An ADMM Approach to the Problem of Nash Equilibrium Seeking

An ADMM Approach to the Problem of Distributed Nash Equilibrium Seeking

Farzad Salehisadaghiani*, Wei Shi† and Lacra Pavel*
*Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada
(e-mails: farzad.salehisadaghiani@mail.utoronto.ca, pavel@control.utoronto.ca)
†School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287, USA
(email: wilbur.shi@asu.edu)

***Abstract*—In this paper, we consider the problem of finding a Nash equilibrium in a multi-player game over generally connected networks. This model differs from a conventional setting in that the communication graph is not necessarily the same as the players' cost dependency graph, and thus players have incomplete information on the actions of their opponents. We develop a relatively fast algorithm within the framework of inexact-ADMM, based on local information exchange between the players. We prove convergence to a Nash equilibrium for fixed step-sizes and analyze its convergence rate. Numerical simulations illustrate its benefits when compared to a consensus-based gradient type algorithm.**

I. Introduction

In this paper we consider the problem of finding a Nash equilibrium (NE) in a distributed game played by a network of players. There is a close connection between this problem and a distributed optimization problem (DOP). In a distributed optimization problem with N agents that communicate over a connected graph, it is desired to minimize a global objective as follows:

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) := \sum_{i=1}^{N} f_i(x) \\ \text{subject to} & x \in \Omega. \end{cases} \tag{1}$$

The agents cooperatively solve (1) over a common optimization variable x. In other words, in a DOP all agents are serving in the public interest so that they reduce the global loss. However, there are many real-world applications that involve selfishness of the players (agents). In these applications, players *selfishly* desire to optimize their own performance even though the global objective may not be minimized, hence play a non-cooperative game. The game is played such that each player (agent) i aims to minimize his own cost function J_i selfishly with respect to (w.r.t) his own action x_i, given a profile of other players' actions, except himself, x_{-i},

$$\begin{cases} \underset{x_i}{\text{minimize}} & J_i(x_i, x_{-i}) \\ \text{subject to} & x_i \in \Omega_i \end{cases} \quad \forall i = 1, \dots, N. \tag{2}$$

where Ω_i is the feasible action (strategy) set of player i, usually assumed to be compact. A Nash equilibrium (NE) $x^* = [x_1^*; x_2^*; \cdots; x_n^*]$ is a solution to the set of coupled optimization problems illustrated in (2).

We are interested in finding the NE of this game as a game theoretic problem defined over a network and solved by N players (agents) selfishly but collaboratively. In the context, by 'selfishly', we mean that for all i, player i minimizes J_i with respect to x_i, since player i is only in charge of his own decision and cannot adjust others' decision directly, while all other players minimize their own cost functions without help in minimizing J_i. By 'collaboratively', we mean that, for all i, player i is willing to share x_i and his estimation of x_{-i} to his neighbours, and at the same time player i receives estimations of x_i and x_{-i} from his neighbours. Note that information/estimation sharing is necessary since player i's cost function is dependent on (coupled to) the other players' decisions, and thus his decision will be affected indirectly by other players' decisions.

Considering the similarities and differences between DOP and distributed NE seeking, we aim to employ an optimization technique referred to as *Alternating Direction Method of Multipliers* (ADMM) to find a Nash equilibrium (NE) in a game. ADMM algorithms have been developed in 1970s to find an optimal point of DOP. This method has become widely used after its re-introduction in [1], [2], [3], [4] . In DOP ADMM takes advantage of dual decomposition and augmented Lagrangian methods. Dual decomposition is a special case of a dual ascent method for solving an optimization problem when the objective function is separable with respect to (w.r.t.) variable x, i.e., $f(x) := \sum_{i=1}^{N} f_i(x_i)$

where $x = [x_1, \ldots, x_N]^T$. This decomposition leads to N parallel dual ascent problems whereby each is to be solved for x_i, $i = 1, \ldots, N$. This parallelism makes the convergence faster. The augmented Lagrangian method involves a penalty term added to the Lagrangian and as a result is more robust and relaxes the assumptions in the dual ascent method. In the coupled DOP (1), an augmented space of local variables with consensus constraints is considered, in order to separate the objective functions, and each agent optimizes his cost function over the *full* argument. Thus the obtained lifted cost function is separable and can be decomposed as a sum of *decoupled cost functions, each being individually convex in its full argument*. This leads to ADMM algorithms, under convexity of the individual cost functions with respect to the *full* argument.

In this work, we aim to exploit the benefits of ADMM in the context of finding an NE of a game (2) over a communication graph. However, there are several differences when compared to a DOP. Here are the difficulties that we need to overcome:

- A Nash game can be seen as a set of parallel coupled optimization problems, each of them associated with the minimization of a player's own cost function w.r.t. his own decision variable x_i, which is *a part* of the full vector (profile) x. However, each optimization problem is dependent on the solution of the other parallel problems. This leads to N Lagrangians whereby each is dependent on the other players' variables.
- Each player i updates only his own variable x_i, however, he also requires an estimate of all other variables $[x_j]_{j=1,\ldots,N,\, j\neq i}$, or x_{-i}, in order to solve his optimization problem. This leads to an extra step in the algorithm.

Thus, while the game setting has an inherent distributed structure (since each player optimizes his own cost function), individual (player-by-player) optimization is over *his own* action. In contrast to DOP, in a game each player's individual action is only *part of the full action profile* and his cost function is coupled to his opponents' actions, which are under their decision. Individual convexity properties w.r.t. the full argument as used in DOP, are too restrictive, unless the game is separable to start with, which is a trivial case. Rather, a typical assumption is individual convexity of each cost w.r.t. his own decision variable x_i, and monotonicity (strict, strong) of the pseudo-gradient (used in gradient-based algorithms for Nash seeking). When we consider an augmented space of actions and estimates of others' actions, the corresponding monotonicity assumption is not automatically satisfied, as it does in DOP. The corresponding monotonicity assumption we use here is cocoercivity of the extended pseudo-gradient.

Related Works. Our work is related to the literature on distributed Nash equilibrium (NE) seeking or NE computation, [5], [6]. Computation of an NE in a distributed manner has recently drawn attention due to many real-world applications. To name only a few, congestion games, [7], sensor network coordination [8], flow control [9], optical networks, [10], [11]. For NE computation in games with monotone pseudo-gradient [12] proposes a distributed algorithm based on proximal best-response map that is shown to be contractive, while [13] proposes a regularized gradient algorithm with diminishing step-sizes under Lipschitz continuity. Reference [14] considers a class of networked aggregative games where the cost functions are quadratic. Computation of a time-varying NE is considered in [15] in zero-sum networked games consisting of two subnetworks with shared objectives. A stochastic gradient-based algorithm for non-cooperative NE seeking is proposed in [16]. Equilibrium seeking of general non-cooperative games is studied in [17], where network unreliability and communication asynchronization issues are mainly addressed. Recently, [18] combines strong monotonicity and Lipschitz continuity to ensure cocoercivity of the pseudo gradient, and proposes primal-dual gradient methods for distributed generalized NE computation with affine constraints.

In above mentioned references, the underlying graph of the game is determined by the dependency of the cost functions over players. The issue of *incomplete information on the opponents' decisions* is not considered. In other words, if all cost functions J_i are generally coupled, the communication graph would need to be a complete graph, and players would need observations of *all their opponents' actions* to compute an NE. However, full, or all-to-all, communication is impractical in many engineering systems (e.g., multi-agent systems, ad hoc networks) and may be inefficient. As pointed out in [19], [20], in large scale multi-agent systems, a player is inherently limited to being observable and communicable to a few other players and has relatively weak computational capabilities.

Recently the NE seeking problem with incomplete information on the opponents' decisions over general graphs has attracted increased attention. The assumption is that players cannot observe all the others' actions directly but agree to exchange information between neighbours over a communication graph. The following references deploy consensus techniques to adapt the NE computation algorithms so that they work *over arbitrary, connected communication* graphs. For the special class of *aggregative* games, where the coupling to opponents' actions is via a common aggregated variable, a consensus-based projected gradient algorithm with diminishing step-sizes was designed in [21], based on a gossiping protocol. The idea on solving games without using all-to-all decision information was generalized in [22], where players' cost functions depend on others' actions in a general manner. The Nash equilibrium was characterized by a variational inequality and an asynchronous gossip-based projected-gradient algorithm was proposed, over a connected communication graph. The adaptation of the algorithm to games with partial coupling was considered in [23]. For diminishing step-sizes, almost-sure convergence to the NE was shown under strict monotonicity of the pseudo-gradient, while, for constant step-sizes, under strong monotonicity, only convergence to a neighbourhood of the NE was shown. We note that very recently [24] consider a similar situation in continuous-time, but for games with unconstrained action sets. A continuous-time NE seeking dynamics is proposed, based on consensus and gradient-type dynamics, which is proved to be exponential stable via a singular perturbation analysis.

Contributions. Different from [24], based on an ADMM approach, we develop a discrete-time NE seeking algorithm for non-cooperative games with compact action sets. This ADMM approach leads to an algorithm with an extra correction term on the actions' and estimates' components, which is beneficial for its convergence rate. Different from [22], [23], the extra correction term facilitates the algorithm's convergence for fixed step-sizes.

First, we reformulate the problem of finding an NE of a game and relate to a set of distributed optimization problems with consensus constraints. This is achieved by introducing local estimates of players' actions and using (virtual) constraints for the consensus of these estimates. We then use the augmented Lagrangian method to derive an ADMM-type algorithm for this set of problems, exploiting the use of a dummy variable to make the constraints separable. This technique can be used for any game which satisfies a set of relatively mild assumptions. The algorithm takes advantage of the speed and robustness of the classical ADMM and reduces the computational costs by using a linear approximation in players' action update rule (inexact-ADMM). Every player updates his action as well as his estimates of the other players' actions by this synchronous, inexact ADMM-type algorithm, using his own estimates and the received information from his neighbours. Compared to consensus-based gradient-type algorithms such as [17], [21]–[23], our ADMM algorithm has an extra penalty term (could be seen as an extra state) which is updated through the iterations and improves the convergence and convergence rate for fixed step-sizes. We prove convergence of the proposed algorithm towards the NE under a cocoercivity assumption on the extended pseudo-gradient, similar to the strong monotonicity assumption in [24]. Moreover, we study its convergence rate and furthermore, we show via simulations that it converges significantly faster than a consensus-based gradient method. To the best of our knowledge, this is the first work on developing an ADMM-type algorithm for distributed NE seeking. A short version of this work without poofs appears in [25], [26].

The paper is organized as follows. The problem statement and assumptions are provided in Section 2. In Section 3, an inexact ADMM-type algorithm is developed. Convergence of the algorithm to a Nash equilibrium and its convergence rate is discussed in Section 4. Simulation results are given in Section 5 and conclusions in Section 6.

A. Notations and Background

A vector $x \in \mathbb{R}^n$ may be represented in multiple equivalent ways $x = [x_1, \ldots, x_n]^T = [x_1; \ldots; x_n]$, or $x = [x_i]_{i=1,\ldots,n}$, or $x = (x_i, x_{-i})$. All vectors are assumed to be column vectors. Given a vector $x \in \mathbb{R}^n$, x^T denotes its transpose and $\|x\| = \sqrt{x^T x}$ denotes its Euclidean norm. Given a symmetric $n \times n$ matrix A, $\|x\|_A$ denotes the weighted norm $\|x\|_A := \sqrt{x^T A x}$. Denote $\mathbf{1}_n = [1, 1, \ldots, 1]^T \in \mathbb{R}^n$ and $\mathbf{0}_n = [0, 0, \ldots, 0]^T \in \mathbb{R}^n$. Let e_i denote the $N \times 1$, i-th unit vector in $\mathbb{R}^n$, whose i-th element is 1 and the rest are 0. Given a vector $x \in \mathbb{R}^n$, the $n \times n$ diagonal matrix with the elements of x, $x_1, \ldots, x_n$ on the diagonal is denoted as $diag(x) = diag(x_1, \ldots, x_n) = diag([x_i]_{i=1,\ldots,n})$. Similarly, given A_i, $i = 1, \ldots, n$ as $(p \times q)$ matrices, denote by $blkdiag([A_i]_{i=1,\ldots,n})$ the $(np \times nq)$ block-partitioned matrix with A_i on the block-diagonal. I_n denotes the identity matrix in $\mathbb{R}^{n \times n}$. The Kronecker product of matrices A and B is denoted by $A \otimes B$. Given matrices $A, B \in \mathbb{R}^{n \times n}$, $A \succ 0$ ($A \succeq 0$) denotes that A is positive (semi-)definite, while $A \succ B$ ($A \succeq B$) denotes that $A - B$ is positive (semi-)definite, respectively. $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ represent the minimum and the maximum eigenvalue of matrix A, respectively. For every $a \in \mathbb{R}^n$, $b \in \mathbb{R}^n$ and $\rho > 0$,

$$-\frac{1}{2\rho}\|a\|^2 - \frac{\rho}{2}\|b\|^2 \leq a^T b \leq \frac{1}{2\rho}\|a\|^2 + \frac{\rho}{2}\|b\|^2. \quad (3)$$

For every $a, b, c \in \mathbb{R}^n$ and $n \times n$ matrix $A \succeq 0$,

$$(a-b)^T A(a-c) = \frac{1}{2}\|a-c\|_A^2 + \frac{1}{2}\|a-b\|_A^2 - \frac{1}{2}\|b-c\|_A^2. \quad (4)$$

Given a set $\Omega \in \mathbb{R}^n$, $|\Omega|$ denotes the cardinality of Ω. The Euclidean projection of $x \in \mathbb{R}^n$ onto $\Omega \subset \mathbb{R}^n$ is denoted by $T_\Omega\{x\}$. Denote by prox_g^a the proximal operator for function g with a constant a, defined as follows:

$$\text{prox}_g^a\{s\} := \arg\min_x \{g(x) + \frac{a}{2}\|x - s\|^2\}. \quad (5)$$

Let $\mathcal{I}_\Omega(x) := \begin{cases} 0 & \text{if } x \in \Omega \\ \infty & \text{otherwise} \end{cases}$, be the indicator function of a set Ω. Then, $\text{prox}_{\mathcal{I}_\Omega}^a\{\cdot\} = T_\Omega\{\cdot\}$.

For an undirected graph $G(V, E)$, we denote by:

- V: Set of vertices in G,
- $E \subseteq V \times V$: Set of all edges in G. $(i, j) \in E$ if and only if i and j are connected by an edge,
- $N_i := \{j \in V | (i, j) \in E\}$: Set of neighbours of i in G,
- $A := [a_{ij}]_{i,j \in V}$: Adjacency matrix of G where $a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ otherwise,
- $\mathbb{D} := \text{diag}(|N_1|, \ldots, |N_N|)$: Degree matrix of G, $d = trace(\mathbb{D}) = \sum_{i=1}^N |N_i|$,
- $L : \mathbb{D} - A$: Laplacian matrix of G,
- $L_N := \mathbb{D}^{-\frac{1}{2}} L \mathbb{D}^{-\frac{1}{2}}$: Normalized Laplacian of G if G has no isolated vertex.

The following hold for a graph G with no isolated vertex: $\mathbb{D} \succ 0$, $\lambda_{\max}(L_N) \leq 2$ and $2\mathbb{D} - L = \mathbb{D}^{\frac{1}{2}}(2I - L_N)\mathbb{D}^{\frac{1}{2}} \succeq 0$, [27]. For a connected and undirected graph G with n vertices, $L \succeq 0$, 0 is a simple eigenvalue of L with the eigenvector $\mathbf{1}_n$ and $L\mathbf{1}_n = \mathbf{0}_n$, $\mathbf{1}_n^T L = \mathbf{0}_n^T$.

II. Problem Statement

Consider a networked game with N players, defined with the following parameters:

- $V = \{1, \ldots, N\}$: Set of all players,
- $\Omega_i \subset \mathbb{R}$: Action set of player i, $\forall i \in V$,
- $\Omega = \prod_{i \in V} \Omega_i \subset \mathbb{R}^N$: Action set of all players, where $\prod$ denotes the Cartesian product,
- $J_i : \Omega \to \mathbb{R}$: Cost function of player i, $\forall i \in V$.

and denoted by $\mathcal{G}(V, \Omega_i, J_i)$. Players' actions are denoted as follows:

- $x_i \in \Omega_i$: Player i's action, $\forall i \in V$,
- $x_{-i} \in \Omega_{-i} := \prod_{j \in V \setminus \{i\}} \Omega_j$: All players' actions except player i's,
- $x = (x_i, x_{-i}) \in \Omega$: All players actions.

The game is played such that for a given $x_{-i} \in \Omega_{-i}$, every player $i \in V$ aims to minimize his own cost function selfishly with respect to (w.r.t.) x_i, i.e., find an optimal solution (action) of his corresponding optimization problem,

$$\begin{cases} \underset{x_i}{\text{minimize}} & J_i(x_i, x_{-i}) \\ \text{subject to} & x_i \in \Omega_i \end{cases} \quad \forall i \in V. \quad (6)$$

Note that each player's optimal action is dependent on the other players' actions. An Nash equilibrium (NE) lies at the intersection of solutions to the set of problems (6) (fixed-point of best-response map), such that no player can reduce his cost by unilaterally deviating from his action. An NE of a game is defined as follows:

Definition 1. *Consider an N-player game $\mathcal{G}(V, \Omega_i, J_i)$, each player i minimizing the cost function $J_i : \Omega \to \mathbb{R}$. An action profile (vector) $x^* = (x_i^*, x_{-i}^*) \in \Omega$ is called a Nash equilibrium (NE) of this game if*

$$J_i(x_i^*, x_{-i}^*) \leq J_i(x_i, x_{-i}^*) \quad \forall x_i \in \Omega_i, \ \forall i \in V.$$

We state a few assumptions for the existence of an NE, [28]–[30].

Assumption 1. *For every $i \in V$,*

- *$\Omega_i \subset \mathbb{R}$ is non-empty, compact and convex,*
- *$J_i(x_i, x_{-i})$ is C^1 and convex in x_i, for every x_{-i}, and jointly continuous in x.*

An NE $x^* = (x_i^*, x_{-i}^*)$ of (6) can be characterized under subdifferential analysis by

$$\nabla_i J_i(x^*) + \partial \mathcal{I}_{\Omega_i}(x_i^*) = 0, \ \forall i \in V, \quad (7)$$

where $\nabla_i J_i(x) = \frac{\partial J_i}{\partial x_i}(x_i, x_{-i})$, $x = (x_i, x_{-i})$, is the gradient of J_i w.r.t. x_i and $\partial \mathcal{I}_{\Omega_i}(\cdot)$ is a subgradient of $\mathcal{I}_{\Omega_i}$, the indicator function of the feasibility constraint $x_i \in \Omega_i$, [31]. Let $F : \Omega \to \mathbb{R}^N$ be the pseudo-gradient of game (6) defined by

$$F(x) := [\nabla_i J_i(x)]_{i \in V}, \quad \nabla_i J_i(x) = \frac{\partial J_i}{\partial x_i}(x_i, x_{-i}) \quad (8)$$

and let $G(x) := [\partial \mathcal{I}_{\Omega_i}(x_i)]_{i \in V}$. Then, (7) can be written in compact form as,

$$F(x^*) + G(x^*) = \mathbf{0}_N. \quad (9)$$

The convexity of Ω_i implies that $\mathcal{I}_{\Omega_i}$ is a convex function and the existence of a bounded subgradient $\partial \mathcal{I}_{\Omega_i}$. Typically another assumption such as monotonicity (strict, strong) of the pseudo-gradient vector F, (8), is used to show that projected-gradient type algorithms converge to x^*, [32].

The challenge is that each optimization problem in (6) is dependent on the solution of the other simultaneous problems. And since this game is distributed, no player is aware of the actions (solutions) of all the other players (problems). We assume that the cost function J_i and the action set Ω are the only information available to player i. Thus, the players need to exchange some information in order to update their estimates. An undirected *communication graph* $G_C(V, E)$ is then defined with no isolated vertex. Let denote the set of neighbours of player i in G_C by N_i. Let also denote $\mathbb{D}$ and L be the degree and Laplacian matrices associated to G_C, respectively. The following assumption is used.

Assumption 2. *G_C is an undirected and connected graph.*

We assume that players maintain estimates of the other players' actions and share them with their neighbours in order to update their estimates. Our goal is to develop a distributed algorithm, based on an ADMM approach, for computing an NE of $\mathcal{G}(V, \Omega_i, J_i)$ using only networked (partial) information over the communication graph $G_C(V, E)$.

III. Distributed Inexact-ADMM Algorithm

We develop a distributed ADMM-type algorithm ([33], page 255) to find an NE of game (6) over G_C by relating it to a set of augmented optimization problems with consensus constraints. These problems are obtained by introducing local estimates of players' actions and using (virtual) constraints for the consensus of these estimates.

We define a few notations for players' estimates.

- $x^i_{-i} \in \mathbb{R}^{N-1}$: Player i's estimate of all other players' actions except him,
- $x^i_i \in \Omega_i \subset \mathbb{R}$: Player i's estimate of his action which is indeed his own action, i.e., $x^i_i = x_i$ for $i \in V$.
- $x^i = (x^i_i, x^i_{-i}) \in \mathbb{R}^N$: Player i's estimate of all players' actions (state variable),
- $\mathbf{x} = [x^i]_{i\in V}$ or $\mathbf{x} = [x^1; \ldots; x^N] \in \mathbb{R}^{N^2}$: Augmented (stacked) vector of all estimates.

Note that all players' actions x can be interchangeably represented as $x = [x^i_i]_{i\in V}$.

Recall that (6) can be regarded as a set of sub-optimization problems, each assigned to a corresponding player, and each dependent on the solutions of the other sub-problems, i.e., each cost J_i is coupled to the others' actions x_{-i}. By employing the actions' estimates x^i, $\forall i \in V$, (also interpreted as local copies of x), we reformulate game (6) so that each cost function is separable. Particularly, from (6), consider the following set of problems:

$$\begin{cases} \underset{x^i_i \in \Omega_i}{\text{minimize}} & J_i(x^i_i, x^i_{-i}) \\ \text{subject to} & x^i = x^j \ \forall i \in V, \forall j \in N_i \end{cases} \qquad \forall i \in V. \tag{10}$$

The equality constraint along with Assumption 2 ensures that all the local copies of x are identical, i.e., $x^1 = x^2 = \ldots = x^N$. Hence (10) recovers (6). Using a slack variable t^{ls} to separate the equality constraints, and the indicator function $\mathcal{I}_{\Omega_i}(x^i_i)$ of the feasibility constraint $x^i_i \in \Omega_i$, we rewrite (10) as:

$$\begin{cases} \underset{x^i_i \in \mathbb{R}}{\text{minimize}} & J_i(x^i_i, x^i_{-i}) + \mathcal{I}_{\Omega_i}(x^i_i) \\ \text{subject to} & x^l = t^{ls} \quad \forall l \in V, \forall s \in N_l \\ & x^s = t^{ls} \quad \forall l \in V, \forall s \in N_l \end{cases} \qquad \forall i \in V. \tag{11}$$

Augmenting the constraints in this way simplifies the algorithm derivation and does not affect the solutions of the problem. One can regard (11) as being the same as (6) but, for each player $i \in V$, considering N estimates (local copies) of players' actions. *Note that in* (11)*, unlike DOP, minimization of J_i is not w.r.t. to the full x^i, but rather w.r.t. to the action component x^i_i. This means that, for the other (estimate) components x^i_{-i}, since only the constraints count, we can consider a zero objective function.* We will develop an ADMM-type algorithm to solve (11) in a distributed manner. First we show that a characterization of an NE for game (6) can be obtained based on KKT conditions of (11). Let $\{u^{ls}, v^{ls}\}_{l\in V, s\in N_l}$ with $u^{ls}, v^{ls} \in \mathbb{R}^N$ be the Lagrange multipliers associated with the two constraints in (11), respectively. The corresponding Lagrangian for player i, $\forall i \in V$ is as follows:

$$L_i\Big(x^i, t^{ls}; \{u^{ls}, v^{ls}\}\Big) := J_i(x^i_i, x^i_{-i}) + \mathcal{I}_{\Omega_i}(x^i_i) + \sum_{l\in V} \sum_{s\in N_l} \Big({u^{ls}}^T (x^l - t^{ls}) + {v^{ls}}^T (x^s - t^{ls}) \Big). \tag{12}$$

with the convention that, according to (11) and as remarked above, J_i is considered for minimization w.r.t. the action component x^i_i, while for minimization w.r.t. the (estimate) components x^i_{-i}, the objective function is considered zero.

Let $\mathbf{x}^* := [x^{i^*}]_{i\in V}$ and $\mathbf{u}^* := [[u^{ij^*}]_{j\in N_i}]_{i\in V}$, $\mathbf{v}^* :=$

$[[v^{ij*}]_{j\in N_i}]_{i\in V}$, be a set of optimal primal and dual solutions to (11). Let also, for notational simplicity,

$$\sum_{j\in N_i} u^{ij} + v^{ji} := w^i, \tag{13}$$

and $\mathbf{w}^* := [w^{i*}]_{i\in V}$, $w^{i*} = \sum_{j\in N_i} u^{ij*} + v^{ji*}$. We show next that for connected graphs, (11) is equivalent to (6) in the sense that, for all $i, j \in V$, the optimal arguments x^{i*} of (11) satisfy $x^{i*} = x^{j*} = t^{ij*} = x^*$, where x^* is an NE of (6).

Lemma 1. *Consider that Assumptions 1, 2 hold. Let $\mathbf{x}^* := [x^{i*}]_{i\in V}$ and $\mathbf{u}^* := [[u^{ij*}]_{j\in N_i}]_{i\in V}$, $\mathbf{v}^* := [[v^{ij*}]_{j\in N_i}]_{i\in V}$, be a set of optimal primal and dual solutions to (11), and let $\mathbf{w}^* := [w^{i*}]_{i\in V}$, where $w^{i*} = \sum_{j\in N_i} u^{ij*} + v^{ji*}$. Then, $\mathbf{x}^* = \mathbf{1}_N \otimes x^*$, where x^* is NE of (6), as in (7) or (9), $\mathbf{u}^* + \mathbf{v}^* = \mathbf{0}_{Nd}$, where $d = \sum_{i=1}^N |N_i|$, and $(\mathbf{1}_N^T \otimes I_N)\mathbf{w}^* = \mathbf{0}_N$.*

Proof. Based on (12), the KKT conditions for (11) are:

$$\nabla_i J_i(x^{i*}) + \partial \mathcal{I}_{\Omega_i}(x_i^{i*}) + w_i^{i*} = 0 \quad \forall i \in V, \tag{14}$$
$$w^{i*}_{-i} = \mathbf{0}_{N-1}, \ \ \forall i \in V, \tag{15}$$
$$x^{i*} = x^{j*} \quad \forall i \in V, \forall j \in N_i, \tag{16}$$
$$u^{ij*} + v^{ij*} = \mathbf{0}_N \quad \forall i \in V, \forall j \in N_i. \tag{17}$$

These are obtained by taking $\frac{\partial}{\partial x_i^i}(\cdot) = 0$, $\frac{\partial}{\partial x_{-i}^i}(\cdot) = 0$, $\frac{\partial}{\partial u^{ls}}(\cdot) = 0$, $\frac{\partial}{\partial v^{ls}}(\cdot) = 0$, $\frac{\partial}{\partial t^{ls}}(\cdot) = 0$ in (12), where, as remarked above, according to (11), the objective function is considered zero when taking $\frac{\partial}{\partial x_{-i}^i}(\cdot) = 0$, for the (estimate) components x_{-i}^i.

From (17) it follows that $\mathbf{u}^* + \mathbf{v}^* = \mathbf{0}_{Nd}$, where $d = \sum_{i=1}^N |N_i|$. We can combine (14) and (15) in vector form, using $w^{i*} = (w_i^{i*}, w^{i*}_{-i}) = (w_i^{i*}, \mathbf{0}_{N-1})$, as

$$\nabla_i J_i(x^{i*})\, e_i + \partial \mathcal{I}_{\Omega_i}(x^{i*}_i)\, e_i + w^{i*} = \mathbf{0}_N, \ \forall i \in V, \tag{18}$$

or, combining for all $i \in V$, in stacked vector form,

$$[\nabla_i J_i(x^{i*})\, e_i]_{i\in V} + [\partial \mathcal{I}_{\Omega_i}(x^{i*}_i)\, e_i]_{i\in V} + \mathbf{w}^* = \mathbf{0}_{N^2}, \tag{19}$$

where e_i is the i-th unit vector in $\mathbb{R}^N$.

By (16) and Assumption 2, it follows that $x^{1*} = \ldots = x^{N*} = \breve{x}$, for some $\breve{x} \in \mathbb{R}^N$. Substituting back in (18), $\breve{x}$ satisfies:

$$\nabla_i J_i(\breve{x})\, e_i + \partial \mathcal{I}_{\Omega_i}(\breve{x}_i)\, e_i + w^{i*} = \mathbf{0}_N, \ \forall i \in V.$$

Summing after $i \in V$, with $F(\breve{x}) := [\nabla_i J_i(\breve{x})]_{i\in V} \in \mathbb{R}^N$, yields

$$F(\breve{x}) + G(\breve{x}) + \sum_{i\in V} w^{i*} = \mathbf{0}_N.$$

For w^{i*}, using (17), it follows that for all $i \in V$, $w^{i*} = \sum_{j\in N_i}(u^{ij*} + v^{ji*}) = \sum_{j\in N_i}(-v^{ij*} + v^{ji*})$. Thus, $\sum_{i\in V} w^{i*} = \sum_{i\in V}\sum_{j\in N_i}(-v^{ij*} + v^{ji*}) = \sum_{i\in V}\sum_{j\in N_i}(-v^{ij*} + v^{ij*}) = \mathbf{0}_N$, where the second equality follows by Assumption 2. Hence, for $\mathbf{w}^* := [w^{i*}]_{i\in V}$,

$$(\mathbf{1}_N^T \otimes I_N)\mathbf{w}^* = \sum_{i\in V} w^{i*} = \mathbf{0}_N. \tag{20}$$

Using (20) into the previous equation yields

$$F(\breve{x}) + G(\breve{x}) = \mathbf{0}_N, \tag{21}$$

hence, by (9), $\breve{x} = x^*$ NE of (6), and $\mathbf{x}^* = \mathbf{1}_N \otimes x^*$. ∎

Recall the pseudo-gradient vector $F : \Omega \to \mathbb{R}^N$, $\Omega \subset \mathbb{R}^N$, defined by $F(x) = [\nabla_i J_i(x)]_{i\in V}$, (8). Let $\mathbf{F} : \Omega^N \to \mathbb{R}^N$, $\Omega^N \subset \mathbb{R}^{N^2}$, be the extension of F to the augmented space, defined as

$$\mathbf{F}(\mathbf{x}) := [\nabla_i J_i(x^i)]_{i\in V}, \ \nabla_i J_i(x^i) = \frac{\partial J_i}{\partial x_i^i}(x_i^i, x_{-i}^i) \tag{22}$$

and called the *extended pseudo-gradient*. Note that $\mathbf{F}(\mathbf{x}^*) = \mathbf{F}(\mathbf{1}_N \otimes x^*) = F(x^*)$. Let also $\mathbf{G}(\mathbf{x}) := [\partial I_{\Omega_i}(x_i^i)]_{i\in V}$, so that $\mathbf{G}(\mathbf{x}^*) = G(x^*)$. With these notations we can write (19) in compact form as,

$$\mathbf{R}\left(\mathbf{F}(\mathbf{x}^*) + \mathbf{G}(\mathbf{x}^*)\right) + \mathbf{w}^* = \mathbf{0}_{N^2}, \tag{23}$$

where $\mathbf{R} = \mathrm{blkdiag}([e_i]_{i\in V})$. Note $(\mathbf{1}_N^T \otimes I_N)\mathbf{R} = I_N$, $\mathbf{R}^T\mathbf{R} = I_N$, and $\mathbf{R}^T\mathbf{x} = x$. Pre-multiplying (23) by $(\mathbf{1}_N^T \otimes I_N)$, yields in compact form,

$$\mathbf{F}(\mathbf{x}^*) + \mathbf{G}(\mathbf{x}^*) + (\mathbf{1}_N^T \otimes I_N)\mathbf{w}^* = \mathbf{0}_N,$$

hence, by (20),

$$\mathbf{F}(\mathbf{x}^*) + \mathbf{G}(\mathbf{x}^*) = \mathbf{0}_N,$$

which is equivalent to (21), since $\mathbf{x}^* = \mathbf{1}_N \otimes x^*$.

A. Derivation of Inexact-ADMM Algorithm

We develop a distributed algorithm, using an *inexact ADMM* approach for computing an NE of $\mathcal{G}(V, \Omega_i, J_i)$ over the communication graph $G_C(V, E)$. The mechanism of the algorithm can be briefly explained as follows: The players maintain estimates of all players' actions and exchange them with their neighbors over G_C. According to an ADMM approach, for each player $i \in V$ we consider an *augmented* Lagrangian associated to (11),

$$L_i^a\Big(x^i, t^{ls}; \{u^{ls}, v^{ls}\}\Big) := J_i(x_i^i, x_{-i}^i) + \mathcal{I}_{\Omega_i}(x_i^i) + \sum_{l\in V}\sum_{s\in N_l}\Big(u^{ls^T}(x^l - t^{ls}) + v^{ls^T}(x^s - t^{ls})\Big) + \frac{c}{2}\sum_{l\in V}\sum_{s\in N_l}(\|x^l - t^{ls}\|^2 + \|x^s - t^{ls}\|^2), \tag{24}$$

where $c > 0$ is a parameter. Using their own estimates as well as the received information from their neighbours they locally update their actions and their own estimates via an inexact ADMM algorithm. According to an *inexact ADMM*, instead of solving an optimization sub-problem, each player uses a method of approximation (typically linear approximation) to reduce the complexity of each sub-problem.

The algorithm is elaborated below in both descriptive and formal representations:

1- ***Initialization:***

For each player $i \in V$, $x_i^i(0) \in \Omega_i$, $x_{-i}^i(0) \in \mathbb{R}^{N-1}$, $u^{ij}(0) = v^{ij}(0) = \mathbf{0}_N$ $j \in N_i$.

At each iteration k, a communication step and an update step is performed.

2- ***Communication Step:*** Each player $i \in V$ exchanges his previous estimate $x^i(k-1)$ with those of his neighbours $x^j(k-1)$, $j \in N_i$.

3- ***Update Step:*** Based on his and his neighbours' previous estimates, $x^i(k-1)$ and $x^j(k-1)$, $j \in N_i$, each player $i \in V$ updates his *action* $x_i^i(k)$ and his *estimates* $x_{-i}^i(k)$ via an inexact ADMM-type algorithm associated to problem (11). This is done based on (24), by alternating the dual and the primal variables and using the most recent information. First, each player updates his *dual Lagrange multipliers*, then using the updated dual variables, each player updates his *action and his estimates*. Prior to the dual variables update, the information of the updated slack variable t^{ij} is required. Based on (24), t^{ij} $\forall i \in V, j \in N_i$ is updated as follows:

$$t^{ij}(k) = \arg\min_{t^{ij}} L_i^a\Big(x^i(k-1), t^{ls}; \{u^{ls}(k-1), v^{ls}(k-1)\}\Big)$$
$$= \arg\min_{t^{ij}}\Big\{-(u^{ij}(k-1) + v^{ij}(k-1))^T t^{ij} + \frac{c}{2}(\|x^i(k-1) - t^{ij}\|^2 + \|x^j(k-1) - t^{ij}\|^2)\Big\},$$

or,

$$t^{ij}(k) = \frac{1}{2c}(u^{ij}(k-1) + v^{ij}(k-1)) + \frac{1}{2}(x^i(k-1) + x^j(k-1)). \tag{25}$$

Each player updates his dual variables by dual ascent with c as the step-size, based on (24), i.e., $\forall i \in V, j \in N_i$,

$$u^{ij}(k) = u^{ij}(k-1) + c\big(x^i(k-1) - t^{ij}(k)\big), \tag{26}$$
$$v^{ij}(k) = v^{ij}(k-1) + c\big(x^j(k-1) - t^{ij}(k)\big).$$

Using (25) in (26) and considering the initial conditions $u^{ij}(0) = v^{ij}(0) = \mathbf{0}_N$ $\forall i \in V$, $j \in N_i$, yields

$$u^{ij}(k) + v^{ij}(k) = \mathbf{0}_N \quad \forall i \in V,\, j \in N_i,\, k \geq 0. \tag{27}$$

so that (25) becomes

$$t^{ij}(k) = \frac{x^i(k-1) + x^j(k-1)}{2}. \tag{28}$$

Substituting (28) into (26), one can obtain the following update rules for the *dual variables*,

$$u^{ij}(k) = u^{ij}(k-1) + \frac{c}{2}\big(x^i(k-1) - x^j(k-1)\big), \tag{29}$$
$$v^{ij}(k) = v^{ij}(k-1) + \frac{c}{2}\big(x^j(k-1) - x^i(k-1)\big).$$

Note that using $u^{ij}(0) = v^{ij}(0) = \mathbf{0}_N, \forall i \in V, j \in N_i$, yields, from (29), that $u^{ij}(k) = -u^{ji}(k)$ and $v^{ij}(k) = -v^{ji}(k)$, $\forall i \in V, j \in N_i$, $\forall k \geq 0$. Thus, for w^i in (13), $w^i(k) = \sum_{j\in N_i} u^{ij}(k) + v^{ji}(k) = \sum_{j\in N_i} u^{ij}(k) - v^{ij}(k)$, using (29) yields that

$$w^i(k) = w^i(k-1) + c\sum_{j\in N_i}(x^i(k-1) - x^j(k-1)). \tag{30}$$

Next, each player $i \in V$ updates his *action* $x_i^i(k)$ and then his *estimates* $x_{-i}^i(k)$, based on the augmented Lagrangian (24), evaluated using the updated dual variables $w^i(k)$, and his and his neighbours' previous estimates, $x^i(k-1)$, $x^j(k-1)$, $j \in N_i$. Thus, minimizing (24) w.r.t. x_i^i and using (28), for each $i \in V$ the local update for $x_i^i(k)$ (action) can be written as:

$$x_i^i(k) = \arg\min_{x_i^i\in\mathbb{R}}\Big\{J_i(x_i^i, x_{-i}^i(k-1)) + \mathcal{I}_{\Omega_i}(x_i^i) + w_i^i(k)^T x_i^i + c\sum_{j\in N_i}\Big\|x_i^i - \frac{x_i^i(k-1) + x_i^j(k-1)}{2}\Big\|^2\Big\}. \tag{31}$$

As in an *inexact* ADMM-type algorithm, we approximate $J_i(x_i^i, x_{-i}^i(k-1))$ in (31) by a proximal first-order approximation around $x_i^i(k-1)$. Then,

$$x_i^i(k) = \arg\min_{x_i^i\in\mathbb{R}}\Big\{\nabla_i J_i(x^i(k-1))^T(x_i^i - x_i^i(k-1)) + \frac{\beta_i}{2}\|x_i^i - x_i^i(k-1)\|^2 + \mathcal{I}_{\Omega_i}(x_i^i) + w_i^i(k)^T x_i^i + c\sum_{j\in N_i}\Big\|x_i^i - \frac{x_i^i(k-1) + x_i^j(k-1)}{2}\Big\|^2\Big\}, \tag{32}$$

where $\beta_i > 0$ is a penalty factor for the proximal first-order approximation of each J_i. By taking $\frac{\partial}{\partial x_i^i}(\cdot) = 0$ on the right-hand side, this yields for $x_i^i(k)$,

$$\begin{aligned}&\partial_i \mathcal{I}_{\Omega_i}(x_i^i(k)) + \alpha_i x_i^i(k) - \beta_i x_i^i(k-1) + w_i^i(k) \\ &-c|N_i| x_i^i(k-1) + \nabla_i J_i(x^i(k-1)) - c \sum_{j \in N_i} x_i^j(k-1) = 0.\end{aligned} \tag{33}$$

where $\alpha_i = \beta_i + 2c|N_i|$. Note that this is equivalent to

$$\begin{aligned}&x_i^i(k) = \arg \min_{x_i^i \in \mathbb{R}} \Big\{ \mathcal{I}_{\Omega_i}(x_i^i) + \frac{\alpha_i}{2} \Big\| x_i^i - \alpha_i^{-1} \Big[\beta_i x_i^i(k-1) \\ &-\nabla_i J_i(x^i(k-1)) - w_i^i(k) \\ &+c \sum_{j \in N_i} \big(x_i^i(k-1) + x_i^j(k-1) \big) \Big] \Big\|^2 \Big\},\end{aligned}$$

or, by (5),

$$\begin{aligned}x_i^i(k) = \mathrm{prox}_{\mathcal{I}_{\Omega_i}}^{\alpha_i} \Big\{ \alpha_i^{-1} \Big[(\beta_i + c|N_i|) x_i^i(k-1) \\ -w_i^i(k) - \nabla_i J_i(x^i(k-1)) + c \sum_{j \in N_i} x_i^j(k-1) \Big] \Big\}.\end{aligned}$$

Thus, each player $i \in V$ updates his *action* $x_i^i(k) = x_i(k)$ as in

$$\begin{aligned}x_i^i(k) = T_{\Omega_i} \Big\{ \alpha_i^{-1} \Big[(\beta_i + c|N_i|) x_i^i(k-1) \\ -w_i^i(k) - \nabla_i J_i(x^i(k-1)) + c \sum_{j \in N_i} x_i^j(k-1) \Big] \Big\}\end{aligned} \tag{34}$$

Next, similar to the update of his action $x_i^i(k) = x_i(k)$, each player $i \in V$ updates his *estimates* of the other players' actions $x_{-i}^i(k)$. This is obtained similarly to (31) by minimizing (24) w.r.t. x_{-i}^i, but ignoring the $J_i(x_i^i, x_{-i}^i)$ term, since this term is minimized only w.r.t. his action x_i^i (see (11)). Thus, similar to (31), $\forall i \in V$

$$\begin{aligned}x_{-i}^i(k) = \arg \min_{x_{-i}^i \in \mathbb{R}^{N-1}} \Big\{ w_{-i}^i(k)^T x_{-i}^i \\ +c \sum_{j \in N_i} \Big\| x_{-i}^i - \frac{x_{-i}^i(k-1) + x_{-i}^j(k-1)}{2} \Big\|^2 \Big\}.\end{aligned}$$

Following the same derivation as for $x_i^i(k)$ update in (31)-(34), and taking $\frac{\partial}{\partial x_{-i}^i}(\cdot) = 0$, the local update for the other players' estimates, $x_{-i}^i(k)$, is obtained as

$$\begin{aligned}x_{-i}^i(k) = \alpha_i^{-1} \Big[(\beta_i + c|N_i|) x_{-i}^i(k-1) - w_{-i}^i(k) \\ +c \sum_{j \in N_i} x_{-i}^j(k-1) \Big],\end{aligned} \tag{35}$$

which is similar to (34), except without the $\nabla_i J_i(x^i(k-1))$ terms, and with no projection needed.

Note that (35) can be interpreted as a weighted average of the estimates with a penalty factor $w_{-i}^i(k)$ associated with the difference between the estimates of neighbouring players (see (30)). Note also that J_i needs to be well-defined at $x^i(k-1)$ so that we can evaluate $\nabla_i J_i(x^i(k-1))$ in (34). Then, based on (29), (34) and (35), and recalling that $c > 0$ is the augmented Lagrange function parameter as in (24), $\beta_i > 0$ is the penalty factor for the proximal first order approximation and $\alpha_i = \beta_i + 2c|N_i|$, the algorithm is summarized as:

Algorithm 1 Inexact-ADMM Algorithm

1: **initialization** $x_i^i(0) \in \Omega_i$, $x_{-i}^i(0) \in \mathbb{R}^{N-1}$, $u^{ij}(0) = v^{ij}(0) = \mathbf{0}_N \ \forall i \in V$, $j \in N_i$
2: **for** $k = 1, 2, \ldots$ **do**
3: **for** each player $i \in V$ **do**
4: players i, j $\forall j \in N_i$ exchange $x^i(k-1)$, $x^j(k-1)$
5: $u^{ij}(k) = u^{ij}(k-1) + \frac{c}{2}\big(x^i(k-1) - x^j(k-1)\big)$
$v^{ij}(k) = v^{ij}(k-1) + \frac{c}{2}\big(x^j(k-1) - x^i(k-1)\big)$
$w^i(k) = \sum_{j \in N_i} u^{ij}(k) - v^{ij}(k)$
6: $x_i^i(k) = T_{\Omega_i} \Big\{ \alpha_i^{-1} \Big[(\beta_i + c|N_i|) x_i^i(k-1) - w_i^i(k)$
$-\nabla_i J_i(x^i(k-1)) + c \sum_{j \in N_i} x_i^j(k-1) \Big] \Big\}$
7: $x_{-i}^i(k) = \frac{(\beta_i + c|N_i|) x_{-i}^i(k-1) + c \sum_{j \in N_i} x_{-i}^j(k-1) - w_{-i}^i(k)}{\alpha_i}$
8: **end for**
9: **end for**

Remark 1. *Unlike DOP algorithms where the minimization is w.r.t. the full vector, in a game each player minimizes his cost function only w.r.t.* x_i^i*. To update his decision (action)* x_i^i *(Step 6), each player uses his estimate of the other players' actions,* x_{-i}^i*, as well as his neighbours' estimates,* x_{-i}^j $\forall j \in N_i$*. Unlike a consensus gradient-based method, [22], [23], the action update has an additional* correction *term (using his dual variables* w_i^i *and the relative error w.r.t to his neighbours' estimates). Each player updates his estimate of the other players' actions,* x_{-i}^i *(Step 7), as a* weighted *average of his neighbours' information* x_{-i}^j $\forall j \in N_i$ *with a* correction *term based on his dual variables,* w_{-i}^i*).*

B. Inexact-ADMM Algorithm in Compact Form

In this section we write Algorithm 1 in a compact vector form for $\mathbf{x} = [x^i]_{i \in V}$, $\mathbf{w} = [w^i]_{i \in V}$, by using the definition of $\mathbf{F}$, $\mathbf{G}$, (22). From Step 6 in Algorithm 1, or equivalently, from (33) with $\alpha_i = \beta_i + 2c|N_i|$, it follows that for the i-th component of x^i, x^i_i,

$$\begin{aligned}&\nabla_i J_i(x^i(k-1)) + \partial_i \mathcal{I}_{\Omega_i}(x^i_i(k)) + w^i_i(k)\\ &+\beta_i(x^i_i(k) - x^i_i(k-1)) + 2c|N_i|x^i_i(k) - 2c|N_i|x^i_i(k-1)\\ &+c\sum_{j\in N_i}(x^i_i(k-1) - x^j_i(k-1)) = 0.\end{aligned}$$

From Step 7 in Algorithm 1, one can obtain for the rest of the components of x^i, x^i_{-i},

$$\begin{aligned}&\beta_i(x^i_{-i}(k) - x^i_{-i}(k-1)) + w^i_{-i}(k)\\ &+2c|N_i|x^i_{-i}(k) - 2c|N_i|x^i_{-i}(k-1)\\ &+c\sum_{j\in N_i}(x^i_{-i}(k-1) - x^j_{-i}(k-1)) = \mathbf{0}_{N-1}.\end{aligned}$$

Combining these two relations into a single vector one for all components of $x^i = (x^i_i, x^i_{-i})$, yields

$$\begin{aligned}&\nabla_i J_i(x^i(k-1))\, e_i + \partial_i \mathcal{I}_{\Omega_i}(x^i_i(k))\, e_i + w^i(k)\\ &+\beta_i(x^i(k) - x^i(k-1)) + 2c|N_i|x^i(k) - 2c|N_i|x^i(k-1)\\ &+c\sum_{j\in N_i}(x^i(k-1) - x^j(k-1)) = \mathbf{0}_N,\ \forall i \in V\end{aligned} \tag{36}$$

where e_i is the i-th unit vector in $\mathbb{R}^N$. Writing (36) in stacked vector form, yields

$$\begin{aligned}&[\nabla_i J_i(x^i(k-1))\, e_i]_{i\in V} + [\partial_i \mathcal{I}_{\Omega_i}(x^i_i(k))\, e_i]_{i\in V} + \mathbf{w}(k)\\ &+(\mathbb{B}\otimes I_N)(\mathbf{x}(k) - \mathbf{x}(k-1)) + 2c(\mathbb{D}\otimes I_N)\mathbf{x}(k)\\ &-2c(\mathbb{D}\otimes I_N)\mathbf{x}(k-1) + c(L\otimes I_N)\mathbf{x}(k-1) = \mathbf{0}_{N^2},\end{aligned}$$

where $\mathbf{x} = [x^i]_{i\in V}$, $\mathbf{w} = [w^i]_{i\in V}$, $\mathbb{B} = \text{diag}([\beta_i]_{i\in V})$, $\mathbb{D} = \text{diag}([|N_i|]_{i\in V})$. Recalling that $\mathbf{R} = \text{blkdiag}([e_i]_{i\in V})$, $\mathbf{F}(\mathbf{x}) = [\nabla_i J_i(x^i)]_{i\in V}$, $\mathbf{G}(\mathbf{x}) = [\partial I_{\Omega_i}(x^i_i)]_{i\in V}$, the previous relation can be written as

$$\begin{aligned}&\mathbf{R}\,(\mathbf{F}(\mathbf{x}(k-1)) + \mathbf{G}(\mathbf{x}(k))) + \mathbf{w}(k)\\ &+(\mathbb{B}\otimes I_N)(\mathbf{x}(k) - \mathbf{x}(k-1)) + 2c(\mathbb{D}\otimes I_N)\mathbf{x}(k)\\ &-c((2\mathbb{D} - L)\otimes I_N))\mathbf{x}(k-1) = \mathbf{0}_{N^2}.\end{aligned} \tag{37}$$

From Step 5 of Algorithm 1, or (30) we can write

$$w^i(k) = w^i(k-1) + c\sum_{j\in N_i}(x^i(k-1) - x^j(k-1)) \tag{38}$$

or, in stacked vector form,

$$\mathbf{w}(k) = \mathbf{w}(k-1) + c(L\otimes I_N)\mathbf{x}(k-1). \tag{39}$$

From (37), (39) for $\mathbf{x} = [x^i]_{i\in V}$, $\mathbf{w} = [w^i]_{i\in V}$, Algorithm 1 is written in stacked vector form as:

Algorithm 2 Vector Form Inexact-ADMM Algorithm

1: **initialization**
$\mathbf{w}(0) = \mathbf{0}_{N^2}$,
$\mathbf{x}(0) = [x^i(0)]_{i\in V}$, $x^i_i(0) \in \Omega_i$, $x^i_{-i}(0) \in \mathbb{R}^{N-1}$,

2: **update rules**
$\mathbf{w}$-update: $\mathbf{w}(k) = \mathbf{w}(k-1) + c\,(L\otimes I_N)\,\mathbf{x}(k-1)$,
$\mathbf{x}$-update:

$$\begin{aligned}2c(\mathbb{D}\otimes I_N)\mathbf{x}(k) &= -\mathbf{R}\,\mathbf{F}(\mathbf{x}(k-1)) - \mathbf{R}\,\mathbf{G}(\mathbf{x}(k))\\ &-(\mathbb{B}\otimes I_N)(\mathbf{x}(k) - \mathbf{x}(k-1)) - \mathbf{w}(k)\\ &+c((2\mathbb{D} - L)\otimes I_N)\mathbf{x}(k-1)\end{aligned}$$

where $\mathbf{R} = \text{blkdiag}([e_i]_{i\in V})$ is the $N^2 \times N$ block-diagonal matrix with unit vectors e_i on the block-diagonal, $\mathbb{D}$ is the degree matrix of G_C, L is the Laplacian matrix of G_C, $\mathbb{B} := \text{diag}((\beta_i)_{i\in V})$ with $\beta_i > 0$ and $c > 0$.

Remark 2. *The compact vector form Algorithm 2 was obtained by stacking the x^i estimate vectors into the $(N^2 \times 1)$ stacked estimate vector $\mathbf{x} = [x^i]_{i\in V}$ or $\mathbf{x} = [x^1; \ldots; x^N]$. An alternative matrix form of the algorithm can be obtained by using x^i vectors as* columns *in an $(N \times N)$ augmented estimate matrix $\mathbf{X} := [x^1, \ldots, x^N] \in \mathbb{R}^{N\times N}$. Note that using the vectorization operation we can write vec$(\mathbf{X}) = \mathbf{x}$. Similarly, let $\mathbf{W} := [w^1, \ldots, w^N] \in \mathbb{R}^{N\times N}$ and vec$(\mathbf{W}) = \mathbf{w}$. Let $\mathbb{F} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ and $\mathbb{G} : \mathbb{R}^N \to \mathbb{R}^{N\times N}$ be diagonal matrices defined by*

$$\mathbb{F}(\mathbf{X}) := \text{diag}(\nabla_i J_i(x^i)_{i\in V}),\ \mathbb{G}(x) := \text{diag}(\partial \mathcal{I}_{\Omega_i}(x_i)_{i\in V}).$$

Hence $\mathbb{F}$ is a $(N\times N)$ matrix with $\nabla_i J_i(x^i)e_i$ as the i-th column, so that we can write vec$(\mathbb{F}(\mathbf{X})) = \mathbf{R}\mathbf{F}(\mathbf{x})$ and $\mathbb{F}(\mathbf{X}) = \text{diag}(\mathbf{F}(\mathbf{x}))$, where $\mathbf{F}(\mathbf{x}) = [\nabla_i J_i(x^i)]_{i\in V} \in \mathbb{R}^N$, (22)*, is the extended pseudo-gradient vector. With these notations, we can write* (36)*,* (38)*, in matrix notation as,*

$$\begin{aligned}&\mathbb{F}(\mathbf{X}(k-1)) + \mathbb{G}(\mathbf{X}(k)) + \mathbb{B}(\mathbf{X}(k) - \mathbf{X}(k-1))\\ &\quad+\mathbf{W}(k) + 2c\mathbb{D}\mathbf{X}(k) - c(2\mathbb{D} - L)\mathbf{X}(k-1) = \mathbf{0}_{N\times N},\\ &\mathbf{W}(k) = \mathbf{W}(k-1) + c\,L\,\mathbf{X}(k-1).\end{aligned}$$

These are the same as the matrix form equations in the LANA Algorithm, [26], which was obtained by employing the players' estimates to augment the NE optimality conditions of game (6)*, comparing these with those of a DOP, and postulating the algorithm based on mimicking the decentralized linearized DOP ADMM algorithm in [34]. In contrast, in Section II the inexact-ADMM algorithm for NE seeking is fully developed, based on augmenting the coupled set of players' optimization* *problems, (which describes the NE problem (6)) into a set of optimization problems with consensus constraints, using the optimality conditions for this set of problems, followed by an ADMM procedure for each of them to derive the algorithm.*

IV. Convergence Analysis

In this section we first analyze the fixed points of Algorithm 1 (or Algorithm 2), then prove its convergence and finally, we provide a convergence rate analysis.

Lemma 2. *Consider that Assumptions 1, 2 hold. Let $\overline{\mathbf{x}} := [\overline{x}^i]_{i\in V}$, $\overline{\mathbf{u}} := [[\overline{u}^{ij}]_{j\in N_i}]_{i\in V}$, $\overline{\mathbf{v}} := [[\overline{v}^{ij}]_{j\in N_i}]_{i\in V}$, $\overline{\mathbf{w}} := [\overline{w}^i]_{i\in V}$ be the fixed points of Algorithm 1 (or Algorithm 2). Then, $\overline{\mathbf{x}} = \mathbf{x}^* = \mathbf{1}_N \otimes x^*$, where x^* is NE of game (6), $\overline{\mathbf{u}} = \mathbf{u}^*$, $\overline{\mathbf{v}} = \mathbf{v}^*$, and $\overline{\mathbf{w}} = \mathbf{w}^*$, where $\mathbf{u}^*$, $\mathbf{v}^*$, $\mathbf{w}^*$ are as in Lemma 1, i.e., $\overline{\mathbf{u}} + \overline{\mathbf{v}} = \mathbf{0}_{Nd}$ and $(\mathbf{1}_N^T \otimes I_N)\overline{\mathbf{w}} = \mathbf{0}_N$.*

Proof. From Step 5 of Algorithm 1, for $i \in V$, $j \in N_i$,

$$\left.\begin{array}{l}\overline{u}^{ij} = \overline{u}^{ij} + \frac{c}{2}\big(\overline{x}^i - \overline{x}^j\big),\\ \overline{v}^{ij} = \overline{v}^{ij} + \frac{c}{2}\big(\overline{x}^j - \overline{x}^i\big)\end{array}\right\} \Rightarrow \overline{x}^i = \overline{x}^j.$$

as in (16), and by Assumption 2 this yields

$$\overline{x}^1 = \ldots = \overline{x}^N = \overline{x}, \tag{40}$$

hence $\overline{\mathbf{x}} = \mathbf{1}_N \otimes \overline{x}$, for some $\overline{x} = [\overline{x}_i]_{i\in V} \in \mathbb{R}^N$. Using (40) in Step 6 of Algorithm 1 yields, $\alpha_i \overline{x}_i = (\beta_i + c|N_i|)\overline{x}_i - \overline{w}_i^i - \nabla_i J_i(\overline{x}) - \partial\mathcal{I}_{\Omega_i}(\overline{x}_i) + c|N_i|\overline{x}_i$. With $\alpha_i = \beta_i + 2c|N_i|$ this leads to

$$\nabla_i J_i(\overline{x}) + \partial\mathcal{I}_{\Omega_i}(\overline{x}_i) + \overline{w}_i^i = 0.\ \forall i \in V$$

Moreover, using (40) in Step 7 of Algorithm 1, we obtain $\alpha_i \overline{x}_{-i}^i = \alpha_i \overline{x}_{-i}^i - \overline{w}_{-i}^i$, hence

$$\overline{w}_{-i}^i = \mathbf{0}_{N-1},\ \forall i \in V$$

as in (15), hence $\overline{w}_{-i}^i = w_{-i}^{i*}$. Combining the previous two relations into a single vector, yields

$$\nabla_i J_i(\overline{x}) e_i + \partial\mathcal{I}_{\Omega_i}(\overline{x}_i) e_i + \overline{w}^i = \mathbf{0}_N,\ \forall i \in V$$

or, in stacked vector form with $\overline{\mathbf{w}} = [\overline{w}^i]_{i\in V}$,

$$[\nabla_i J_i(\overline{x}) e_i]_{i\in V} + [\partial\mathcal{I}_{\Omega_i}(\overline{x}_i) e_i]_{i\in V} + \overline{\mathbf{w}} = \mathbf{0}_{N^2}.$$

This can be written compactly as

$$\mathbf{R}\left(\mathbf{F}(\overline{\mathbf{x}}) + \mathbf{G}(\overline{\mathbf{x}})\right) + \overline{\mathbf{w}} = \mathbf{0}_{N^2}, \tag{41}$$

where $\mathbf{F}(\overline{\mathbf{x}}) = [\nabla_i J_i(\overline{x})]_{i\in V}$, $\mathbf{G}(\overline{\mathbf{x}}) = [\partial I_{\Omega_i}(\overline{x}_i)]_{i\in V}$ and $\mathbf{R} = \text{blkdiag}([e_i]_{i\in V})$. On the other hand, by (27),

$$\overline{u}^{ij} + \overline{v}^{ij} = \mathbf{0}_N \quad \forall i \in V,\ j \in N_i,$$

hence $\overline{\mathbf{u}} + \overline{\mathbf{v}} = \mathbf{0}_{Nd}$. Comparing this with (17) it follows that $\forall i \in V$, $j \in N_i$, $\overline{u}^{ij} = u^{ij*}$, $\overline{v}^{ij} = v^{ij*}$, hence $\overline{\mathbf{u}} = \mathbf{u}^*$, $\overline{\mathbf{v}} = \mathbf{v}^*$. From Step 5 of Algorithm 1, we can write (38), and with initial conditions $w^i(0) = \mathbf{0}_N$ $\forall i \in V$ it follows that

$$w^i(k) = c\sum_{t=0}^{k-1}\sum_{j\in N_i}(x^i(t) - x^j(t))$$

Equivalently, in stacked vector form, or from (39),

$$\mathbf{w}(k) = c(L\otimes I_N)\mathbf{q}(k-1),\ \text{ for }\ \mathbf{q}(k) := \sum_{t=0}^{k}\mathbf{x}(t), \tag{42}$$

where $\mathbf{q}(k)$ is an auxiliary variable. Then

$$\overline{w}^i = c\sum_{t=0}^{\infty}\sum_{j\in N_i}(x^i(t) - x^j(t))$$

or,

$$\overline{\mathbf{w}} = c(L\otimes I_N)\overline{\mathbf{q}},$$

where $\overline{\mathbf{q}}$ is a fixed point associated with $\mathbf{q}(k)$. Left multiplying both sides by $\mathbf{1}_N^T \otimes I_N$ and using $\mathbf{1}_N^T L = \mathbf{0}_N^T$,

$$(\mathbf{1}_N^T \otimes I_N)\overline{\mathbf{w}} = c(\mathbf{1}_N^T L \otimes I_N)\overline{\mathbf{q}} = \mathbf{0}_N \tag{43}$$

and comparing it to (20), it follows that $\overline{\mathbf{w}} = \mathbf{w}^*$.

Pre-multiplying (41) by $(\mathbf{1}_N^T \otimes I_N)$ with $(\mathbf{1}_N^T \otimes I_N)\mathbf{R} = I_N$ yields,

$$\mathbf{F}(\overline{\mathbf{x}}) + \mathbf{G}(\overline{\mathbf{x}}) + (\mathbf{1}_N^T \otimes I_N)\overline{\mathbf{w}} = \mathbf{0}_N.$$

Using (43), $\overline{\mathbf{x}} = \mathbf{1}_N \otimes \overline{x}$, $\mathbf{F}(\mathbf{1}_N \otimes \overline{x}) = F(\overline{x})$, $\mathbf{G}(\mathbf{1}_N \otimes \overline{x}) = G(\overline{x})$, it follows that

$$F(\overline{x}) + G(\overline{x}) = \mathbf{0}_N,$$

hence, by (9), $\overline{x} = x^*$ and $\overline{\mathbf{x}} = \mathbf{x}^* = \mathbf{1}_N \otimes x^*$. ∎

A. Convergence Proof

We now make the following assumption.

Assumption 3. *Let $\mathbf{F}$ be the extended pseudo-gradient vector as in (22). $\mathbf{F}$ is cocoercive , i.e., $\forall \mathbf{x}, \mathbf{y} \in \Omega^N$,*

$$(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y}))^T(x - y) \geq \sigma_F\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})\|^2 \tag{44}$$

where $\sigma_F > 0$.

Remark 3. *Note that Assumption 3 is the extension of the monotonicity assumption on the pseudo-gradient F, (8), to the augmented space of actions and estimates. Note that the extended pseudo-gradient satisfies $\mathbf{F}(\mathbf{1}_N \otimes x) = F(x)$. Assumption 3 is not usually required in distributed optimization problems (DOP);*

there instead the individual (strong) convexity of each objective function is assumed to be w.r.t. the full vector x, as well as Lipschitz continuity of its gradient (see [4]). As a result, in the augmented space of local copies, the obtained cost functions are decoupled, and each one is individually (strong) convex, or cocoercive, in its full argument, thus automatically satisfying Assumption 3 or a similar one.

*In contrast, in a game, each player minimizes his cost w.r.t. his own action x_i (*part *of the full argument on which his own cost function depends on, due to coupling to others' actions). Individual convexity of each cost function w.r.t. the full argument is too strong in a game context; rather, individual convexity w.r.t. x_i, for any given x_{-i}, is used (see Assumption 1). For convergence to an NE, an additional (strict, strong) monotonicity assumption of the pseudo-gradient F,* (8) *is typically used (see [17], [21], [22], [12], [32]). Because of coupling to others' actions and because of partial convexity, monotonicity of the pseudo-gradient is* not automatically satisfied *when extended to the augmented space, as it happens in DOP. Thus Assumption 3 is the naturally extended monotonicity assumption for the augmented space of actions and estimates. This is similar to the strong monotonicity Assumption 5 used in [24]. As we will show, this leads to an algorithm with ADMM algorithm benefits in terms of convergence speed. In addition, we show that it can be relaxed to hold only at $y = x^*$ and $\mathbf{y} = \mathbf{1}_N \otimes x^*$ (see Remark 4).*

Theorem 1. *Under Assumptions 1, 2 and 3, if*

$$\sigma_F > \frac{1}{2\lambda_{\min}(c(2\mathbb{D} - L) + \mathbb{B})}, \tag{45}$$

then the sequence $\{x^i(k)\}\ \forall i \in V$, or $\{\mathbf{x}(k)\}$, generated by Algorithm 1 (or Algorithm 2) converges to x^* (or $\mathbf{1}_N \otimes x^*$), with x^* an NE of game (6)*. In* (45)*, $c > 0$, $\mathbb{B} := diag((\beta_i)_{i\in V})$, $\beta_i > 0$ is player i's penalty factor, $\mathbb{D}$ and L are the degree and Laplacian matrices of G_C, respectively.*

Proof. From the $\mathbf{x}$-update in the vector form Algorithm 2, it follows that

$$\begin{aligned}
&\mathbf{R}\left(\mathbf{F}(\mathbf{x}(k-1)) + \mathbf{G}(\mathbf{x}(k))\right) + \mathbf{w}(k) \\
&+(\mathbb{B} \otimes I_N)(\mathbf{x}(k) - \mathbf{x}(k-1)) \\
&+2c(\mathbb{D} \otimes I_N)\mathbf{x}(k) - c((2\mathbb{D} - L) \otimes I_N))\mathbf{x}(k-1) = \mathbf{0}_{N^2}
\end{aligned}$$

or, denoting $H := \mathbb{B} + c(2\mathbb{D} - L)$ and substituting $\mathbf{w}(k)$ by $\mathbf{w}(k+1) - c(L \otimes I_N)\mathbf{x}(k)$ (from the $\mathbf{w}$-update),

$$\begin{aligned}
&\mathbf{R}\,\mathbf{F}(\mathbf{x}(k-1)) + \mathbf{R}\,\mathbf{G}(\mathbf{x}(k)) + \mathbf{w}(k+1) \\
&+(H \otimes I_N)\left(\mathbf{x}(k) - \mathbf{x}(k-1)\right) = \mathbf{0}_{N^2}.
\end{aligned} \tag{46}$$

Note that $H \otimes I_N \succeq 0$, since $\mathbb{B} \succeq 0$ and $c(2\mathbb{D} - L) \succeq 0$.

Recall the KKT conditions in compact form (23)

$$\mathbf{R}\,\mathbf{F}(\mathbf{x}^*) + \mathbf{R}\,\mathbf{G}(\mathbf{x}^*) + \mathbf{w}^* = \mathbf{0}_{N^2},$$

where $\mathbf{x}^* = \mathbf{1}_N \otimes x^*$. Subtracting this from the previous relation, multiplying both sides by $(\mathbf{x}(k) - \mathbf{x}^*)^T$ and using $\mathbf{R}^T\mathbf{x} = x$, yields

$$\begin{aligned}
&\Big(\mathbf{R}\left(\mathbf{F}(\mathbf{x}(k-1)) - \mathbf{F}(\mathbf{x}^*)\right)\Big)^T(\mathbf{x}(k) - \mathbf{x}^*) \\
&+\Big(G(x(k)) - G(x^*)\Big)^T(x(k) - x^*) \\
&+(\mathbf{w}(k+1) - \mathbf{w}^*)^T(\mathbf{x}(k) - \mathbf{x}^*) \\
&+(\mathbf{x}(k) - \mathbf{x}(k-1))^T(H \otimes I_N)(\mathbf{x}(k) - \mathbf{x}^*) = 0.
\end{aligned} \tag{47}$$

For the first term in (47) we write,

$$\begin{aligned}
&\Big(\mathbf{R}\left(\mathbf{F}(\mathbf{x}(k-1)) - \mathbf{F}(\mathbf{x}^*)\right)\Big)^T(\mathbf{x}(k) - \mathbf{x}^*) \\
&= \Big(\mathbf{R}\left(\mathbf{F}(\mathbf{x}(k-1)) - \mathbf{F}(\mathbf{x}^*)\right)\Big)^T(\mathbf{x}(k-1) - \mathbf{x}^*) \\
&+\Big(\mathbf{R}\,\mathbf{F}(\mathbf{x}(k-1)) - \mathbf{F}(\mathbf{x}^*)\Big)^T(\mathbf{x}(k) - \mathbf{x}(k-1)).
\end{aligned}$$

We use Assumption 3 with $\mathbf{R}^T\mathbf{x} = x$ for the first term on the right-hand side, and (3) for $\rho = \frac{1}{2\sigma_F}$ for the second term, to yield

$$\begin{aligned}
&\Big(\mathbf{R}\left(\mathbf{F}(\mathbf{x}(k-1)) - \mathbf{F}(\mathbf{x}^*)\right)\Big)^T(\mathbf{x}(k) - \mathbf{x}^*) \\
&\geq \sigma_F\|\mathbf{F}(\mathbf{x}(k-1)) - \mathbf{F}(\mathbf{x}^*)\|^2 \\
&-\sigma_F\|\mathbf{R}\left(\mathbf{F}(\mathbf{x}(k-1)) - \mathbf{F}(\mathbf{x}^*)\right)\|^2 - \frac{1}{4\sigma_F}\|\mathbf{x}(k) - \mathbf{x}(k-1)\|^2 \\
&= -\frac{1}{4\sigma_F}\|\mathbf{x}(k) - \mathbf{x}(k-1)\|^2
\end{aligned} \tag{48}$$

where the equality follows from $\mathbf{R}^T\mathbf{R} = I_N$.

For the second term in (47),where $G(x) := (\partial\mathcal{I}_{\Omega_i}(x_i))_{i\in V}$, by the convexity of $\mathcal{I}_{\Omega_i}$ (Assumption 1) it follows that

$$\Big(G(x(k)) - G(x^*)\Big)^T(x(k) - x^*) \geq 0. \tag{49}$$

Finally for the third term in (47), we use the following. From (42) it follows that $\mathbf{w}(k+1) = c(L \otimes I_N)\mathbf{q}(k)$, where $\mathbf{q}(k) = \sum_{t=0}^{k}\mathbf{x}(t)$, and $\overline{\mathbf{w}} = c(L \otimes I_N)\overline{\mathbf{q}} = \mathbf{w}^*$ (by Lemma 2, and (43)). Thus,

$$\mathbf{q}(k) - \mathbf{q}(k-1) = \mathbf{x}(k) \tag{50}$$

Then, we can write

$$(\mathbf{w}(k+1) - \mathbf{w}^*)^T(\mathbf{x}(k) - \mathbf{x}^*)$$

$$= (\mathbf{q}(k) - \overline{\mathbf{q}}))^T (cL \otimes I_N)(\mathbf{q}(k) - \mathbf{q}(k-1) - \mathbf{x}^*)$$
$$= (\mathbf{q}(k) - \overline{\mathbf{q}}))^T (cL \otimes I_N)(\mathbf{q}(k) - \mathbf{q}(k-1))$$

by $(L \otimes I_N)\mathbf{x}^* = (L \otimes I_N)(\mathbf{1}_N \otimes x^*) = L\mathbf{1}_N \otimes x^* = \mathbf{0}_{N^2}$. Using (48), (49) and this last relation in (47) yields

$$\begin{aligned} 0 \geq & -\frac{1}{4\sigma_F}\|\mathbf{x}(k) - \mathbf{x}(k-1)\|^2 \\ & +(\mathbf{q}(k) - \overline{\mathbf{q}}))^T (cL \otimes I_N)(\mathbf{q}(k) - \mathbf{q}(k-1)) \\ & +(\mathbf{x}(k) - \mathbf{x}(k-1))^T (H \otimes I_N)(\mathbf{x}(k) - \mathbf{x}^*), \end{aligned}$$

where $H \otimes I_N \succeq 0$. Using (4) to deal with all cross-terms in the previous inequality, yields

$$\begin{aligned} & \|\mathbf{x}(k) - \mathbf{x}^*\|^2_{H \otimes I_N} + \|\mathbf{q}(k) - \overline{\mathbf{q}}\|^2_{cL \otimes I_N} \\ & -\|\mathbf{x}(k-1) - \mathbf{x}^*\|^2_{H \otimes I_N} - \|\mathbf{q}(k-1) - \overline{\mathbf{q}}\|^2_{cL \otimes I_N} \leq \\ & -\|\mathbf{x}(k) - \mathbf{x}(k-1)\|^2_{H \otimes I_N} - \|\mathbf{q}(k) - \mathbf{q}(k-1)\|^2_{cL \otimes I_N} \\ & +\frac{1}{2\sigma_F}\|\mathbf{x}(k) - \mathbf{x}(k-1)\|^2. \end{aligned}$$

Let us make the following notations:

$$\mathbf{z}(k) := \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{q}(k) \end{pmatrix}, \ \mathbf{z}^* := \begin{pmatrix} \mathbf{1}_N \otimes x^* \\ \overline{\mathbf{q}} \end{pmatrix}, \tag{51}$$

$$\Phi \triangleq \begin{pmatrix} H & \mathbf{0}_{N \times N} \\ \mathbf{0}_{N \times N} & cL \end{pmatrix}. \tag{52}$$

Then we can write the last inequality as

$$\begin{aligned} & \|\mathbf{z}(k) - \mathbf{z}^*\|^2_{\Phi \otimes I_N} - \|\mathbf{z}(k-1) - \mathbf{z}^*\|^2_{\Phi \otimes I_N} \leq \\ & -\|\mathbf{z}(k) - \mathbf{z}(k-1)\|^2_{\Phi \otimes I_N} + \frac{1}{2\sigma_F}\|\mathbf{x}(k) - \mathbf{x}(k-1)\|^2. \end{aligned}$$

Moreover, since $\frac{1}{2\sigma_F}\|\mathbf{x}(k) - \mathbf{x}(k-1)\|^2 \leq \frac{1}{2\sigma_F \lambda_{\min}(H)}\|\mathbf{x}(k) - \mathbf{x}(k-1)\|^2_{H \otimes I_N} \leq \frac{1}{2\sigma_F \lambda_{\min}(H)}\|\mathbf{z}(k) - \mathbf{z}(k-1)\|^2_{\Phi \otimes I_N}$, from the foregoing it follows that

$$\begin{aligned} & \|\mathbf{z}(k) - \mathbf{z}^*\|^2_{\Phi \otimes I_N} - \|\mathbf{z}(k-1) - \mathbf{z}^*\|^2_{\Phi \otimes I_N} \leq \\ & -\zeta\|\mathbf{z}(k) - \mathbf{z}(k-1)\|^2_{\Phi \otimes I_N} \leq 0, \end{aligned} \tag{53}$$

where $\zeta = 1 - \frac{1}{2\sigma_F \lambda_{\min}(H)}$, and $0 < \zeta < 1$ by (45). Summing (53) over k from 1 to ∞ yields

$$\begin{aligned} & \sum_{k=1}^{\infty} \|\mathbf{z}(k) - \mathbf{z}(k-1)\|^2_{\Phi \otimes I_N} \\ & \leq \frac{1}{\zeta}(\|\mathbf{z}(0) - \mathbf{z}^*\|^2_{\Phi \otimes I_N} - \|\mathbf{z}(\infty) - \mathbf{z}^*\|^2_{\Phi \otimes I_N}) < \infty. \end{aligned} \tag{54}$$

From (54) it follows that $\|\mathbf{z}(k) - \mathbf{z}(k-1)\|^2_{\Phi \otimes I_N} \to 0$. From (53), since $\|\mathbf{z}(k) - \mathbf{z}^*\|^2_{\Phi \otimes I_N}$ is bounded and non-increasing, it follows that $\|\mathbf{z}(k) - \mathbf{z}^*\|^2_{\Phi \otimes I_N} \to \theta$, for some $\theta \geq 0$, or $\|\mathbf{x}(k) - \mathbf{x}^*\|^2_{H \otimes I_N} + \|\mathbf{q}(k) - \overline{\mathbf{q}}\|^2_{cL \otimes I_N} \to \theta$. Thus, the sequences $\mathbf{x}$ and $\mathbf{q}$ are bounded and have limit (fixed) points denoted by $\overline{\mathbf{x}}$ and $\overline{\mathbf{q}}$. Then,

$$\theta = \|\overline{\mathbf{x}} - \mathbf{x}^*\|^2_{H \otimes I_N}.$$

By Lemma 2, $\overline{\mathbf{x}} = \mathbf{x}^*$ where $\mathbf{x}^* = \mathbf{1} \otimes x^*$, with x^* the NE of the game by Lemma 1, and thus $\theta = 0$. One can conclude that $\mathbf{x}(k) \to \mathbf{1} \otimes x^*$. ■

Remark 4. *Assumption 3 is used only in equation* (48) *and only for $y = x^*$. Hence, a sufficient condition is a restricted cocoercivity assumption, where* (44) *is satisfied only for $y = x^*$, $\mathbf{y} = \mathbf{1}_N \otimes x^*$ (a similar type of assumption as in Assumption 4.4 in [5]).*

Remark 5. *Note that while similarly based on augmenting the action space, the methodology used here is different that the one used in [10]. There the game was reformulated as a (set of) optimization problems in a doubly augmented space $(x, y) \in \Omega^2$, with a fixed-point solution, i.e., enforcing $x = y$ in the solution after solving them. In the methodology developed here, we use an N-fold augmented space, $[x^i]_{i \in V} \in \Omega^N$, (each player $i \in V$ having his own state variable x^i in Ω). In this augmented space, we also reformulated the game as a set of optimization problems, but we imposed (virtual) consensus constraints $x^i = x^j$ prior to solving them. By following then an ADMM approach to solve them, we developed an algorithm for which we proved convergence on a single time-scale, unlike [10], [11].*

B. Convergence Rate Analysis

Next we investigate the convergence rate of Algorithm 1 (or Algorithm 2). We use the following result.

Proposition 1 ([35]). *If a sequence $\{a(k)\} \subset \mathbb{R}$ is:* (i) *nonnegative,* $a(k) \geq 0$, (ii) *summable,* $\sum_{k=1}^{\infty} a(k) < \infty$, *and* (iii) *monotonically non-increasing,* $a(k+1) \leq a(k)$, *then we have:* $a(k) = o\left(\frac{1}{k}\right)$, *i.e.,* $\lim_{k \to \infty} k\, a(k) = 0$.

For $\|\mathbf{z}(k-1) - \mathbf{z}(k)\|^2_{\Phi \otimes I_N}$, with $\mathbf{z}(k)$, $\Phi \otimes I_N$ as in (51), (52), summability follows from (54) in Theorem 1. Next we provide a lemma showing that $\|\mathbf{z}(k-1) - \mathbf{z}(k)\|^2_{\Phi \otimes I_N}$ is monotonically non-increasing. The non-ergodic $o\left(\frac{1}{k}\right)$ convergence rate is established based on these results and Proposition 1.

Lemma 3. *Under the assumptions of Theorem 1, the sequence $\{\mathbf{z}(k)\}$,* (51)*, generated by Algorithm 1 (or Algorithm 2) satisfies for all $k \geq 1$,*

$$\|\mathbf{z}(k) - \mathbf{z}(k+1)\|^2_{\Phi \otimes I_N} \leq \|\mathbf{z}(k-1) - \mathbf{z}(k)\|^2_{\Phi \otimes I_N},$$

where $\Phi \otimes I_N \succeq 0$ is as in (52).

Proof. The proof follows similarly to that of Theorem 1. To simplify the equations we define $\Delta\mathbf{x}(k+1) := \mathbf{x}(k) - \mathbf{x}(k+1)$, $\Delta\mathbf{q}(k+1) := \mathbf{q}(k) - \mathbf{q}(k+1)$, $\Delta\mathbf{z}(k+1) := \mathbf{z}(k) - \mathbf{z}(k+1)$, $\Delta\mathbf{F}(\mathbf{x}(k+1)) := \mathbf{F}(\mathbf{x}(k)) - \mathbf{F}(\mathbf{x}(k+1))$, $\Delta\mathbf{G}(x(k+1)) = \mathbf{G}(x(k)) - \mathbf{G}(x(k+1))$.

By Assumption 3 and using $\mathbf{R}^T\mathbf{x} = x$ and (3) we have

$$\begin{aligned}&\sigma_F\|\Delta\mathbf{F}(\mathbf{x}(k))\|^2 \leq \Delta\mathbf{x}(k)^T\mathbf{R}\Delta\mathbf{F}(\mathbf{x}(k))\\ &= \Delta\mathbf{x}(k+1)^T\mathbf{R}\Delta\mathbf{F}(\mathbf{x}(k))\\ &\ +(\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k+1))^T\mathbf{R}\Delta\mathbf{F}(\mathbf{x}(k))\\ &\leq \Delta\mathbf{x}(k+1)^T\mathbf{R}\Delta\mathbf{F}(\mathbf{x}(k)) + \tfrac{1}{4\sigma_F}\|\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k+1)\|^2\\ &\ +\sigma_F\|\mathbf{R}\Delta\mathbf{F}(\mathbf{x}(k))\|^2\end{aligned}$$

Using $\mathbf{R}^T\mathbf{R} = I_N$, this implies that

$$\Delta\mathbf{x}(k+1)^T\mathbf{R}\Delta\mathbf{F}(\mathbf{x}(k)) + \frac{1}{4\sigma_F}\|\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k+1)\|^2 \geq 0. \tag{55}$$

By Assumption 1 as in (49) it follows that

$$\Delta\mathbf{x}(k+1)^T\mathbf{R}\Delta\mathbf{G}(x(k+1)) \geq 0. \tag{56}$$

By (46) and (42),

$$\begin{aligned}&\mathbf{RF}(\mathbf{x}(k-1)) + \mathbf{RG}(x(k))\\ &= (H\otimes I_N)(\mathbf{x}(k-1) - \mathbf{x}(k)) - c(L\otimes I_N)\mathbf{q}(k),\end{aligned}$$

and similarly we obtain,

$$\begin{aligned}&\mathbf{R}\Delta\mathbf{F}(\mathbf{x}(k)) + \mathbf{R}\Delta\mathbf{G}(x(k+1)) = \\ &-(H\otimes I_N)(\Delta\mathbf{x}(k+1) - \Delta\mathbf{x}(k)) - c(L\otimes I_N)\Delta\mathbf{q}(k+1).\end{aligned} \tag{57}$$

Substituting $\mathbf{R}\Delta\mathbf{F}(\mathbf{x}(k))$, (57), and (56) into (55), yields

$$\begin{aligned}&\frac{1}{4\sigma_F}\|\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k+1)\|^2\\ &-c\Delta\mathbf{x}(k+1)^T(L\otimes I_N)\Delta\mathbf{q}(k+1)\\ &-\Delta\mathbf{x}(k+1)^T(H\otimes I_N)(\Delta\mathbf{x}(k+1) - \Delta\mathbf{x}(k)) \geq 0.\end{aligned} \tag{58}$$

Using (50), $\Delta\mathbf{q}(k+1) = -\mathbf{x}(k+1)$, hence

$$\Delta\mathbf{q}(k) - \Delta\mathbf{q}(k+1) = -\Delta\mathbf{x}(k+1).$$

Substituting this into the second term of (58), yields

$$\begin{aligned}&\frac{1}{4\sigma_F}\|\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k+1)\|^2\\ &+(\Delta\mathbf{q}(k) - \Delta\mathbf{q}(k+1))^Tc(L\otimes I_N)\Delta\mathbf{q}(k+1)\\ &+(\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k+1))^T(H\otimes I_N)\Delta\mathbf{x}(k+1) \geq 0,\end{aligned}$$

or, equivalently,

$$\begin{aligned}&\frac{1}{4\sigma_F}\|\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k+1)\|^2\\ &+(\Delta\mathbf{z}(k) - \Delta\mathbf{z}(k+1))^T(\Phi\otimes I_N)\Delta\mathbf{z}(k+1) \geq 0\end{aligned}$$

where Φ is as in (52). Using (4) for the cross-term yields

$$\begin{aligned}&\|\Delta\mathbf{z}(k)\|^2_{\Phi\otimes I_N} - \|\Delta\mathbf{z}(k+1)\|^2_{\Phi\otimes I_N}\\ &\geq \|\Delta\mathbf{z}(k) - \Delta\mathbf{z}(k+1)\|^2_{\Phi\otimes I_N} - \frac{1}{2\sigma_F}\|\Delta\mathbf{x}(k) - \Delta\mathbf{x}(k+1)\|^2\\ &\geq \zeta\|\Delta\mathbf{z}(k) - \Delta\mathbf{z}(k+1)\|^2_{\Phi\otimes I_N} \geq 0,\end{aligned}$$

where $\zeta = 1 - \frac{1}{2\sigma_F\lambda_{\min}(H)} > 0$, by (45). ■

Theorem 2. *Under the same assumptions of Theorem 1, the following rate holds for Algorithm 1 (or Algorithm 2):*

$$\|\mathbf{x}(k) - \mathbf{1}_N\otimes x^*\|^2_{L\otimes I_N} = o\left(\tfrac{1}{k}\right),$$

where $(L\otimes I_N) \succeq 0$, and L is the Laplacian matrix of G_C.

Proof. By Theorem 1 (equation (54)), $\sum_{k=0}^{\infty}\|\mathbf{z}(k-1) - \mathbf{z}(k)\|^2_{\Phi\otimes I_N} < \infty$. Moreover, Lemma 3 proves that $\|\mathbf{z}(k-1) - \mathbf{z}(k)\|^2_{\Phi\otimes I_N}$ is monotonically non-increasing. Then, it directly follows by Proposition 1 that,

$$\|\mathbf{z}(k-1) - \mathbf{z}(k)\|^2_{\Phi\otimes I_N} = o\left(\tfrac{1}{k}\right).$$

Expanding this by using (51) and (52), yields,

$$\|\mathbf{x}(k-1) - \mathbf{x}(k)\|^2_{H\otimes I_N} + c\|\mathbf{q}(k-1) - \mathbf{q}(k)\|^2_{L\otimes I_N} = o\left(\frac{1}{k}\right)$$

If the sum of two positive sequences is $o(\frac{1}{k})$, then each sequence is $o(\frac{1}{k})$, hence, by the foregoing, $\|\mathbf{q}(k-1) - \mathbf{q}(k)\|^2_{L\otimes I_N} = o\left(\frac{1}{k}\right)$. By (50), $\mathbf{q}(k-1) - \mathbf{q}(k) = \mathbf{x}(k)$, so,

$$\|\mathbf{x}(k)\|^2_{L\otimes I_N} = o\left(\frac{1}{k}\right).$$

By Assumption 2, $(L\otimes I_N)(\mathbf{1}_N\otimes x^*) = \mathbf{0}_{N^2}$, hence $\|\mathbf{x}(k) - \mathbf{1}_N\otimes x^*\|^2_{L\otimes I_N} = o\left(\frac{1}{k}\right)$. ■

V. Simulation Results

In this section, we present two numerical examples to illustrate the performance of Algorithms 1 (or 2). The first one is a quadratic example, while the second one is as in [23].

A. Example 1

Consider a quadratic game over $N = 20$-player network. 20 producers are involved in the production of a homogeneous commodity. The quantity produced by firm i (decision made by agent i) is denoted as x_i. The cost function of producing the commodity at firm

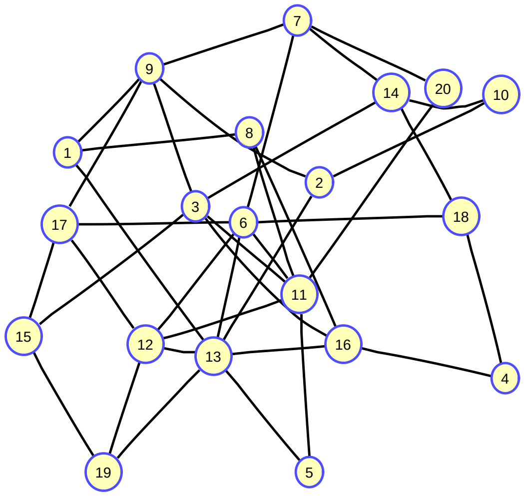

Fig. 1: Communication graph G_C for Example 1.

i is denoted as $u_i(x_i)$. The demand price is denoted as $f(x_i, x_{-i})$. The cost function is

$$J_i(x_i, x_{-i}) = u_i(x_i) - x_i f(x_i, x_{-i}), x_i \in \Omega_i, \forall i \in V,$$

where $V = \{1, 2, \cdots, N\}$, $N = 20$, $u_i = c_i x_i$, $c_i = 20 + 10(i-1)$, and $f(x_i, x_{-i}) = 1200 - \sum_{j=1}^{N} x_j$. To demonstrate that the algorithm still works when iterates hit the boundary, we set $\Omega_i = [7, 100], \forall i \in V$. The underlying communication graph G_C is randomly generated according to a connectivity ratio $r_c = \frac{4}{N}$, (see Fig. 1). Recall that the connectivity ratio of the network, r_c, is defined as its actual number of edges divided by $\frac{N(N-1)}{2}$. Clearly, the minimal possible r_c is $\frac{2}{N}$ and the maximal possible one is 1. It can be checked that Assumptions 1, 2 and 3 are satisfied.

The results for the implementation of Algorithm 1, or 2 are shown in Fig. 2. Initial conditions are uniformly randomly selected. The 'true' NE is obtained by (centralized) gradient method and the overall relative error of the generated sequence is illustrated in the top figure in Fig. 2. We used different values for c to verify that the rate of convergence is not affected. We also report the first 300 iterations of the convergence paths of firms 1, 5, 8, 11, and 14 (bottom figure).

B. Example 2

Consider a wireless ad-hoc network (WANET) with 16 nodes and 16 wireless links as in [23]. $N = 15$

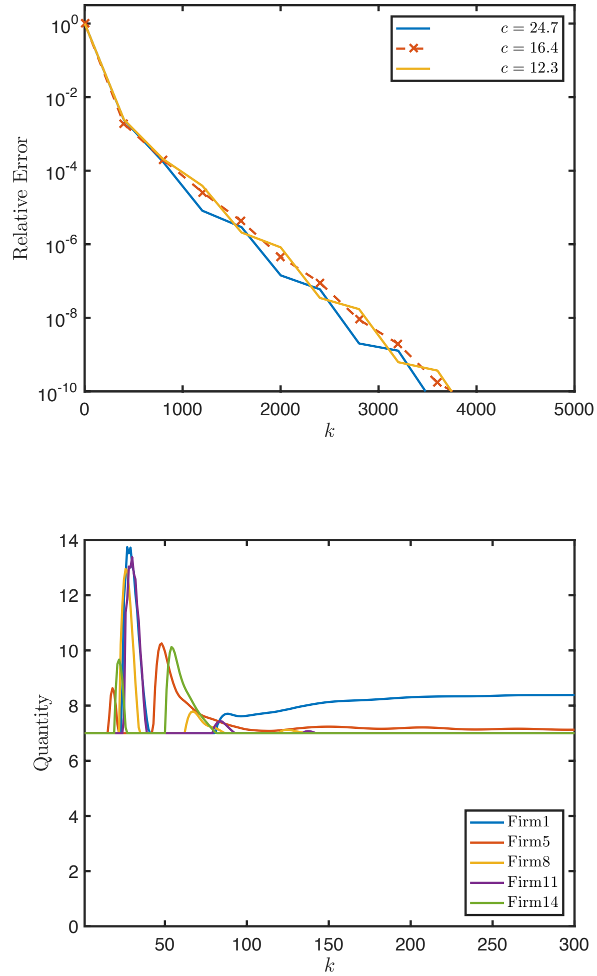

Fig. 2: Convergence of Algorithm 1, or 2 over G_C in Fig. 1. The top figure plots relative error $\frac{\|\mathbf{x}(k)-\mathbf{x}^*\|}{\|\mathbf{x}(0)-\mathbf{x}^*\|}$ v.s. iteration number k. The bottom figure shows the convergence paths of 5 randomly selected firms.

users want to transfer data from a source node to a destination node via this WANET. Fig. 3 (a) shows the topology of the WANET in which solid lines represent links and dashed lines display paths that assigned to users to transfer data. Each link has a positive capacity that restricts the users' data flow . Here is the list of WANET notations:

1) L_j: Link j, $j = 1, \ldots, 16$,
2) R_i: The path assigned to user i, $i = 1, \ldots, 15$,
3) $C_j > 0$: Link j's capacity, $j = 1, \ldots, 16$,
4) $0 \leq x_i \leq 10$: The data flow of user i, $i = 1, \ldots, 15$.

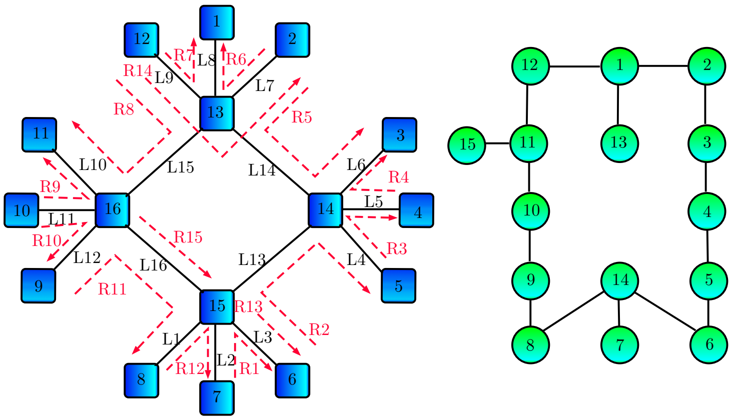

Fig. 3: (a) Wireless Ad-Hoc Network (left). (b) Communication graph G_C (right) for Example 2.

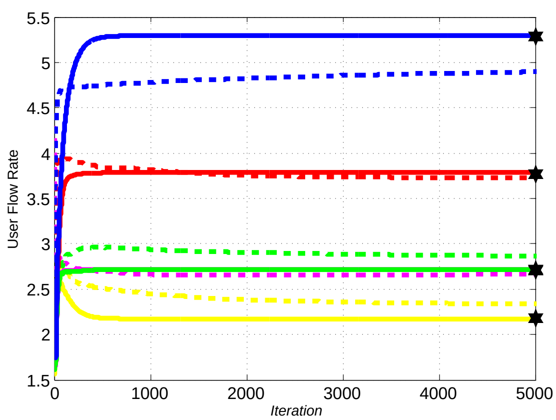

Fig. 4: Flow rates of users 1, 3, 5, 8, 13 using Algorithm 1, or 2 (solid lines) vs. algorithm in [23] (dashed lines). NE represented by black stars.

Each path consists of a set of links, e.g., $R_1 = \{L_2, L_3\}$. For each user i, the cost function J_i is defined by:

$$J_i(x_i, x_{-i}) := \sum_{j:L_j \in R_i} \frac{\kappa}{C_j - \sum_{w:L_j \in R_w} x_w} - \chi_i \log(x_i+1),$$

where $\kappa > 0$ and $\chi_i > 0$ are network-wide known and user-specific parameters, respectively. The communication graph G_C is depicted in Fig. 3 (b). In this case Assumptions 1, 2 hold, while (44) in Assumption 3 is satisfied for $y = x^*$, (see Remark 4). We compare our Algorithm 1, or 2, with the gradient-based one proposed in [23]. The results are shown in Fig. 4, for $\chi_i = 10\ \forall i = 1, \dots, 15$ and $C_j = 10\ \forall j = 1, \dots, 16$, uniform randomly selected initial conditions, $c = 1$ and $\beta_i = 1$, $\forall i = 1, \dots, 15$. The simulation results show that Algorithm 1, or 2 is about two orders of magnitude faster than the one in [23]. The factors that lead to this improvement are as follows: 1) According to the inexact-ADMM algorithm, we used the difference between the estimates of the users as a correction term to update each user's action and his estimates. 2) Unlike the gossiping protocol used in [22], here every user communicates with all of his neighbours (not only one of them) at each iteration.

VI. Conclusions

We designed a distributed NE seeking algorithm in general games by using an inexact-ADMM approach. Each player maintains action estimates for his opponents, which are exchanged with his neighbours over a connected, undirected communication graph. The game was reformulated as a set of augmented optimization problems with virtual consensus constraints for the action estimates, and solved within the framework of inexact-ADMM. An inexact-ADMM algorithm was designed and its convergence to an NE of the game, as well as its convergence rate, was analyzed. The convergence rate of the algorithm was compared with that of an existing gossip-based NE seeking algorithm.

References

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

[2] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *2012 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 5445–5450.

[3] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[4] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2015.

[5] P. Frihauf, M. Krstic, and T. Basar, "Nash equilibrium seeking in noncooperative games," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1192–1207, 2012.

[6] B. Gharesifard and J. Cortes, "Distributed convergence to Nash equilibria in two-network zero-sum games," *Automatica*, vol. 49, no. 6, pp. 1683–1692, 2013.

[7] H. Yin, U. V. Shanbhag, and P. G. Mehta, "Nash equilibrium problems with scaled congestion costs and shared constraints," *IEEE Transactions on Automatic Control*, vol. 56, no. 7, pp. 1702–1708, 2011.

[8] M. Stanković, K. Johansson, and D. Stipanović, "Distributed Seeking of Nash Equilibria with Applications to Mobile Sensor Networks," *IEEE Transactions on Automatic Control*, vol. 57, no. 4, pp. 904–919, 2012.

[9] T. Alpcan and T. Başar, "Distributed algorithms for Nash equilibria of flow control games," in *Advances in Dynamic Games*. Springer, 2005, pp. 473–498.

[10] L. Pavel, "An extension of duality to a game-theoretic framework," *Automatica*, vol. 43, no. 2, pp. 226–237, 2007.

[11] Y. Pan and L. Pavel, "Games with coupled propagated constraints in optical networks with multi-link topologies," *Automatica*, vol. 45, no. 4, pp. 871–880, 2009.

[12] G. Scutari, F. Facchinei, J. S. Pang, and D. P. Palomar, "Real and Complex Monotone Communication Games," *IEEE Trans. on Information Theory*, vol. 60, no. 7, pp. 4197–4231, 2014.

[13] A. Kannan and U. Shanbhag, "Distributed computation of equilibria in monotone Nash games via iterative regularization techniques," *SIAM Journal on Optimization*, vol. 22, no. 4, pp. 1177–1205, 2012.

[14] F. Parise, B. Gentile, S. Grammatico, and J. Lygeros, "Network aggregative games: Distributed convergence to Nash equilibria," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2295–2300.

[15] Y. Lou, Y. Hong, L. Xie, G. Shi, and K. H. Johansson, "Nash equilibrium computation in subnetwork zero-sum games with switching communications," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2920–2935, 2016.

[16] F. Yousefian, A. Nedich, and U. V. Shanbhag, "Self-Tuned Stochastic Approximation Schemes for Non-Lipschitzian Stochastic Multi-User Optimization and Nash Games," *IEEE Transactions on Automatic Control*, vol. 61, no. 7, pp. 1753–1766, 2016.

[17] M. Zhu and E. Frazzoli, "Distributed robust adaptive equilibrium computation for generalized convex games," *Automatica*, vol. 63, pp. 82–91, 2016.

[18] P. Yi and L. Pavel, "A distributed primal-dual algorithm for computation of generalized Nash equilibria with shared affine coupling constraints via operator splitting methods," *arXiv preprint arXiv:1703.05388*, 2017.

[19] J. Marden, "Learning in large-scale games and cooperative control," *Ph.D. Dissertation*, 2007.

[20] N. Li and J. R. Marden, "Designing games for distributed optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 230–242, 2013.

[21] J. Koshal, A. Nedic, and U. V. Shanbhag, "Distributed algorithms for aggregative games on graphs," *Operations Research*, vol. 64, no. 3, pp. 680–704, 2016.

[22] F. Salehisadaghiani and L. Pavel, "Distributed Nash equilibrium seeking: A gossip-based algorithm," *Automatica*, vol. 72, pp. 209–216, 2016.

[23] ——, "Distributed Nash equilibrium seeking by gossip in games on graphs," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 6111–6116.

[24] M. Ye and G. Hu, "Distributed Nash Equilibrium Seeking by A Consensus Based Approach," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2017.

[25] F. Salehisadaghiani and L. Pavel, "Distributed Nash equilibrium seeking via the alternating direction method of multipliers," in *the 20-th IFAC World Congress*, 2017, p. to appear.

[26] W. Shi and L. Pavel, "LANA: an ADMM-like Nash equilibrium seeking algorithm in decentralized environment," in *American Control Conference*, 2017, p. to appear.

[27] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.

[28] J. Nash, "Equilibrium Points in n-Person Games," *Proc. Nat. Acad. Sci. USA*, vol. 36, no. 1, pp. 48–49, 1950.

[29] G. Debreu, "A Social Equilibrium Existence Theorem," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, no. 10, p. 886, 1952.

[30] I. Glicksberg, "A Further Generalization of the Kakutani Fixed Point Theorem, with Application to Nash Equilibrium Points," *Proceedings of the American Mathematical Society*, vol. 3, no. 1, pp. 170–174, 1952.

[31] R. Rockafellar, "On the Maximality of Sums of Nonlinear Monotone Operators," *Transactions of the American Mathematical Society*, vol. 149, no. 1, pp. 75–88, 1970.

[32] F. Facchinei and J.-S. Pang, *Finite-dimensional variational inequalities and complementarity problems*. Springer, 2003.

[33] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, Massachusetts, 1997.

[34] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized Linearized Alternating Direction Method of Multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[35] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6013–6023, 2015.