A Saddle Point Algorithm for Networked Online Convex Optimization

Alec Koppel, Felicia Y. Jakubiec and Alejandro Ribeiro

***Abstract*—An algorithm to learn optimal actions in convex distributed online problems is developed. Learning is online because cost functions are revealed sequentially and distributed because they are revealed to agents of a network that can exchange information with neighboring nodes only. Learning is measured in terms of the global network regret, which is defined here as the accumulated loss of causal prediction with respect to a centralized clairvoyant agent to which the information of all times and agents is revealed at the initial time. A variant of the Arrow-Hurwicz saddle point algorithm is proposed to control the growth of global network regret. This algorithm uses Lagrange multipliers to penalize the discrepancies between agents and leads to an implementation that relies on local operations and exchange of variables between neighbors. We show that decisions made with this saddle point algorithm lead to regret whose order is not larger than $O(\sqrt{T})$, where T is the total operating time. Numerical behavior is illustrated for the particular case of distributed recursive least squares. An application to computer network security in which service providers cooperate to detect the signature of malicious users is developed to illustrate the practical value of the proposed algorithm.**

I. Introduction

In distributed online learning problems, agents of a network want to causally learn a strategy that is as good as the one they could learn if they had access to the information of all other agents while communicating with neighboring nodes only. The specific setting considered here consists of convex cost functions that are sequentially revealed to individual agents. In offline centralized learning the functions of all agents and all times are known beforehand and a *constant* and *common* action is selected for agents to play. In online centralized learning, functions are still available at a central location but are revealed sequentially. The *common* action to be played by agents is selected ex ante using past observations and incurs a cost ex post after the current functions become available. In distributed online learning the agents select actions based on previous cost functions observed locally and messages received from neighbors in past communication exchanges. This papers proposes the use of a saddle point algorithm so that distributed online strategies achieve comparable performance to centralized offline strategies.

Centralized online learning problems can be formulated in the language of regret minimization [2], [3]. In this setting, a learner makes a sequence of plays to which Nature provides the answer in the form of a loss function. Regret is defined as the accumulation over time of the loss difference between the online learner and a clairvoyant offline learner to which cost functions have been revealed beforehand. We interpret regret as a measure of the price for causal prediction. When loss functions are convex, several algorithms are known to achieve regret whose growth with the accumulated operating time T is sublinear – which entails vanishing cost differences between online and offline plays at specific times. Germane to this paper is online gradient descent in which plays are updated by descending on the gradient of observed costs. Despite the mismatch of descending on the prior function while incurring a cost in the current function, online gradient descent achieves regret that grows not faster than a function of order $O(\sqrt{T})$ in general and not faster than $O(\log T)$ under more stringent conditions [4]. Other methods to control regret growth are proximal maps [5], mirror descent [6], [7], and dual averaging [8]. All of these strategies may be understood as special cases of a strategy known as "follow the regularized leader" [2].

Often, data is gathered by agents of a network. Nodes would like to cooperate to learn global optimal strategies but collecting data at a clearinghouse is slow, costly, and sometimes unsafe. This motivates the use of distributed online learning algorithms. In deterministic settings, optimal actions of separable convex costs are computed using distributed optimization algorithms which can be categorized into primal methods, dual methods, and primal-dual methods. In primal methods agents descent along their local gradients while averaging their signals with those of their neighbors, [7], [9]–[11]. In dual methods agents reformulate distributed optimization as an agreement constrained optimization problem and ascend in the dual domain using the fact that dual function gradients can be computed while cooperating with neighboring nodes only [12], [13]. Variations of dual methods include the alternating direction method of multipliers [14], [15] and the incipient development of second order methods that rely on separable approximations of global Newton steps [16]. Primal-dual methods combine primal descent with dual ascent [9], [17], [18]. Primal methods have been generalized to distributed online learning and have proved effective for particular cases where averaging is advantageous [19], [20].

In this paper we propose a variant of the saddle point method [17], [21] to solve distributed online learning problems. Our main technical contribution is to demonstrate that saddle point iterates achieve a regret that grows at a rate not faster than $O(\sqrt{T})$. We begin the paper in Section II with a discussion of the concept of regret and extend it to networked settings by defining the concepts of centralized, uncoordinated, local, and global networked regret. The concepts of local and global networked regret, as well as their practical meaning, are illustrated with applications to decentralized recursive least squares (Section II-A) and decentralized online support vector machines (Section II-B). The saddle point algorithm is developed in Section III by drawing parallels with deterministic and stochastic optimization. The method relies on the addition of equality constraints and the definition of an online Lagrangian associated with the instantaneous cost function. Primal descent steps on the Lagrangian are used to update actions and dual ascent steps are used to update the multipliers associated with the equality constraints. As in the case of online gradient descent, there is

Work in this paper is supported by NSF CCF-1017454, NSF CCF-0952867 and ONR N00014-12-1-0997. The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104. Email: {akoppel, life, aribeiro}@seas.upenn.edu. Part of the results in this paper appeared in [1]

a mismatch between descending on past online Lagrangians to find an action to be played at the current time and incur a cost associated with a function that can be arbitrarily different. Despite this mismatch, and again, analogous to online gradient descent, the saddle point method achieves regret of order $O(\sqrt{T})$ (Section IV). This result is first established in terms of the global networked regret (Theorem 1) and then shown to hold for the regrets of individual agents as well (Theorem 2).

Numerical analyses are undertaken in Section V using the recursive least squares problem of Section II-A as an example. Local and global networked regrets are shown to decline at the rates guaranteed by the theoretical analysis. Agents are also shown to learn the same globally optimal estimate that would had been learnt had all the information being available at a central location. We also illustrate the effects of network size, connectivity, and topology in learning rates and compare the performance of saddle point with the methods developed in [19], [20]. An application of the proposed algorithm to train a support vector machine in the context of computer network security is presented in Section VI. Each agent of the network represents a service provider to which friendly and hostile users connect sequentially. The service providers aim to learn the signature of intrusions so as to deny service to attackers. Different service providers want to collaborate to detect intrusions but sharing information is problematic in terms of cost, delay, and the possibility of revealing sensitive proprietary information. The saddle point method allows provider cooperation while sharing multipliers and classification vectors without explicitly exchanging feature vectors. Concluding remarks are provided in Section VII.

II. Regret Minimization for Distributed Learning

We consider formulations of learning problems with instantaneous actions $\tilde{\mathbf{x}}_t \in X \subseteq \mathbb{R}^p$ chosen by a player, instantaneous functions $\mathbf{f}_t : \mathbb{R}^p \to \mathbb{R}^q$ chosen by nature, and associated losses $l_t(\tilde{\mathbf{x}}_t, \mathbf{f}_t)$ that indicate how good the choice of playing $\tilde{\mathbf{x}}_t$ is when nature selects the function $\mathbf{f}_t$. In *offline* learning the functions $\mathbf{f}_t$ for times $t = 1, \dots T$ are known beforehand at time $t = 0$ and used to select a fixed strategy $\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}$ for all times. The total loss associated with the selection of $\tilde{\mathbf{x}}$ is $\sum_{t=1}^{T} l_t(\tilde{\mathbf{x}}, \mathbf{f}_t)$. In *online* learning the function $\mathbf{f}_t$ is revealed at time t and we are required to choose $\tilde{\mathbf{x}}_t$ without knowing $\mathbf{f}_t$ but rather the functions $\mathbf{f}_u$ that nature played at earlier times $u < t$. The total loss associated with the variables $\tilde{\mathbf{x}}_t$ played for times $1 \leq t \leq T$ is the sum $\sum_{t=1}^{T} l_t(\tilde{\mathbf{x}}_t, \mathbf{f}_t)$. The regret associated with these plays is defined as the difference between their aggregate cost and the minimum aggregate cost achievable by offline policies

$$\mathbf{Reg}_T^{\mathrm{C}} := \sum_{t=1}^{T} l_t(\tilde{\mathbf{x}}_t, \mathbf{f}_t) - \inf_{\tilde{\mathbf{x}} \in X} \sum_{t=1}^{T} l_t(\tilde{\mathbf{x}}, \mathbf{f}_t). \tag{1}$$

In regret formulations of online learning the goal is to design strategies that observe past functions $\mathbf{f}_u$ played by nature at times $u < t$ to select and action $\tilde{\mathbf{x}}_t$ that makes the regret $\mathbf{Reg}_T^C$ in (1) small. In particular, we say that a strategy learns optimal plays if $\mathbf{Reg}_T^C/T$ vanishes with growing T. We emphasize that the functions $\mathbf{f}_t$ are arbitrary and that while the offline strategy has the advantage of knowing all functions beforehand, online strategies have the advantage of being allowed to change their plays at each time slot.

In this paper we subsume the functions $\mathbf{f}_t$ and the loss l_t into the function $f_t : \mathbb{R}^p \to \mathbb{R}$ such that $f_t(\tilde{\mathbf{x}}) = l_t(\tilde{\mathbf{x}}, \mathbf{f}_t)$ and consider cases in which functions f_t are written as a sum of components available at different nodes of a network. To be specific start by defining the optimal offline strategy as $\tilde{\mathbf{x}}^* = \operatorname{argmin}_{\tilde{\mathbf{x}}} \sum_{t=1}^{T} f_t(\tilde{\mathbf{x}})$ and rewrite the regret in (1) as

$$\mathbf{Reg}_T^{\mathrm{C}} = \sum_{t=1}^{T} f_t(\tilde{\mathbf{x}}_t) - \sum_{t=1}^{T} f_t(\tilde{\mathbf{x}}^*). \tag{2}$$

Further consider a symmetric and connected network $\mathcal{G} = (V, \mathcal{E})$ with N nodes forming the vertex set $V = \{1, \dots, N\}$ and $M = |\mathcal{E}|$ directed edges of the form $e = (j, k)$. That the network is symmetric means that if $e = (j, k) \in \mathcal{E}$ it must also be that $e' = (k, j) \in \mathcal{E}$. That the network is connected means that all pairs of nodes are connected by a chain of edges. We also define the neighborhood of j as the set of nodes $n_j := \{k : (j, k) \in \mathcal{E}\}$ that share an edge with j. Each node in the network is associated with a sequence of cost functions $f_{i,t} : \mathbb{R}^p \to \mathbb{R}$ for all times $t \geq 1$. If a common variable $\tilde{\mathbf{x}}$ is played for all these functions the global network cost at time t is then given by

$$f_t(\tilde{\mathbf{x}}) = \sum_{i=1}^{N} f_{i,t}(\tilde{\mathbf{x}}). \tag{3}$$

The functions $f_{i,t}$ in (4), and as a consequence the functions f_t are assumed convex for all times t but are otherwise arbitrary.

Combining the definitions in (2) and (3) we can consider a coordinated game where all agents play a common variable $\tilde{\mathbf{x}}_t$ at time t. The accumulated regret associated with playing the coordinated sequence $\{\tilde{\mathbf{x}}_t\}_{t=1}^{T}$, as opposed to playing the optimal $\tilde{\mathbf{x}}^* = \operatorname{argmin}_{\tilde{\mathbf{x}}} \sum_{t=1}^{T} f_t(\tilde{\mathbf{x}})$ for all times t, can then be expressed as

$$\mathbf{Reg}_T^{\mathrm{C}} = \sum_{t=1}^{T} \sum_{i=1}^{N} f_{i,t}(\tilde{\mathbf{x}}_t) - \sum_{t=1}^{T} \sum_{i=1}^{N} f_{i,t}(\tilde{\mathbf{x}}^*). \tag{4}$$

An alternative formulation is to consider that agents play their own variables $\mathbf{x}_{i,t}$ to incur their own local cost $f_{i,t}(\mathbf{x}_{i,t})$. In this case we have the aggregate cost $\sum_{i=1}^{N} f_{i,t}(\mathbf{x}_{i,t})$ which leads to the definition of the uncoordinated regret by time T as

$$\mathbf{Reg}_T^{\mathrm{U}} = \sum_{t=1}^{T} \sum_{i=1}^{N} f_{i,t}(\mathbf{x}_{i,t}) - \sum_{t=1}^{T} \sum_{i=1}^{N} f_{i,t}(\tilde{\mathbf{x}}^*). \tag{5}$$

This formulation is of little interest because agents are effectively independent of each other. Indeed, to reduce the regret in (5) it suffices to let agents learn strategies that are good with respect to their local costs $\sum_{t=1}^{T} f_{i,t}(\mathbf{x}_{i,t})$. A simple local gradient descent policy can achieve small regret with respect to the optimal *local* action $\mathbf{x}_i^* = \operatorname{argmin}_{\mathbf{x}_i} \sum_{t=1}^{T} f_{i,t}(\mathbf{x}_i)$ [22]. This uncoordinated strategy is likely to result in negative regret in (5) since the variable $\tilde{\mathbf{x}}^*$ is chosen as common across all agents.

A more appropriate formulation is to consider games where agents have an incentive to learn the cost functions of their peers. Suppose then that each agent in the network plays his own variables $\mathbf{x}_{i,t}$ which are not necessarily identical to the variables $\mathbf{x}_{j,t}$ played by other agents $j \neq i$ in the same time slot. However, we still want each agent to learn a play that is optimal with respect to the global cost in (3). Thus, we formulate a problem in which the *local regret* of agent j is defined as

$$\mathbf{Reg}_T^{j} = \sum_{t=1}^{T} \sum_{i=1}^{N} f_{i,t}(\mathbf{x}_{j,t}) - \sum_{t=1}^{T} \sum_{i=1}^{N} f_{i,t}(\tilde{\mathbf{x}}^*). \tag{6}$$

The regret formulations in (4) and (6) are identical. This means that (6) corresponds to a problem in which agent j aspires to learn a play that is as good as the play that can be learned by a centralized agent that has access to the cost functions $f_{i,t}$ of all agents i. However, the assumption here is that only the local functions $f_{j,t}$ are known to agent j.

By further considering the sum of all local regrets in (6) we define *global networked regret* as

$$\begin{aligned}\mathbf{Reg}_T &:= \frac{1}{N}\sum_{j=1}^{N}\mathbf{Reg}_T^j \\ &= \frac{1}{N}\sum_{t=1}^{T}\sum_{i,j=1}^{N} f_{i,t}(\mathbf{x}_{j,t}) - \sum_{t=1}^{T}\sum_{i=1}^{N} f_{i,t}(\tilde{\mathbf{x}}^*), \end{aligned} \tag{7}$$

where we used (6) and simplified terms to write the second equality. In this paper we develop a variation of the saddle point algorithm of Arrow and Hurwicz [17] to find a strategy whose local and global network regrets [cf. (6) and (7)] are of order not larger than $O(\sqrt{T})$. We also show that the proposed algorithm can be implemented by agents that have access to their local cost functions only and perform causal variable exchanges with peers in their network neighborhood. This saddle point algorithm is presented in Section III after the discussion of two representative examples.

A. Distributed recursive least squares

As an example problem that admits the formulation in (7) consider a distributed version of recursive least squares (RLS). Suppose we want to estimate a signal $\tilde{\mathbf{x}} \in \mathbb{R}^p$ when agents collect observations $\mathbf{y}_{it} \in \mathbb{R}^q$ that relate to $\tilde{\mathbf{x}}$ according to the model $\mathbf{y}_{it} = \mathbf{H}_{i,t}\tilde{\mathbf{x}} + \mathbf{w}_{i,t}$, where the noise $\mathbf{w}_{i,t}$ is Gaussian and independent and identically distributed across nodes and time. The optimal estimator $\tilde{\mathbf{x}}^*$ given the observations $\mathbf{y}_{i,t}$ for all i and t is the least mean squared error estimator $\tilde{\mathbf{x}}^* = \operatorname{argmin}_x \sum_{t=1}^{T}\sum_{i=1}^{N} \|\mathbf{H}_{i,t}\tilde{\mathbf{x}} - \mathbf{y}_{i,t}\|^2$. If the signals $\mathbf{y}_{i,t}$ are known for all nodes i and times t the optimal estimator $\tilde{\mathbf{x}}^*$ can be easily computed. In this paper we are interested in cases where the signal $\mathbf{y}_{j,t-1}$ is revealed at time $t-1$ to sensor j which then proceeds to determine the causal signal estimate $\mathbf{x}_{j,t} \in \mathbb{R}^p$ as a function of past observations $y_{j,u}$ for $u = 1, \ldots, t-1$ and information received from neighboring nodes in previous time slots. This is tantamount to a distributed RLS problem because signals are revealed sequentially to agents of a distributed network. Setting aside for the moment the issue of how to select $\mathbf{x}_{j,t}$ the regret in (6) is a measure of goodness for $\mathbf{x}_{j,t}$ with respect to a clairvoyant centralized estimator. Indeed, the particular form of (6) becomes

$$\begin{aligned}\mathbf{Reg}_T^j = &\sum_{t=1}^{T}\sum_{i=1}^{N} \|\mathbf{H}_{i,t}\mathbf{x}_{j,t} - \mathbf{y}_{i,t}\|^2 \\ &- \sum_{t=1}^{T}\sum_{i=1}^{N} \|\mathbf{H}_{i,t}\tilde{\mathbf{x}}^* - \mathbf{y}_{i,t}\|^2. \end{aligned} \tag{8}$$

The regret $\mathbf{Reg}_T^j$ in (8) is measuring the mean squared error penalty that agent j is incurring by estimating the signal $\tilde{\mathbf{x}}$ as $\mathbf{x}_{j,t}$ instead of the optimal estimator $\tilde{\mathbf{x}}^*$. In that sense it can be interpreted as the penalty for distributed causal operation with respect to centralized clairvoyant operation – the estimate $\tilde{\mathbf{x}}^*$ is centralized because it has access to the observations of all nodes and clairvoyant because it has access to the current observation $\mathbf{y}_{i,t}$. The algorithms developed in this paper are such that the regret penalty $\mathbf{Reg}_T^j$ in (8) grows at a sub-linear rate not larger than $O(\sqrt{T})$ – see Sections III and IV.

B. Decentralized Online Support Vector Machines

As a second example consider the problem of training a support vector machine (SVM) for binary classification [23]. Suppose that each agent i is given a training data set $\mathcal{S}_i$ with T elements that are revealed sequentially. The elements of this set are pairs $(\mathbf{z}_{i,t}, y_{i,t})$ where $\mathbf{z}_{i,t} \in \mathbb{R}^p$ is a feature vector having a known binary label $y_{i,t} \in \{-1, 1\}$. Given the aggregate training set $\mathcal{S} = \cup_{i=1}^{N}\mathcal{S}_i$ we seek a decision hyperplane which best separates data points with distinct labels. That is, we seek a vector $\tilde{\mathbf{x}} \in \mathbb{R}^p$ such that $\tilde{\mathbf{x}}^T\mathbf{z}_{i,t} > 0$ whenever $y_{i,t} = 1$ and $\tilde{\mathbf{x}}^T\mathbf{z}_{i,t} > 0$ for $y_{i,t} = -1$. Since data sets are not separable in general, we consider a soft margin formulation which penalizes misclassifications through the hinge loss $l((\mathbf{z}_{i,t}, y_{i,t}); \tilde{\mathbf{x}}) := \max(0, 1 - y_{i,t}\tilde{\mathbf{x}}^T\mathbf{z}_{i,t})$. The hinge loss $l((\mathbf{z}_{i,t}, y_{i,t}); \tilde{\mathbf{x}})$ is null if the label $y_{i,t}$ is correctly classified by the hyperplane defined by $\tilde{\mathbf{x}}$ – which happens when $\tilde{\mathbf{x}}^T\mathbf{z}_{i,t} > 0$ for $y_{i,t} = 1$ and $\tilde{\mathbf{x}}^T\mathbf{z}_{i,t} < 0$ for $y_{i,t} = -1$ – and grows linearly with the distance between the point $\mathbf{z}_{i,t}$ and the classifying hyperplane otherwise. To balance model complexity with training error we further add a quadratic regularization term so that the optimal classifier $\tilde{\mathbf{x}}^*$ is the one that minimizes the cost

$$\tilde{\mathbf{x}}^* = \operatorname*{argmin}_{\tilde{\mathbf{x}} \in X} \frac{\zeta}{2}\|\tilde{\mathbf{x}}\|_2^2 + \frac{1}{NT}\sum_{t=1}^{T}\sum_{i=1}^{N} \max\left(0, 1 - y_{i,t}\cdot\tilde{\mathbf{x}}^T\mathbf{z}_{i,t}\right), \tag{9}$$

where ζ is the regularization constant tuning classifier bias and variance. The classifier $\tilde{\mathbf{x}}^*$ that results from solving (9) is the centralized batch classifier.

To consider distributed online versions of SVM training define functions $f_{i,t} : \mathbb{R}^p \to \mathbb{R}$ with values

$$f_{i,t}(\tilde{\mathbf{x}}) = \frac{\zeta}{2}\|\tilde{\mathbf{x}}\|_2^2 \; + \; \max\left(0, 1 - y_{i,t}\tilde{\mathbf{x}}^T\mathbf{z}_{i,t}\right), \tag{10}$$

so that the minimization argument in (9) can be written as $\sum_{i,t} f_{i,t}(\tilde{\mathbf{x}})$. This modification amounts to considering the case where each agent in the network has access only to a distinct local labeled data set.

The total penalty $\sum_{i,t} f_{i,t}(\tilde{\mathbf{x}}^*)$ of the optimal batch or offline classifier $\tilde{\mathbf{x}}^*$ quantifies the number of misclassifications incurred when the observations of all nodes and all times are known beforehand. The various online classifiers whose performances are described by (4), (5), and (6) quantify the number of misclassifications incurred when the class corresponding to feature vectors $\mathbf{z}_{i,t}$ is predicted causally using features $\mathbf{z}_{i,u}$ and associated classes $y_{i,u}$ observed at past times $u < t$. The centralized regret $\mathbf{Reg}_T^{\mathrm{C}}$ in (4) corresponds to the case when all observations are causally available at a central location. The uncoordinated regret $\mathbf{Reg}_T^{\mathrm{U}}$ in (5) corresponds to the case where classification is based on past local observations only. The local regrets $\mathbf{Reg}_T^j$ in (6) corresponds to cases when each of the agents is trying to accumulate past global network knowledge through communication with neighboring agents.

III. Arrow-Hurwicz Saddle Point Algorithm

We turn to developing a saddle point algorithm to control the growth of the local and global network regrets [cf. (6) and (7)]. Since the regret functions $\mathbf{Reg}_T^j$ defined in (6) are the same

for all agents j, plays $\mathbf{x}_{j,t}$ that are good for one agent are also good for another. Thus, a suitable strategy is to select actions $\mathbf{x}_{j,t}$ which are the same for every agent. Since the network $\mathcal{G}$ is assumed to be connected, this relationship can be attained by imposing the constraint $\mathbf{x}_{j,t} = \mathbf{x}_{k,t}$ for all pairs of neighboring nodes $(j,k) \in \mathcal{E}$. To write more compactly define the column vector $\mathbf{x}_t := [\mathbf{x}_{1,t}; \dots; \mathbf{x}_{N,t}] \in \mathbb{R}^{Np}$ and the augmented graph edge incidence matrix $\mathbf{C} \in \mathbb{R}^{Mp \times Np}$. The matrix $\mathbf{C}$ is formed by $M \times N$ square blocks of dimension p. If the edge $e = (j,k)$ links node j to node k the block (e,j) is $[\mathbf{C}]_{ej} = \mathbf{I}_p$ and the block $[\mathbf{C}]_{ek} = -\mathbf{I}_p$, where $\mathbf{I}_p$ denotes the identity matrix of dimension p. All other blocks are identically null, i.e., $[\mathbf{C}]_{ek} = \mathbf{0}_p$ for all edges $e \neq (j,k)$. With this definitions the constraint $\mathbf{x}_{j,t} = \mathbf{x}_{k,t}$ for all pairs of neighboring nodes can be written as

$$\mathbf{C}\mathbf{x}_t = \mathbf{0}, \qquad \forall t = 1, \dots, T. \tag{11}$$

The edge incidence matrix $\mathbf{C}$ has exactly p null singular values. We denote as $0 < \gamma$ the smallest nonzero singular value of $\mathbf{C}$ and as Γ the largest singular value of $\mathbf{C}$. The singular values γ and Γ are measures of network connectedness.

Imposing the constraint in (11) for all times t requires global coordination – indeed, the formulation would be equivalent to the centralized regret problem in (4). Instead, we consider a modification of (3) in which we add a linear penalty term to incentivize the selection of coordinated actions. Introduce then dual variables $\boldsymbol{\lambda}_{e,t} = \boldsymbol{\lambda}_{jk,t} \in \mathbb{R}^p$ associated with the constraint $\mathbf{x}_{j,t} - \mathbf{x}_{k,t} = \mathbf{0}$ and consider the addition of penalty terms of the form $\boldsymbol{\lambda}_{jk,t}^T(\mathbf{x}_{j,t} - \mathbf{x}_{k,t})$. For an edge that starts at node j, the multiplier $\boldsymbol{\lambda}_{jk,t}$ is assumed to be kept at node j. Further introduce the stacked vector $\boldsymbol{\lambda}_t := [\boldsymbol{\lambda}_{1,t}; \dots; \boldsymbol{\lambda}_{M,t}] \in \mathbb{R}^{Mp}$ and define the online Lagrangian at time t as

$$\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t) \;=\; \sum_{i=1}^{N} f_{i,t}(\mathbf{x}_{i,t}) + \boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{x}_t \;= f_t(\mathbf{x}) + \boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{x}_t. \tag{12}$$

The definition in (12) corresponds to the Lagrangian associated with the minimization of the instantaneous function $\sum_{i=1}^N f_{i,t}(\mathbf{x}_{i,t})$ subject to the agreement constraint $\mathbf{C}\mathbf{x}_t = \mathbf{0}$. Using this interpretation of the online Lagragian we consider the use of the Arrow-Hurwicz saddle point method. This method exploits the fact that primal-dual optimal pairs are saddle points of the Lagrangian to work through successive primal gradient descent steps and dual gradient ascent steps. Particularized to the online Lagrangian in (12) the saddle point algorithm takes the form

$$\mathbf{x}_{t+1} = \mathcal{P}_X[\mathbf{x}_t - \epsilon \nabla_{\mathbf{x}} \mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)], \tag{13}$$

$$\boldsymbol{\lambda}_{t+1} = \mathcal{P}_\Lambda[\boldsymbol{\lambda}_t + \epsilon \nabla_{\boldsymbol{\lambda}} \mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)], \tag{14}$$

where ϵ is a given stepsize. The notation $\mathcal{P}_\Lambda(\boldsymbol{\lambda})$ denotes projection of dual variables on a given convex compact set Λ. We assume that the set of multipliers Λ can be written as a Cartesian product of sets Λ_{jk} so that the projection of $\boldsymbol{\lambda}$ into Λ is equivalent to the separate projection of the components $\boldsymbol{\lambda}_{jk}$ into the sets Λ_{jk}. The notation $\mathcal{P}_X(\mathbf{x})$ denotes projection onto the set of feasible primal variables so that we have $\mathbf{x}_j \in X$ for all the N components of the vector $\mathbf{x}_t := [\mathbf{x}_1; \dots; \mathbf{x}_N]$.

The pair of iterations in (13)-(14) can be implemented in a distributed manner such that the variables kept at node j, namely, $\mathbf{x}_{j,t}$ and $\boldsymbol{\lambda}_{jk,t}$, are updated using the values of other local variables and variables of neighboring nodes, namely, $\mathbf{x}_{k,t}$ and $\boldsymbol{\lambda}_{kj,t}$ for $k \in n_j$. To see that this is true, take the gradient with respect to $\mathbf{x}_{j,t}$ in (13) and observe that only the term $\nabla_{\mathbf{x}_{j,t}} f_{j,t}(\mathbf{x}_{j,t})$ is not null in the sum in (12). Further observe that when taking the gradient of the linear penalty term $\boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{x}_t$ the variable $\mathbf{x}_j$ appears only in the terms associated with edges of the form $e = (j,k)$ or $e = (k,j)$. Thus, the gradient of this penalty term with respect to $\mathbf{x}_j$ can be written as $\nabla_{\mathbf{x}_{j,t}}(\boldsymbol{\lambda}_t^T \mathbf{C}\mathbf{x}_t) = \sum_{k \in n_j} \boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t}$. Combining these two observations it follows that the gradient of the online Lagrangian with respect to the primal variable $\mathbf{x}_{j,t}$ of node j can be written as

$$\nabla_{\mathbf{x}_j} \mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t) = \nabla_{\mathbf{x}_j} f_{j,t}(\mathbf{x}_{j,t}) + \sum_{k \in n_j} (\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t}). \tag{15}$$

The computation of this gradient only depends on the local gradient of the local loss function $f_{j,t}$, the local primal variable $\mathbf{x}_{j,t}$, the local dual variables $\boldsymbol{\lambda}_{jk,t}$ and the dual variables $\boldsymbol{\lambda}_{kj,t}$ of neighboring nodes $k \in n_j$. Similarly, to determine the gradient of the online Lagrangian with respect to the dual variable $\boldsymbol{\lambda}_{jk}$ observe that the only term in (12) that involves this variable is the one associated with the constraint $\mathbf{x}_{j,t} - \mathbf{x}_{k,t}$. Therefore, the gradient with respect to $\boldsymbol{\lambda}_{jk}$ can be written as

$$\nabla_{\boldsymbol{\lambda}_{jk}} \mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t) = \mathbf{x}_{j,t} - \mathbf{x}_{k,t}. \tag{16}$$

To compute this gradient at node j we use the local primal variable $\mathbf{x}_{j,t}$ and the neighboring play $\mathbf{x}_{k,t}$. Separating (13) along the components $\mathbf{x}_{j,t}$ associated with node j it follows that the primal iteration is equivalent to the N parallel updates

$$\mathbf{x}_{j,t+1} = \mathcal{P}_X\Big[\mathbf{x}_{j,t} - \epsilon\Big(\nabla_{\mathbf{x}_j} f_{j,t}(\mathbf{x}_{j,t}) + \sum_{k \in n_j} (\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t})\Big)\Big], \tag{17}$$

where $\mathcal{P}_X(\mathbf{x}_{j,t})$ denotes projection of $\mathbf{x}_{j,t}$ into the feasible primal set X. Likewise, separating (14) into the subcomponents along the $\boldsymbol{\lambda}_{jk}$ direction yields the M parallel updates

$$\boldsymbol{\lambda}_{jk,t+1} = \mathcal{P}_{\Lambda_{jk}}\Big[\boldsymbol{\lambda}_{jk,t} + \epsilon\,(\mathbf{x}_{j,t} - \mathbf{x}_{k,t})\Big], \tag{18}$$

where $\mathcal{P}_{\Lambda_{jk}}$ denotes projection of $\boldsymbol{\lambda}_{jk}$ into the dual set Λ_{jk}. Node j can implement (17)-(18) by using local variables and receiving variables $\boldsymbol{\lambda}_{kj,t}$ and $\mathbf{x}_{k,t}$ maintained at neighboring nodes $k \in n_j$.

As an example application consider the distributed RLS problem in Section II-A. From (8), we glean that local functions are $f_{i,t}(\mathbf{x}_{i,t}) = \|\mathbf{H}_{i,t}\mathbf{x}_{i,t} - \mathbf{y}_{i,t}\|^2$ to conclude that the primal update at agent j shown in (17) takes the specific form

$$\mathbf{x}_{j,t+1} = \mathcal{P}_{X_j}\Big[\mathbf{x}_{j,t} - \epsilon\Big(2\mathbf{H}_{j,t}^T\big(\mathbf{H}_{j,t}\mathbf{x}_{j,t} - \mathbf{y}_{j,t}\big) + \sum_{k \in n_j} (\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t})\Big)\Big]. \tag{19}$$

As a second application consider the SVM classification problem of Section II-B. In this case the functions $f_{i,t}$ are given in (10) and the specific form of (17) is case the functions $f_{i,t}$ are given in (10) and the specific form of (17) is

$$\mathbf{x}_{j,t+1} = \mathcal{P}_{X_j}\Big[\mathbf{x}_{j,t} - \epsilon\Big(\zeta\mathbf{x}_{j,t} - y_{i,t}\mathbf{z}_{i,t}\mathbb{I}\big(y_{i,t}\tilde{\mathbf{x}}^T\mathbf{z}_{i,t} < 1\big) + \sum_{k \in n_j} (\boldsymbol{\lambda}_{jk,t} - \boldsymbol{\lambda}_{kj,t})\Big)\Big], \tag{20}$$

where $\mathbb{I}\big(y_{i,t}\tilde{\mathbf{x}}^T\mathbf{z}_{i,t} < 1\big) = 1$ when $y_{i,t}\tilde{\mathbf{x}}^T\mathbf{z}_{i,t} < 1$ and $\mathbb{I}\big(y_{i,t}\tilde{\mathbf{x}}^T\mathbf{z}_{i,t} < 1\big) = 0$ otherwise. The conditional subtraction in the third term on the right hand side of (20) comes from the computation of the subgradient of the hinge loss, and moves

the current classifier in the direction of mistaken feature vectors weighted by the label. This update may be interpreted as a projected version of the Perceptron algorithm [24], [25] with a dual correction term that incorporates side information about neighbors' classifiers. For both, RLS and SVM, the dual iteration is as given in (18) because the form of this update is independent of the specific form of the cost functions $f_{i,t}$.

Remark 1 Recursive application of the primal and dual iterations in (13)-(14), or, equivalently, (17)-(18), would result in the minimization of the instantaneous global cost $\sum_{i=1}^{N} f_{i,t}(\mathbf{x}_{i,t})$ subject to the agreement constraint $\mathbf{C}\mathbf{x}_t = \mathbf{0}$. However, (13) and (14) are applied only once for each online Lagrangian and, moreover, this instantantaneous minimization is *not* the optimization problem that specifies the optimal action $\tilde{\mathbf{x}}^*$ which we defined as the minimizer of the accumulated cost $\sum_{t=1}^{T}\sum_{i=1}^{N} f_{i,t}(\tilde{\mathbf{x}})$. In fact, the variables $\mathbf{x}_{t+1}$ are obtained upon descending on the online Lagrangian $\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)$ associated with the functions $f_{i,t}$ – that are observed at time t – but their contribution to the regrets in (6) and (7) is determined by the functions $f_{i,t+1}$ – which are to be observed after playing $\mathbf{x}_{t+1}$ at time $t+1$. It is thus not obvious that (13)-(14) is a viable strategy to control regret, even though it will turn out to be so under mild assumptions; see Section IV. The justification for the use of these iterations comes from modeling the functions $f_{i,t}$ as drawn from a stationary distribution. This renders the problem of regret minimization equivalent to the solution of a stochastic optimization problem and (13)-(14) equivalent to a stochastic saddle point algorithm. In general, methods that work in stochastic optimization tend to work for regret minimization. Do observe, however, that no stochastic model is assumed in this paper. The functions $f_{i,t}$ are arbitrary.

IV. Regret Bounds

We turn to establishing that the local and global network regrets in (6) and (7) associated with plays $\mathbf{x}_{j,t}$ generated by the saddle point algorithm in (13)-(14) grow not faster than $O(\sqrt{T})$. In order to obtain these results, some conditions are required of the primal and dual variables, cost functions, and network. We state these assumptions below.

Assumption 1 *The network $\mathcal{G}$ is connected. The smallest nonzero singular value of the incidence matrix $\mathbf{C}$ is γ, the largest singular value is Γ, and the network diameter is D.*

Assumption 2 *The gradients of the loss functions for any $\mathbf{x}$ is bounded by a constant L, i.e.*

$$\|\nabla f_t(\mathbf{x})\| \leq L. \tag{21}$$

Assumption 3 *The loss functions $f_{i,t}(\mathbf{x})$ are Lipschitz continuous with modulus $K_{i,t} \leq K$,*

$$\|f_{i,t}(\mathbf{x}) - f_{i,t}(\mathbf{y})\| \;\leq\; K_{i,t}\|\mathbf{x}-\mathbf{y}\| \;\leq\; K\|\mathbf{x}-\mathbf{y}\|. \tag{22}$$

Assumption 4 *The set X of feasible plays is included in the 2-norm ball of radius $C_{\mathbf{x}}/N$.*

$$X \;\subseteq\; \big\{\tilde{\mathbf{x}} \in \mathbb{R}^p : \|\tilde{\mathbf{x}}\| \leq C_{\mathbf{x}}/N\big\}. \tag{23}$$

Assumption 5 *The convex set Λ_{jk} onto which the dual variables $\boldsymbol{\lambda}_{jk,t}$ are projected is included in a 1-norm ball of radius $C_{\boldsymbol{\lambda}}$,*

$$\Lambda_{jk} \;\subseteq\; \big\{\boldsymbol{\lambda} \in \mathbb{R}^p : \|\boldsymbol{\lambda}\|_1 \;\leq C_{\boldsymbol{\lambda}}\big\}, \tag{24}$$

for some constant $C_{\boldsymbol{\lambda}} \geq DNK + 1$.

Assumption 1 is standard in distributed algorithms. Assumptions 2 and 3 are typical in the analysis of saddle point algorithms. The bounds on the sets X and Λ_{jk} in assumptions 4 and 5 are constructed so that the iterates $\mathbf{x}_{j,t}$ and $\boldsymbol{\lambda}_{jk,t}$ are bounded by the respective constants in (23) and (24). The constant $C_{\mathbf{x}}/N$ in Assumption 4 is chosen so that the 2-norm of the stacked primal iterates $\mathbf{x}_t := [\mathbf{x}_{1,t}; \ldots; \mathbf{x}_{N,t}]$ are bounded as $\|\mathbf{x}_t\| \leq C_{\mathbf{x}}$.

The various bounds in Assumptions 1 - 5 permit bounding the norm of the gradients of the online Lagrangians in (12). For the gradient with respect to the primal variable $\mathbf{x}$, use of the triangle and Cauchy-Schwarz inequalities yields

$$\|\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)\| = \|\nabla f_t(\mathbf{x}_t) + \mathbf{C}^T\boldsymbol{\lambda}_t\| \leq \|\nabla f_t(\mathbf{x}_t)\| + \|\mathbf{C}^T\|\|\boldsymbol{\lambda}_t\|. \tag{25}$$

Use now the bounds in (21) and (24) and the definition of Γ as the largest singular value of $\mathbf{C}$ to simplify (25) to

$$\|\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)\| \leq L + \Gamma\sqrt{M}C_{\boldsymbol{\lambda}} := L_{\mathbf{x}}, \tag{26}$$

where we defined $L_{\mathbf{x}}$ for future reference. For the gradient with respect to the dual variable $\boldsymbol{\lambda}$, we can similarly write

$$\|\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)\| \;=\; \|\mathbf{C}\mathbf{x}_t\| \leq \|\mathbf{C}\|\|\mathbf{x}_t\| \;\leq\; \Gamma C_{\mathbf{x}} \;:=\; L_{\boldsymbol{\lambda}}. \tag{27}$$

Our results concerning local and global networked regret are both derived from the following lemma that simultaneously bounds the uncoordinated regret in (5) and the weighted penalty disagreement $\sum_{t=1}^{T}\boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t$ as we formally state next.

Lemma 1 *Consider the sequence $\mathbf{x}_t := [\mathbf{x}_{1,t}; \ldots; \mathbf{x}_{N,t}]$ generated by the saddle point algorithm in* (17)-(18)*. Let $\tilde{\mathbf{x}}^*$ be the optimal offline action in* (6)*, assume $\boldsymbol{\lambda}_1 = \mathbf{0}$ and further assume that assumptions 1 - 5 hold. If we select $\epsilon = 1/\sqrt{T}$ we have that for all $\boldsymbol{\lambda} \in \Lambda$ it holds*

$$\sum_{t=1}^{T}\sum_{i=1}^{N}\Big[f_{i,t}(\mathbf{x}_{i,t}) - f_{i,t}(\tilde{\mathbf{x}}^*)\Big] + \sum_{t=1}^{T}\boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t \leq \frac{\sqrt{T}}{2}\left(\|\mathbf{x}_1 - \tilde{\mathbf{x}}^*\|^2 + \|\boldsymbol{\lambda}\|^2 + L_{\mathbf{x}}^2 + L_{\boldsymbol{\lambda}}^2\right). \tag{28}$$

Proof: The proof is broken up into three parts. In the first part, we use the definition of the saddle point primal iterate and the first order characterization of convexity to bound the difference between the current algorithmic choice $\mathbf{x}_t$ and an arbitrary $\mathbf{x} \in X$. In the second, we mirror the first step in the dual variable $\boldsymbol{\lambda}$. We wrap up by combining the bounds obtained in the previous two steps, summing over time and using feasibility and boundedness properties to simplify expressions.

Begin then by considering the squared 2-norm of the difference between the iterate $\mathbf{x}_{t+1}$ at time $t+1$ and an arbitrary point $\mathbf{x} \in X$ and use (13) to express $\mathbf{x}_{t+1}$ in terms of $\mathbf{x}_t$,

$$\|\mathbf{x}_{t+1} - \mathbf{x}\|^2 = \|\mathcal{P}_X[\mathbf{x}_t - \epsilon\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)] - \mathbf{x}\|^2. \tag{29}$$

Since $\mathbf{x} \in X$ the distance between the projected vector $\mathcal{P}_X[\mathbf{x}_t - \epsilon\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)]$ and $\mathbf{x}$ is smaller than the distance before projection. Use this fact in (29) and expand the square to write

$$\begin{aligned}\|\mathbf{x}_{t+1} - \mathbf{x}\|^2 \;&\leq\; \|\mathbf{x}_t - \epsilon\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t) - \mathbf{x}\|^2 \\ &= \|\mathbf{x}_t - \mathbf{x}\|^2 - 2\epsilon\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t, \lambda_t)^T(\mathbf{x}_t - \mathbf{x}) \\ &\quad + \epsilon^2\|\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)\|^2.\end{aligned} \tag{30}$$

Further note that as stated in (26) the norm of the primal gradient of the online Lagrangian is bounded by $L_{\mathbf{x}}$. Substitute this bound for the corresponding term in (30) and reorder terms to write

$$\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)^T(\mathbf{x}_t-\mathbf{x}) \leq \frac{1}{2\epsilon}\left(\|\mathbf{x}_t-\mathbf{x}\|^2-\|\mathbf{x}_{t+1}-\mathbf{x}\|^2\right)+\frac{\epsilon L_{\mathbf{x}}^2}{2}. \tag{31}$$

Observe now that since the functions $f_{i,t}(\mathbf{x}_i)$ are convex, the online Lagrangian is a convex function of $\mathbf{x}$ [cf. (12)]. Thus, it follows from the first order convexity condition that

$$\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)-\mathcal{O}_t(\mathbf{x},\boldsymbol{\lambda}_t)\leq\nabla_{\mathbf{x}}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)^T(\mathbf{x}_t-\mathbf{x}). \tag{32}$$

Substituting the upper bound in (31) for the right hand side of the inequality in (32) yields

$$\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)-\mathcal{O}_t(\mathbf{x},\boldsymbol{\lambda}_t) \leq \frac{1}{2\epsilon}\left(\|\mathbf{x}_t-\mathbf{x}\|^2-\|\mathbf{x}_{t+1}-\mathbf{x}\|^2\right)+\frac{\epsilon L_{\mathbf{x}}^2}{2}. \tag{33}$$

We set this analysis aside and proceed to repeat the steps in (29)-(33) for the distance between the iterate $\boldsymbol{\lambda}_{t+1}$ at time $t+1$ and an arbitrary multiplier $\boldsymbol{\lambda}$.

$$\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2=\|\mathcal{P}_{\Lambda}[\boldsymbol{\lambda}_t+\epsilon\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)]-\boldsymbol{\lambda}\|^2, \tag{34}$$

where we have substituted (14) to express $\boldsymbol{\lambda}_{t+1}$ in terms of $\boldsymbol{\lambda}_t$. Using the non-expansive property of the projection operator in (34) and expanding the square, we obtain

$$\begin{aligned}\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2 &\leq \|\boldsymbol{\lambda}_t+\epsilon\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)-\boldsymbol{\lambda}\|^2. \\ &= \|\boldsymbol{\lambda}_t-\boldsymbol{\lambda}\|^2+2\epsilon\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{x}_t,\lambda_t)^T(\boldsymbol{\lambda}_t-\boldsymbol{\lambda}) \\ &\quad+\epsilon^2\|\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)\|^2.\end{aligned} \tag{35}$$

Now we reorder terms and substitute the bound $L_{\boldsymbol{\lambda}}$ for the norm of the dual subgradient of the online Lagrangian given in (27) to write

$$\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)^T(\boldsymbol{\lambda}_t-\boldsymbol{\lambda}) \geq \frac{1}{2\epsilon}\left(\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2-\|\boldsymbol{\lambda}_t-\boldsymbol{\lambda}\|^2\right)-\frac{\epsilon}{2}L_{\boldsymbol{\lambda}}^2. \tag{36}$$

Note that the online Lagrangian [cf. (12)] is a linear function of its Lagrange multipliers, which implies that online Lagrangian differences for fixed $\mathbf{x}_t$ satisfy

$$\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)-\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda})\geq\nabla_{\boldsymbol{\lambda}}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)^T(\boldsymbol{\lambda}_t-\boldsymbol{\lambda}). \tag{37}$$

Substitute the lower bound (36) into the right hand side of (37) to obtain

$$\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)-\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}) \geq \frac{1}{2\epsilon}\left(\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2-\|\boldsymbol{\lambda}_t-\boldsymbol{\lambda}\|^2\right)-\frac{\epsilon}{2}\ L_{\boldsymbol{\lambda}}^2. \tag{38}$$

We now turn to combining the bounds in (33) and (38). To do so observe that the term $\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)$ appears in both inequalities. Thus, subtraction of the terms in inequality (38) from those in (33) followed by reordering terms yields

$$\begin{aligned}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda})-\mathcal{O}_t(\mathbf{x},\boldsymbol{\lambda}_t) &\leq \frac{1}{2\epsilon}\big(\|\mathbf{x}_t-\mathbf{x}\|^2-\|\mathbf{x}_{t+1}-\mathbf{x}\|^2 \\ &\quad+\|\boldsymbol{\lambda}_t-\boldsymbol{\lambda}\|^2-\|\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}\|^2\big)+\frac{\epsilon}{2}\left(L_{\mathbf{x}}^2+L_{\boldsymbol{\lambda}}^2\right).\end{aligned} \tag{39}$$

Now sum (39) over time to write

$$\sum_{t=1}^{T}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda})-\mathcal{O}_t(\mathbf{x},\boldsymbol{\lambda}_t) \leq \frac{1}{2\epsilon}\left(\|\mathbf{x}_1-\mathbf{x}\|^2+\|\boldsymbol{\lambda}\|^2\right)+\frac{\epsilon}{2}T(L_{\mathbf{x}}^2+L_{\boldsymbol{\lambda}}^2). \tag{40}$$

Here we have used the telescopic property of the summand on the right hand side of (40) and omitted the subtraction of the nonnegative quantity $\|\boldsymbol{\lambda}_T-\boldsymbol{\lambda}\|^2$. Using the explicit expression for the online Lagrangian in (12) we can write the online Lagrangian difference on the left side of (39) as

$$\begin{aligned}\mathcal{O}_t(\mathbf{x}_t,\boldsymbol{\lambda})-\mathcal{O}_t(\mathbf{x},\boldsymbol{\lambda}_t) &= \sum_{i=1}^{N}f_{i,t}(\mathbf{x}_{i,t})+\boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t-\sum_{i=1}^{N}f_{i,t}(\mathbf{x})-\boldsymbol{\lambda}_t^T\mathbf{C}\mathbf{x}.\end{aligned} \tag{41}$$

Let now $\mathbf{x}$ be an arbitrary feasible point for the coordinated regret game, i.e., one for which $\mathbf{x}_i=\mathbf{x}_j$ for all i and j, or, equivalently, one for which $\mathbf{C}\mathbf{x}=\mathbf{0}$. For these feasible points the last term in (41) vanishes. Substituting the resulting expression for the left hand side of (40) yields, after reordering terms,

$$\begin{aligned}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(f_{i,t}(\mathbf{x}_{i,t})-f_{i,t}(\mathbf{x})\right)+\sum_{t=1}^{T}\boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t \\ \leq \frac{1}{2\epsilon}\left(\|\mathbf{x}_1-\mathbf{x}\|^2+\|\boldsymbol{\lambda}\|^2\right)+\frac{\epsilon}{2}T(L_{\mathbf{x}}^2+L_{\boldsymbol{\lambda}}^2),\end{aligned} \tag{42}$$

for arbitrary feasible point $\mathbf{x}$ satisfying $\mathbf{C}\mathbf{x}=\mathbf{0}$. The bound in (42) holds for $\tilde{\mathbf{x}}^*$ because $\tilde{\mathbf{x}}^*$ is optimal for coordinated regret – thus feasible, in particular. The result in (28) follows by making $\mathbf{x}=\tilde{\mathbf{x}}^*$ and $\epsilon=1/\sqrt{T}$ in (42).

■

From Lemma 1 we obtain a bound for the uncoordinated regret $\mathbf{Reg}_T^{\mathrm{U}}$ defined in (5). To do so simply note that $\boldsymbol{\lambda}=\mathbf{0}$ belongs to the set Λ. Using this particular value of $\boldsymbol{\lambda}$ in (28) yields

$$\begin{aligned}\mathbf{Reg}_T^{\mathrm{U}} &= \sum_{t=1}^{T}\sum_{j=1}^{N}f_{j,t}(\mathbf{x}_{j,t})-\sum_{t=1}^{T}\sum_{i=1}^{N}f_{j,t}(\tilde{\mathbf{x}}^*) \\ &\leq \frac{\sqrt{T}}{2}\left(\|\mathbf{x}_1-\mathbf{x}\|^2+L_{\mathbf{x}}^2+L_{\boldsymbol{\lambda}}^2\right).\end{aligned} \tag{43}$$

This bound is of little use because, as we mentioned in Section II, agents can reduce uncoordinated regret by just operating independently of each other. Observe, however, that the relationship in (28) also includes the weighted penalty disagreement $\sum_{t=1}^{T}\boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t$. The presence of this term indicates that the actions of different users can't be too different and that it should be possible to relate global networked regret to uncoordinated regret. This is indeed possible and leads to the regret bound that we introduce in the following theorem.

Theorem 1 *Let $\mathbf{x}_t:=[\mathbf{x}_{1,t};\ldots;\mathbf{x}_{N,t}]$ denote the sequence generated by the saddle point algorithm in* (17)-(18) *and let $\tilde{\mathbf{x}}^*$ be the optimal offline action in* (6)*. If Assumptions 1-5 hold, with the initialization $\boldsymbol{\lambda}_1=\mathbf{0}$ and step size $\epsilon=1/\sqrt{T}$, the global network regret [cf.* (7)*] is bounded by*

$$\mathbf{Reg}_T \ \leq\ \frac{\sqrt{T}}{2}\left(\|\mathbf{x}_1-\tilde{\mathbf{x}}^*\|^2+MC_{\boldsymbol{\lambda}}^2+L_{\mathbf{x}}^2+L_{\boldsymbol{\lambda}}^2\right)\ =\ O(\sqrt{T}). \tag{44}$$

Proof: We begin by writing the expression for $\mathbf{Reg}_T$, and add and subtract the left hand side of (28), the uncoordinated regret plus a constraint slack penalizing node disagreement to write

$$\begin{aligned}\mathbf{Reg}_T &= \sum_{t=1}^{T}\frac{1}{N}\sum_{j,k=1}^{N} f_{k,t}(\mathbf{x}_{j,t}) - \sum_{t=1}^{T}\sum_{k=1}^{N} f_{k,t}(\tilde{\mathbf{x}}^*) \qquad (45)\\ &= \sum_{t=1}^{T}\left(\frac{1}{N}\sum_{j,k=1}^{N} f_{k,t}(\mathbf{x}_{j,t}) - \sum_{k=1}^{N} f_{k,t}(\mathbf{x}_{k,t}) - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t\right)\\ &\quad + \sum_{t=1}^{T}\left(\sum_{k=1}^{N} f_{k,t}(\mathbf{x}_{k,t}) - \sum_{k=1}^{N} f_{k,t}(\tilde{\mathbf{x}}^*) + \boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t\right).\end{aligned}$$

The second time summation on the right side of (45) may be bounded with Lemma 1. Thus we turn to providing an upper estimate of the first sum. Assumption 3 regarding the Lipschitz continuity of the loss functions implies

$$\sum_{j,k=1}^{N}\Big[f_{k,t}(\mathbf{x}_{j,t}) - f_{k,t}(\mathbf{x}_{k,t})\Big] \leq \sum_{j,k=1}^{N} K_{k,t}\|\mathbf{x}_{j,t} - \mathbf{x}_{k,t}\|. \quad (46)$$

Maximize over the right hand side of (46) to obtain an expression for the magnitude of the worst case node discrepancy

$$\sum_{j,k=1}^{N} K_{k,t}\|\mathbf{x}_{j,t} - \mathbf{x}_{k,t}\| \leq N^2 K \max_{j,k}\|\mathbf{x}_{j,t} - \mathbf{x}_{k,t}\|. \quad (47)$$

Using Assumption 1 regarding the diameter of the network, the worst case node discrepancy on the right hand side of (47) may be further bounded above by the magnitude of the constraint violation as $\max_{j,k}\|\mathbf{x}_{j,t} - \mathbf{x}_{k,t}\| \leq D\|\mathbf{C}\mathbf{x}_t\|$. Substituting this bound into (47) yields

$$\sum_{j,k=1}^{N}\Big[f_{k,t}(\mathbf{x}_{j,t}) - f_{k,t}(\mathbf{x}_{k,t})\Big] \leq DN^2K\|\mathbf{C}\mathbf{x}_t\|. \quad (48)$$

We return to bounding the first sum in the right hand side of (45). To do so write $\sum_{k=1}^{N} f_{k,t}(\mathbf{x}_{k,t}) = (1/N)\sum_{j,k=1}^{N} f_{k,t}(\mathbf{x}_{k,t})$, which we can do because $f_{k,t}(\mathbf{x}_{k,t})$ is independent of j. Use this to substitute $(1/N)\sum_{j,k=1}^{N} f_{k,t}(\mathbf{x}_{j,t}) - \sum_{k=1}^{N} f_{k,t}(\mathbf{x}_{k,t})$ for the bound in (48) to write

$$\begin{aligned}&\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{j,k=1}^{N}\Big[f_{k,t}(\mathbf{x}_{j,t}) - f_{k,t}(\mathbf{x}_{k,t})\Big] - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t\right) \qquad (49)\\ &\qquad\leq \sum_{t=1}^{T}\Big(DNK\|\mathbf{C}\mathbf{x}_t\| - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t\Big)\\ &\qquad= \sum_{t=1}^{T}\left(DNK\frac{\mathbf{C}\mathbf{x}_t}{\|\mathbf{C}\mathbf{x}_t\|} - \boldsymbol{\lambda}\right)^T\mathbf{C}\mathbf{x}_t.\end{aligned}$$

where the last equality follows from grouping terms. The difference between node losses evaluated at other nodes' predictions and their own is bounded by the magnitude of the constraint violation and a Lagrangian penalty term. We annihilate the right hand side of (49) by constructing a dual feasible $\tilde{\boldsymbol{\lambda}}$ as follows. Partition the edge set $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ with $\mathcal{E}^+ = \{e : [\sum_{t=1}^{T}\mathbf{C}[\mathbf{x}_t]_e \geq \mathbf{0}\}$. and $\mathcal{E}^- = \{e : [\sum_{t=1}^{T}[\mathbf{C}\mathbf{x}_t]_e < \mathbf{0}\}$. Define $\tilde{\boldsymbol{\lambda}}$ as

$$\tilde{\boldsymbol{\lambda}}_e = \begin{cases} DNK[\mathbf{C}\mathbf{x}_t]_e/\|\mathbf{C}\mathbf{x}_t\| + 1 & \text{for } e \in \mathcal{E}^+ \text{ and all } t\\ DNK[\mathbf{C}\mathbf{x}_t]_e/\|\mathbf{C}\mathbf{x}_t\| - 1 & \text{for } e \in \mathcal{E}^- \text{ and all } t\end{cases} \quad (50)$$

Our ability to construct such a fixed finite $\tilde{\boldsymbol{\lambda}}$ follows from the compactness of X and hence the boundedness of $\|\mathbf{C}\mathbf{x}_t\|$. Note that

$$\begin{aligned}\|\tilde{\boldsymbol{\lambda}}\| &\leq \sqrt{|\mathcal{E}^+|(DNK+1)^2 + |\mathcal{E}^-|(DNK-1)^2} \qquad (51)\\ &\leq \sqrt{M}(DNK+1) \leq \sqrt{M}C_{\boldsymbol{\lambda}}.\end{aligned}$$

so $\tilde{\boldsymbol{\lambda}}$ is dual feasible in the sense of (24). The first inequality in (51) follows from the fact that computing $\|\tilde{\boldsymbol{\lambda}}\|$ is a sum over the entries of a unit vector, while the second inequality uses the relationship $|\mathcal{E}^+| - |\mathcal{E}^-| \leq M$. Now plug $\boldsymbol{\lambda} = \tilde{\boldsymbol{\lambda}}$ into (49) to write

$$\begin{aligned}&\sum_{t=1}^{T}\sum_{e\in\mathcal{E}^+}\left(DNK\frac{[\mathbf{C}\mathbf{x}_t]_e}{\|\mathbf{C}\mathbf{x}_t\|} - DNK\frac{[\mathbf{C}\mathbf{x}_t]_e}{\|\mathbf{C}\mathbf{x}_t\|} - 1\right)[\mathbf{C}\mathbf{x}_t]_e \qquad (52)\\ &\quad+\sum_{t=1}^{T}\sum_{e\in\mathcal{E}^-}\left(DNK\frac{[\mathbf{C}\mathbf{x}_t]_e}{\|\mathbf{C}\mathbf{x}_t\|} - DNK\frac{[\mathbf{C}\mathbf{x}_t]_e}{\|\mathbf{C}\mathbf{x}_t\|} + 1\right)[\mathbf{C}\mathbf{x}_t]_e\\ &= \sum_{t=1}^{T}\left(\sum_{e\in\mathcal{E}^+} -[\mathbf{C}\mathbf{x}_t]_e + \sum_{e\in\mathcal{E}^-}[\mathbf{C}\mathbf{x}_t]_e\right) = 0.\end{aligned}$$

With the dual variable selection given by (50), we have made first three terms on the right hand side of (45) null. Now apply Lemma 1 to the last three terms on the right hand side of (45) and substitute in the bound for the magnitude of $\tilde{\boldsymbol{\lambda}}$ in (51), which allows us to conclude (44). ■

Theorem 1 provides a guarantee that the saddle point iterates achieve a global networked regret that grows not faster than $O(\sqrt{T})$. This rate is the same that can be guaranteed in centralized problems when functions are not strongly convex. The learning rate depends on primal initialization, network size and topology, as well as smoothness properties of the loss functions. The learning rate result established in Theorem 1 is a bound on the global networked regret which is the average the local regrets incurred by each agent. By relating the uncoordinated regret bound in (43) with the local regret defined in (6) we obtain a similar bound on the regret of each individual agent as we formally state next.

Theorem 2 *Let $\mathbf{x}_t := [\mathbf{x}_{1,t}; \ldots; \mathbf{x}_{N,t}]$ be the sequence generated by the saddle point algorithm in* (17)-(18) *and let $\tilde{\mathbf{x}}^*$ be the global batch learner in* (6)*. If Assumptions 1-5 hold, with the initialization $\boldsymbol{\lambda}_1 = \mathbf{0}$ and step size $\epsilon = 1/\sqrt{T}$, the local regret of node j [cf.* (6)*] is bounded by*

$$\mathbf{Reg}_T^j \;\leq\; \frac{\sqrt{T}}{2}\big(\|\mathbf{x}_1 - \tilde{\mathbf{x}}^*\|^2 + MC_{\boldsymbol{\lambda}}^2 + L_{\mathbf{x}}^2 + L_{\boldsymbol{\lambda}}^2\big) \;=\; O(\sqrt{T}). \quad (53)$$

Proof: Begin writing the expression for local regret of node j and add and subtract the left hand side of (43).

$$\begin{aligned}\mathbf{Reg}_T^j &= \sum_{t=1}^{T}\sum_{k=1}^{N} f_{k,t}(\mathbf{x}_{j,t}) - \sum_{t=1}^{T}\sum_{k=1}^{N} f_{k,t}(\tilde{\mathbf{x}}^*) \qquad (54)\\ &= \sum_{t=1}^{T}\left(\sum_{k=1}^{N}[f_{k,t}(\mathbf{x}_{j,t}) - f_{k,t}(\mathbf{x}_{k,t})] - \boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t\right)\\ &\quad+\sum_{t=1}^{T}\left(\sum_{k=1}^{N}[f_{k,t}(\mathbf{x}_{k,t}) - f_{k,t}(\tilde{\mathbf{x}}^*)] + \boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t\right)\end{aligned}$$

The last three terms of (54) were bounded in Lemma 1, so we turn our focus to the first tree terms in an analogous manner to

the proof of Theorem 1. The Lipschitz continuity of the losses in Assumption 3 yields

$$\sum_{k=1}^{N}\left(f_{k,t}(\mathbf{x}_{j,t})-f_{k,t}(\mathbf{x}_{k,t})\right)\leq\sum_{k=1}^{N}K_{k,t}\|\mathbf{x}_{j,t}-\mathbf{x}_{k,t}\|. \tag{55}$$

Now, maximize over the right hand side of (55) to write an expression for the maximum difference between node predictions

$$\sum_{k=1}^{N}K_{k,t}\|\mathbf{x}_{j,t}-\mathbf{x}_{k,t}\|\leq NK\max_{k}\|\mathbf{x}_{j,t}-\mathbf{x}_{k,t}\|. \tag{56}$$

The quantity on the right hand side of (56) can be expressed in terms of the magnitude of the constraint violation. In particular, the definition of the diameter as the maximum of shortest paths between nodes combined with the triangle inequality allows us to write

$$NK\max_{k}\|\mathbf{x}_{j,t}-\mathbf{x}_{k,t}\|\leq DNK\|\mathbf{C}\mathbf{x}_t\|. \tag{57}$$

We substitute in the right hand side of (57) into (55), and apply the resulting inequality to the first three terms on the right hand side of (54) to obtain

$$\begin{aligned}\sum_{t=1}^{T}\left(\sum_{k=1}^{N}\left[f_{k,t}(\mathbf{x}_{j,t})-f_{k,t}(\mathbf{x}_{k,t})\right]-\boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t\right) \\ \leq\sum_{t=1}^{T}\left(DNK\|\mathbf{C}\mathbf{x}_t\|-\boldsymbol{\lambda}^T\mathbf{C}\mathbf{x}_t\right) \\ =\sum_{t=1}^{T}\left(DNK\frac{\mathbf{C}\mathbf{x}_t}{\|\mathbf{C}\mathbf{x}_t\|}-\boldsymbol{\lambda}\right)^T\mathbf{C}\mathbf{x}_t.\end{aligned} \tag{58}$$

Using $\boldsymbol{\lambda}=\tilde{\boldsymbol{\lambda}}$ defined in (50), we make the right hand side of (58) null in precisely the same manner as (52). Returning to the first three terms of (54), we apply Lemma 1 and substitute in the expression for the magnitude of $\tilde{\boldsymbol{\lambda}}$ in (51) to yield (53). ■

Theorem 2 establishes that the local regret of each individual agent in the network grows at a rate not larger than $O(\sqrt{T})$, which is equivalent to saying that its time average vanishes as $O(1/\sqrt{T})$. It follows that individuals learn global information while only having access to local observations and the strategies of neighboring agents. The constants that bound the regret growth depend on the initial condition, network connectivity, and properties of the loss functions.

V. Numerical Analysis

We study the numerical behavior of the saddle point algorithm in (17)-(18) when used to solve the distributed recursive least squares problem in Section II-A for a variety of network sizes, topologies, and levels of connectivity (sections V-A - V-C). We also investigate how saddle point iterates compare against other networked online learning methods (Section V-D). The primal iteration for recursive least squares is given by the explicit expression in (19). Besides the local and global network regrets in (6) and (7) that we know grow not faster than $O(\sqrt{T})$ [cf. theorems 1 and 2] we also study the relative error of the estimates $\mathbf{x}_{j,t}$ relative to the optimal batch estimator $\tilde{\mathbf{x}}^*$ and the relative agreement between estimates $\mathbf{x}_{j,t}$ and $\mathbf{x}_{k,t}$ of different agents. Specifically, the relative error associated with the estimate $\mathbf{x}_{j,t}$ of agent j at time t is defined as

$$\text{RE}(\mathbf{x}_{j,t}):=\frac{\|\mathbf{x}_{j,t}-\tilde{\mathbf{x}}^*\|}{\|\tilde{\mathbf{x}}^*\|}. \tag{59}$$

The agreement between estimates of different nodes is defined in terms of the variable time averages $\bar{\mathbf{x}}_{j,t}:=(1/t)\sum_{u=1}^{t}\mathbf{x}_{j,u}$. For the average estimate $\bar{\mathbf{x}}_{j,t}$ of agent j at time t we define the average relative variation as

$$\text{RV}(\bar{\mathbf{x}}_{j,t}):=\frac{1}{N}\sum_{k=1}^{N}\frac{\|\bar{\mathbf{x}}_{j,t}-\bar{\mathbf{x}}_{k,t}\|}{\|\tilde{\mathbf{x}}^*\|}. \tag{60}$$

The average relative variation $\text{RV}(\bar{\mathbf{x}}_{j,t})$ denotes the average Euclidean error between $\bar{\mathbf{x}}_{j,t}$ and all others, relative to the magnitude of the offline strategy $\tilde{\mathbf{x}}^*$. The reason to focus on time averages $\bar{\mathbf{x}}_{j,t}$ instead of the plain estimates $\mathbf{x}_{j,t}$ is that the latter tend to oscillate around the batch estimate $\tilde{\mathbf{x}}^*$ and agreement between estimates of different agents is difficult to visualize.

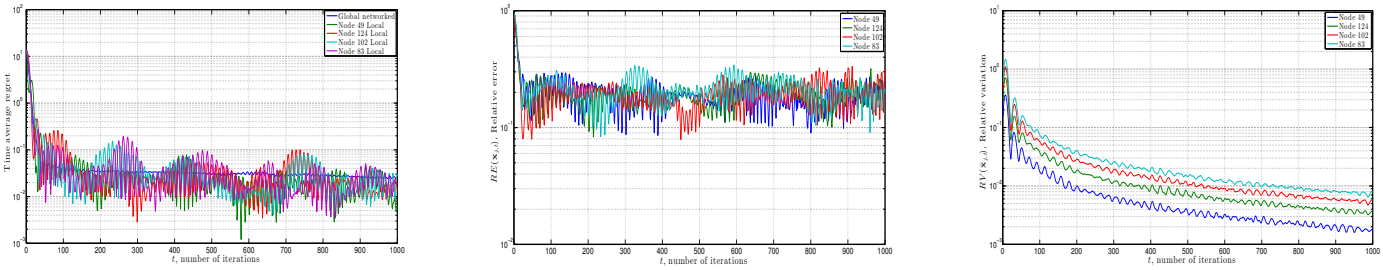
For all of the subsequent numerical experiments, we consider $q=1$ and $p=10$ – i.e., observations $\mathbf{y}_{it}=\mathbf{H}_{i,t}\tilde{\mathbf{x}}+\mathbf{w}_{i,t}$ are scalar and the signal $\tilde{\mathbf{x}}$ has dimension $p=10$. The matrices $\mathbf{H}_{i,t}=\mathbf{H}_i\in\mathbb{R}^{1\times p}$ are constant across time but vary across agents. The components of the vector $\mathbf{H}_i$ are chosen with equal probability from $\{1/p,2/p,\ldots,1\}$. The random noise terms $\mathbf{w}_{i,t}\in\mathbb{R}$ are Gaussian distributed with zero mean and variance $\sigma^2=0.1$ and the true signal is $\tilde{\mathbf{x}}=\mathbf{1}$. Further observe that since $q<p$ it is impossible to estimate $\tilde{\mathbf{x}}$ without cooperation between members of the network because the individual signals of each agent are not sufficient to determine $\tilde{\mathbf{x}}$. In all cases we run (19) - (18) for a total of $T=10^3$ iterations with step size $\epsilon=1/\sqrt{T}=0.03$. Initial iterates are chosen as $\mathbf{x}_{j,1}=\mathbf{0}$ for all j and $\boldsymbol{\lambda}_{jk,1}=\mathbf{0}$ for all j and k.

The trajectories of a sample run for a random network with $N=200$ nodes in which the probability of connecting two nodes is $\rho=0.2$ are shown in Figure 1. The time average of the global and local regrets, $\mathbf{Reg}_t/t$ and $\mathbf{Reg}_t^j/t$, respectively, for representative nodes are shown in Figure 1a. Observe that $\mathbf{Reg}_t/t$ decreases until $t\approx 200$ iterations and then stabilizes at $\mathbf{Reg}_t/t\approx 5\times 10^{-2}$. This is consistent with the result in Theorem 1 in which regret of order $O(\sqrt{T})$ is attained by selecting a stepsize of order $O(T)$. To obtain smaller regret values the algorithm has to be run with smaller stepsize. The same decline is observed for the average local regrets $\mathbf{Reg}_t^j/t$. The only difference is that the $\mathbf{Reg}_t^j/t$ exhibit oscillating variations as iterations progress. These are not present in $\mathbf{Reg}_t/t$ which averages values across the whole network.

Learning of the global batch strategy can be corroborated by reduction of the Euclidean distance to $\tilde{\mathbf{x}}^*$ at each node and the achievement of primal variable consensus. Figure 1b shows that the relative error $\text{RE}(\mathbf{x}_{j,t})$ declines with the iteration index t and stabilizes below 0.4 for $t\geq 100$, demonstrating that the former goal is achieved, though the noise in the observations yields persistent oscillations. Figure 1c plots $\text{RV}(\bar{\mathbf{x}}_{j,t})$ versus iteration t. Observe that agents also converge towards a common value, i.e. $\text{RV}(\bar{\mathbf{x}}_{j,t})\leq 10^{-2}$ for $t\geq 700$, as exchange of local information successfully allows agents to learn a globally optimal strategy.

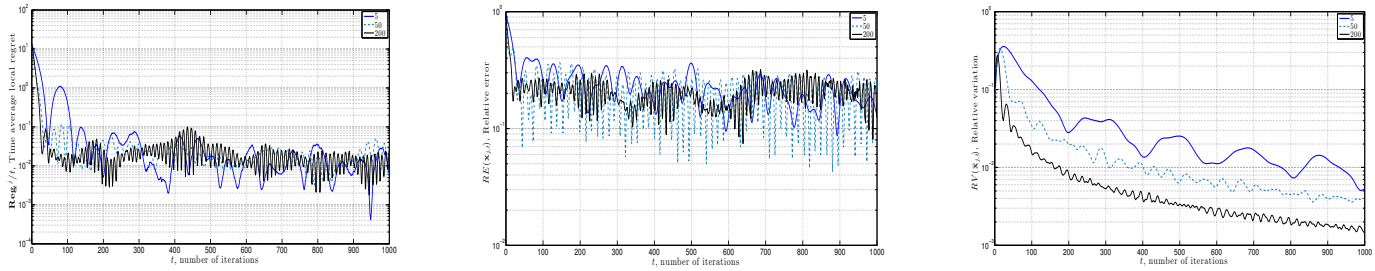
A. Network size

To investigate the dependence of the learning rates in theorems 1 and 2 with the network size N we run (19) - (18) for problem instances with $N=5$, $N=50$, and $N=200$ nodes. Connections between nodes are random, with the probability of two nodes being connected set to $\rho=0.2$. Figure 2 shows the results of this numerical experiment for an arbitrary agent in the network. In Figure 2a, we show $\mathbf{Reg}_t^j/t$ over iteration t. Observe that as N increases, $\mathbf{Reg}_t^j/t$ declines at comparable rates for the different



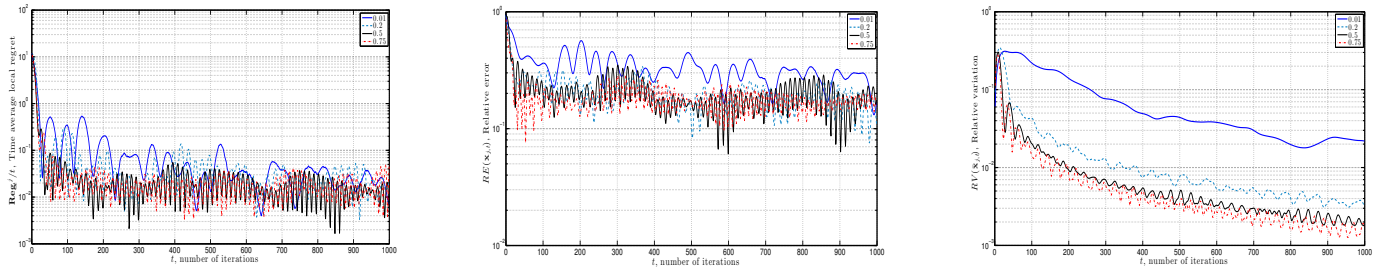
(a) Average regret $\mathbf{Reg}_t^i/t$ versus iteration t (b) Relative error $\mathrm{RE}(\mathbf{x}_{j,t})$ versus iteration t (c) Relative variation $\mathrm{RV}(\bar{\mathbf{x}}_{j,t})$ versus iteration t

Fig. 1: In a $N = 200$ node random network with connection probability $\rho = 0.2$, figures 1a-1b show average global networked regret $\mathbf{Reg}_T/T$ and local regrets $\mathbf{Reg}_t^j/t$ for a representative sample of agents, and $\mathrm{RE}(\mathbf{x}_{j,t})$, respectively, versus iteration t. Average regrets sharply decline and then stabilize, consistent with the regret bounds dependence on a fixed step size $\epsilon = 1/\sqrt{T}$. Decentralized online learning is corroborated by both distance to the global batch learner, measured by $\mathrm{RE}(\mathbf{x}_{j,t})$, decreasing, as shown in 1b, and consensus in the primal variable, which is shown in Figure 2c. In the later, we plot $\mathrm{RV}(\bar{\mathbf{x}}_{j,t})$ versus iteration t, which goes to null as agents learn all information available throughout the network.



(a) Average regret $\mathbf{Reg}_t^j/t$ versus iteration t (b) Relative error $\mathrm{RE}(\mathbf{x}_{j,t})$ versus iteration t (c) Relative variation $\mathrm{RV}(\bar{\mathbf{x}}_{j,t})$ versus iteration t

Fig. 2: Learning achieved by an arbitrary agent in networks of size $N = 5$, $N = 50$, and $N = 200$ with nodes randomly connected with prob. $\rho = 0.2$. 2a-2b show $\mathbf{Reg}_t^j/t$, the time average local regret, and $\mathrm{RE}(\mathbf{x}_{j,t})$, the relative error, respectively, as compared with iteration t. Both $\mathbf{Reg}_t^j/t$ and $\mathrm{RE}(\mathbf{x}_{j,t})$ decline sharply, but with more instability in smaller networks, and stabilize near 10^{-2} and 10^{-1}, respectively. Figure 2c shows $RV(\bar{\mathbf{x}}_{j,t})$ versus iteration t, and illustrate that node j's average prediction remain close to that of other nodes. Network disagreement becomes more stable and declines faster with increasing N, as information contained per individual required for learning the global batch strategy declines.



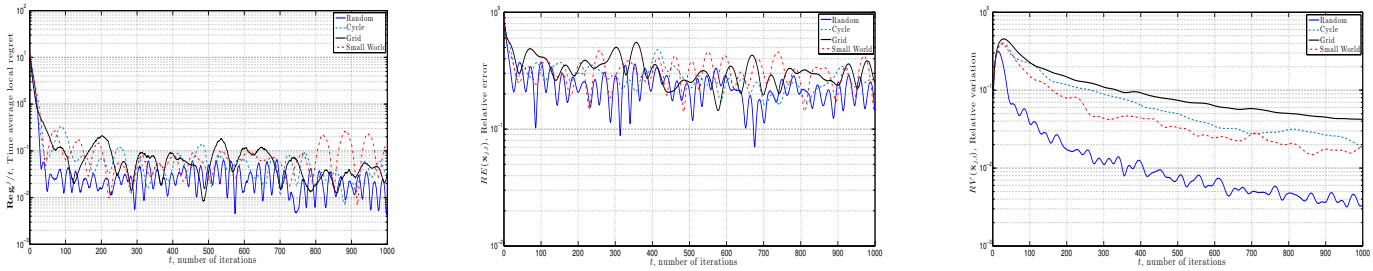
(a) Average regret $\mathbf{Reg}_t^j/t$ versus iteration t (b) Relative error $\mathrm{RE}(\mathbf{x}_{j,t})$ versus iteration t (c) Relative variation $\mathrm{RV}(\mathbf{x}_{j,t})$ versus iteration t

Fig. 3: Saddle point algorithm learning rates and discrepancy on a random $N = 50$ node network with connection probability $\rho \in \{0.01, 0.2, 0.5, 0.75\}$. Figure 3a-3b show $\mathbf{Reg}_T^j/T$, the time average local regret of an arbitrary node in the network, and $\mathrm{RE}(\mathbf{x}_{j,t})$, the relative error, respectively, as compared with iteration t. Both $\mathbf{Reg}_T^j/T$ and $\mathrm{RE}(\mathbf{x}_{j,t})$ are more oscillatory in less connected networks. Figure 3c shows $RV(\bar{\mathbf{x}}_{j,t})$ versus iteration t. Primal variable consensus is more difficult to achieve in networks with fewer communication links.

network sizes, and this rate similarity is also reflected in the trajectory of $\mathrm{RE}(\mathbf{x}_{j,t})$ over time t, as shown in Figure 2b. To reach the benchmark $\mathrm{RE}(\mathbf{x}_{j,t}) \leq 10^{-3}$ we require $t = 354$, $t = 279$, and $t = 232$ for $N = 5$, $N = 50$, and $N = 200$, respectively.

While learning occurs at comparable rates in the different networks, the trajectories are more oscillatory in smaller networks. To be specific, in Figure 2b. we note that to achieve $\mathrm{RE}(\mathbf{x}_{j,t}) \leq 0.2$, the algorithm requires $t = 106$, $t = 30$, and $t = 18$ iterations for $N = 5$, $N = 50$, and $N = 200$, respectively. Moreover, average wavelength of oscillations of $\mathrm{RE}(\mathbf{x}_{j,t})$ is $\tau = 50$, $\tau = 20$, and $\tau = 10$ for $N = 5$, $N = 50$, and $N = 200$, respectively. This stability difference reflects the fact that as N increases, the fraction of information per node contained in each agent's prediction decreases. Hence each dual variable must compensate for a larger relative level of discrepancy per communication link in smaller networks. Equivalently, for agents in larger networks to achieve comparable learning rates, information must diffuse faster.

Figure 2c shows that the network reaches consensus, as measured with $\mathrm{RV}(\bar{\mathbf{x}}_{j,t})$, faster with larger N. In particular, for the benchmark $\mathrm{RV}(\bar{\mathbf{x}}_{j,t}) \leq 10^{-2}$, the algorithm requires $t = 719$, $t = 317$, and $t = 179$ iterations for $N = 5$, $N = 50$, and $N = 200$ node networks, respectively. This suggests that the agreement constraint plays a larger role in maintaining a comparable learning rate in larger networks, as the amount of disagreement must



(a) Average regret $\mathbf{Reg}_t^j/t$ versus iteration t (b) Relative error $\text{RE}(\mathbf{x}_{j,t})$ versus iteration t (c) Relative variation $\text{RV}(\bar{\mathbf{x}}_{j,t})$ versus iteration t

Fig. 4: Saddle point algorithm run on $N = 50$ node cycle, grid, random and small world networks, where edges are generated randomly between agents with probability $\rho = 0.2$ in the later two. Model noise is sampled from $\mathbf{w}_{i,t} \sim \mathcal{N}(0, 0.1)$. Figure 4a-4b show $\mathbf{Reg}_T^j/T$ and $\text{RE}(\mathbf{x}_{j,t})$, respectively, over iteration t. Learning slows and numerical oscillations become more prevalent with increasing network diameter. Grid and cycle networks have larger diameter than small world and random networks, resulting in slower information propagation. Figure 4c shows that the agents reach consensus slower in terms of $\text{RV}(\bar{\mathbf{x}}_{j,t})$ with increasing network diameter.

be smaller for individuals to learn global information in larger networks.

B. Node connectivity

To understand the impact of network connectivity on algorithm performance we fix the network size to $N = 50$ and run (19) - (18) on random networks where the probability of connecting two nodes takes values $\rho \in \{0.01, 0.2, 0.5, 0.75\}$. Figure 3 shows the results of this experimental setup. Figure 3a depicts $\mathbf{Reg}_t^j/t$ versus iteration t, and illustrates that the difference in connectivity levels leads to a negligible difference in the learning rate. However, we see that numerical stability varies substantially. The sparsely connected networks experience more oscillatory behavior, as may be observed in the plot of of relative error versus iteration t in Figure 3b. This stability difference follows from the slower rate of information diffusion, and also coincides with slowing convergence to the batch strategy. Figure 3c shows the evolution of $\text{RV}(\bar{\mathbf{x}}_{j,t})$ over time. The achievement of primal variable consensus is more challenging in sparsely connected networks. That is, for the benchmark $\text{RV}(\bar{\mathbf{x}}_{j,t}) \leq 2 \times 10^{-2}$, the algorithm requires $t = 875$, $t = 183$, $t = 120$, and $t = 43$ iterations for the cases $\rho = 0.01$, $\rho = 0.2$, $\rho = 0.5$, and $\rho = 0.75$, respectively. Intuitively, the discrepancy in agents' predictions is smaller when more communication links are present in the network.

C. Topology and Diameter

To study the interplay of network topology and diameter on the learning rates established in Theorems 1 and 2, we fix the network size to $N = 50$ and run (19) - (18) over random graphs, small world graphs, cycles, and grids. In the first two, the probability that node pairs are randomly connected is fixed at $\rho = 0.2$. The latter two are deterministically generated. A cycle is a closed directed chain of nodes. Grids are formed by taking the two-dimensional integer lattice of size $\sqrt{N} \times \sqrt{N}$, with $\sqrt{N}$ rounded to the nearest integer. Connections are drawn between adjacent nodes in the lattice as well as between remainder nodes at the boundary. Cycles, grids and random networks have progressively larger number of connections per node and smaller diameter. Random networks have small degree and small diameter; see [26], [27].

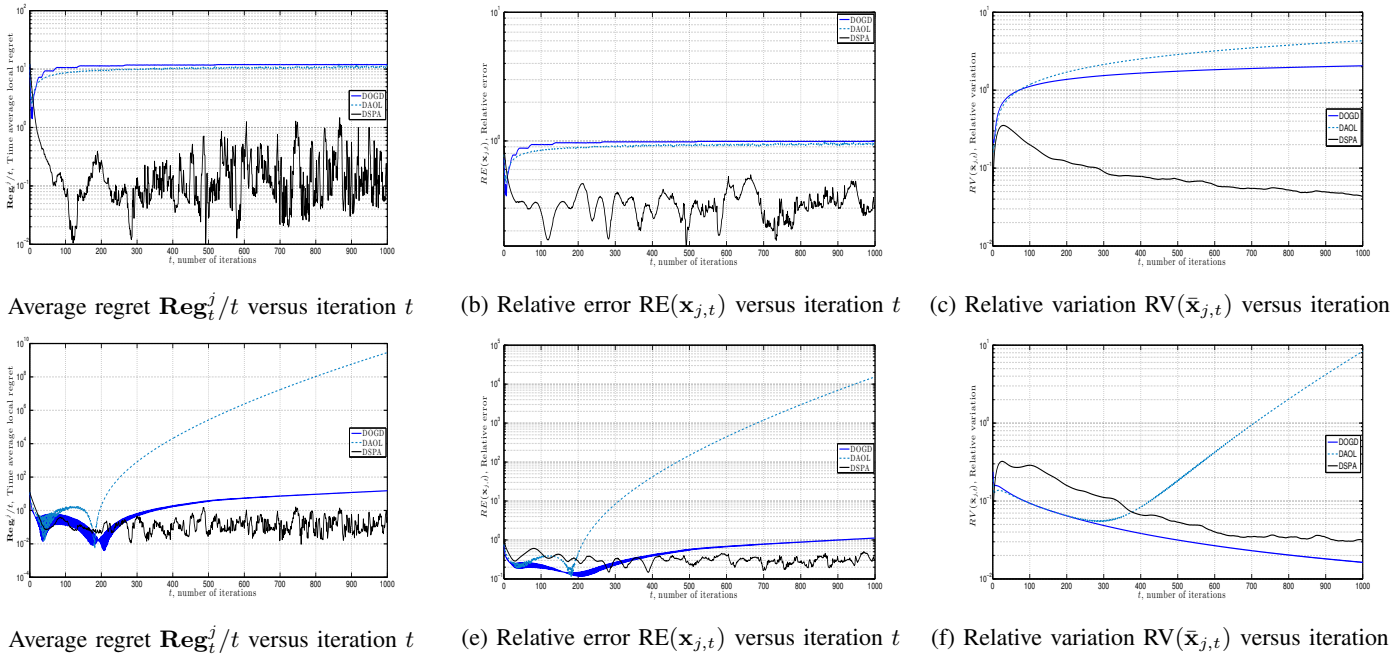
We present the results of this experiment in Figure 4. In Figure 4a, we plot $\mathbf{Reg}_t^j/t$ compared with iteration t. Observe that the rate at which $\mathbf{Reg}_t^j/t$ decreases is comparable across the different networks, yet we can differentiate the learning achieved in the different settings by the benchmark $\mathbf{Reg}_t^j/t \leq 10^{-2}$. To surpass this bound, the algorithm requires $t = 293$, $t = 221$ iterations for random and small world networks, respectively, whereas for grids and cycles it requires $t = 483$, $t = 865$ iterations. This indicates that structured deterministic networks are a more difficult setting for networked online learning, and the randomness present in random and small world networks allows more effective information flow.

In the plot of $\text{RE}(\mathbf{x}_{j,t})$ over time t shown in Fig 4b, we see a slower rate of convergence towards the batch learner in the structured deterministic networks: $\text{RE}(\mathbf{x}_{j,t}) \leq 0.2$ requires $t = 81$, $t = 176$, $t = 556$, and $t = 578$ iterations for random, small world, grid, and cycle networks, respectively, which validates the relationship observed in Figure 4a. We observe this rate difference more readily in Fig 4c, which plots $\text{RV}(\bar{\mathbf{x}}_{j,t})$ over time t. To obtain $\text{RV}(\bar{\mathbf{x}}_{j,t}) \leq 5 \times 10^{-2}$, the algorithm requires $t = 49$, $t = 301$, $t = 809$, and $t = 525$ iterations respectively for random, small world, grid, and cycle networks. These experiments indicate that information propagation slows in large diameter networks, causing more numerical oscillations and decreasing the learning rate. Put another way, networks in which agents may communicate more effectively reach consensus.

D. Algorithm Comparison

We turn to comparing the saddle point method against other recent works in networked online convex optimization. To that end, we consider grid and cycle topologies with $N = 50$ agents. We implement Distributed Online Gradient Descent (DOGD) [19], and Distributed Autonomous Online Learning (DAOL) [28]. Both of these are consensus protocols based on an iterative weighted averaging process. The primary difference between these two methods is that DOGD performs many gradient averaging steps per node update, whereas DAOL only does a single local averaging per iterate.

Figure 5 shows the results of this comparison. In figures 5a and 5d, we plot $\mathbf{Reg}_t^j/t$ versus the iteration t on a grid and cycle network, respectively. Both DOGD and DAOL fail to achieve learning: for all $t \geq 100$, $\mathbf{Reg}_t^j/t \approx 10$ in the grid network. Moreover, in the cycle case $\mathbf{Reg}_t^j/t \approx 10$ for all $t \geq 500$ for DOGD, while DAOL suffers unbounded regret as t increases. On the other hand, $\mathbf{Reg}_t^j/t \leq 5 \times 10^{-2}$ for $t \geq 580$ for the saddle point algorithm (DSPA). DSPA experiences a more substantial edge in learning performance in cycle networks, as compared with



(a) Average regret $\mathbf{Reg}_t^j/t$ versus iteration t (b) Relative error $\mathrm{RE}(\mathbf{x}_{j,t})$ versus iteration t (c) Relative variation $\mathrm{RV}(\bar{\mathbf{x}}_{j,t})$ versus iteration t

(d) Average regret $\mathbf{Reg}_t^j/t$ versus iteration t (e) Relative error $\mathrm{RE}(\mathbf{x}_{j,t})$ versus iteration t (f) Relative variation $\mathrm{RV}(\bar{\mathbf{x}}_{j,t})$ versus iteration t

Fig. 5: Comparison of saddle point method (DSPA) against other decentralized online learning methods as measured by $\mathbf{Reg}_T^j/T$, $\mathrm{RE}(\mathbf{x}_{j,t})$, and $\mathrm{RV}(\bar{\mathbf{x}}_{j,t})$ versus iteration t on grid (top) and cycle (bottom) networks of size $N = 50$ with $\mathbf{w}_{i,t} \sim \mathcal{N}(0, 0.1)$ signal noise. DSPA yields a more effective learning strategy in both network settings, measured in terms of $\mathbf{Reg}_T^j/T$, $\mathrm{RE}(\mathbf{x}_{j,t})$, when compared with consensus methods. DSPA and DOGD achieve comparable levels of consensus in cycle networks, though DOGD fails to learn the batch strategy, as seen in Figure 5d and Figure 5f. Gradient averaging methods (DOGD, DAOL) fail to learn in the grid network, and DAOL diverges in the cycle setting. Such methods may not be an appropriate tool for networked online learning problems since they seek a consensus which may diverge from the global batch strategy.

the diffusion methods.

The dynamics apparent in the regret plots appear in the relative error performance metric as well, as may be observed in Figures 5b and 5e, which plot $\mathrm{RE}(\mathbf{x}_{j,t})$ versus time t for grid and cycle networks, respectively. In the grid network, for all $t \geq 100$, $\mathrm{RE}(\mathbf{x}_{j,t}) \approx 10$ for DOGD, DAOL. In cycle networks DOGD achieves a near constant error after $t = 300$ iterations, while DAOL incurs an unbounded $\mathrm{RE}(\mathbf{x}_{j,t})$ with increasing t. Averaging the predictions of neighbors does not yield an effective strategy for this problem setting.

The gradient averaging methods fail to achieve consensus in the primal variable in the grid network, as may be seen in Figure 5c, which plots $\mathrm{RV}(\bar{\mathbf{x}}_{j,t})$ versus t. Moreover, while DOGD reaches a comparable level of agent discrepancy to DSPA in the cycle network, as may be seen in Figure 5f, DAOL experiences an unbounded growth in the average relative variation. Thus, in the later setting, DOGD yields a strategy which achieves consensus but diverges from the strategy of the batch learner. This follows form the fact that if the dimension of the signal to be estimated is less that of the observations, the averaging process of the consensus algorithms fails to move towards the optimum since averaging node predictions does not yield the average of individual loss functions' optima.

VI. Computer Network Security

We test the use of the saddle point algorithm in (17)-(18) to train a SVM for detecting security breaches in computer networks. The setting is one in which a number of service providers track user connectivity information in order to predict which users may be potentially harmful. This scenario is naturally cast as an online learning problem since users connect sequentially. If we further consider a network of interconnected service providers we see that each of them would benefit from additional information from other hosts, yet direct information sharing is problematic in terms of communication cost, delay, and the possibility of revealing sensitive proprietary information. This is casted naturally as a *networked* online learning problem where the service providers train their classifiers based on their local information and communication with neighboring peers. Instead of sharing the values of their feature vectors the different service providers exchange multipliers and classification vectors.

In the language of sections II-B and III we consider service providers that collect feature vectors $\mathbf{z}_{i,t}$ that they tag as friendly or malicious by setting $y_{i,t} = 1$ or $y_{i,t} = -1$, respectively. Starting with the local feature $\mathbf{z}_{i,t}$ and class $y_{i,t}$ given, as well as with the current local classifier $\mathbf{x}_{i,t}$ and multipliers $\boldsymbol{\lambda}_{ij,t}$ and $\boldsymbol{\lambda}_{ji,t}$ also given, we use the primal iteration in (17), which for the particular case of SVM classification takes the specific form in (20), to update the local classifier. The vector $\mathbf{x}_{i,t+1}$ is then used to predict the label $y_{i,t+1}$ corresponding to feature $\mathbf{z}_{i,t+1}$. The correct label is observed and recorded for use in the subsequent iteration. The updated classifier $\mathbf{x}_{i,t+1}$ is also shared with neighboring providers that use it to update their Lagrange multipliers using (18). The updated multipliers are then shared with neighbors as well. This permits updating of the classifier $\mathbf{x}_{i,t+1}$ through the use of (20). The feature vectors $\mathbf{z}_{i,t}$ that we use in our numerical experiments are described next.

A. Feature Vectors

We use the feature vectors in the data set in [29] which is constructed from approximately seven weeks of tcpdump data of network traffic that is processed into connection records. The training set on which we test the saddle point algorithm consists of $d = 4.94 \times 10^5$ single sample points of size $\tilde{p} = 41$ which contain

TABLE I: Components of feature vector for detecting computer network attacks: Standard user connection features.

Number	Feature Type	Type	Range	Description
1.	Duration	Integer	$[0, 5.84 \times 10^4]$	Connection duration
2-4.	Protocol Type	Binary	$\{0, 1\}$	Binary indicators for whether protocol type is TCP, UDP, or ICMP
5-9.	Service	Binary	$\{0, 1\}$	Binary indicators for service types: http, ftp, smtp, telnet, otherwise "other"
10-25.	Flag	Binary	$\{0, 1\}$	Binary indicators for connection statuses: SF, S0, S1, S2, S3, OTH, REJ, RSTO, RSTOS0, SH, RSTRH, SHR, RSTOS0, SH, RSTRH, SHR
26.	Source Bytes	Integer	$[0, 6.93 \times 10^8]$	Bytes sent from user
27.	Destination Bytes	Integer	$[0, 5.16 \times 10^6]$	Bytes received by host
28.	Land	Binary	$\{0, 1\}$	Indicator: 1 if source/destination IP addresses and port Number equal, 0 else
29.	Wrong Fragment	Integer	$[0, 3]$	Number of bad checksum packets
30.	Urgent Packets	Integer	$[0, 3]$	Number of packets with urgent bit activated

TABLE II: Components of feature vector for detecting computer network attacks: Content features tracking suspicious user to host behavior.

Number	Feature Type	Type	Range	Description
31.	Hot	Integer	$[0, 30]$	Number of "hot" actions: enter system directory, or create/execute programs
32.	Number of Failed Logins	Integer	$[0, 5]$	Number of failed logins per connection
33.	Login	Binary	$\{0, 1\}$	1 if login is correct, 0 otherwise
34.	Number Compromised	Integer	$[0, 884]$	Number of "not found" connection errors
35.	Root Shell	Binary	$\{0, 1\}$	1 if root gets the shell, 0 otherwise
36.	su Attempted	Binary	$\{0, 1\}$	1 if su command used, 0 otherwise
37.	Number Root Commands	Integer	$[0, 993]$	Number of user operations done as root
38.	Number File Creations	Integer	$[0, 28]$	Number files user created during session
39.	Number Shell Accesses	Integer	$[0, 2]$	Number of logins of normal users
40.	Number access files	Integer	$[0, 8]$	Number of operations on control files
41.	Number Outbound Commands	Integer	0	Number of outbound ftp commands
42.	Hot Login	Binary	$\{0, 1\}$	1 if admin/root accessed, 0 else
43.	Guest Login	Binary	$\{0, 1\}$	1 if guest login used, 0 else

TABLE III: Components of feature vector for detecting computer network attacks: Time traffic features derived from user behavior in the last two seconds.

Number	Feature Type	Type	Range	Description
44.	Count	Integer	$[0, 511]$	Number requests for same dest. IP
45.	Server Count	Integer	$[0, 511]$	Number requests for same dest. port
46.	Server Rate	Real	$[0, 1]$	Prop. of connections flagged (4) s0 - s3, among those in Count (23)
47.	Server S. Error Rate	Real	$[0, 1]$	Prop. of users flagged in (4) as s0 - s3, per Server Count (24)
48.	REJ Error Rate	Real	$[0, 1]$	Prop. of users flagged in (4) as REJ, compared with Count (23)
49.	Server Error Rate	Real	$[0, 1]$	Prop. of users flagged (4) as REJ, compared with Server Count (24)
50.	Same Server Rate	Real	$[0, 1]$	Prop. of connections to same service, compared to Count (23)
51.	Different Server Rate	Real	$[0, 1]$	Proportion of connections to different services, per Count (23)
52.	Server Different from Host Rate	Real	$[0, 1]$	Prop. of connections to diff. dest. compared to Server Count (24)

TABLE IV: Components of feature vector for detecting computer network attacks: Machine traffic features derived from the past 100 connections to host. These features are computed with respect same host/client indicators as time traffic features.

Number	Feature Type	Type	Range	Description
53.	Dst. Host Count	Integer	$[0, 255]$	Number requests for same dest. IP
54.	Dst. Host Srv. Count	Integer	$[0, 255]$	Number requests for same dest. port
55.	Dst. Host Same Srv. Rate	Real	$[0, 1]$	Prop. users to same service, compared to Dst. Host Count (32)
56.	Dst. Host Diff. Srv. Rate	Real	$[0, 1]$	Prop. users to diff. services, compared to Dst. Host Count (32)
57.	Dst. Host. Same Src. Port Rate	Real	$[0, 1]$	Prop. users to same source port, compared to Dst. Host Srv. Count (33)
58.	Dst. Host Srv. Diff Host Rate	Real	$[0, 1]$	Prop. users to diff. dest. machine, compared to Dst. Host Srv. Count (32)
59.	Dst. Host Serror Rate	Real	$[0, 1]$	Prop. users flagged (4) as s0 - s3, compared to Dst. Host Count (32)
60.	Dst. Host Srv. Serror Rate	Real	$[0, 1]$	Prop. users flagged (4) as s0 - s3, compared to Dst. Host Srv. Count (33)
61.	Dst. Host R. Error Rate	Real	$[0, 1]$	Proportion of users flagged (4) as REJ, as compared to Dst. Host Count (32)
62.	Dst. Host Srv. Error Rate	Real	$[0, 1]$	Prop. users flagged (4) as REJ, compared to Dst. Host Srv. Count (33)

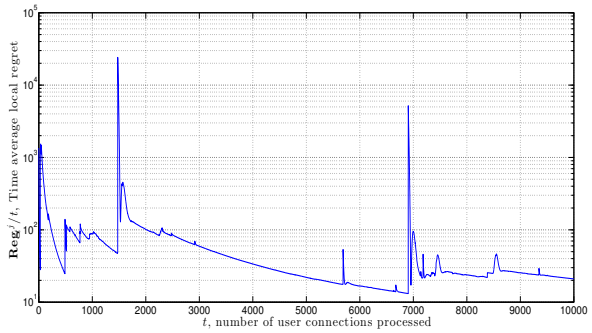


Fig. 6: Average local regret $\mathbf{Reg}_t^j/t$ vs. number of user connections processed t on a $N = 50$ node cycle network for the network security application of the distributed online SVM saddle point algorithm (20)-(18). Local regret of node $j = 29$ vanishes with t as node j's classifier converges to the global batch classifier computed with Liblinear [31]. Spikes in $\mathbf{Reg}_t^j/t$ correspond to misclassifications, and follow from nondifferentiability of the hinge loss. Large spikes at $t = 1476, 6905$ correspond to attacker examples not previously seen by the service provider that compromise its security, from which it quickly recovers.

client connectivity information, whose features fall into three categories: basic, content, and traffic features; see tables I - IV and [30]. The basic features in Table I consist of information contained in a TCP/IP connection, such as protocol type and user and host information. The content features in Table II consist of those that are most useful for detecting attacks related to user to root and remote to local attacks, examples of which include number of failed login attempts and root access attempts. The traffic features in Table III are computed with respect to a window interval around the connection, and consist of two groups: same host features and same service features. The former tracks connections in the prior two seconds that have the same host destination as the current connection and compute relevant statistics. The latter examines connections in the past two seconds that have the same service type as the current connection. We also record this same information averaged from the perspective of hosts over the last 100 user connections. These metrics are the traffic features shown in Table IV.

Basic, content, and traffic information are recorded for each user connection to construct a set of feature vectors $\{\mathbf{v}_k\}_{k=1}^d$, with labels $y_k \in \{-1, 1\}$ denoting whether a user is harmless or an attacker, respectively. The labels are formed by modifying the data in [29] to merge all the attacker types into one group, and adjusting the number of positive and negative training examples to be approximately equal. Feature statistics reported in Tables I - IV reflect these adjustments. Many features in data set in [29] are categorical (nominal), which we modify to obtain binary features. In particular, for each possible value the categorical variable may take, we construct a binary indicator variable denoting whether the variable takes on a particular value. For example, the feature Protocol Type in [29] takes integer values $1, 2, 3$ corresponding to TCP, UDP, or ICMP, from which construct three separate indicator variables for protocol type of the individual connection. With this modification, the feature vectors $\mathbf{z}_{i,t}$ are extended to dimension $p = 62$.

B. Numerical Results

We implement (20)-(18) for this intrusion detection problem in a cycle network with $N = 50$ nodes. We randomly partition the adjusted data from [29] into N blocks such that each service provider (node) trains a classifier online on its own data subset. We run the simulation over the entire one-percent adjusted training set, i.e. using $NT = 5 \times 10^5$ data points, with a constant step size $\epsilon = 1/\sqrt{T} \approx 0.01$. The regularization parameter $\zeta = \log(62)$ is chosen after 10-fold cross-validation and the Residual Information Criterion (RIC) as in [32]. The primal and dual variables are initialized at time $t = 1$ as zero vectors $\mathbf{x}_{j,1} = \mathbf{0}$ for all j and $\boldsymbol{\lambda}_{jk,1} = \mathbf{0}$ for all j and k. We compute the global batch classifier $\tilde{\mathbf{x}}^*$ with Liblinear [31].

Fig. 6 shows $\mathbf{Reg}_t^j/t$ the time average local regret for the (arbitrarily chosen) node $j = 29$. The iteration index t corresponds to the number of user connections processed. Observe that $\mathbf{Reg}_t^j/t$ decays with the number of processed user connections at the rate guaranteed by Theorem 1. The large instantaneous magnitude of $\mathbf{Reg}_t^j/t$ is a result of the large ranges of features such as Source and Destination Bytes. Large spikes at $t = 1,476$ and $t = 6,905$ correspond to attacker examples not previously observed. Observe that by $t = 10^5$, $\mathbf{Reg}_t^j/t \leq 21$, indicating that the service provider effectively learns an intrusion classifier as good as the one with user information aggregated at a central location for all times in advance. Each time an attacker compromises the host, which correspond to a spike in the local regret trajectory, the intrusion detection protocol recovers quickly.

We turn to studying the classifier error rates. Denote the vector of predictions $\hat{y}_{j,t}$, which is of length t and whose uth entry is given by $[\hat{y}_{j,t}]_u = \text{sgn}(\mathbf{x}_{j,t}^T \mathbf{z}_{j,u})$ for users $u \leq t$. We break misclassifications into two categories: (i) the false alarm rate $\alpha_{j,t} := P(\hat{y}_{j,u} = 1 \mid y_{j,u} = -1)$ which tracks the proportion of friendly users predicted as attackers; (ii) the error rate $\beta_{j,t} := P(\hat{y}_{j,u} = -1 \mid y_{j,u} = 1)$ which accounts for the attackers that were not detected. These quantities are computed as the number of entries of $\hat{y}_{j,t}$ that equal 1 over the number of associated users $u \leq t$ with label -1, and vice versa. We consider the average false alarm rate $\bar{\alpha}_{j,t} = \sum_{u=1}^t P(\hat{y}_{j,u} = 1 \mid y_{j,u} = -1)/t$, and the average error rate $\bar{\beta}_{j,t} = \sum_{u=1}^t P(\hat{y}_{j,u} = -1 \mid y_{j,u} = 1)/t$ on a test data set of size $T = 1 \times 10^4$.

Fig. 7 shows the evolution of $\bar{\alpha}_{j,t}$, the average false alarm rate, versus the number of connections processed t. The expected classifier accuracy $(1 - \bar{\alpha}_{j,t})$ stabilizes between $[0.67, 0.70]$ after a burn-in period $t \geq 5 \times 10^3$ (false alarm rate $[0.30, 0.33]$), indicating that an inordinate proportion of friendly users are not denied service in this intrusion detection protocol. A node's ability to flag harmful users is the essential performance metric. Fig. 8 plots at $\bar{\beta}_{j,t}$, the error rate of service provider $j = 29$ on a test set of fixed size $T = 1 \times 10^4$, with the number of connections processed t. The average rate of correctly detecting an attacker, or power, $(1 - \bar{\beta}_{j,t})$ begins near null and stabilizes between $[0.86, 0.90]$ for $t \geq 3 \times 10^3$, which is competitive given the difficulty predicting attacks in commercial settings. The price for this accuracy level is its conservative treatment of normal users.

VII. Conclusion

We extended the idea of online convex optimization to networked settings, where nodes are allowed to make autonomous learning decisions while incorporating neighbors' information. We developed regret formulations illustrating the distributed learning goal and proposed the use of a saddle point algorithm to solve such problems. Theorem 1 showed that the saddle point iterates achieve the networked online learning goal, which is the sub linear growth

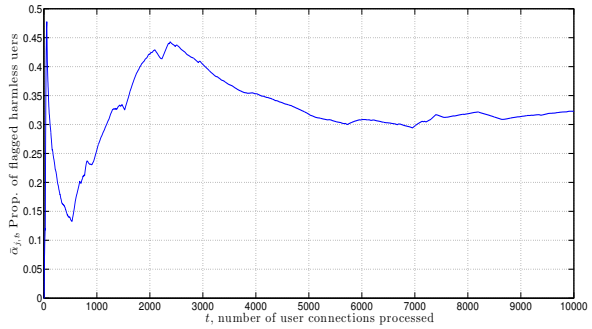


Fig. 7: Time average error rate of incorrectly flagging a benign user $\bar{\alpha}_{j,T} = \sum_{t=1}^{T} P(\hat{y}_{j,t} = 1 \mid y_{j,t} = -1)$ on a test set of $T = 1 \times 10^4$ user connections. The error rate stabilizes between $[0.30, 0.33]$ as the server learns to not flag benign users unnecessarily, despite widely varying connectivity information.

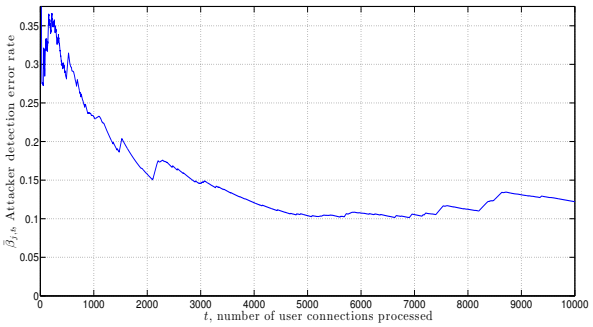


Fig. 8: Time average empirical probability of failing to detect an attacker $\bar{\beta}_{j,T} = \sum_{t=1}^{T} P(\hat{y}_{j,t} = -1 \mid y_{j,t} = 1)$ on a test set of $T = 1 \times 10^4$ user connections. The error rate stabilizes between $[0.10, 0.15]$ as the host learns to deny service to a variety of attacker profiles.

rate of global networked regret: the time average regret goes to null at a rate of $O(1/\sqrt{T})$. Theorem 2 guaranteed that individual agents also achieve this learning rate as well.

Numerical analysis demonstrated the algorithm performance dependency on network size, connectivity, and topology: learning rates are comparable across different network sizes but more prone to numerical oscillations in smaller networks. Similarly, network topologies with smaller diameter yield more stable predictions. We applied this algorithm to the problem of training a SVM classifier online over a network, and consider an attacker detection problem in a computer network security application. Empirically this method yields competitive prediction accuracy and is able to maintain the privacy of distinct nodes' users connectivity data.

REFERENCES

[1] A. Koppel, F. Y. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

[2] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, Feb. 2012.

[3] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[4] Martin Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," 2003.

[5] C. B. Do, Q. V. Le, and C. Foo, "Proximal regularization for online and batch learning.," in *ICML*, A.P. Danyluk, L. Bottou, and M.L. Littman, Eds. 2009, vol. 382 of *ACM International Conference Proceeding Series*, p. 33, ACM.

[6] S. Shalev-shwartz and Y. Singer, "Logarithmic regret algorithms for strongly convex repeated games," in *The Hebrew University*, 2007.

[7] S.S. Ram, A. Nedic, and V.V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of Optimization Theory and Applications*, vol. 147, no. 3, pp. 516–545, 2010.

[8] T. Suzuki, "Dual averaging and proximal gradient descent for online alternating direction multiplier method," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Atlanta, GA, 2013, vol. 28, pp. 392–400, JMLR Workshop and Conference Proceedings.

[9] A. Nedic and A. Ozdaglar, "Subgradient methods for saddle-point problems," *Journal of Optimization Theory and Applications*, pp. 205–228, 2009.

[10] Dusan Jakovetic, Jo ao Manuel Freitas Xavier, and José M. F. Moura, "Fast distributed gradient methods," *CoRR*, vol. abs/1112.2972, 2011.

[11] Kun Yuan, Qing Ling, and Wotao Yin, "On the convergence of decentralized gradient descent," *arXiv preprint arXiv:1310.7063*, 2013.

[12] M.G. Rabbat, R.D. Nowak, and J.A Bucklew, "Generalized consensus computation in networked systems with erasure links," in *Signal Processing Advances in Wireless Communications, 2005 IEEE 6th Workshop on*, June 2005, pp. 1088–1092.

[13] F. Jakubiec and A. Ribeiro, "D-map: Distributed maximum a posteriori probability estimation of dynamic systems," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 450–466, February 2013.

[14] I. Schizas, A. Ribeiro, and G. Giannakis, "Consensus in ad hoc wsns with noisy links - part i: distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, January 2008.

[15] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "Dlm: Decentralized linearized alternating direction method of multipliers," *IEEE Trans. Signal Process.*, August 2014.

[16] M. Zargham, A. Ribeiro, A. Jadbabaie, and A. Ozdaglar, "Accelerated dual descent for network optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 905 – 920, April 2014.

[17] K.J. Arrow, L. Hurwicz, and H. Uzawa, *Studies in linear and non-linear programming*, With contributions by H. B. Chenery, S. M. Johnson, S. Karlin, T. Marschak, R. M. Solow. Stanford Mathematical Studies in the Social Sciences, vol. II. Stanford University Press, Stanford, 1958.

[18] Y. Nesterov, "Primal-dual subgradient methods for convex problems," Tech. Rep., 2005.

[19] K. I. Tsianos and M. G. Rabbat, "Distributed strongly convex optimization," *CoRR*, vol. abs/1207.3031, 2012.

[20] F. Yan, S. V. N. Vishwanathan, and Y. Qi, "Cooperative autonomous online learning," *CoRR*, vol. abs/1006.4039, 2010.

[21] A. Nedic and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods,," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2008.

[22] I. Lobel and A. Ozdaglar, "Distributed subgradient methods for convex optimization," *LIDS Report*, vol. 2800, 2009.

[23] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, June 1998.

[24] H. Block, "The perceptron: a model for brain functioning," *Reviews of Modern Physics*, vol. 34, pp. 123–135, 1962.

[25] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, Dec. 1999.

[26] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, June 4 1998.

[27] D. B. West, *Introduction to Graph Theory*, Prentice Hall, 2 edition, September 2000.

[28] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Trans. on Knowl. and Data Eng.*, vol. 25, no. 11, pp. 2483–2493, Nov. 2013.

[29] "KDD Cup 1999 Data," http://archive.ics.uci.edu/ml/databases/kddcup99 /kddcup99.html.

[30] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, Piscataway, NJ, USA, 2009, CISDA'09, pp. 53–58, IEEE Press.

[31] R-E Fan, K-W Chang, C-J Hsieh, X-R Wang, and C-J Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[32] P. Shi and C-L Tsai, "Regression model selection-a residual likelihood approach," *Journal of the Royal Statistical Society Series B*, vol. 64, no. 2, pp. 237–252, 2002.