf{x}_i(t))\, dt - \int_0^T f_0(t, \mathbf{x}^*)\, dt \leq C h(T). \tag{9}$$

In the next section we develop the details of a distributed and online version of the Arrow-Hurwicz algorithm, such that its generated trajectories are feasible and optimal in the sense of Definition 1. The latter is formally stated and proved in Section IV along with an intermediate result that claims that the disagreement across agents is sublinear with respect to the time horizon, hence suggesting consensus.

III. Distributed Online Saddle point

Problem (1) can be solved in a distributed manner with a variety of methods, one of which is the saddle point algorithm of Arrow and Hurwicz [7], which in fact admits a distributed implementation. Since each agent $i \in \mathcal{V}$ has access only to the local cost and constraints, a more natural representation of the problem (1) is one where each agent selects a local decision vector $\mathbf{x}_i \in \mathbb{R}^n$. Nodes then try to achieve the minimum of their local objective functions $f_{0i}(t, \mathbf{x}_i)$ while satisfying the local constraints $f_i(t, \mathbf{x}_i) \preceq 0$ and keeping their variables equal to the variables $\mathbf{x}_j$ of neighboring nodes $j \in \mathcal{N}_i$. By defining $\mathbf{x} = [\mathbf{x}_1^\top, \dots, \mathbf{x}_N^\top]^\top$, this formulation can be written as

$$\begin{aligned} \mathbf{x}^* := \operatorname*{argmin}_{\mathbf{x} \in \mathcal{X}^N} \int_0^T f_0(t, \mathbf{x})\, dt = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{X}^N} \int_0^T \sum_{i=1}^N f_{0i}(t, \mathbf{x}_i)\, dt \\ \text{s.t.} \quad f_i(t, \mathbf{x}_i) \preceq 0, \forall t \in [0, T], \forall i \in \mathcal{V}, \\ \mathbf{x}_i = \mathbf{x}_j, \forall i \in \mathcal{V}, j \in \mathcal{N}_i. \end{aligned} \tag{10}$$

With the assumption that the network is connected, the constraints $\mathbf{x}_i = \mathbf{x}_j$ for all i and $j \in \mathcal{N}_i$ imply that (1) and (10) are equivalent. We formalize this assumption next.

Assumption 4. The network is connected with diameter D, i.e., the shortest distance between the two most distant nodes in the network is D.

In this work we aim to extend the saddle point algorithm to control the growth of regret and fit. In doing so it is convenient to relax the consensus constraints $\mathbf{x}_i = \mathbf{x}_j$ in (10) to allow for some controlled disagreement. We accomplish this by defining the set of constraints

$$g_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2 - \gamma \leq 0, \tag{11}$$

where γ is a positive constant limiting how much constraint violation is allowed. Notice that the parameter γ could be set to be arbitrarily small. The advantage of using a controlled disagreement is that it allows for agents to achieve a good global performance without damaging excessively the local performance, which in some applications might be important as well. By allowing larger values of γ, we allow more disagreement and therefore we prioritize the local performance, whereas by making γ closer to zero the goal is set in the centralized performance. The same relaxation is considered in [20] in the case of unconstrained distributed optimization. With this modification, we can construct the following time varying Lagrangian

$$\mathcal{L}(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f_0(t, \mathbf{x}) + \sum_{i=1}^{N} \left(\boldsymbol{\lambda}_i^\top f_i(t, \mathbf{x}_i) + \boldsymbol{\mu}_i^\top g_i(\mathbf{x}) \right), \tag{12}$$

where $\boldsymbol{\lambda}_i \in \mathbb{R}_+^{m_i}$ for $i = 1 \dots N$ and $\boldsymbol{\mu}_i \in \mathbb{R}_+^{|\mathcal{N}_i|}$ for $i = 1 \dots N$ are the Lagrange multipliers and where $g_i(\mathbf{x}) \in \mathbb{R}^{|\mathcal{N}_i|}$ is the vector with components $g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ for all $j \in \mathcal{N}_i$. Saddle point methods rely on the fact that for a constrained convex optimization problem, a pair is a primal-dual solution if and only if it is a saddle point of the Lagrangian associated with the problem, see e.g. [30]. This is the case in problem (10) since $f_0(\cdot, \mathbf{x})$, $f(\cdot, \mathbf{x})$ and $g(\cdot, \mathbf{x})$ are convex (c.f. Assumption 1). In addition, because $\boldsymbol{\lambda}, \boldsymbol{\mu} \succeq 0$, the Lagrangian is convex with respect to $\mathbf{x}$ and therefore the subgradient with respect to $\mathbf{x}$ exists for all time $t \geq 0$, let us denote it by $\mathcal{L}_x(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$. The Lagrangian is linear with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ and therefore its partial derivatives with respect to these variables exist. Let us denote them by $\mathcal{L}_\lambda(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ and $\mathcal{L}_\mu(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ respectively. The actions $\mathbf{x}$ are updated – as in the classic Arrow-Hurwicz algorithm – by following the negative subgradient of the Lagrangian with respect to $\mathbf{x}$

$$\begin{aligned} \dot{\mathbf{x}} &= -\mathcal{L}_x(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &= -f_{0,x}(t, \mathbf{x}) - \sum_{i=1}^{N} f_{i,x}(t, \mathbf{x}_i)^\top \boldsymbol{\lambda}_i - \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \mu_{ij} g_{ij,x}(\mathbf{x}), \end{aligned} \tag{13}$$

where $\mathcal{N}_i$ is the set of neighbors of node i. The primal update interprets the constraints as potentials with corresponding weights $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ and descends along a linear combination of the gradients of said potentials. The multipliers are then updated by following the subgradient of the Lagrangian with respect to them

$$\dot{\boldsymbol{\lambda}} = \mathcal{L}_\lambda(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}), \tag{14a}$$

$$\dot{\boldsymbol{\mu}} = \mathcal{L}_\mu(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = g(\mathbf{x}). \tag{14b}$$

The intuition behind the latter update is that if a constraint is violated, for instance $f_{1,1}(\mathbf{x}) > 0$ the corresponding multiplier, $\boldsymbol{\lambda}_{1,1}$ will be increased, thus augmenting the relative weight of this potential in the linear combination in (13). Which in turn pushes the action towards satisfying said constraint. On the other hand, if the constraint is satisfied, the weight of that potential will be reduced, thus making the direction of the gradient of the function less important in the weighted linear combination. Observe that the multipliers need to remain positive at all time to ensure the convexity of the Lagrangian with respect to $\mathbf{x}$, yet if a multiplier takes the value zero and its corresponding constraint is satisfied, the previous update turns the multiplier negative. To avoid this issue, we will require a projection over the positive orthant. We formalize this idea next, after making the observation that the update (13)–(14) is indeed distributed. To see this, write the Lagrangian as a sum of the following local Lagrangians

$$\begin{aligned} \mathcal{L}^i(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f_0^i(t, \mathbf{x}_i) + \boldsymbol{\lambda}_i^\top f_i(t, \mathbf{x}_i) \\ + \sum_{j \in \mathcal{N}_i} \mu_{ij} \left(\|\mathbf{x}_i - \mathbf{x}_j\|^2 - \gamma \right), \end{aligned} \tag{15}$$

where to compute each local Lagrangian, agent i needs only information regarding its variables and those of its neighbors. Then, each agent can compute locally the gradient of the Lagrangian with respect to its local variable $\mathbf{x}_i$ and perform the update described in (13)–(14), with the caveat that to ensure that the multipliers are always positive we need to consider a projected dynamical system. We formalize this idea next.

Definition 2 (Projection of a vector at a point). *Let $K \subset \mathbb{R}^n$ be a compact convex set. Then, for any $\mathbf{y} \in K$ and $\mathbf{v} \in \mathbb{R}^n$, we defined the projection of $\mathbf{v}$ over the set K at the point $\mathbf{y}$ as*

$$\Pi_K[\mathbf{y}, \mathbf{v}] = \lim_{\xi \to 0^+} \frac{P_K(\mathbf{y} + \xi \mathbf{v}) - \mathbf{y}}{\xi}, \tag{16}$$

where the standard projection $P_K(\mathbf{z}) = \operatorname{argmin}_{\mathbf{y} \in K} \|\mathbf{y} - \mathbf{z}\|^2$ is always well defined because K is convex.

The intuition behind the projection is that, if the point $\mathbf{y}$ is in the interior of the set K then the projection of the vector $\mathbf{v}$ is the vector itself. In cases where $\mathbf{y}$ is in the boundary of the set K, the projection of $\mathbf{v}$ is its component tangental to the boundary of K. With this definition at hand, and by defining the gain of the controller to be $\varepsilon > 0$ we define the distributed online saddle point controller as follows. Each agent updates its action by following the negative subgradient

of the Lagrangian with respect to its local copy of the action $\mathbf{x}_i$

$$\dot{\mathbf{x}}_i = \Pi_{\mathcal{X}} \left[\mathbf{x}_i, -\varepsilon \mathcal{L}_{x_i}(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right], \tag{17}$$

Likewise, the multipliers $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_i$ are updated by ascending along the direction of the gradient of the Lagrangian with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ respectively, i.e.,

$$\dot{\boldsymbol{\lambda}}_i = \Pi_{+} \left[\boldsymbol{\lambda}_i, \varepsilon \mathcal{L}_{\lambda_i}(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right], \tag{18a}$$

$$\dot{\mu}_{ij} = \Pi_{+} \left[\mu_{ij}, \varepsilon \mathcal{L}_{\mu_{ij}}(t, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \right]. \tag{18b}$$

The three gradients can be computed in a distributed fashion since they only depend on each agent's own variables and those of their neighbors. In [15] it was shown that in the centralized case, a saddle point algorithm akin to the one described by (17)–(18) achieves feasible and strongly optimal trajectories, i.e., fit bounded by a sublinear function of the time horizon and regret bounded by function that is constant with respect to the time horizon. In this work we show that the distributed version of said algorithm (c.f. (17) and (18)) achieves feasible and optimal trajectories in the sense of Definition 1. Moreover, the network disagreement is bounded by a function that is sublinear with respect to the time horizon. These results are the subject of the next section.

IV. Feasible and Optimal trajectories

Let us consider an energy-like function which will be used in subsequent analysis. Let $\tilde{\mathbf{x}} \in \mathcal{X}^N$, $\tilde{\boldsymbol{\lambda}} \in \mathbb{R}_{+}^{\sum_i m_i}$, $\tilde{\boldsymbol{\mu}} \in \mathbb{R}_{+}^{\sum_i |\mathcal{N}_i|}$, where we denote by $|\mathcal{N}_i|$ the cardinality of the set of neighbors of node i, and define the function

$$V_{\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2} \left(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2 + \|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|^2 + \|\boldsymbol{\mu} - \tilde{\boldsymbol{\mu}}\|^2 \right). \tag{19}$$

By considering the time derivative of the previous function along the dynamics (17)–(18) we establish that the integral of the difference of the Lagrangian evaluated at $(\mathbf{x}(t), \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$ and the Lagrangian evaluated at $(\tilde{\mathbf{x}}, \boldsymbol{\lambda}(t), \boldsymbol{\mu}(t))$ is bounded by a constant independent of the time horizon T. This idea is the key to establish that the saddle point dynamics yield feasible and optimal trajectories. We first claim that the saddle point dynamics (17)–(18) yields sublinear network disagreement for all $T > 0$. We formalize this result in Proposition 1.

Proposition 1 (Sublinear Network Disagreement). *Let Assumptions 1–4 hold. Then for any $T \geq 0$ the solutions of the dynamical system* (17)–(18) *are such that the network disagreement is sublinear with respect to T. In particular for $\boldsymbol{\lambda}(0) = 0$ and $\boldsymbol{\mu}(0) = 0$, for any $i, j \in \mathcal{V}$ we have that*

$$\begin{aligned} &\int_0^T \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| \, dt \\ &\leq D \sqrt{(K + \gamma)T + \frac{1}{2\varepsilon} \left(1 + \|\mathbf{x}^* - \mathbf{x}(0)\|^2 \right)}, \end{aligned} \tag{20}$$

where D is the network diameter defined in Assumption 4.

Proof. See [31]. ∎

The sublinear network disagreement that the previous proposition establishes suggests that the solution of the distributed implementation of the algorithm is not different than the centralized. Since for the latter results of sublinear fit and regret have been established [15] it is unsurprising that the same holds here. The latter means that the trajectories that arise from the Distributed Online Saddle Point Dynamics (17)–(18) are feasible and optimal in the sense of definition 1. We formalize these results in the following theorem.

Theorem 1 (Feasibility). *Let Assumptions 1–4 hold. Then for any $T \geq 0$ the solutions of the dynamical system* (17)–(18)*, with $\varepsilon > 1/2$, are such that the k-th component of the local fit $\mathcal{F}_{Tj}^i$ for any $i, j \in \mathcal{V}$ is bounded by $\left(\mathcal{F}_{Tj}^i\right)_k \leq O(\sqrt{T})$. Likewise, the local regret $\mathcal{R}_T^i$ for any $i \in \mathcal{V}$ is bounded by a function of $O(\sqrt{T})$.*

Proof. See [31]. ∎

The previous theorem establishes that the local fit achieved by a system that follows saddle point dynamics (17)–(18) is bounded by a function whose rate of growth is sublinear, thus suggesting vanishing penalties. The fact that the fit grows sublinearly is equivalent to achieving trajectories that are feasible in the sense of Definition 1. Likewise, we have established that the regret is sublinear and thus, the trajectories have in average the same cost that solution of the centralized clairvoyant solution (10). Hence, they are optimal in the sense of Definition 1.In the next section we present numerical examples that support the theoretical conclusions.

V. Numerical Examples

In this section we consider a team of N robots tasked with classifying in real time and in a distributed manner the different objects and terrains that compose the environment in which they are deployed. This problem, has been studied in [2], although the method presented here is different. Each robot has only access to information about the environment based on the path it has traversed and the images gathered. Therefore, its local information may not be enough to achieve the task of classification since the information gathered may omit regions of the feature space that are crucial. See for instance Figure 1 where we depict random trajectories of twenty agents driving around an intersection. When the agent is on the pavement i.e., the absolute value of its horizontal or vertical coordinate is less than five, then it observes pavement images. On the other hand, outside this region it observes grass. As it can be observed in that figure only some of the agents visit both regions and the interest is that the whole team can learn a common classifier. The advantage of learning such classifier is that a robot can identify if it is on grass even if it has not seen grass in the training process. In particular we consider a problem in which each robot receives features $\mathbf{z}_i(t) \in \mathbb{R}^n$ from the scene and corresponding labels $y_i(t) \in \{-1, 1\}$ depending on whether the terrain is grass or pavement. The details of the feature extraction from image data is provided in Section V-A. The common objective of the agents can be formulated as training a common linear classifier $\mathbf{x} \in \mathbb{R}^n$ that minimizes a loss function. The value of the loss function is small when the

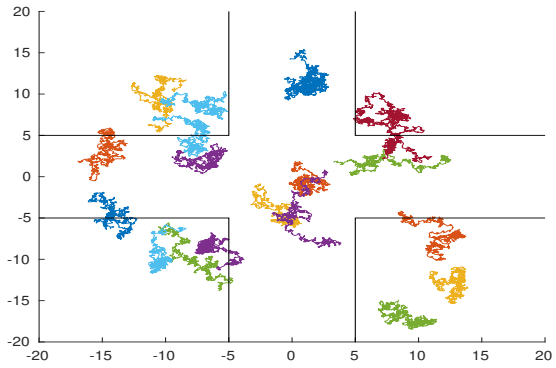

Fig. 1: Example of 20 robots driving randomly at an intersection. When the robot is on the street it is observing pavement images, whereas when it outside of the intersection it has acces to grass images.

classification is accurate and it takes large values in the opposite case. In particular, we consider logistic regression

$$f_i(t, \mathbf{x}) = \log\left(1 + e^{-y_i(t)\mathbf{x}^\top \mathbf{z}_i(t)}\right). \tag{21}$$

The classifier is designed so that the prediction is defined by the sign of the inner product between the classifier $\mathbf{x}$ and the feature vector $\mathbf{z}_i(t)$ observed by robot i at time t. This is, the predicted label is given by $\hat{y}_i(t) = \text{sign}(\mathbf{x}^\top \mathbf{z}_i(t))$. Notice that if the prediction is correct, both $\hat{y}_i(t)$ and $y_i(t)$ have the same sign and thus, the exponential in (21) takes a small value. Which in turn results in $f_i(t, \mathbf{x})$ being small. On the other hand, if the classification is incorrect, the sign of the exponential is positive, which results in a large value of $f_i(t, \mathbf{x})$. Hence, the expression in (21) is a surrogate of the error function since it results in small values when the prediction is correct and on large values on the other hand. Notice that agents need to exchange their current actions $\mathbf{x}(t)$ with their neighbors to solve the minimization of (21) using the algorithm defined by (17) and (18). Since the dimensionality of the actions is as large as the feature vector we want to find a sparse classifier in order to reduce the communication cost. A way of doing so is to include a ℓ_1 norm regularization in the cost, which is known to promote sparsity. Let, $\alpha > 0$ and define the following local cost

$$\tilde{f}_i(t, \mathbf{x}) = f_i(t, \mathbf{x}) + \alpha \left\|\mathbf{x}\right\|_1. \tag{22}$$

The previous objective introduces a tradeoff between classification performance and sparsity. Instead, one can define a desired tolerance for classification error – by imposing that $f_i(t, \mathbf{x})$ is smaller than a given tolerance $\delta > 0$ for all $i = 1 \dots N$– and by minimizing the objective $\|\mathbf{x}\|_1$, so to get the sparsest of the solutions. With this idea we define the following centralized problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{x}\|_1 \\ \text{s.t.} \quad & f_i(t, \mathbf{x}) - \delta < 0 \quad \forall i = 1 \dots N. \end{aligned} \tag{23}$$

To solve this problem in a distributed manner, we define – as done in Section II – local copies of the classifier $\mathbf{x}_i$ for each agent. The decentralized version of the previous problem then yields

$$\begin{aligned} \min_{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N} \quad & \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i\|_1 \\ \text{s.t.} \quad & f_i(t, \mathbf{x}_i) - \delta \leq 0 \quad \forall i = 1 \dots N \\ & \|\mathbf{x}_i - \mathbf{x}_j\|^2 - \gamma \leq 0 \quad \forall i = 1 \dots N. \end{aligned} \tag{24}$$

We evaluate the performance of the saddle-point algorithm (17)–(18) by solving the problem (24) applied to the team of robots navigating around the intersection depicted in Figure 1. The positions of the N agents is initialized by drawing it from a uniform distribution on the square $[-L, L]^2$ and their paths are random walks updated every T_s seconds, where each step is drawn from a two-dimensional Gaussian variable, with zero mean and covariance matrix $\text{diag}(\sigma_w, \sigma_w)$. Ever T_s seconds each agent has observed I images in the IRA[1] database [2] of either grass or pavement. Do notice that even though the algorithm proposed is derived in continuous time, for this application we propose to work with a discrete time system. In Section V-B we present the results achieved by the saddle-point algorithm in the previously described problem. Before doing so, we describe in the next section the feature extraction from the images.

A. Data from image database

The feature extraction is done as in [2], a procedure inspired in the two-dimensional texton [32]. We describe it next for completeness. The texture features $\mathbf{z}_i(t)$ are generated as the sum of a sparse dictionary representation of subpatches of size 24-by-24. This is, each robot classifies images patches of size 24-by-24 by first extracting the nine non-overlapping 8-by-8 sub-patches within it. Each sub-patch is then vectorized, the sample mean subtracted off and divided by its norm. Such that the resulting sub-patch j observed by agent i, yields a zero-mean vector $\mathbf{z}_i^j$ with norm one. The 9 vectors resulting from each sub-patch are stacked as columns in a matrix $\mathbf{Z}_i = [\mathbf{z}_i^1; \dots; \mathbf{z}_i^9]$. On the other hand, the agents have a dictionary of textures that has been trained offline following [2]. An example of this dictionary can be observed in Figure 2. The dictionary can be represented by a matrix $\mathbf{D} \in \mathcal{M}^{n \times 64}$, where n is the number of features that one wants to extract. The feature used for classification by agent i is the aggregate sparse coding $\mathbf{z}_i(t)$, defined as $\mathbf{z}_i(t) = \sum_{j=1}^{9} \mathbf{z}^*(\mathbf{D}; \mathbf{z}_i^j(t))$, where $\mathbf{z}^*(\mathbf{D}; \mathbf{z}_i^j(t))$ is the solution to the following optimization problem.

$$\mathbf{z}^*(\mathbf{D}; \mathbf{z}_i^j(t)) = \underset{\mathbf{z}}{\text{argmin}} \frac{1}{2}\|\mathbf{z}_i^j(t) - \mathbf{D}\mathbf{z}\|_2^2 + \zeta\|\mathbf{z}\|_1, \tag{25}$$

where $\zeta > 0$ the coefficient of the regularization.

B. Results

In this section we present the bahavior of the Online Distributed Online Algorithm (17)–(18) for a team of robots that drive in the intersection as the one depicted in Figure

[1]Integrated Research Assessment for the U.S. Armys Robotics Collaborative Technology Alliance

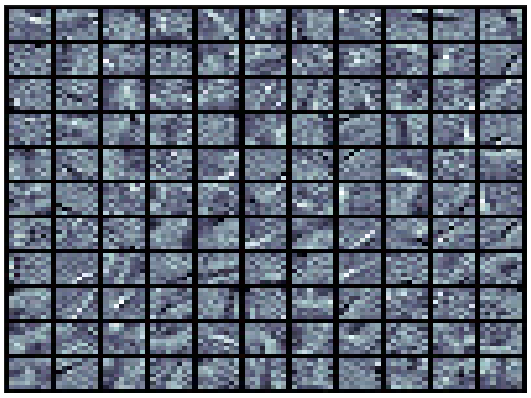

Fig. 2: Example of dictionary for 8-by-8 gray scale patches.

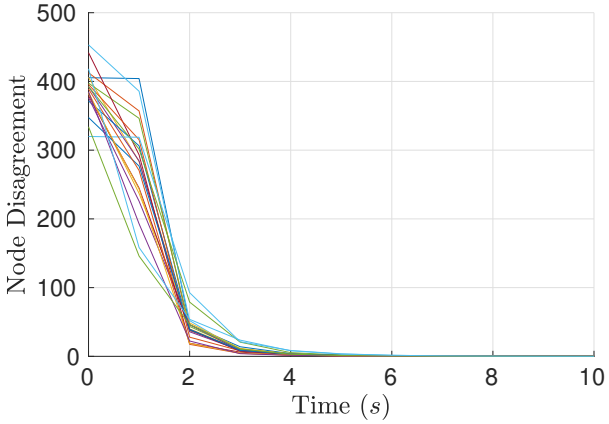

Fig. 3: Network disagreement per node for a network of 20 agents that follow the dynamics (17)–(18). The feature vectors $\mathbf{z}_i(t) \in \mathbb{R}^n$ are extracted from images of the IRA texture database as described in section V-A. The disagreement is sublinear as expected by virtue of Proposition 1.

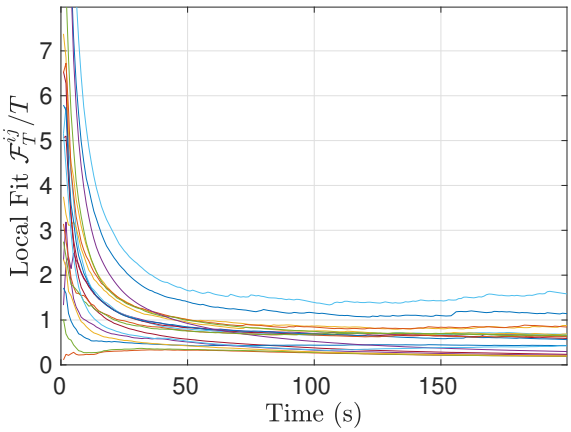

Fig. 4: Network fit $\mathcal{F}_T^{ij}/T$ of one randomly selected agent in a network of $N = 20$ agents that follow the dynamics (17)–(18). The feature vectors $\mathbf{z}_i(t) \in \mathbb{R}^n$ are extracted from images of the IRA texture database as described in Section V-A. As predicted by Theorem 1 the average local fit is sublinear with the time horizon.

1. For this particular example we consider $N = 20$ agents, $L = 15$, σ_w and $T_s = 1$. The parameters of the feature extraction are set to $\zeta = 0.125$, $n = 128$. We chose $\delta = 0.001$, $\gamma = 10$ and the algorithm step size to be $\eta = 0.02$, and we consider that each agent has acces to 24 images per sampling period, in this case, 24 images per second. In Figure 3 we observe that the network disagreement converges to zero in approximately 6 seconds, which implies consensus among the agents. This observation supports the theoretical result in Proposition 1. In Figure 4 we depict the network fit of one random agent. As predicted by Theorem 1 the fit is sublinear. The effectiveness of the algorithm can be observed in the classification accuracy achieved by the agents in Figure 5. Notice that the classification error of all the agents is bellow 30%. It can be observed as well, that some agents classify with accuracy above 90%. The latter is the case for agents that are observing grass. There seems to be an intrinsic difficulty in classifying pavement in the current data set. To support this claim we compute the covariance matrix of 512 features of images selected randomly. We then project the 192-dimensional feature vector onto the first two principal components. This projection is depicted in Figure 6. As it can be observed the points corresponding to pavement cannot be separated from points corresponding to grass, yet there is a cluster of grass points that is away from the points containing pavement. This suggests that it is indeed harder to classify pavement images.

VI. Conclusion

We considered the problem of constrained distributed online learning. Each agent only has access to its local constraints and objective function, and the aim is to coordinate the actions among the agents such that the resulting trajectories are feasible and optimal for the team as a whole. We showed that a distributed online version of the saddle point algorithm achieves global fit, regret and network disagreement bounded by functions whose growth rate is bounded by $\sqrt{T}$. The latter result suggests vanishing constraint violation, optimality and network agreement in average as time evolves. We evaluate the performance of the algorithm for a team of robots driving through an urban environment to perform real time texture classification for the purpose of terrain recognition.

References

[1] C. Shen, T.-H. Chang, K.-Y. Wang, Z. Qiu, and C.-Y. Chi, "Distributed robust multicell coordinated beamforming with imperfect csi: An admm approach," *IEEE Transactions on signal processing*, vol. 60, no. 6, pp. 2988–3003, 2012.

[2] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "D4l: Decentralized dynamic discriminative dictionary learning," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 2966–2973, IEEE, 2015.

[3] A. Koppel, S. Paternain, C. Richard, and A. Ribeiro, "Decentralized online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 66, no. 12, pp. 3240–3255.

[4] C. Freundlich, S. Lee, and M. M. Zavlanos, "Distributed active state estimation with user-specified accuracy," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 418–433, 2018.

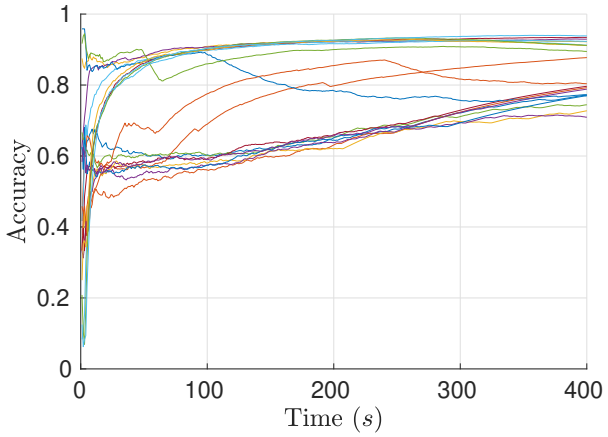

Fig. 5: The accuracy of the prediction per node reaches a minimum of 70% for a network of $N = 20$ agents that follow the dynamics (17)–(18). The feature vectors $\mathbf{z}_i(t) \in \mathbb{R}^n$ are extracted from images of the IRA texture database as described in Section V-A. As it can be observed some agents achieve accuracy of 90%. This agents are observing grass images, whereas those that perform worst are classifying pavement. The fact that one of the classes is poorly classified can be understood as not having a cluster of points where there is no grass as it can be seen in the study of the two principal components of the data set in Figure 6.

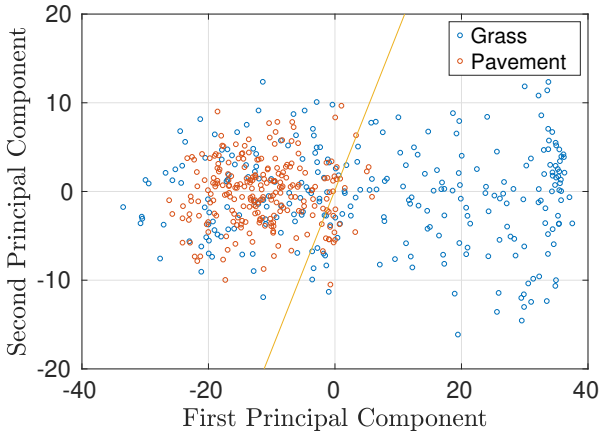

Fig. 6: We depict the projection of the features extracted from 512 images onto the two principal components of the data set. As it can be observed in this picture the images of grass are easier to classify since they present a distinct cluster without any pavement images. On the other hand, the cluster of points corresponding to pavement are intertwined with grass images, which makes its classification harder. We depict as well the projection of the classifier trained by node 1 after 400 seconds.

[5] A. Ghosh and S. Sarkar, "Pricing for profit in internet of things," in *Information Theory (ISIT), 2015 IEEE International Symposium on*, pp. 2211–2215, IEEE, 2015.

[6] K. Gatsis and G. J. Pappas, "Wireless control for the iot: Power, spectrum, and security challenges," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pp. 341–342, ACM, 2017.

[7] K. J. Arrow and L. Hurwicz, *Studies in linear and nonlinear programming*. CA: Stanford University Press, 1958.

[8] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[9] D. Yuan, S. Xu, and H. Zhao, "Distributed primal–dual subgradient method for multiagent optimization via consensus algorithms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 6, pp. 1715–1724, 2011.

[10] T.-H. Chang, A. Nedic, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1524–1538, 2014.

[11] D. Feijer and F. Paganini, "Stability of primal–dual gradient dynamics and applications to network optimization," *Automatica*, vol. 46, no. 12, pp. 1974–1981, 2010.

[12] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.

[13] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *ICML*, pp. 928–936, 2003.

[14] V. N. Vapnik, "The nature of statistical learning theory," 1995.

[15] S. Paternain and A. Ribeiro, "Online learning of feasible strategies in unknown environments," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2807–2822, 2017.

[16] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Transactions on Signal Processing*, 2017.

[17] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed optimization via dual averaging," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 1484–1489, IEEE, 2013.

[18] S. Lee and M. M. Zavlanos, "Distributed primal-dual methods for online constrained optimization," in *American Control Conference (ACC), 2016*, pp. 7171–7176, IEEE, 2016.

[19] S. Lee, A. Ribeiro, and M. M. Zavlanos, "Distributed continuous-time online optimization using saddle-point methods," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pp. 4314–4319, IEEE, 2016.

[20] A. Koppel, B. M. Sadler, and A. Ribeiro, "Proximity without consensus in online multi-agent optimization," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 3726–3730, IEEE, 2016.

[21] F. Y. Jakubiec and A. Ribeiro, "D-map: Distributed maximum a posteriori probability estimation of dynamic systems," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 450–466, 2013.

[22] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[23] R. D. Braun, "Collaborative optimization: an architecture for large-scale distributed design," 1996.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[25] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[26] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[27] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2-3, pp. 169–192, 2007.

[28] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro, "A class of prediction-correction methods for time-varying convex optimization.," *IEEE Trans. Signal Processing*, vol. 64, no. 17, pp. 4576–4591, 2016.

[29] M. Fazlyab, S. Paternain, V. M. Preciado, and A. Ribeiro, "Prediction-correction interior-point method for time-varying convex optimization," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1973–1986, 2018.

[30] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[31] S. Paternain, S. Lee, M. M. Zavlanos, and A. Ribeiro, "Distributed constrained online learning," *arXiv preprint arXiv:1903.06310*, 2019.

[32] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International journal of computer vision*, vol. 43, no. 1, pp. 29–44, 2001.