Distributed Solutions of Convex-Concave Games on Networks

Yingying Xiao, Xiaodong Hou, and Jianghai Hu

***Abstract*— In this paper, we study the convex-concave games on agent networks played by two teams. Each agent has local variables from both teams and a local payoff function dependent on the variables from its neighbors that is convex in the variables of one team and concave in the variables of the other. The goal is to find a saddle point (if it exists) of the sum of all local payoff functions. The problem can be converted to a fixed point problem by using the saddle differential operator. As oftentimes direct fixed point iterations are computationally expensive, using the operator splitting technique, we propose two general-purpose iterative algorithms that can find the saddle points of a convex-concave payoff function by splitting it into several parts each of which is more amenable for computation. Applying these algorithms to the convex-concave games on agent networks, we obtain efficient synchronous distributed solution algorithms. Their randomized, asynchronous implementations are also discussed. Numerical examples are provided to illustrate the proposed algorithms.**

I. INTRODUCTION

This paper studies the zero-sum games on agent networks between two teams where the global payoff function is of the form $K = \sum_{i=1}^{m} K_i$ with K_i being the local payoff function of agent i. It is assumed that the decision variables from both teams are distributed among the different agents, and that the local payoff functions are coupled: each K_i depends on not only its own decision variables from the two teams but also the decisions variables of its neighboring agents. As a special case, each agent may hold the variables of only one team, and the games are played between two opposing teams of agents. We focus on convex-concave games where each K_i (hence K) is assumed to be convex in the variables of one team and concave in the variables of the other team. Under fairly mild conditions (see, e.g., Proposition 1), saddle points of K exist. The objective of this paper is to design distributed algorithms that can compute a saddle point of K.

One motivation of this study is our previous work [1] on the distributed solutions of locally coupled optimization problems on agent networks, where each agent tries to solve a local optimization problem whose objective function/constraints depend on the variables of its neighboring agents. The Lagrangian function K_i of agent i's optimization problem is convex-concave in the primal and dual variables; and the global primal-dual solution is a saddle point of $K = \sum_i K_i$. Thus, algorithms developed in this paper can lead to new primal-dual algorithms for solving the distributed optimization problems in [1]. We note that the Lagrangian K is linear in the dual variables, while in this paper we consider the more general convex-concave payoff functions K (e.g., those resulted from variational inequalities [2]).

There has been an enormous amount of existing work on the applications of saddle point problems to constrained optimization, zero-sum games, partial differential equations, machine learning and so on (see the survey [3]). Methods developed for computing saddle points include iterative methods such as the Arrow-Hurwicz and Uzawa method [4] and its variants, Krylov subspace methods and associated preconditioning techniques [5], Hermitian and skew-Hermitian splitting based methods [6], to name a few classes. Several (sub)gradient-based algorithms have been developed that achieve $O(1/k)$ convergence, e.g., [7] using smoothing techniques, [8] using prox method, and [9] by approximating the saddle point using the running average with a constant stepsize. On the other hand, these algorithms are centralized and not amenable for distributed implementation.

A distributed projected subgradient method is proposed in [10] to solve the saddle point problem arising from a networked optimization problem where the objective function and constraints are separable in agents' individual variables, while both depend on a global decision variable. By introducing local copies of the global decision variable and the dual variable, the problem is formulated as a saddle point problem for the Lagrangian function. When applied to the locally coupled networked optimization problem considered in [1], the algorithms developed in [10] could result in excessive communication and storage requirements.

This paper is organized as follows. Some useful facts on convex-concave (or saddle) functions and saddle points are reviewed in Section II. In Section III, two centralized algorithms based on splitting methods are proposed to compute the saddle points. Section IV presents the formulation of convex-concave games on agent networks and the proposed distributed solution algorithms. Simulation results of a convex-concave games with 7 agents are given in Section V. Finally, Section VI concludes the paper.

II. SADDLE FUNCTIONS AND SADDLE POINTS

We first review some basic facts from convex analysis. An extended-real-valued function $f : \mathbb{R}^n \to \overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$ is convex if, for all $x_1, x_2 \in \mathbb{R}^n$ and all $\lambda \in [0,1]$, $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$ holds whenever $\{f(x_1), f(x_2)\} \neq \{\pm\infty\}$. For instance, for any convex

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA {xiao106,hou39,jianghai}@purdue.edu

subset $C \subset \mathbb{R}^n$, the two functions

$$f(x) = \begin{cases} -\infty & \text{if } x \in C, \\ +\infty & \text{if } x \notin C; \end{cases} \tag{1}$$

$$\mathbf{1}_C(x) := \begin{cases} 0 & \text{if } x \in C, \\ +\infty & \text{if } x \notin C, \end{cases} \tag{2}$$

are convex. A convex function f is called *proper* if: (i) $f(x) > -\infty$ for all x; and (ii) $f(x) < +\infty$ for at least one x. It is called *closed* if its epigraph $\{(x,t) \,|\, f(x) \le t\}$ is a closed set, or equivalently, if f is lower semicontinuous. By these definitions, f in (1) is never proper; and it is closed if and only if the convex set C is closed.

A. Saddle Functions

Let X and Y be two Euclidean spaces (possibly of different dimensions) and let $K : X \times Y \to \overline{\mathbb{R}}$ be an extended-real-valued function.

Definition 1 ([11]): $K : X \times Y \to \overline{\mathbb{R}}$ is called a *saddle function* on $X \times Y$ if $K(x,y)$ is a convex function of x for each fixed y and a concave function of y for each fixed x. The saddle function K is *closed* if $K(x,y)$ is lower semicontinuous in x for each fixed y and upper semicontinuous in y for each fixed x. The effective domain of K is defined as

$$\begin{aligned} \operatorname{dom} K := \{(x,y) \in X \times Y \,|\, K(x,y') < +\infty,\ \forall y' \in Y \\ \text{and } K(x',y) > -\infty,\ \forall x' \in X\}. \end{aligned}$$

K is called *proper* if $\operatorname{dom} K \neq \emptyset$.

Example 1: Some examples of saddle functions are given below.

(i) $K(x,y) = f(x) - g(y) + h(x,y)$ is a saddle function where $f \in \overline{\mathbb{R}}$ and $g \in \mathbb{R}$ are convex functions and $h(x,y) = x^T A y$ is bilinear for some matrix A. K is closed and proper if both f and g are CCP functions.

(ii) For the optimization problem $\min_{x \in \mathbb{R}^n}\{f(x) \,|\, g(x) \preceq 0\}$ where $f \in \mathbb{R}$ and $g \in \mathbb{R}^m$ are convex functions, the Lagrange function $L(x,y) = f(x) + \langle y, g(x)\rangle - \mathbf{1}_D(y)$ where $D = \{y \,|\, y \succeq 0\}$ is a saddle function. It is closed and proper if both f and g are CCP functions.

(iii) Let $C \subset X$ and $D \subset Y$ be convex subsets. Define

$$\begin{aligned} \mu_{C\times D}(x,y) &:= \mathbf{1}_C(x) - \mathbf{1}_{(C\times D^c)^c}(x,y) \qquad (3) \\ &= \begin{cases} 0 & \text{if } x \in C \text{ and } y \in D \\ -\infty & \text{if } x \in C \text{ and } y \notin D \\ +\infty & \text{if } x \notin C, \end{cases} \\ \nu_{C\times D}(x,y) &:= -\mathbf{1}_D(y) + \mathbf{1}_{(C^c\times D)^c}(x,y) \qquad (4) \\ &= \begin{cases} 0 & \text{if } x \in C \text{ and } y \in D \\ +\infty & \text{if } x \notin C \text{ and } y \in D \\ -\infty & \text{if } y \notin D. \end{cases} \end{aligned}$$

Here, $(C\times D^c)^c := (X\times Y)\backslash(C\times(Y\backslash D))$; similarly for $(C^c \times D)^c$. Then $\mu_{C\times D}$ and $\nu_{C\times D}$ are saddle functions on $X\times Y$ with the domain $C\times D$. Further, both of them are closed and proper if C and D are nonempty closed subsets. For a real-valued saddle function K, $K+\mu$ and $K+\nu$ are two saddle functions whose values agree with K on their domain $C \times D$.

B. Saddle Points

A point $(x^*, y^*) \in X \times Y$ is called a *saddle point* of the saddle function K if

$$K(x^*, y) \le K(x^*, y^*) \le K(x, y^*), \ \ \forall x \in X,\, y \in Y. \tag{5}$$

In other words, x^* is a minimizer of $K(\cdot, y^*)$ and y^* is a maximizer of $K(x^*, \cdot)$. In this case, we must have $\sup_y \inf_x K(x,y) = \inf_x \sup_y K(x,y) = K(x^*, y^*)$. If K is interpreted as the payoff function of a zero-sum two-player convex-concave game, then saddles points are exactly the Nash equilibria.

General saddle functions may have no saddle points. A sufficient condition for the existence of saddle points based on the Sion's Minimax Theorem [12] is given below.

Proposition 1: ([12]) Suppose K is a real-valued closed saddle function on $X \times Y$. Let $C \subset X$ and $D \subset Y$ be two nonempty compact convex subsets. Then

$$\min_{x\in C} \max_{y \in D} K(x,y) = \max_{y\in D} \min_{x \in C} K(x,y).$$

Moreover, (x^*, y^*) with $x^* = \arg\min_{x\in C} \max_{y\in D} K(x,y)$ and $y^* = \arg\max_{y\in D} \min_{x\in C} K(x,y)$ is a saddle point of $K + \mu_{C\times D}$ (or $K + \nu_{C\times D}$).

C. Saddle Subdifferential Operator

For the saddle function K on $X\times Y$, a set-valued operator $T_K : X \times Y \to 2^{X\times Y}$ can be defined by

$$T_K(x,y) = \begin{bmatrix} \partial_x K(x,y) \\ \partial_y(-K)(x,y) \end{bmatrix}, \quad \forall (x,y) \in X \times Y.$$

Here, $\partial_x K(x,y)$ denotes the subdifferentials (set of subgradients) of the convex function $K(\cdot, y)$ at the point x; similarly for $\partial_y(-K)(x,y)$. In [13], T_K is referred to as the saddle subdifferential operator of K. The domain of T_K is defined as $\operatorname{dom} T_K := \{(x,y) \,|\, T_K(x,y) \neq \emptyset\}$.

The zero set of T_K is defined as $\operatorname{zer}(T_K) := \{(x,y) \,|\, 0 \in T_K(x,y)\}$. The condition (5) for a point (x^*, y^*) to be a saddle point of K is equivalent to $0 \in \partial_x K(x^*, y^*)$ and $0 \in \partial_y(-K)(x^*, y^*)$, i.e., $0 \in T_K(x^*, y^*)$. We thus have the following result.

Proposition 2: The set of saddle points of K is $\operatorname{zer}(T_K)$.

Recall that a set-valued operator $T : Z \to 2^Z$ for $Z = \mathbb{R}^n$ is called *monotone* if $(w_2 - w_1)^T(z_2 - z_1) \ge 0$ for all $z_1, z_2 \in Z$ and all $w_1 \in T(z_1)$, $w_2 \in T(z_2)$. It is called *maximally monotone* if it is monotone and its graph is not properly contained in that of any other monotone operator. Given $\lambda > 0$, the resolvent of T is the (set-valued) operator $R : Z \to 2^Z$ defined by $R = (I + \lambda T)^{-1}$ with its domain being $\operatorname{dom} R := \{x \,|\, R(x) \neq \emptyset\}$. If T is monotone, then R is an *averaged operator*: $R = (I+S)/2$ for some nonexpansive operator S, and hence single-valued. If T is further maximally monotone, then $\operatorname{dom} R = Z$ and the set of fixed points of R, $\operatorname{Fix}(R)$, is exactly the zero set of T, $\operatorname{zer}(T)$. See [13] for further details.

Theorem 1 ([11]): Let K be a saddle function on $X \times Y$. If K is proper, then T_K is a monotone operator with the domain $\operatorname{dom} T_K \subset \operatorname{dom} K$. If K is proper and closed, then T_K is a maximally monotone operator.

By the above result, T_K is maximally monotone for a closed proper saddle function K. Its resolvent, denoted by R_K, is an averaged operator with the fixed point set $\operatorname{Fix}(R_K) = \operatorname{zer}(T_K)$ being exactly the set of saddle points of K. The iteration $(x^{k+1}, y^{k+1}) = R_K(x^k, y^k)$ will converge to a point in $\operatorname{Fix}(R_K)$ and hence a saddle point of K.

We next characterize how R_K can be computed. For any $(x, y) \in X \times Y$, $(p, q) = R_K(x, y)$ if and only if

$$\begin{aligned}
&(x, y) \in (I + \lambda T_K)(p, q) \\
\Leftrightarrow\ &(x, y) \in (p + \lambda \partial_p K(p, q), q + \lambda \partial_q(-K)(p, q)) \\
\Leftrightarrow\ &\begin{cases} 0 \in \partial_p K(p, q) + (p - x)/\lambda \\ 0 \in \partial_q(-K)(p, q) + (q - y)/\lambda \end{cases} \qquad (6a) \\
\Leftrightarrow\ &\begin{cases} p = \arg\min_{p \in X}\ K(p, q) + \frac{1}{2\lambda}\|p - x\|^2 \\ q = \arg\max_{q \in Y}\ K(p, q) - \frac{1}{2\lambda}\|q - y\|^2 \end{cases} \qquad (6b) \\
\Leftrightarrow\ &(p, q) \text{ is a saddle point of} \\
&K(p, q) + \frac{1}{2\lambda}\left(\|p - x\|^2 - \|q - y\|^2\right). \qquad (6c)
\end{aligned}$$

By letting $p = x$ and $q = y$ in (6c), we see that, as expected, fixed points of R_K are exactly the saddle points of K.

The above conditions can be used to compute R_K for certain families of saddle functions.

Example 2:

(i) Suppose $K(x, y) = f(x) + \langle y, Ax - b\rangle$ is the Lagrange function for the optimization problem of minimizing $f(x)$ subject to $Ax = b$. The corresponding T_K is called the KKT operator [13]. In particular, (6a) becomes

$$0 \in (p - x)/\lambda + \partial f(p) + A^T q, \quad (q - y)/\lambda = Ap - b.$$

The second equation implies $q = y + \lambda(Ap - b)$, which when plugged into the first one yields $0 \in \partial f(p) + (p - x)/\lambda + A^T y + \lambda A^T(Ap - b)$. In other words,

$$\begin{cases} p &= \arg\min_z \mathcal{L}_\lambda(z) + \frac{1}{2\lambda}\|z - x\|^2 \\ q &= y + \lambda(Ap - b). \end{cases}$$

Here, $\mathcal{L}_\lambda(z) := f(z) + \langle y, Az - b\rangle + \frac{\lambda}{2}\|Az - b\|^2$ is the augmented Lagrange function.

(ii) Consider the saddle function

$$K(x, y) = \frac{1}{2}\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} \Sigma_1 & \Sigma_2 \\ \Sigma_2^T & -\Sigma_3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} \qquad (7)$$

where $\Sigma_1, \Sigma_3 \succeq 0$. Then (6a) becomes

$$\begin{cases} (p - x)/\lambda + \Sigma_1 p + \Sigma_2 q + b_1 = 0 \\ (q - y)/\lambda + \Sigma_3 q - \Sigma_2^T p - b_2 = 0 \end{cases} \qquad (8)$$

$$\Rightarrow \quad R_K(x, y) = (I + \lambda\Sigma)^{-1}\begin{bmatrix} x - \lambda b_1 \\ y + \lambda b_2 \end{bmatrix}, \qquad (9)$$

where $\Sigma = \begin{bmatrix} \Sigma_1 & \Sigma_2 \\ -\Sigma_2^T & \Sigma_3 \end{bmatrix}$. Note that $(I + \lambda\Sigma)^{-1}$ exists since it is the resolvent of a maximally monotone linear operator Σ whose monotonicity follows from the fact that $\Sigma + \Sigma^T \succeq 0$. Fixed points of R_K are those points (x^*, y^*) satisfying $\Sigma\begin{bmatrix}(x^*)^T & (y^*)^T\end{bmatrix}^T = \begin{bmatrix}-b_1^T & b_2^T\end{bmatrix}^T$, the exact same condition for (x^*, y^*) to be a saddle point of K. If Σ is nonsingular, there is a unique saddle point $\Sigma^{-1}\begin{bmatrix}-b_1^T & b_2^T\end{bmatrix}^T$. In the special case of $\Sigma_1 = 0$ and $\Sigma_3 = 0$, the result becomes:

$$\begin{cases} p = \left(I + \lambda^2\Sigma_2\Sigma_2^T\right)^{-1}\left(x - \lambda b_1 - \lambda\Sigma_2 y - \lambda^2\Sigma_2 b_2\right) \\ q = y + \lambda b_2 + \lambda\Sigma_2^T p. \end{cases} \qquad (10)$$

(iii) For the function μ defined in (3), we have $R_\mu(x, y) = \Pi_C \times \Pi_D(x, y) = (\Pi_C(x), \Pi_D(y))$. Here, for a convex set Ω, Π_Ω denotes the orthogonal projection onto Ω. This follows from (6c) as the saddle points of $\mu(p, q) + \frac{1}{2\lambda}(\|p - x\|^2 - \|q - y\|^2)$ on $X \times Y$ are exactly the saddle points of $\frac{1}{2\lambda}(\|p - x\|^2 - \|q - y\|^2)$ on $C \times D$. Similarly, for ν defined in (4), R_ν is also $\Pi_C \times \Pi_D$.

The following result can be easily proved and will be useful later on.

Proposition 3 (Separable K): Suppose $K(x, y) = K_1(x_1, y_1) + \cdots + K_m(x_m, y_m)$ is separable. Here, $K_i(x_i, y_i)$ is a closed proper saddle function on $X_i \times Y_i$ for each i; $x = (x_1, \ldots, x_m) \in X = X_1 \times \cdots \times X_m$; and $y = (y_1, \ldots, y_m) \in Y = Y_1 \times \cdots \times Y_m$. Then, $(p, q) = R_K(x, y)$ is given by $p = (p_1, \ldots, p_m)$ and $q = (q_1, \ldots, q_m)$ where $(p_i, q_i) = R_{K_i}(x_i, y_i)$ for each i.

III. Splitting Method

Throughout this section we assume that K is an extended-real-valued closed proper saddle function on $X \times Y$ and it has at least one saddle point (x^*, y^*).

Suppose K admits the decomposition $K(x, y) = K_1(x, y) + \cdots + K_m(x, y)$ where each K_i is a closed proper saddle function on $X \times Y$ (note that this is different from the separable K case in Proposition 3). By Theorem 1, T_K and all T_{K_i}'s are maximal monotone operators, whose resolvents R_K and R_{K_i}'s are averaged maps, all well defined on $X \times Y$. Although K is assumed to have a saddle point, K_i's may have none. In other words, the set of fixed points is nonempty for R_K, but could be empty for R_{K_i}'s.

In this section, we consider the scenario that each R_{K_i} has a much lower computational cost than R_K. We will focus on the two cases that $m = 2$ and $m = 3$ and derive in each case iteration algorithms that can compute a fixed point of R_K (i.e., a saddle point of K) using the operators R_{K_i}'s.

A. Two-Operator Splitting

Suppose $K = K_1 + K_2$ where R_{K_1} and R_{K_2} are much easier to compute that R_K. Several examples of such a case are given in the following.

Example 3:

(i) Consider Example 1 (i), $K(x, y) = f(x) - g(y) + x^T A y$, which has $T_K = [\partial f(x)^T + y^T A^T \quad \partial g(y)^T - x^T A]^T$.

Even if both f and g are differentiable, to compute R_K using (6a), one would need to solve two nonlinear equations that are coupled due to the bilinear term. On the other hand, we can write K as $K = K_1 + K_2$ with $K_1 = f(x) + x^T Ay$ and $K_2 = -g(y)$. Then, R_{K_1} can be computed as in Example 2 (i), while $R_{K_2} = I \times \text{prox}_{\lambda g}$. Here, $\text{prox}_{\lambda g}$ is the proximal operator of g, $\text{prox}_{\lambda g}(y) := \arg\min_q g(q) + (1/2\lambda)\|q - y\|^2$, that can often be computed very efficiently [14].

(ii) Let $K = K_1 + K_2$ where K_1 is given in (7) and $K_2 = \mu_{C\times D}$ for some nonempty compact convex subsets $C \subset X$ and $D \subset Y$. By Example 2 (ii) and (iii), R_{K_1} and R_{K_2} can be easily computed. However, to compute R_K directly, e.g., via (6b), two coupled constrained quadratic optimization problems need to be solved. In particular, the solution may not satisfy (8).

(iii) Suppose $K(x, y) = G_0(x)+\sum_{i=1}^m G_i(x_i, y)$ where $x = (x_1, \ldots, x_m) \in X = X_1 \times \cdots \times X_m$; $y \in Y$; G_0 is a real-valued closed convex function; and G_i, $i = 1, \ldots, m$, are real-valued closed saddle function on $X_i \times Y$. By introducing duplicate variables $\tilde{y}_i$, $i = 1, \ldots, m$, of y, K can be rewritten as $K(x, \tilde{y}) = K_1 + K_2$ where $K_1 = G_1(x_1, \tilde{y}_1) + \cdots + G_m(x_m, \tilde{y}_m)$ is separable and $K_2 = G_0(x) - \mathbf{1}_{\mathcal{A}}(\tilde{y})$ with $\mathcal{A} = \{\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_m) \in Y^m \,|\, y_1 = \cdots = y_m\}$. Then, R_{K_1} can be computed by Proposition 3 and $R_{K_2} = \text{prox}_{\lambda G_0} \times \Pi_{\mathcal{A}}$. Note that $\Pi_{\mathcal{A}}(\tilde{y}) = (\bar{y}, \ldots, \bar{y})$ with $\bar{y} = \frac{1}{m}(y_1 + \cdots + y_m)$.

Next we adopt the Douglas-Rachford splitting to derive an algorithm for finding a saddle point of K using R_{K_1} and R_{K_2}. By Proposition 2, the saddle points of $K = K_1 + K_2$ are exactly the points in the set $\text{zer}(T_{K_1}+T_{K_2})$. Applying the Peaceman-Rachford method [15], a point $w^* = (x^*, y^*)$ in $\text{zer}(T_{K_1}+T_{K_2})$ can be obtained as $w^* = R_{K_2}(z^*)$ where z^* is a fixed point of the nonexpansive map $(2R_{K_1}-I)(2R_{K_2}-I)$. Such a z^* can be obtained by iterations using the averaged map $(1-\alpha)I + \alpha(2R_{K_1} - I)(2R_{K_2} - I)$ for $\alpha \in (0, 1)$. This leads to the well known Douglas-Rachford splitting algorithm [16]

$$w^{k+1} = R_{K_2}(z^k); \tag{11a}$$

$$z^{k+1} = z^k + 2\alpha\Big(R_{K_1}(2w^{k+1} - z^k) - w^{k+1}\Big) \tag{11b}$$

for $k = 0, 1, \ldots$. Starting from any z^0, the sequence w^k will converge to a saddle point $w^* = (x^*, y^*)$ of K. Note that in (11), the roles of K_1 and K_2 can be switched.

Example 4 (Sparse bilinear games): Let $K(x, y) = x^T Ay + b_1^T x + b_2^T y + \beta_x\|x\|_1 - \beta_y\|y\|_1$ where $\|\cdot\|_1$ is the L_1-norm and $\beta_x, \beta_y > 0$. The two norm regulation terms are added to promote sparsity in x^* and y^* for the saddle points $w^* = (x^*, y^*)$ of K. Write $K = K_1 + K_2$ where $K_1 = x^T Ay + b_1^T x + b_2^T y$ and $K_2 = \beta_x\|x\|_1 - \beta_y\|y\|_1$. Then R_{K_1} is given in (10) and $R_{K_2} = \text{prox}_{\beta_x\lambda\|\cdot\|_1} \times \text{prox}_{\beta_y\lambda\|\cdot\|_1}$ where $\text{prox}_{\eta\lambda\|\cdot\|_1}$ for $\eta \in \{\beta_x, \beta_y\}$ is given by the elementwise soft thresholding operator $(s_{\eta\lambda}(v))_i := \max\{v_i - \eta\lambda, 0\} - \max\{-v_i - \eta\lambda, 0\}$ where i indexes the entries of v and $s_{\eta\lambda}(v)$ (see [14]).

Consider the following simple example for $x, y \in \mathbb{R}^3$,

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix},\ b_1 = \begin{bmatrix} -30 \\ -33 \\ -60 \end{bmatrix},\ \text{and } b_2 = \begin{bmatrix} -117 \\ -126 \\ -45 \end{bmatrix}.$$

It can be verified that K_1 has the unique saddle point $(x_1^*, y_1^*) = (30, 124, -73, -5, 7, 7)$. When keeping $\beta_x = \beta_y$ and gradually increasing their value, the saddle point of $K = K_1 + K_2$ starts from (x_1^*, y_1^*) at $\beta_x = \beta_y = 0$, becomes $(0, 0, 5.75, 0, -2.5, 0)$ at $\beta_x = \beta_y = 80$, and eventually becomes all zeros when $\beta_x = \beta_y = 130$.

B. Three-Operator Splitting

When $m = 3$, $K = K_1 + K_2 + K_3$. In this case, a solution algorithm can be developed by using the recently derived Davis-Yin three-operator splitting technique [17]. Suppose K_3 is differentiable; thus, the subdifferential operator T_{K_3} is single-valued. Further assume that T_{K_3} is Lipschitz continuous in (x, y) with the Lipschitz parameter L_3. A point $w^* = (x^*, y^*)$ in $\text{zer}(T_{K_1} + T_{K_2} + T_{K_3})$, i.e., a saddle point of K, can be obtained by $w^* = R_{K_2}(z^*)$ with z^* being a fixed point of the operator

$$S := (2R_{K_1} - I)(2R_{K_2} - I - \gamma T_{K_3} R_{K_2}) - \gamma T_{K_3} R_{K_2}$$

for $\gamma \in (0, 2/L_3)$. Although S may not be nonexpansive in general, $1/2I + 1/2S$ is an averaged operator [17]. Thus, iterations using $1/2I+1/2S$ will converge to a fixed point z^* of $1/2I+1/2S$ and hence of S. This results in the following iterative algorithm:

$$w^{k+1} = R_{K_2}(z^k); \tag{12a}$$

$$z^{k+1} = z^k - w^{k+1} + R_{K_1}\left(2w^{k+1} - z^k - \gamma T_{K_3}(w^{k+1})\right) \tag{12b}$$

for $k = 0, 1, \ldots$. Starting at any z^0, the sequence w^{k+1} generated by (12) converges to a saddle point $w^* = (x^*, y^*)$ of K. The roles of K_1 and K_2 in (12) can be switched.

Example 5: The following example is modified from the multi-channel communication capacity problem in [18]. Denote by $\Delta = \{w \in \mathbb{R}^n \,|\, w_i \geq 0, \forall i,\ \sum_i w_i = 1\}$ the standard simplex in $\mathbb{R}^n$ and consider $x, y \in \mathbb{R}^n$. Let $K(x, y) = \sum_{i=1}^n \log(1 + \beta_i y_i/(\sigma_i + x_i)) + \mu_{\Delta\times\Delta}(x, y) + x^T Ay$ where $\beta_i, \sigma_i > 0$ and $A \in \mathbb{R}^{n\times n}$. Denote the three terms in K as K_1, K_2 and K_3. Then R_{K_1} can be computed by Proposition 3 as K_1 is separable; $R_{K_2} = \Pi_\Delta \times \Pi_\Delta$; K_3 is differentiable and T_{K_3} is Lipschitz continuous with $L_3 = \|A\|$. Therefore, a saddle point of K can be computed by the iteration (12).

IV. Convex-Concave Games on Networks and Their Distributed Solution

In this section, we will formulate convex-concave games on agent networks where the saddle function K is the sum of local saddle functions each depending on the local variables of an agent and some of its neighbors. Using the splitting methods in Section III, we will propose distributed algorithms that can find a saddle point of K through iterations using only locally available information for each agent.

Consider a set of m agents indexed by $[m] := \{1, \ldots, m\}$. Each agent i holds two local variables, $x_i \in \mathbb{R}^{\check{n}_i}$ and $y_i \in \mathbb{R}^{\hat{n}_i}$, as well as a local extended-real-valued saddle function K_i that depends on not only (x_i, y_i) but also the variables of its neighboring agents, specifically, x_j for $j \in \check{\mathcal{N}}_i^+$ and y_l for $l \in \hat{\mathcal{N}}_i^+$. The two (possibly different) sets $\check{\mathcal{N}}_i^+, \hat{\mathcal{N}}_i^+ \subset [m]$ are called the x-in-neighbor and y-in-neighbor sets of agent i, respectively. As a result, the local saddle function K_i of agent i is of the form $K_i(x_i, (x_j)_{j\in\check{\mathcal{N}}_i^+}, y_i, (y_l)_{l\in\hat{\mathcal{N}}_i^+})$, which is convex in $(x_i, (x_j)_{j\in\check{\mathcal{N}}_i^+})$ for each fixed $(y_i, (y_l)_{l\in\hat{\mathcal{N}}_i^+})$ and concave in $(y_i, (y_l)_{l\in\hat{\mathcal{N}}_i^+})$ for each fixed $(x_i, (x_j)_{j\in\mathcal{N}_i^+})$.

The above dependency relation can be modeled by two directed graphs both with the vertex set $[m]$. The x-dependency graph has the edge set $\mathcal{E}_x \subset [m] \times [m]$ such that an edge $(j, i) \in \mathcal{E}_x$ exists if and only if agent i's function K_i depends on the variable x_j of agent j. Thus, agent i's x-out-neighbor set can be defined as $\check{\mathcal{N}}_i^- = \{j | (i, j) \in \mathcal{E}_x\}$ and its x-in-neighbor set defined above can be expressed as $\check{\mathcal{N}}_i^+ = \{j | (j, i) \in \mathcal{E}_x\}$. Similarly, we can define the y-dependency graph $([m], \mathcal{E}_y)$, the y-out-neighbor set $\hat{\mathcal{N}}_i^- = \{l | (i, l) \in \mathcal{E}_y\}$ of agent i and have its y-in-neighbor set expressed as $\hat{\mathcal{N}}_i^+ = \{l | (l, i) \in \mathcal{E}_y\}$.

Denote $x := (x_i)_{i\in[m]}$ and $y := (y_i)_{i\in[m]}$, i.e., the concatenated vectors of all x_i's and y_i's, respectively. The problem we aim to solve is:

$$\text{Find a saddle point of } K(x, y) := \sum_{i\in[m]} K_i. \tag{13}$$

Assumption 1: Assume that each K_i is closed and proper, and that $K = \sum_i K_i$ has at least one saddle point.

We note that in the above assumption it is not necessary that saddle points exist for any of the K_i.

Assumption 2 (Communicability): Two agents i and j can communicate with each other whenever $(i, j) \in \mathcal{E}_x \cup \mathcal{E}_y$ or $(j, i) \in \mathcal{E}_x \cup \mathcal{E}_y$, i.e., each agent can communicate bi-directionally with all of its neighbors.

Our goal is to solve the problem (13) in a distributed iterative way that, at each iteration, each agent only uses the variables from itself and its neighbors to update and the iterated values in aggregate converge to a saddle point $z^* = (x^*, y^*)$ of K. Towards this goal, for each agent i, we introduce the local auxiliary variables $(x_{ij})_{j\in\check{\mathcal{N}}_i^+}$ and $(y_{il})_{l\in\hat{\mathcal{N}}_i^+}$ representing the copies held by agent i for the x-variables of its x-in-neighbors and the y-variables of its y-in-neighbors, respectively. Denote by $\mathbf{x}_i := (x_i, (x_{ij})_{j\in\check{\mathcal{N}}_i^+}) \in \mathbb{R}^{\check{N}_i}$ and $\mathbf{y}_i := (y_i, (y_{il})_{l\in\hat{\mathcal{N}}_i^+}) \in \mathbb{R}^{\hat{N}_i}$ the augmented x-variable and y-variable of agent i, and by $\mathbf{x} := (\mathbf{x}_i)_{i\in[m]} \in \mathbb{R}^{\check{N}}$ and $\mathbf{y} := (\mathbf{y}_i)_{i\in[m]} \in \mathbb{R}^{\hat{N}}$ their concatenations.

Let $\mathcal{A}_x := \{\mathbf{x} \,|\, x_i = x_{ji},\ \forall i \in [m], j \in \check{\mathcal{N}}_i^-\} \subset \mathbb{R}^{\check{N}}$ and $\mathcal{A}_y := \{\mathbf{y} \,|\, y_i = y_{li},\ \forall i \in [m], l \in \hat{\mathcal{N}}_i^-\} \subset \mathbb{R}^{\hat{N}}$ be the x-consensus and y-consensus subspaces for $\mathbf{x}$ and $\mathbf{y}$, respectively. Then, the problem (13) is equivalent to

$$\text{Find a saddle point of } K(\mathbf{x}, \mathbf{y}) + \mu_{\mathcal{A}_x\times\mathcal{A}_y}(\mathbf{x}, \mathbf{y}). \tag{14}$$

where $K = K_1(\mathbf{x}_1, \mathbf{y}_1) + \cdots + K_m(\mathbf{x}_m, \mathbf{y}_m)$ is separable. Note that R_K can be computed using Proposition 3 and $R_{\mu_{\mathcal{A}_x\times\mathcal{A}_y}} = \Pi_{\mathcal{A}_x} \times \Pi_{\mathcal{A}_y}$. Here, $\Pi_{\mathcal{A}_x}(\mathbf{x})$ is the projection of $\mathbf{x}$ onto the subspace $\mathcal{A}_x$ defined as follows. First, let

$$\bar{x}_i := (x_i + \textstyle\sum_{j\in\check{\mathcal{N}}_i^-} x_{ji})/(1 + |\check{\mathcal{N}}_i^-|),\ \ \forall i \in [m]. \tag{15}$$

Then, $\bar{\mathbf{x}} := \Pi_{\mathcal{A}_x}(\mathbf{x})$ is given by $\bar{\mathbf{x}} = (\bar{\mathbf{x}}_i)_{i\in[m]}$ where $\bar{\mathbf{x}}_i = (\bar{x}_i, (\bar{x}_{ij})_{j\in\check{\mathcal{N}}_i^+})$ satisfies $\bar{x}_{ij} = \bar{x}_j$ for $j \in \check{\mathcal{N}}_i^+$. Similarly, we can define the projection of $\mathbf{y}$ onto $\mathcal{A}_y$, $\bar{y} := \Pi_{\mathcal{A}_y}(\mathbf{y})$.

A. Synchronous Implementation

Adopting the widely used two-operator splitting method, (generalized) Douglas-Rachford (D-R) splitting algorithm in (11), we have the following synchronous iterations (summarized in Algorithm 1) for $k = 0, 1, \ldots$,

$$\mathbf{w}_i^{k+1} = \bar{\mathbf{z}}_i^k,\ \ i \in [m]; \tag{16a}$$

$$\mathbf{z}_i^{k+1} = \mathbf{z}_i^k + 2\alpha\Big(R_{K_i}(2\mathbf{w}_i^{k+1} - \mathbf{z}_i^k) - \mathbf{w}_i^{k+1}\Big),\ \ i \in [m]. \tag{16b}$$

where $\alpha \in (0, 1)$, $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)$, and $\bar{\mathbf{z}}^k = \Pi_{\mathcal{A}_x} \times \Pi_{\mathcal{A}_y}(\mathbf{z}^k)$. Under Assumption 2, the iterations $\mathbf{w}^k$ generated by the algorithm (16) will converge to a solution of problem (14).

Algorithm 1 Synchronous Douglas-Rachford Algorithm

Initialize $\mathbf{z}^0$, and let $k \leftarrow 0$
repeat
  for $i = 1, \ldots, m$ **do**
    $\mathbf{w}_i^{k+1} \leftarrow \bar{\mathbf{z}}_i^k$
  for $i = 1, \ldots, m$ **do**
    $\mathbf{z}_i^{k+1} \leftarrow \mathbf{z}_i^k + 2\alpha\left(R_{K_i}(2\mathbf{w}_i^{k+1} - \mathbf{z}_i^k) - \mathbf{w}_i^{k+1}\right)$
  $k \leftarrow k + 1$
until $|\mathbf{z}^k - \mathbf{z}^{k-1}|$ is sufficiently small
return $\mathbf{w}^k$

The step (16a) requires bi-directional communications between neighboring agents: each agent i first collects variables x_{ji}'s from its x-out-neighbors $j \in \check{\mathcal{N}}_i^-$ and variables y_{li}'s from its y-out-neighbors $l \in \hat{\mathcal{N}}_i^-$; it then computes $\bar{x}_i$ by (15) and $\bar{y}_i = (y_i + \sum_{l\in\hat{\mathcal{N}}_i^-} y_{li})/(1 + |\hat{\mathcal{N}}_i^-|)$; finally it sends $\bar{x}_i$ and $\bar{y}_i$ back to its out-neighbors to be the updated values of x_{ji}'s and y_{li}'s as part of their augmented variables, respectively. In other words, the step (16a) requires two synchronous rounds of communications among neighboring agents. The step (16b) does not require inter-agent communications.

In some cases the resolvent R_{K_i} may not be easily computed directly. An example is $K_i = K_{i,1} + K_{i,3}$ where $K_{i,3}$ is the differential part and $K_{i,1}$ represents the nonsmooth part (e.g., the indicator function of the local feasible set). Since $\sum_{i\in[m]} K_{i,3}$ is separable in variables $\mathbf{z}_i, i \in [m]$, by treating it as the "K_3" in the Davis-Yin three-operator splitting, we have the following iterations from (12): for $k = 0, 1, \ldots$,

$$\mathbf{w}_i^{k+1} = \bar{\mathbf{z}}_i^k,\ \ i \in [m]; \tag{17a}$$

$$\mathbf{z}_i^{k+1} = \mathbf{z}_i^k - \mathbf{w}_i^{k+1} + R_{K_{i,1}}\Big(2\mathbf{w}_i^{k+1} - \mathbf{z}_i^k - \gamma\nabla K_{i,3}(\mathbf{w}_i^{k+1})\Big),\ \ i \in [m]. \tag{17b}$$

where $\gamma \in (0, 2/L_3)$. Here L_3 is the largest of the Lipschitz constants of the derivatives of $K_{i,3}$ w.r.t. $\mathbf{z}_i$ for $i \in [m]$. Under Assumption 2, the iterations $\mathbf{w}^k$ generated by the algorithm (17) will converge to a solution of problem (14). The algorithm summary is the same as Algorithm 1 except that Step 6 is replaced by (17b), hence it is omitted here.

B. Randomized Implementation

Here by utilizing the following theorem, we derive an asynchronous version of the iterations above.

Theorem 2 ([19]): Let $S : \mathbb{R}^n \to \mathbb{R}^n$ be an α-averaged operator with $\mathrm{Fix}(S) \neq \emptyset$. Partition $x \in \mathbb{R}^n$ into $(x_1, \ldots, x_m)$ and Sx into $(S_1 x, \ldots, S_m x)$ where $x_i, S_i x \in \mathbb{R}^{n_i}$ for $i \in [m]$. Consider the following iteration. At each step $k = 0, 1, \ldots$, first an index $i^k \in [m]$ is chosen randomly and independently with the probabilities $\mathbb{P}\left(i^k = i\right) = p_i \geq \varepsilon$, $i \in [m]$, for some positive ε; then x^k is updated to x^{k+1} where $x_{i^k}^{k+1} = S_{i^k} x^k$ and $x_\ell^{k+1} = x_\ell^k$ for $\ell \neq i^k$. Then, x^k converges almost surely to some $x^* \in \mathrm{Fix}(S)$ as $k \to \infty$.

Applying Theorem 2 to iterations (16), at round k, suppose only one agent $i \in [m]$ is activated with the i.i.d. probability $p_i \geq \varepsilon > 0$ to perform the following update:

$$\mathbf{w}_i^{k+1} = \bar{\mathbf{z}}_i^k; \tag{18a}$$

$$\mathbf{z}_i^{k+1} = \mathbf{z}_i^k + 2\alpha\Big(R_{K_i}(2\mathbf{w}_i^{k+1} - \mathbf{z}_i^k) - \mathbf{w}_i^{k+1}\Big). \tag{18b}$$

To compute $\bar{\mathbf{z}}_i^k = \left(\bar{x}_i^k, \bar{y}_i^k, (\bar{x}_j^k)_{j\in\check{\mathcal{N}}_i^+}, (\bar{y}_l^k)_{l\in\hat{\mathcal{N}}_i^+}\right)$ in (18a), it is required that agent i communicates between its out-neighbors in $\check{\mathcal{N}}_i^- \cup \hat{\mathcal{N}}_i^-$ (to compute $\bar{x}_i^k$ and $\bar{y}_i^k$ and send them back to out-neighbors), as well as all of its in-neighbors $j \in \check{\mathcal{N}}_i^+$ and $l \in \hat{\mathcal{N}}_i^-$ (to use their $\bar{x}_j$ and $\bar{y}_l$ for updating x_{ij} and y_{il}). The latter requires agent i's in-neighbors to further communicate with their respective out-neighbors. To avoid this, we can let each agent i hold two extra variables: $\bar{x}_i \in \mathbb{R}^{\check{n}_i}$, which is the latest average of x_i and its copies x_{ji} held by agent i's x-out-neighbors; and $\bar{y}_i \in \mathbb{R}^{\hat{n}_i}$, which is the latest averages of y_i and its copies y_{li} held by agent i's y-out-neighbors. With these modification, an asynchronous version of the Algorithm 1 is summarized in the following Algorithm 2.

In each round of Algorithm 2, only the activated agent i communicates with its x- and y- in-neighbors in (i) step 7 to collect the latest averages $\bar{x}_j^k$, $\bar{y}_l^k$; and (ii) steps 11-14 to send back the differences $x_{ij}^{k+1} - x_{ij}^k$, $y_{il}^{k+1} - y_{il}^k$ while all the other agents remain idle. By Theorem 2, starting from any initial $\mathbf{z}^0$, the sequence $\mathbf{w}^k$ generated by Algorithm 2 converges with probability one to a solution $\mathbf{w}^* = (\mathbf{x}^*, \mathbf{y}^*)$ to the problem (14).

Remark 1: By carrying out the update (17b) for a randomly activated agent i, the algorithm (17) can also be implemented in a randomized way.

V. NUMERICAL EXAMPLES

In this section, we present the simulation results of a convex-concave game with 7 agents to demonstrate the effectiveness of the proposed Algorithms 1 and 2.

Algorithm 2 Randomized Douglas-Rachford Algorithm

1: Choose any $\mathbf{z}^0$, and let $k \leftarrow 0$
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \bar{x}_i^0 \leftarrow (x_i^0 + \sum_{j\in\check{\mathcal{N}}_i^-} x_{ji}^0)/(1 + |\check{\mathcal{N}}_i^-|)$
4: $\quad \bar{y}_i^0 \leftarrow (y_i^0 + \sum_{l\in\hat{\mathcal{N}}_i^-} y_{li}^0)/(1 + |\hat{\mathcal{N}}_i^-|)$
5: **repeat**
6: $\quad$ Pick $i \in [m]$ with i.i.d. probability $p_i > 0$
7: $\quad \mathbf{w}_i^{k+1} \leftarrow \left(\bar{x}_i^k, \bar{y}_i^k, (\bar{x}_j^k)_{j\in\check{\mathcal{N}}_i^+}, (\bar{y}_l^k)_{l\in\hat{\mathcal{N}}_i^+}\right)$
8: $\quad \mathbf{z}_i^{k+1} \leftarrow \mathbf{z}_i^k + 2\alpha\left(R_{K_i}(2\mathbf{w}_i^{k+1} - \mathbf{z}_i^k) - \mathbf{w}_i^{k+1}\right)$
9: $\quad \bar{x}_i^{k+1} \leftarrow \bar{x}_i^k + (x_i^{k+1} - x_i^k)/(|\check{\mathcal{N}}_i^-| + 1)$
10: $\quad \bar{y}_i^{k+1} \leftarrow \bar{y}_i^k + (y_i^{k+1} - y_i^k)/(|\hat{\mathcal{N}}_i^-| + 1)$
11: $\quad$ **for** $j \in \check{\mathcal{N}}_i^+$ **do**
12: $\quad\quad \bar{x}_j^{k+1} \leftarrow \bar{x}_j^k + (x_{ij}^{k+1} - x_{ij}^k)/(|\check{\mathcal{N}}_j^-| + 1)$
13: $\quad$ **for** $l \in \hat{\mathcal{N}}_i^+$ **do**
14: $\quad\quad \bar{y}_l^{k+1} \leftarrow \bar{y}_l^k + (y_{il}^{k+1} - y_{il}^k)/(|\hat{\mathcal{N}}_l^-| + 1)$
15: $\quad k \leftarrow k + 1$
16: **until** k is sufficiently large
17: **return** $\mathbf{w}^k$

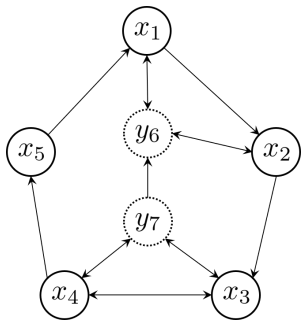

Fig. 1: Dependence graphs $\mathcal{E}_x \cup \mathcal{E}_y$ of the simulation example.

As shown in Fig. 1, each agent i holds a local variable x_i or y_i (but not both), and each local saddle function K_i depends on the variables of its neighboring agents as depicted by the arrows and is further assumed to be a quadratic function of the form (7) with randomly generated parameters. Thus, the problem is a quadratic convex-concave game played between two groups of agents with coupled payoff functions, where the x-group $\{1, 2, 3, 4, 5\}$ cooperates to minimize the global payoff function $K = \sum_{i=1}^7 K_i$ while the y-group $\{6, 7\}$ cooperates to maximize it.

We further assume that, each K_i associated with the agents in the x-group is strictly convex in its own and its neighbors' x-variables and linear in its neighbors' y-variables; and each K_i belonging to the agents in the y-group is strictly concave in its own and its neighbors' y-variables and linear in its neighbors' x-variables. Therefore each K_i may have no saddle points but their sum K has a unique saddle point (x^*, y^*).

We first apply Algorithm 1 to solve this problem with four sets of parameters, $(\lambda, \alpha) = (0.01, 0.5)$, $(1, 0.5)$, $(1, 0.98)$, $(100, 0.5)$, respectively. As shown in Fig. 2, starting at an arbitrary initial point $\mathbf{z}^0$, the iteration values $(\bar{x}^k, \bar{y}^k) :=$

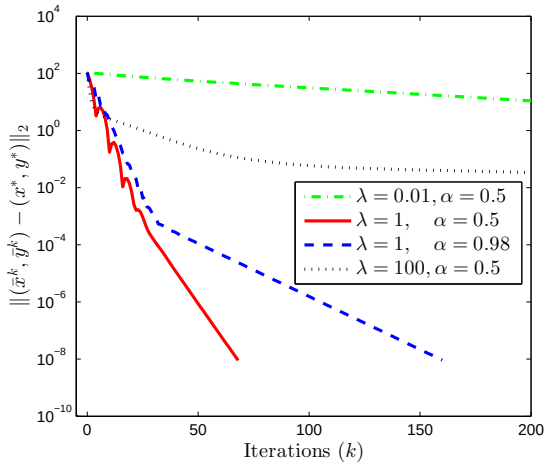

Fig. 2: Comparison of the convergence rates when applying Algorithm 1 with different (λ, α) to solve the quadratic game on the 7-agent network.

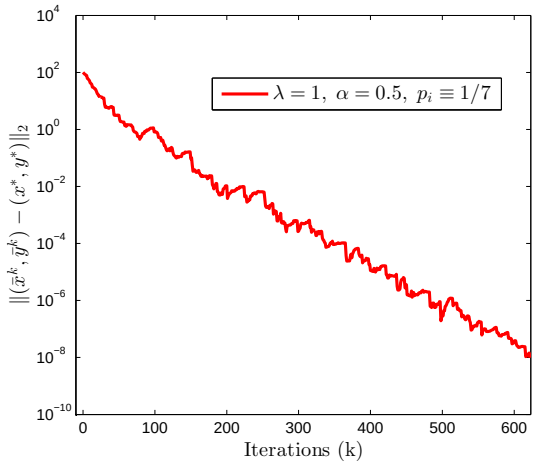

Fig. 3: Convergence of applying Algorithm 2 to solve the same quadratic minimax problem on networks.

$((\bar{x}_i)_{i=1}^5, (\bar{y}_i)_{i=6}^7)$ extracted from the computation of $\bar{\mathbf{z}}$ in (16a) converge to the unique solution (x^*, y^*) in all cases. For this example, the convergence with a moderate $\lambda = 1$, which is used to construct the resolvent operators, is faster than that with the smaller λ ($\lambda = 0.01$) and the larger λ ($\lambda = 100$), and the convergence rate is linear.

Next we use Algorithm 2 to solve the same problem with agent activation probability $p_i = 1/7, i = 1, \ldots, 7$, i.e., each agent is activated to update at each round with i.i.d, equal probabilities. Compared to Algorithm 1 using the same parameters $(\lambda, \alpha) = (1, 0.5)$, Algorithm 2 requires much more iterations to achieve the same convergence accuracy (10^{-8}) since at each round only one agent performs update in contrast to 7 agents in Algorithm 1.

VI. CONCLUSIONS AND FUTURE WORKS

This paper is concerned with developing distributed solutions to the convex-concave games on agent networks. Synchronous and randomized solution algorithms are proposed, both of which have low communication and storage overheads. In future work we will quantify the proposed algorithms' convergence rates in terms of the local payoff functions' convexity/concavity and the dependency graphs' connectivity.

REFERENCES

[1] J. Hu, Y. Xiao, and J. Liu, "Distributed algorithms for solving locally coupled optimization problems on agent networks," in *Proc. 57th IEEE Int. Conf. Decision and Control*, 2018, to appear.

[2] G. Scutari, D. P. Palomar, F. Facchinei, and J.-s. Pang, "Convex optimization, game theory, and variational inequality theory," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 35–49, 2010.

[3] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta numerica*, vol. 14, pp. 1–137, 2005.

[4] K. J. Arrow, L. Hurwicz, and H. Uzawa, "Studies in linear and non-linear programming," 1958.

[5] M. Rozložník and V. Simoncini, "Krylov subspace methods for saddle point problems with indefinite preconditioning," *SIAM journal on matrix analysis and applications*, vol. 24, no. 2, pp. 368–391, 2002.

[6] Z.-Z. Bai and G. H. Golub, "Accelerated hermitian and skew-hermitian splitting iteration methods for saddle-point problems," *IMA Journal of Numerical Analysis*, vol. 27, no. 1, pp. 1–23, 2007.

[7] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical programming*, vol. 103, no. 1, pp. 127–152, 2005.

[8] A. Nemirovski, "Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 229–251, 2004.

[9] A. Nedić and A. Ozdaglar, "Subgradient methods for saddle-point problems," *Journal of optimization theory and applications*, vol. 142, no. 1, pp. 205–228, 2009.

[10] D. Mateos-Núnez and J. Cortés, "Distributed saddle-point subgradient algorithms with laplacian averaging," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2720–2735, 2017.

[11] R. T. Rockafellar, "Monotone operators associated with saddle-functions and minimax problems," *Nonlinear functional analysis*, vol. 18, no. Part 1, pp. 397–407, 1970.

[12] M. Sion *et al.*, "On general minimax theorems." *Pacific Journal of mathematics*, vol. 8, no. 1, pp. 171–176, 1958.

[13] E. K. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. Comput. Math*, vol. 15, no. 1, pp. 3–43, 2016.

[14] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[15] D. W. Peaceman and H. H. Rachford, Jr, "The numerical solution of parabolic and elliptic differential equations," *Journal of the Society for industrial and Applied Mathematics*, vol. 3, no. 1, pp. 28–41, 1955.

[16] J. Douglas and H. H. Rachford, "On the numerical solution of heat conduction problems in two and three space variables," *Trans. American Mathematical Society*, vol. 82, no. 2, pp. 421–439, 1956.

[17] D. Davis and W. Yin, "A three-operator splitting scheme and its optimization applications," *Set-Valued and Variational Analysis*, vol. 25, no. 4, pp. 829–858, 2017.

[18] A. Ghosh and S. Boyd, "Minimax and convex-concave games," *Lecture Notes for Course EE392, Stanford Univ., Stanford, CA*, 2003.

[19] P. Bianchi, W. Hachem, and F. Iutzeler, "A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2947–2957, 2016.