An asynchronous distributed and scalable generalized Nash equilibrium seeking algorithm for strongly monotone games

Cenedese, Carlo; Belgioioso, Giuseppe; Grammatico, Sergio; Cao, Ming

DOI
10.1016/j.ejcon.2020.08.006

Publication date
2021

Document Version
Final published version

Published in
European Journal of Control

European Journal of Control 58 (2021) 143–151

Contents lists available at ScienceDirect

European Journal of Control

journal homepage: www.elsevier.com/locate/ejcon

An asynchronous distributed and scalable generalized Nash equilibrium seeking algorithm for strongly monotone games

Carlo Cenedese[a,*], Giuseppe Belgioioso[b], Sergio Grammatico[c], Ming Cao[a]

[a] Engineering and Technology Institute Groningen (ENTEG), University of Groningen, the Netherlands
[b] Control System group, TU Eindhoven, Eindhoven, the Netherlands
[c] Delft Center for Systems and Control, TU Delft, the Netherlands

ARTICLE INFO

Article history:
Received 12 March 2020
Revised 3 August 2020
Accepted 8 August 2020
Available online 25 August 2020

Recommended by Prof. T. Parisini

Keywords:
Game theory
Variational GNE
Monotone games
Asynchronous update
Delayed communication
Operator theory

ABSTRACT

In this paper, we present three distributed algorithms to solve a class of Generalized Nash Equilibrium (GNE) seeking problems in strongly monotone games. The first one (SD-GENO) is based on synchronous updates of the agents, while the second and the third (AD-GEED and AD-GENO) represent asynchronous solutions that are robust to communication delays. AD-GENO can be seen as a refinement of AD-GEED, since it only requires node auxiliary variables, enhancing the scalability of the algorithm. Our main contribution is to prove convergence to a v-GNE *variational*-GNE (vGNE) of the game via an operator-theoretic approach. Finally, we apply the algorithms to network Cournot games and show how different activation sequences and delays affect convergence. We also compare the proposed algorithms to a state-of-the-art algorithm solving a similar problem, and observe that AD-GENO outperforms it.

1. Introduction

In modern society, multi-agent network systems arise in several areas, leading to increasing research activities. When self-interested agents interact between each other, one of the best mathematical tools to study the emerging collective behavior is noncooperative game theory over networks. In fact, networked games emerges in several application domains, such as smart grids [8,12], social networks [10,19,20] and distributed robotics [6]. In a game setup, the players (or agents) aim at minimizing a local and private cost function which represents their individual interest, and, at the same time, satisfy local and global constraints, limiting the possible decisions (or strategies/actions). The cost function and constraints of a single player are influenced by the behavior of a fraction of the others, called "neighbors". Thus, each decision is influenced by some local information, which is typically exchanged with the neighbors. One popular notion of solution for these games is the GNE, where no player benefits from unilaterally changing its strategy, see [16].

In [3,20,30], the authors focused on developing synchronous and distributed equilibrium seeking algorithms for noncooperative games, namely, the case in which all the agents update their strategies at the same time. Even though this assumption is quite common, it may lead to sever limitations in the case of agents with heterogeneous computational capabilities in the game. For example, consider an allocation game between several processors, as in [31], and assume that they are of two types: high and low performances. A synchronous update scheme implies that all the players must complete their current update, before a new one can start. Thus, the low performance processors create a bottleneck in the overall performance. To overcome this problem, we focus on developing asynchronous update rules. In fact, it is known that asynchronicity can speed up the convergence, facilitate the insertion of new agents in the network and even increase robustness w.r.t. communication faults, see [5] and references therein.

Among the very first works on asynchronous distributed optimization, the one of Bertsekas and Tsitsiklis in [4] stands out. From there onward, several authors elaborated on these ideas and produced novel results for convex optimization [11,23,27]. In [31], Yi and Pavel developed an asynchronous algorithm to solve noncooperative generalized games subject to equality coupling constraints. This result was enabled by the framework (ARock), recently introduced by Peng et al. in [26], that provides a wide range of asynchronous variations of the classical fixed point iterative algorithms.

In this paper, we propose an asynchronous algorithm robust to delayed information to solve noncooperative games subject to

* Corresponding author.
E-mail address: c.cenedese@rug.nl (C. Cenedese).

https://doi.org/10.1016/j.ejcon.2020.08.006

affine coupling constraints. This work generalizes and extends the current literature on the topic in the following ways.

- We tackle the case of a game subject to inequality coupling constraint (rather than only equality constraint). This drastically broaden the type of problems that can be solved by the proposed approach. For example in signal processing [29] and smart grids [12] inequality constraints arise naturally. This extension cannot be achieved via an extension of the results currently available due to the different control structure considered.
- The algorithms that we develop rely on node variables only, rather than edge variables as in [31]. This, apparently subtle difference, leads to a solution that adopts (almost always) a lower number of variables. So, it is lighter from a computational point of view, requires less memory and involves lighter communication between agents. All these features make the proposed solution achieve overall better performances than [31].

We conclude the paper comparing the proposed algorithms to that in [31], for the case of a Cournot game, showing that our algorithms achieve faster convergence. A preliminary and partial version of these results were presented in [7].

2. Notation

2.1. Basic notation

The set of real, positive, and non-negative numbers are denoted by $\mathbb{R}$, $\mathbb{R}_{>0}$, $\mathbb{R}_{\geq 0}$, respectively; $\overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$. The set of natural numbers is $\mathbb{N}$. For a square matrix $A \in \mathbb{R}^{n\times n}$, its transpose is $A^\top$, $[A]_i$ is the ith row of the matrix and $[A]_{ij}$ represents the element in ith row and jth column. $A \succ 0$ ($A \succeq 0$) stands for a positive definite (semidefinite) matrix. $A \otimes B$ is the Kronecker product of the matrices A and B. The identity matrix is denoted by $I_n \in \mathbb{R}^{n\times n}$. $\boldsymbol{0}$ (resp. $\boldsymbol{1}$) is the vector/matrix with only 0 (resp. 1) elements. For $x_1, \dots, x_N \in \mathbb{R}^n$, the collective vector is denoted by $\boldsymbol{x} := \mathrm{col}((x_i)_{i\in(1,\dots,N)}) := [x_1^\top, \dots, x_N^\top]^\top$. $\mathrm{diag}((A_i)_{i\in(1,\dots,N)})$ describes a block-diagonal matrix with $A_1, \dots, A_N$ on the main diagonal.

2.2. Operator-theoretic notation

The identity operator is denoted by $\mathrm{Id}(\cdot)$. The set valued mapping $N_{\mathcal{C}} : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ denotes the normal cone to the set $\mathcal{C} \subseteq \mathbb{R}^n$, that is $N_{\mathcal{C}}(x) = \{u \in \mathbb{R}^n \mid \sup\langle \mathcal{C} - x, u\rangle \leq 0\}$ if $x \in \mathcal{C}$ and $\varnothing$ otherwise. The graph of a set valued mapping $\mathcal{A} : \mathcal{X} \rightrightarrows \mathcal{Y}$ is $\mathrm{gra}(\mathcal{A}) := \{(x, u) \in \mathcal{X} \times \mathcal{Y} \mid u \in \mathcal{A}(x)\}$. The projection operator over a closed set $S \subseteq \mathbb{R}^n$ is $\mathrm{proj}_S(x) : \mathbb{R}^n \to S$ and it is defined as $\mathrm{proj}_S(x) := \mathrm{argmin}_{y\in S}\|y - x\|^2$. A set valued mapping $\mathcal{F} : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is ℓ-Lipschitz continuous with $\ell > 0$, if $\|u - v\| \leq \ell\|x - y\|$ for all (x, u), $(y, v) \in \mathrm{gra}(\mathcal{F})$; $\mathcal{F}$ is (strictly) monotone if for all $(x, u), (y, v) \in \mathrm{gra}(\mathcal{F})$ $\langle u - v, x - y\rangle \geq (>)0$ holds true, and maximally monotone if it does not exist a monotone operator with a graph that strictly contains $\mathrm{gra}(\mathcal{F})$. Moreover, it is α-strongly monotone if for all $(x, u), (y, v) \in \mathrm{gra}(\mathcal{F})$ it holds $\langle x - y, u - v\rangle \geq \alpha\|x - y\|^2$. The operator $\mathcal{F}$ is η-averaged (η-AVG) with $\eta \in (0, 1)$ if $\|\mathcal{F}(x) - \mathcal{F}(y)\|^2 \leq \|x - y\|^2 - \frac{1-\eta}{\eta}\|(\mathrm{Id} - \mathcal{F})(x) - (\mathrm{Id} - \mathcal{F})(y)\|^2$ for all $x, y \in \mathbb{R}^n$; $\mathcal{F}$ is β-cocoercive if $\beta\mathcal{F}$ is $\frac{1}{2}$-averaged, i.e. firmly nonexpansive (FNE). The resolvent of an operator $\mathcal{A} : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is $\mathrm{J}_{\mathcal{A}} := (\mathrm{Id} + \mathcal{A})^{-1}$.

3. Problem formulation

3.1. Mathematical formulation

We consider a noncooperative game Γ between N agents (or players) subject to affine coupling constraints. We define the game as the triplet $\Gamma := (\boldsymbol{\mathcal{X}}, \{f_i\}_{i\in\{1\dots N\}}, \mathcal{G})$, where its elements are respectively: the collective feasible decision set, the players' local cost functions and the graph describing the communication network. In the following subsections, each one of them is introduced.

3.1.1. Feasible strategy set

Every agent $i \in \mathcal{N} := \{1, \dots, N\}$ has a local decision variable (or strategy) x_i belonging to its private decision set $\Omega_i \subset \mathbb{R}^{n_i}$, namely the set of all those strategies that satisfy the local constraints of player i. The collective vector of all the strategies, or strategy profile of the game, is denoted as $\boldsymbol{x} := \mathrm{col}(x_1, \dots, x_N) \in \mathbb{R}^n$, where $n := \sum_{i\in\mathcal{N}} n_i$. Then, all the decision variables of all the players other than i are represented via the compact notation $\boldsymbol{x}_{-i} := \mathrm{col}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N)$. We assume that the agents are subject to m affine coupling constraints described by the affine function $\boldsymbol{x} \mapsto A\boldsymbol{x} + b$, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Thus, the collective feasible decision set can be written as

$$\boldsymbol{\mathcal{X}} := \boldsymbol{\Omega} \cap \{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \leq b\}, \tag{1}$$

where $\boldsymbol{\Omega} = \prod_{i\in\mathcal{N}} \Omega_i \subset \mathbb{R}^n$, is the Cartesian product of the local constraints sets Ω_i's. Accordingly, the set of all the feasible strategies of each agent $i \in \mathcal{N}$ reads as

$$\mathcal{X}_i(\boldsymbol{x}_{-i}) := \left\{ y \in \Omega_i \mid A_i y - b_i \leq \sum_{j\in\mathcal{N}\setminus\{i\}} \left(b_j - A_j x_j\right) \right\},$$

where $A = [A_1, \dots, A_N]$, $A_i \in \mathbb{R}^{m\times n_i}$ and $\sum_{j=1}^N b_j = b$. The choice of affine coupling constraints is widely spread in the literature of noncooperative games, see e.g., [10,24,30]. Moreover, in [20], Remark 3, it is highlighted that separable and convex coupling constraints can always be rewritten in an affine form. Finally, we introduce some blanket assumptions on this set of feasible strategy, standard in the literature [9,10,16,30,31].

Standing Assumption 1 (Convex constraint sets)**.** For each player $i \in \mathcal{N}$, the set Ω_i is convex, nonempty and compact. The collective feasible set $\boldsymbol{\mathcal{X}}$ satisfies Slater's constraint qualification.

3.1.2. Cost functions

Each player $i \in \mathcal{N}$ has a local cost function $f_i(x_i, \boldsymbol{x}_{-i}) : \Omega_i \times \boldsymbol{\Omega}_{-i} \to \mathbb{R}$, where $\boldsymbol{\Omega}_{-i} := \prod_{j\in\mathcal{N}\setminus\{i\}} \Omega_j$. The coupling between the players appears not only in the constraints but also in the cost function, due to the dependency on both x_i and $\boldsymbol{x}_{-i}$. Next, we assume some properties for these functions that are extensively used in the literature [16,30].

Standing Assumption 2 (Convex and differentiable cost functions)**.** For all $i \in \mathcal{N}$, the cost function $f_i(x_i, \boldsymbol{x}_{-i})$ is continuously differentiable and convex in x_i.

3.1.3. Communication network

The communication between agents is described by an *undirected and connected* graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of edges. Given two agents $i, j \in \mathcal{N}$, the couple (i, j) belongs to $\mathcal{E}$, if agent i shares information with agent j and vice versa. Then we say that j is a neighbour of i, i.e., $j \in \mathcal{N}_i$ where $\mathcal{N}_i$ is the neighbourhood of i. The number of edges in the graph is denoted by $E := |\mathcal{E}|$. To define the *incidence matrix* $V \in \mathbb{R}^{E\times N}$ associated to $\mathcal{G}$, let us label the edges as e_l, for $l \in \{1, \dots, E\}$. We define the entry $[V]_{li} := 1$ (resp. -1) if $e_l = (i, \cdot)$ (resp. $e_l = (\cdot, i)$) and 0 otherwise. The decision of which of the two agents composing an edge is the sink and which the source is arbitrary. By construction, $V\mathbf{1}_N = \mathbf{0}_N$. Then, we define $\mathcal{E}_i^{\mathrm{out}}$ (resp. $\mathcal{E}_i^{\mathrm{in}}$) as the set of all the indexes l of the edges e_l that start from (resp. end in) node i, and hence $\mathcal{E}_i = \mathcal{E}_i^{\mathrm{out}} \cup \mathcal{E}_i^{\mathrm{in}}$. The *node Laplacian* $L \in \mathbb{R}^{N\times N}$ of an undirected graph is a symmetric matrix defined by $L := V^\top V$. Another important property of L, used in the remainder, is $L\mathbf{1}_N = \mathbf{0}_N$.

3.2. Generalized Nash equilibrium

In summary, the considered generalized game is described by the following set of inter-dependent optimization problems:

$$\forall i \in \mathcal{N} \; : \; \begin{cases} \underset{y \in \mathbb{R}^{n_i}}{\text{argmin}} & f_i(y, \boldsymbol{x}_{-i}) \\ \text{s.t.} & y \in \mathcal{X}_i(\boldsymbol{x}_{-i}). \end{cases} \tag{2}$$

The most popular equilibrium concept considered for noncooperative games with coupling constraints is the *generalized Nash equilibrium*, thus the configuration in which all the relations in (2) simultaneously hold.

Definition 1 (Generalized Nash Equilibrium)**.** A collective strategy $\boldsymbol{x}^* \in \boldsymbol{\mathcal{X}}$ is a generalized Nash equilibrium (GNE) if, for each player i, it holds

$$f_i(x_i^*, \boldsymbol{x}_{-i}^*) \leq \inf\{ f_i(y, \boldsymbol{x}_{-i}^*) \mid y \in \mathcal{X}_i(\boldsymbol{x}_{-i}^*) \}.$$

In this work, we focus on a subset of GNE, the so called variational GNE (vGNE), a class of equilibria that is considered in many other works throughout the literature – see [3,16,21,22]. The name of these equilibria derives from the fact that they can be formulated as the solutions to a *variational inequality* (VI). An important property of these equilibria is that each agent faces the same penalty to fulfill the coupling constraints, which is particularly useful to represent a "fair" competition between agents [16]. Variational GNE can be seen as a particular case of the concept of *normalized equilibrium points*, firstly introduced by Rosen in [28] and further studied in [10,25].

To properly characterize this set, we define the *pseudo-gradient* mapping (or game mapping) of (2), as

$$F(\boldsymbol{x}) = \operatorname{col}((\nabla_{x_i} f_i(x_i, \boldsymbol{x}_{-i}))_{i \in \mathcal{N}}). \tag{3}$$

The pseudo-gradient gathers in a collective vector form the gradients of the cost functions each w.r.t. the local decision variable. Next, we introduce some standard technical assumptions, e.g., [2,13].

Standing Assumption 3. The mapping F in (3) is α-strongly monotone and ℓ-Lipschitz continuous, for some $\alpha, \ell > 0$.

When Standing assumption 2 holds true, the mapping F is single valued and the set of vGNE of the game in (2) corresponds to the solution to VI$(F, \boldsymbol{\mathcal{X}})$, namely the problem of finding a vector $\boldsymbol{x}^* \in \boldsymbol{\mathcal{X}}$ such that

$$\langle F(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq 0, \quad \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}. \tag{4}$$

The continuity of F (Assumption 2) and compactness of $\boldsymbol{\mathcal{X}}$ (Assumption 1) imply the existence of a solution to VI$(F, \boldsymbol{\mathcal{X}})$, while the strong monotonicity (Assumption 3) entails uniqueness, see [15], Th. 2.3.3.

Next, let us define the KKT conditions associated to the game in (2). The strong duality of the problem (Assumptions 1 and 2) implies that, if $\boldsymbol{x}^*$ is a GNE of (2), then there exist N dual variables $\lambda_i^* \in \mathbb{R}^m_{\geq 0}$, for all $i \in \mathcal{N}$, such that the following inclusions are satisfied:

$$\forall i \in \mathcal{N} \; : \; \begin{cases} \mathbf{0} \in \nabla_{x_i} f_i(x_i^*, x_{-i}^*) + A_i^\top \lambda_i^* + N_{\Omega_i}(x_i^*), \\ \mathbf{0} \in b - A\boldsymbol{x}^* + N_{\mathbb{R}^m_{\geq 0}}(\lambda_i^*). \end{cases} \tag{5}$$

Instead of looking for the solution of the general case where $\lambda_1^*, \dots, \lambda_N^*$ may be different, we examine the special case when $\lambda^* := \lambda_1^* = \cdots = \lambda_N^*$, namely

$$\forall i \in \mathcal{N} \; : \; \begin{cases} \mathbf{0} \in \nabla_{x_i} f_i(x_i^*, x_{-i}^*) + A_i^\top \lambda^* + N_{\Omega_i}(x_i^*), \\ \mathbf{0} \in b - A\boldsymbol{x}^* + N_{\mathbb{R}^m_{\geq 0}}(\lambda^*). \end{cases} \tag{6}$$

It follows from [17], Th. 3.1(ii), that the KKT inclusions in (6) correspond to the solution set to VI$(F, \boldsymbol{\mathcal{X}})$. Thus, every solution $\boldsymbol{x}^*$ to VI$(F, \boldsymbol{\mathcal{X}})$ is also a GNE of the game in (2), [17, Th. 3.1(i)]. Since the solution set to VI$(F, \boldsymbol{\mathcal{X}})$ is a singleton, we conclude that there exists a unique vGNE of the game (2).

4. Synchronous distributed GNE seeking algorithm

We first introduce the synchronous counterpart of AD-GENO, i.e., the *Synchronous Distributed GNE Seeking Algorithm with Node variables* (SD-GENO). The derivation of the algorithm is based on an operator splitting approach to solve the KKT system in (6). A similar approach was also adopted in [3,30] in the contest of GNE finding problems.

4.1. Algorithm design

The KKT conditions of each agent i in (5) are satisfied by a couple (x_i, λ_i), where the dual variables λ_i may be different among the players. If we enforce the consensus among the dual variables, then the unique solution of the inclusions is the vGNE of the game. This is achieved by exploiting the fact that $\ker(V) = \operatorname{span}(\mathbf{1})$ and introducing the auxiliary variables σ_l, $l \in \{1, \dots, E\}$, one for every edge in the graph. Using the notations $\boldsymbol{\lambda} := \operatorname{col}((\lambda_i)_{i \in \mathcal{N}}) \in \mathbb{R}^{mN}$, $\Lambda := \operatorname{diag}((A_i)_{i \in \mathcal{N}}) \in \mathbb{R}^{mN \times n}$, $\bar{b} := \operatorname{col}((b_i)_{i \in \mathcal{N}}) \in \mathbb{R}^{mN}$, $\boldsymbol{\sigma} := \operatorname{col}((\sigma_l)_{l \in \{1 \dots E\}}) \in \mathbb{R}^{mE}$, $\boldsymbol{V} := V \otimes I_m \in \mathbb{R}^{mE \times mN}$ and $\boldsymbol{L} := L \otimes I_m \in \mathbb{R}^{mE \times mN}$, we cast the augmented version of the inclusions in (5) by

$$\begin{aligned} \mathbf{0} &\in F(\boldsymbol{x}) + \Lambda^\top \boldsymbol{\lambda} + N_{\boldsymbol{\Omega}}(\boldsymbol{x}) \\ \mathbf{0} &\in \bar{b} - \Lambda \boldsymbol{x} + N_{\mathbb{R}^{mN}_{\geq 0}}(\boldsymbol{\lambda}) + \boldsymbol{L}\boldsymbol{\lambda} + \rho \boldsymbol{V}^\top \boldsymbol{\sigma} \\ \mathbf{0} &= -\rho \boldsymbol{V} \boldsymbol{\lambda}, \end{aligned} \tag{7}$$

where $\rho \in \mathbb{R}_{>0}$. In (7), the term $\boldsymbol{L}\boldsymbol{\lambda}$ accelerates the convergence of the dual variables to consensus.

A solution $\varpi = \operatorname{col}(\boldsymbol{x}^*, \boldsymbol{\sigma}^*, \boldsymbol{\lambda}^*)$ of the above inclusions can be equivalently recast as a zero of the sum of two mappings $\mathcal{A}$ and $\mathcal{B}$ defined as

$$\begin{aligned} \mathcal{A} : \varpi \mapsto & \begin{bmatrix} 0 & 0 & \Lambda^\top \\ 0 & 0 & -\rho \boldsymbol{V} \\ -\Lambda & \rho \boldsymbol{V}^\top & 0 \end{bmatrix} \varpi + \begin{bmatrix} N_{\boldsymbol{\Omega}}(\boldsymbol{x}) \\ 0 \\ N_{\mathbb{R}^{mN}_{\geq 0}}(\boldsymbol{\lambda}) \end{bmatrix} \\ \mathcal{B} : \varpi \mapsto & \begin{bmatrix} F(\boldsymbol{x}) \\ 0 \\ \bar{b} + \boldsymbol{L}\boldsymbol{\lambda} \end{bmatrix}. \end{aligned} \tag{8}$$

In fact, $\varpi^* \in \operatorname{zer}(\mathcal{A} + \mathcal{B})$ if and only if ϖ^* satisfies (7).

Next, we show that the zeros of $\mathcal{A} + \mathcal{B}$ characterize the vGNE of the original game.

Proposition 1. *Let $\mathcal{A}$ and $\mathcal{B}$ be as in (8). Then the following hold:*

(i) $\operatorname{zer}(\mathcal{A} + \mathcal{B}) \neq \varnothing$,

(ii) *if $\operatorname{col}(\boldsymbol{x}^*, \boldsymbol{\sigma}^*, \boldsymbol{\lambda}^*) \in \operatorname{zer}(\mathcal{A} + \mathcal{B})$ then $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ satisfies the KKT conditions in (5), with $\lambda_1^* = \cdots = \lambda_N^*$, hence $\boldsymbol{x}^*$ is the unique vGNE of the game in (2).*

The proof is attained by exploiting the property that $\ker(V) = \ker(L)$, for the graph described in Section 3.1.3. The steps are similar to those in [30, Th. 2]. We omit them here for brevity reasons.

Several researchers have analyzed the problem of finding a zero of the sum of two monotone operators. The so called *splitting methods* represent one of the most popular approach developed to attain an iterative algorithm to solve this class of problem - see [14], [1, Ch. 26].

Lemma 1. *The mappings $\mathcal{A}$ and $\mathcal{B}$ in (8) are maximally monotone. Moreover, $\mathcal{B}$ is χ-cocoercive, where $\chi := \min\left\{ \frac{\alpha}{\ell^2}, \lambda_{\max}(L)^{-1} \right\}$.*

The properties of the operators proved above drive us to select the *preconditioned forward-backward* splitting (PFB) to derive a distributed and iterative algorithm seeking $\operatorname{zer}(\mathcal{A} + \mathcal{B})$. This approach was previously adopted by other researchers, e.g., [30].

The PFB splitting operator reads as

$$T := \mathrm{J}_{\Phi^{-1}\mathcal{A}} \circ (\mathrm{Id} - \Phi^{-1}\mathcal{B}) . \tag{9}$$

The so-called preconditioning matrix Φ is defined by

$$\Phi := \begin{bmatrix} \boldsymbol{\tau}^{-1} & 0 & -\Lambda^\top \\ 0 & \delta^{-1} I_{mM} & \rho \boldsymbol{V} \\ -\Lambda & \rho \boldsymbol{V}^\top & \boldsymbol{\varepsilon}^{-1} \end{bmatrix} \tag{10}$$

where $\delta \in \mathbb{R}_{>0}$, $\boldsymbol{\varepsilon} = \mathrm{diag}((\varepsilon_i)_{i\in\mathcal{N}}) \otimes I_m$ with $\varepsilon_i > 0$, for all $i \in \mathcal{N}$ and $\boldsymbol{\tau}$ is defined in a similar way.

The update rule of the algorithm is obtained by including a relaxation step, i.e.,

$$\begin{aligned} \tilde{\varpi}(k) &= T\varpi(k) \\ \varpi(k+1) &= \varpi(k) + \eta(\tilde{\varpi}(k) - \varpi(k)). \end{aligned} \tag{11}$$

It comes from (9) that $\mathrm{fix}(T) = \mathrm{zer}(\mathcal{A}+\mathcal{B})$, in fact $\varpi \in \mathrm{fix}(T) \Leftrightarrow \varpi \in T\varpi \Leftrightarrow 0 \in \Phi^{-1}(\mathcal{A}+\mathcal{B})\varpi \Leftrightarrow \varpi \in \mathrm{zer}(\mathcal{A}+\mathcal{B})$, see [1, Th. 26.14].

In the remainder of this section, we provide the complete derivation of SD-GENO, obtained directly from (11). In the following, we denote $\varpi := \varpi(k)$, $\varpi^+ := \varpi(k+1)$ and $\tilde{\varpi} := \tilde{\varpi}(k)$ to simplify the notation. Consider $\tilde{\varpi} = T\varpi$. From (9) it holds that $\tilde{\varpi} = \mathrm{J}_{\Phi^{-1}\mathcal{A}} \circ (\mathrm{Id} - \Phi^{-1}\mathcal{B})\varpi \Leftrightarrow \Phi(\varpi - \tilde{\varpi}) \in \mathcal{A}\tilde{\varpi} + \mathcal{B}\varpi$, thus

$$\mathbf{0} \in \mathcal{A}\tilde{\varpi} + \mathcal{B}\varpi + \Phi(\tilde{\varpi} - \varpi). \tag{12}$$

The update rule of each components of ϖ is attained by analyzing the row blocks of (12). The first reads as $\mathbf{0} \in F(\boldsymbol{x}) + N_{\boldsymbol{\Omega}}(\tilde{\boldsymbol{x}}) + \boldsymbol{\tau}^{-1}(\tilde{\boldsymbol{x}} - \boldsymbol{x}) + \Lambda^\top\boldsymbol{\lambda}$. By solving this inclusion by $\tilde{\boldsymbol{x}}$, we attain the update rule for the primal variables:

$$\tilde{\boldsymbol{x}} = \mathrm{J}_{N_{\boldsymbol{\Omega}}} \circ \big(\boldsymbol{x} - \boldsymbol{\tau}(F(\boldsymbol{x}) + \Lambda^\top\boldsymbol{\lambda})\big) . \tag{13}$$

Similarly, from the second row block of (12), we attain the update for $\tilde{\boldsymbol{\sigma}}$, i.e.,

$$\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma} + \delta\rho\boldsymbol{V}\boldsymbol{\lambda}. \tag{14}$$

Finally, the third row block of (12) is $\mathbf{0} \in \bar{b} + \boldsymbol{L}\boldsymbol{\lambda} + N_{\mathbb{R}^{mN}_{\geq 0}}(\tilde{\boldsymbol{\lambda}}) + \Lambda(2\tilde{\boldsymbol{x}} - \boldsymbol{x}) + \rho\boldsymbol{V}^\top(2\tilde{\boldsymbol{\sigma}} - \boldsymbol{\sigma}) + \boldsymbol{\varepsilon}^{-1}(\tilde{\boldsymbol{\lambda}} - \boldsymbol{\lambda})$, from which we obtain

$$\begin{aligned} \tilde{\boldsymbol{\lambda}} &= \mathrm{J}_{N_{\mathbb{R}^{mN}_{\geq 0}}} \circ \big(\boldsymbol{\lambda} + \boldsymbol{\varepsilon}(\Lambda(2\tilde{\boldsymbol{x}} - \boldsymbol{x}) - \bar{b} - \rho\boldsymbol{V}^\top(2\tilde{\boldsymbol{\sigma}} - \boldsymbol{\sigma}) - \boldsymbol{L}\boldsymbol{\lambda})\big) \\ &\overset{(14)}{=} \mathrm{proj}_{\mathbb{R}^{mN}_{\geq 0}}\big(\boldsymbol{\lambda} + \boldsymbol{\varepsilon}(\Lambda(2\tilde{\boldsymbol{x}} - \boldsymbol{x}) - \bar{b} \\ &\quad -\rho\boldsymbol{V}^\top\boldsymbol{\sigma} - (2\delta\rho^2 + 1)\boldsymbol{L}\boldsymbol{\lambda})\big). \end{aligned} \tag{15}$$

Note that, the update of $\tilde{\boldsymbol{\lambda}}$ depends only on the aggregate information $\boldsymbol{V}^\top\boldsymbol{\sigma}$. We can exploit this feature to replace the edge auxiliary variables σ_l's, with a single variable for each agent i defined by $z_i := \big([V^\top]_i \otimes I_m\big)\boldsymbol{\sigma} \in \mathbb{R}^{mN}$. Recalling that $\boldsymbol{V}^\top\boldsymbol{V} = L \otimes I_m =: \boldsymbol{L}$, we compute the update rule of these new variables and replace (14) by

$$\tilde{\boldsymbol{z}} = \boldsymbol{V}^\top\boldsymbol{\sigma} + \delta\rho\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\lambda} = \boldsymbol{z} + \delta\rho\boldsymbol{L}\boldsymbol{\lambda}. \tag{16}$$

Consequently, (15) is modified accordingly as

$$\begin{aligned} \tilde{\boldsymbol{\lambda}} = \mathrm{proj}_{\mathbb{R}^{mN}_{\geq 0}}\big(&\boldsymbol{\lambda} + \boldsymbol{\varepsilon}(\Lambda(2\tilde{\boldsymbol{x}} - \boldsymbol{x}) - \bar{b} \\ &- \rho\boldsymbol{z} - (2\delta\rho^2 + 1)\boldsymbol{L}\boldsymbol{\lambda})\big). \end{aligned} \tag{17}$$

To ensure that this change of variables does not affect the equilibrium of the game, we introduce the following result proving that an equilibrium point of the new set of equations is indeed a vGNE of (2).

Theorem 1. *If* $\mathrm{col}(\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{\lambda}^*)$ *is a solution to the Eqs. (13), (16), (17), with* $\mathbf{1}^\top\boldsymbol{z}^* = 0$, *then* $\boldsymbol{x}^*$ *is a vGNE of the game in (2).*

Remark 1. In [31], the algorithm SYDNEY achieves convergence to the vGNE of the game (2), when this is subject to equality coupling constraints only. This solution relies on edge auxiliary variables to enforce the consensus of the λ_i's. Therefore, the number of variables that each agent has to store is $\mathcal{O}(N)$.

The change of "variables", from $\boldsymbol{\sigma}$ to $\boldsymbol{z}$, is convenient when the edges outnumber the nodes, which is almost always the case. In fact, a lower number of variables leads to an overall increment in the algorithmic efficiency and to a fixed memory requirement for each player that does not increase with N. Furthermore, if SYDNEY in [31] is modified to address inequality constraints, it would require an additional round of communication between the agents, making it more demanding and slower than SD-GENO.

4.2. Synchronous, distributed algorithm with node variables (SD-GENO)

The complete formulation of the algorithm is obtained by gathering together all the update rules introduced in the previous section, i.e., (13), (16), (17) and adding a relaxation step. The algorithm in compact form is expressed as

$$\begin{cases} \tilde{\boldsymbol{x}} = \mathrm{proj}_{\boldsymbol{\Omega}}\big(\boldsymbol{x} - \boldsymbol{\tau}(F(\boldsymbol{x}) + \Lambda^\top\boldsymbol{\lambda})\big) \\ \tilde{\boldsymbol{z}} = \boldsymbol{z} + \rho\delta\boldsymbol{L}\boldsymbol{\lambda} \\ \tilde{\boldsymbol{\lambda}} = \mathrm{proj}_{\mathbb{R}^{mN}_{\geq 0}}\big(\boldsymbol{\lambda} + \boldsymbol{\varepsilon}(\Lambda(2\tilde{\boldsymbol{x}} - \boldsymbol{x}) - \bar{b} \\ \quad -\rho\boldsymbol{z} - (2\delta\rho^2 + 1)\boldsymbol{L}\boldsymbol{\lambda})\big) \\ \boldsymbol{x}^+ = \boldsymbol{x} + \eta(\tilde{\boldsymbol{x}} - \boldsymbol{x}) \\ \boldsymbol{z}^+ = \boldsymbol{z} + \eta(\tilde{\boldsymbol{z}} - \boldsymbol{z}) \\ \boldsymbol{\lambda}^+ = \boldsymbol{\lambda} + \eta(\tilde{\boldsymbol{\lambda}} - \boldsymbol{\lambda}), \end{cases} \tag{18}$$

while the local updates and the initial condition of SD-GENO are provided in Algorithm 1. It is composed of two main phases:

Algorithm 1: SD-GENO.

Input: $k = 0$, for all $i \in \mathcal{N}$, $x_i(0) \in \mathbb{R}^{n_i}$, $\lambda_i(0) \in \mathbb{R}^m$, $z_i(0) = \mathbf{0}_m$. Choose δ, ε_i, τ_i satisfying (19), while $\eta \in (0, 1)$ and $\rho \in (0, 1]$.

Iteration k:

Communication: each $i \in \mathcal{N}$ gathers $\lambda_j(k)$ from the neighbors and updates the disagreement vector $d_i(k) := \sum_{j\in\mathcal{N}_i}(\lambda_i - \lambda_j)$

Local update, for $i \in \mathcal{N}$ **do**

$$\begin{aligned} \tilde{x}_i &= \mathrm{proj}_{\Omega_i}\big(x_i - \tau_i(\nabla_i f_i(x_i, \boldsymbol{x}_{-i}) + A_i^\top\lambda_i)\big) \\ \tilde{z}_i &= z_i + \rho\,\delta\,d_i(k) \\ \tilde{\lambda}_i &= \mathrm{proj}_{\mathbb{R}^m_{\geq 0}}\big(\lambda_i + \varepsilon_i(A_i(2\tilde{x}_i - x_i) - b_i \\ &\qquad -\rho z_i - (2\delta\rho^2 + 1)d_i(k))\big) \\ x_i^+ &= x_i + \eta(\tilde{x}_i - x_i) \\ z_i^+ &= z_i + \eta(\tilde{z}_i - z_i) \\ \lambda_i^+ &= \lambda_i + \eta(\tilde{\lambda}_i - \lambda_i) \end{aligned}$$

$k \leftarrow k + 1$

the communication with the neighbors and the local update. First, each agent gathers the information about the strategies and the dual variables of the neighbors. Next, the local update is performed, based on a gradient descent and dual ascend structure. It is worth noticing that only one round of communication is required at each iteration of SD-GENO.

The convergence of SD-GENO to the vGNE of the game in (2) is proven in the following theorem.

Theorem 2. *Set the step sizes* ε_i, δ, τ_i, *for all* $i \in \mathcal{N}$, *and* $\vartheta \in \mathbb{R}$ *such that*

$$\tau_i \leq (\|A_i\| + \vartheta)^{-1} \tag{19a}$$

$$\delta \leq (2\rho + \vartheta)^{-1} \tag{19b}$$

$$\varepsilon_i \leq (\rho|\mathcal{N}_i| + \|A_i\| + \vartheta)^{-1}, \tag{19c}$$

$$\vartheta > \frac{1}{2\chi} \tag{19d}$$

with χ as in Lemma 1 and $\eta \in \left(0, \frac{4\chi\vartheta-1}{2\chi\vartheta}\right)$. Then, the sequence $(\boldsymbol{x}(k))_{k\in\mathbb{N}}$ generated by SD-GENO (Algorithm 1) converges to the vGNE of the game in (2).

5. Asynchronous distributed algorithm with edge variables (AD-GEED)

In the case of heterogeneous agents with very different update rates, SD-GENO can converge slowly, due to its synchronous structure. To overcome this limitation, we introduce here the *Asynchronous Distributed GNE Seeking Algorithm with Edge variables* (AD-GEED). It uses edge auxiliary variables $\{\sigma_l\}_{l\in\{1\dots E\}}$ and an asynchronous update to compute the vGNE of the game in (2). As discussed in the previous section, the use of edge-based auxiliary variables may lead to a limited scalability of the final algorithm. In Section 6, we use AD-GEED as a starting point to develop an algorithm relying on node variables only. From a technical point of view, the asynchronicity is achieved by exploiting an asynchronous framework for fixed-point iterations, the so called "ARock" framework, developed in [26].

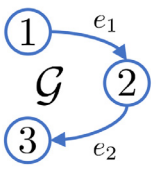
5.1. Algorithm design

The update rule in the asynchronous case, is similar to that in (11), with the main difference that, at each iteration, only one agent $i \in \mathcal{N}$ updates its strategy x_i, dual variable λ_i and local auxiliary variables $\{\sigma_l\}_{l\in\mathcal{E}_i^{\text{out}}}$. To mathematically formulate this concept, we introduce N diagonal matrices $\boldsymbol{H}_i$, where $[\boldsymbol{H}_i]_{jj}$ is 1 if the j-th element of $\text{col}(\boldsymbol{x},\boldsymbol{\sigma},\boldsymbol{\lambda})$ is an element of $\text{col}(x_i, \{\sigma_l\}_{l\in\mathcal{E}_i^{\text{out}}}, \lambda_i)$ and 0 otherwise. The matrix $\boldsymbol{H}_i$ triggers the update of those elements in ϖ that are associated to agent i. We assume that the choice of which agent performs the update during the iteration k is ruled by an i.i.d. random variable $\zeta(k)$, taking values in $\boldsymbol{H} := \{\boldsymbol{H}_i\}_{i\in\mathcal{N}}$. Given a discrete probability distribution $(p_1, \dots, p_N)$, let $\mathbb{P}[\zeta(k) = \boldsymbol{H}_i] = p_i$, for all $i \in \mathcal{N}$. Therefore, the update rule in the asynchronous case is cast as

$$\varpi(k+1) = \varpi(k) + \eta\zeta(k)(T\varpi(k) - \varpi(k)). \tag{20}$$

An illustrative example is now provided to clarify how to construct the set $\boldsymbol{H}$.

Example 1. Consider a game with $N = 3$, $E = 2$, $m = 1$, $n_i = 1$, $i = 1, 2, 3$ and ϖ is the collective vector of all the strategies and auxiliary variables in the game. The communication network is described by the undirected graph $\mathcal{G}$, where the arrows describe the convention adopted for the edges.



$$\mathcal{E}_1^{\text{out}} = \{e_1\} \qquad \mathcal{E}_2^{\text{out}} = \{e_2\} \qquad \mathcal{E}_3^{\text{out}} = \varnothing \qquad \varpi = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \sigma_1 \\ \sigma_2 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

In this case, $\boldsymbol{H}$ is a set of three 8×8 matrices, namely

$$\boldsymbol{H}_1 := \text{diag}((1,0,0,1,0,1,0,0))$$
$$\boldsymbol{H}_2 := \text{diag}((0,1,0,0,1,0,1,0))$$
$$\boldsymbol{H}_3 := \text{diag}((0,0,1,0,0,0,0,1)).$$

If during iteration k agent 2 is updating, (20) turns into

$$\varpi(k+1) = \varpi(k) + \eta\boldsymbol{H}_2(T\varpi(k) - \varpi(k)). \tag{21}$$

So, the only elements of ϖ that change are $(x_2, \sigma_2, \lambda_2)$, precisely the variables associated to agent 2.

We assume that each agent i is equipped with public and private memory, the former is used by the neighbors to write their strategies (and dual/auxiliary variables) when they complete an update. The latter instead is used by i to store a copy of the public memory, when it is performing a local update. This memory is not accessible to the neighbors, so it ensures the consistency of the local updates, refer also to [26]. If an agent $j \in \mathcal{N}_i$ concludes its update while agent i is still computing its future strategy during iteration k, then the value of the strategy of j, which agent i is using, becomes outdated. We denote the vector of possibly outdated strategy used for the update during iteration k as $\hat{\varpi}(k)$. All the variables updated by an agent i, i.e., x_i, λ_i and $\{\sigma_l\}_{l\in\mathcal{E}_i^{\text{out}}}$, share the same delay $\varphi_i(k) \in \mathbb{N}$, since they are written at the same moment in the neighbors' public memories of its neighbors. Technically, the components of $\hat{\varpi}(k)$ associated to agent $j \neq i$ used during the k-th iteration by agent i for the update are $\text{col}(x_j(k-\varphi_j(k)), \{\sigma_\ell(k-\varphi_j(k))\}_{\ell\in\mathcal{E}_j^{\text{out}}}, \lambda_j(k-\varphi_j(k)))$, hence $\hat{\varpi}_j(k) = \varpi_j(k-\varphi_j(k))$.

According to this, the final formulation of the update rule (20) becomes

$$\varpi(k+1) = \varpi(k) + \eta\zeta(k)(T - \text{Id})\hat{\varpi}(k). \tag{22}$$

The only assumption that we impose over the delay, is boundedness, as formalized next.

Assumption 4 (Bounded maximum delay)**.** The delays are uniformly upper bounded, i.e. there exists $\bar{\varphi} > 0$ such that $\sup_{k\in\mathbb{N}_{\geq 0}} \max_{i\in\mathcal{N}}\{\varphi_i(k)\} \leq \bar{\varphi} < +\infty$.

The local update rules of AD-GEED are presented in Algorithm 2 and they are achieved via steps similar to those introduced in Section 4.1 for SD-GENO. To ease the notation, for each agent $j \in \mathcal{N}$, we define $\hat{x}_j := x_j(k+\varphi_j(k))$, $\hat{\lambda}_j := \lambda_j(k-\varphi_j(k))$ and $\hat{\sigma}_l := \sigma_l(k-\varphi_j(k))$, for all $l \in \mathcal{E}_j^{\text{out}}$, and furthermore $\hat{\boldsymbol{x}} := \text{col}((\hat{x}_j)_{j\in\mathcal{N}})$, $\hat{\boldsymbol{\lambda}} := \text{col}((\hat{\lambda}_j)_{j\in\mathcal{N}})$, $\hat{\boldsymbol{\sigma}} := \text{col}((\hat{\sigma}_j)_{j\in\mathcal{N}})$. Notice that each agent has always access to the most recent value of its variables, i.e., $\varphi_i(k) = 0$ for every agent $i \in \mathcal{N}$.

The following convergence theorem is achived by exploiting the results in [26] for a Krasnosel'skiĭ asynchronous iteration.

Theorem 3. *For every $i \in \mathcal{N}$, choose ε_i, δ, τ_i as in (19), and let $\eta \in \left(0, \frac{cNp_{\min}}{2\bar{\varphi}\sqrt{p_{\min}}+1}\left(2 - \frac{1}{2\chi\vartheta}\right)\right]$ and $c \in (0, 1)$. Then, the sequence $(\boldsymbol{x}(k))_{k\in\mathbb{N}}$ generated by AD-GEED (Algorithm 2) converges to the vGNE of the game in (2) almost surely.*

Remark 2. If the probability distribution is uniform, i.e., $p_{\min} = \frac{1}{N}$, and we choose $\vartheta = \frac{1}{\chi}$, then the bounds on the relaxation step simplify as $\eta \in \left(0, \frac{3}{2}\frac{c\sqrt{N}}{2\bar{\varphi}+\sqrt{N}}\right]$. Moreover, if there is no delay, so $\bar{\varphi} = 0$, or the number of agents is very high, the bounds may be chosen independently from the number of players, e.g., as $\eta \in (0, 1]$.

The structure of AD-GEED is similar to that of ADAGNES in [31, Algorithm 1], where edge auxiliary variables are used to achieve consensus over the dual variables. However, unlike ADAGNES, our algorithm can handle inequality coupling constraints. Moreover, it has better performances, in terms of convergence time, according to our numerical experience, see Figure 3.

Algorithm 2: AD-GEED.

Input: $k=0$, $\boldsymbol{x}^0 \in \mathbb{R}^n$, $\boldsymbol{\lambda}^0 \in \mathbb{R}^{mN}$, $\boldsymbol{\sigma}^0 = \mathbf{0}_{mM}$, chose δ, ε_i, τ_i satisfying (**??**) and $\eta \in (0,1)$.

Iteration k**:** Select the agent i_k with probability $\mathbb{P}[\zeta(k) = \boldsymbol{H}_{i_k}] = p_{i_k}$

Reading: Agent i_k copies in its private memory the current values of the public memory, i.e. $\hat{x}_j$, $\hat{\lambda}_j$, $\forall j \in \mathcal{N}_{i_k}$ and $\hat{\sigma}_l$, $\forall l \in \mathcal{E}^{\text{in}}_{i_k}$ and $l \in \mathcal{E}^{\text{out}}_j$.

Update:

$$\tilde{x}_{i_k} = \text{proj}_{\Omega_{i_k}}\left(x_{i_k} - \tau_{i_k}(\nabla_{i_k} f_{i_k}(x_{i_k}, \hat{\boldsymbol{x}}_{-i_k}) + A_{i_k}^\top \lambda_{i_k})\right)$$

$$\tilde{\sigma}_l = \sigma_l + \delta\rho([V]_l \otimes I_m)\hat{\boldsymbol{\lambda}}\,, \quad \forall l \in \mathcal{E}^{\text{out}}_{i^k}$$

$$\tilde{\lambda}_{i_k} = \text{proj}_{\mathbb{R}^m_{\geq 0}}\Big(\lambda_{i_k} + \varepsilon_{i_k}\big(A_{i_k}(2\tilde{x}_{i_k} - x_{i_k}) - b_{i_k} - \rho([V^\top]_{i_k} \otimes I_m)\hat{\boldsymbol{\sigma}} - (2\delta\rho^2 + 1)\textstyle\sum_{j\in\mathcal{N}_{i_k}}(\lambda_i - \hat{\lambda}_j)\big)\Big)$$

$$x^+_{i_k} = x_{i_k} + \eta(\tilde{x}_{i_k} - x_{i_k})$$
$$\sigma^+_l = \sigma_l + \eta(\tilde{\sigma}_l - \sigma_l)\,, \quad \forall l \in \mathcal{E}^{\text{out}}_{i^k}$$
$$\lambda^+_i = \lambda_{i_k} + \eta(\tilde{\lambda}_{i_k} - \lambda_{i_k})$$

Writing: in the public memories of each $j \in \mathcal{N}_{i_k}$

$$(x_{i_k}, \lambda_{i_k}) \leftarrow (x^+_{i_k}, \lambda^+_{i_k})$$
$$\{\sigma_l\}_{l\in\mathcal{E}^{\text{out}}_{i_k}} \leftarrow \{\sigma^+_l\}_{l\in\mathcal{E}^{\text{out}}_{i_k}}$$
$$k \leftarrow k+1$$

6. Asynchronous, distributed algorithm with node variables (AD-GENO)

This section presents the main result of the paper, namely, we use AD-GEED as a backbone to design an algorithm converging in the same number of iteration, but relying on node auxiliary variables only, and therefore intrinsically lighter from a computational point of view. We name it *Asynchronous Distributed GNE Seeking Algorithm with Node variables* (AD-GENO). It is based on an idea akin to the one used to develop SD-GENO. In fact, the local update of λ_i in AD-GEED requires only the aggregate quantity $([V^\top]_i \otimes I_m)\boldsymbol{\sigma}$. We introduce a variable z_i to capture the variation of this aggregate quantity and show that it does not affect the dynamics of the pair (x_i, λ_i), thus preserving the convergence proved in Theorem 3. Unlike the synchronous case, we cannot directly define $\boldsymbol{z} = \boldsymbol{V}^\top\boldsymbol{\sigma}$, due to the different update frequencies of $\{\sigma_l\}_{l\in\mathcal{E}_i}$ and z_i that would affect the dynamics of $\boldsymbol{\lambda}$. This mismatch is clarified via the following example.

Example 2. Consider the communication network in Example 1 and assume that in the first three time instances, agent 2 updates twice and then 3 updates once, i.e., $i_0 = i_1 = 2$ and $i_2 = 3$. For $k=1$, according to Algorithm 2 it holds

$$\begin{aligned}\sigma_2(2) &= \sigma_2(1) + \eta\rho\delta(\lambda_2(1) - \lambda_3(0))\\ \lambda_2(2) &\propto \rho(\sigma_2(1) - \sigma_1(0)),\end{aligned} \tag{23}$$

where $\propto$ is used to describe dependency. Next, for $k=2$ only λ_3 is updated, then

$$\lambda_3(3) \propto -\rho\sigma_2(2). \tag{24}$$

If we substitute the edge variables σ_1, σ_2 with $z_i = [V^\top]_i\boldsymbol{\sigma}$ for $i = 1,2,3$, and apply the same activation sequence, it leads to

$$\begin{aligned}z_3(3) &= z_3(0) + \eta\rho\delta(\lambda_3(0) - \lambda_2(2))\\ \lambda_3(3) &\propto \rho z_3(0).\end{aligned} \tag{25}$$

From the comparison of (24) and (25), it is clear that the value of $\lambda_3(3)$ would be different in the two cases. This is explained by the fact that σ_2 is updated twice, while z_3 only once.

To bridge the gap between $\boldsymbol{\sigma}$ and $\boldsymbol{z}$, we introduce an extra variable $\mu_i \in \mathbb{R}^m$ for each node i. The role of μ_i is to store the changes of the neighbors dual variable λ_j, during the time between the last update of i and the next one.

In Algorithm 3 we present the local update rules of AD-GENO.

Algorithm 3: AD-GENO.

Input: $k=0$, $\boldsymbol{x}(0) \in \mathbb{R}^n$, $\boldsymbol{\lambda}(0) \in \mathbb{R}^{mN}$, $\boldsymbol{z}(0) = \mathbf{0}_{mN}$. For every $i \in \mathcal{N}$, choose δ, ε_i, τ_i satisfying (19), $\eta \in (0,1)$ and set $\mu_i = \mathbf{0}_m$.

Iteration k**:** Select the agent i_k with probability $\mathbb{P}[\zeta(k) = H_{i_k}] = p_{i_k}$

Reading: Agent i_k copies its public memory in the private one, i.e., the values $\hat{x}_j$, $\hat{\lambda}_j$, $\forall j \in \mathcal{N}_{i_k}$, and μ_{i_k}.

Reset the public values of μ_{i_k} to $\mathbf{0}_m$.

Update:

$$\tilde{x}_{i_k} = \text{proj}_{\Omega_{i_k}}\left(x_{i_k} - \tau_{i_k}(\nabla_{i_k} f_{i_k}(x_{i_k}, \hat{\boldsymbol{x}}_{-i_k}) + A_{i_k}^\top \lambda_{i_k})\right)$$
$$\tilde{z}_{i_k} = z_{i_k} + \delta\eta\mu_{i_k}$$
$$\tilde{\lambda}_{i_k} = \text{proj}_{\mathbb{R}^m_{\geq 0}}\Big(\lambda_{i_k} + \varepsilon_{i_k}\big(A_{i_k}(2\tilde{x}_{i_k} - x_{i_k}) - b_{i_k} - \rho\tilde{z}_{i_k} + (2\delta\rho^2 - 1)\textstyle\sum_{j\in\mathcal{N}_{i_k}\setminus\{i_k\}}(\lambda_{i_k} - \hat{\lambda}_j)\big)\Big)$$

$$x^+_{i_k} = x_{i_k} + \eta(\tilde{x}_{i_k} - x_{i_k})$$
$$z^+_{i_k} = \tilde{z}_{i_k} + \eta\delta\rho\textstyle\sum_{l\in\mathcal{E}^{\text{out}}_{i_k}}([V]_l \otimes I_m)\hat{\boldsymbol{\lambda}}$$
$$\lambda^+_{i_k} = \lambda_{i_k} + \eta(\tilde{\lambda}_{i_k} - \lambda_{i_k})$$

Writing: in the public memory of each $j \in \mathcal{N}_{i_k}$

$$(x_{i_k}, \lambda_{i_k}) \leftarrow (x^+_{i_k}, \lambda^+_{i_k})$$
$$\mu_j \leftarrow \mu_j + \hat{\lambda}_j - \lambda_{i_k}$$
$$k \leftarrow k+1$$

The convergence of AD-GENO is proven by the following theorem. Essentially, we show that introducing $\boldsymbol{z}$ and μ does not affect the dynamics of $(\boldsymbol{x}, \boldsymbol{\lambda})$.

Theorem 4. *For every $i \in \mathcal{N}$ choose ε_i, δ, τ_i as in (19). Let $\eta \in \left(0, \frac{cNp_{\min}}{2\bar{\varphi}\sqrt{p_{\min}+1}}\left(2 - \frac{1}{2\chi\vartheta}\right)\right]$ with $p_{\min} := \min\{p_i\}_{i\in\mathcal{N}}$ and $c \in (0,1)$. Then, the sequence $(\boldsymbol{x}(k))_{k\in\mathbb{N}}$ generated by AD-GENO (Algorithm 3) converges to the vGNE of the game in (2) almost surely.*

Remark 3. Only one extra scalar variable μ_i is used for every agent $i \in \mathcal{N}$, and hence the benefits of adopting only node variables, discussed in Remark 1, hold also in this asynchronous counterpart. Furthermore, the number of required communication rounds between agents does not increase, since the variable μ_i is updated by the neighbors of agent i during their writing phase.

7. Simulations

We conclude by proposing two sets of simulations to validate the theoretical results in the previous sections. First, we apply AD-GENO on a network Cournot game and study how delays and different activation sequences affect the convergence. Then, we compare the total computation time required by AD-GENO, AD-GEED

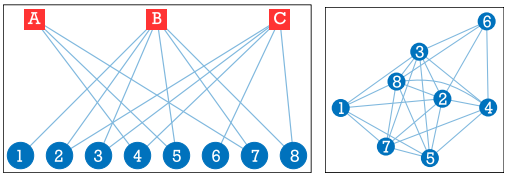


Fig. 1. (a) Action of players $\{1,\dots,8\}$ over the three markets A, B, C, D, (b) Communication network arising from the competition over the markets.

and ADAGNES (in [31, Algorithm 1]), over different communication graphs.

7.1. AD-GENO convergence

In a network Cournot game, N firms compete over m markets and the coupling constraints arise from the maximum markets capacities. We consider a smilar formulation to that proposed in [30]. Here, we considered $N = 8$ firms, with the possibility to act over $m = 3$ markets, i.e., $x_i \in \mathbb{R}^3$, for all $i \in \mathcal{N}$. The local production is bounded in $0 \le x_i \le \bar{x}_i$, where each component of $\bar{x}_i \in \mathbb{R}^3$ is randomly drawn from [10,45]. In Fig. 1a, the interaction of each firm with the markets is shown, where an edge is drawn between a firm and a market if one of former's strategies is applied to the latter. Two firms are neighbors if they compete over the same market, therefore the communication network between the firms is the one in Fig. 1b. The coupling constraints are defined by $\boldsymbol{A}\boldsymbol{x} \le b$, where $\boldsymbol{A} := [A_1, \dots, A_N] \in \mathbb{R}^{3\times 24}$ while $b \in \mathbb{R}^3$. The element $[A_i]_{jk}$ is nonzero, if $[x_i]_k > 0$ and it is applied to market j. Each nonzero element in $\boldsymbol{A}$ is randomly chosen from [0.6,1], this value can be seen as the efficiency of a strategy on a market. The components of $b \in \mathbb{R}^3$ are the capacities of the markets, randomly drawn from [20,100]. The local cost function is defined as $f_i(x_i, \boldsymbol{x}_{-i}) := c_i(\boldsymbol{x}) - P(\boldsymbol{A}\boldsymbol{x})^\top A_i x_i$; $c_i(\boldsymbol{x})$ and it describes the cost of opting for a certain strategy, while $P(\boldsymbol{A}\boldsymbol{x})$ is the reward attained. The price is assumed linear in its argument $P(z) = \bar{P} - Dz$, where $\bar{P} \in \mathbb{R}^3$ and $D \in \mathbb{R}^{3\times 3}$ is a diagonal matrix, their non zero components are randomly chosen from [250,500] and [1,5] respectively. The function $c_i(\boldsymbol{x}) = x_i^\top Q_i x_i + q_i^\top x_i$ is quadratic, where $Q_i \in \mathbb{R}^{4\times 4}$ is diagonal and $q_i \in \mathbb{R}^4$. Their values are randomly chosen from [1,8] and [1,4], respectively.

In order to explore different setups we simulate three different cases:

(A) The communication is delay free, i.e., $\bar{\varphi} = 0$, and the activation sequence is alphabetic, and hence $P[\zeta(k) = \boldsymbol{H}_i] = \frac{1}{N}$, for every $i \in \mathcal{N}$.
(B) The activation sequence is still alphabetic, but the communication may be delayed of 3 time instants at most, i.e., $\bar{\varphi} = 3$.
(C) The communication has no delay, but the probability of update is different between agents, half of them have $p_i = \frac{1}{6}$, while the rest $p_i = \frac{1}{12}$.

The outcome of these scenarios are presented in Fig. 2. The main difference from case (A) can be noticed if there is a non-uniform update probability, i.e., case (C). In fact, we notice that a skewer probability implies slower converge. From simulations, we noticed that the convergence of the dual variables is often the bottleneck to high convergence performances. In all our algorithms, we mitigated this effect by an appropriate tuning of ρ.

7.2. Comparison between algorithms

Next, we compare the performance of AD-GENO with respect to AD-GEED and ADAGNES, from a computational time point of view. For the comparison with ADAGNES, we consider a modified version of the Nash–Cournot game presented in Section 7.1 with coupling equality constraints, i.e., only $\boldsymbol{A}\boldsymbol{x} = b$. Here, we consider $N = 40$ firms, each with at most $n_i = 2$ products. To provide an extensive comparison, we considered many instances of this game varying the communication between agents, from a complete to a sparse graph. The other quantities in the games are chosen as in the previous section. We compared the algorithms over 160 different graphs. The computational time required to obtain convergence is compared in the three cases.[1]

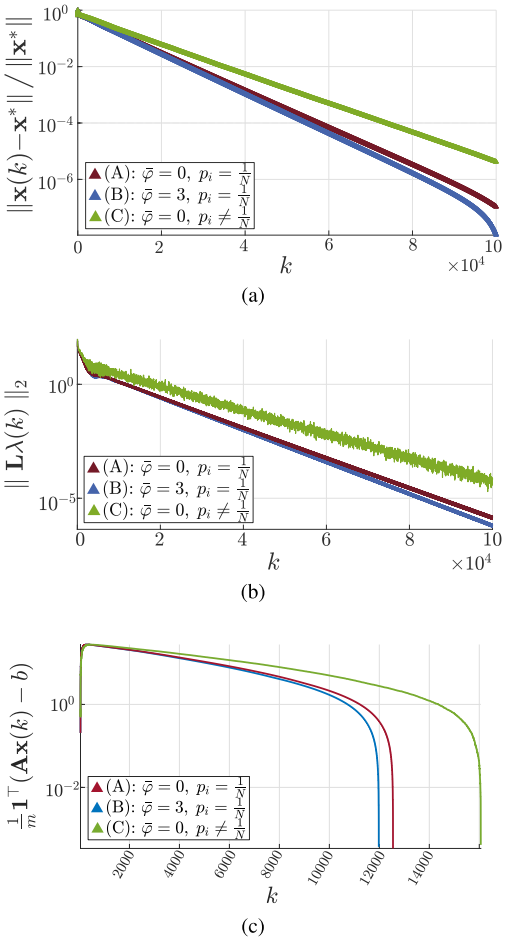


Fig. 2. (a) Normalized distance from the vGNE, (b) Norm of the disagreement between the dual variables, (c) Constraints violation.

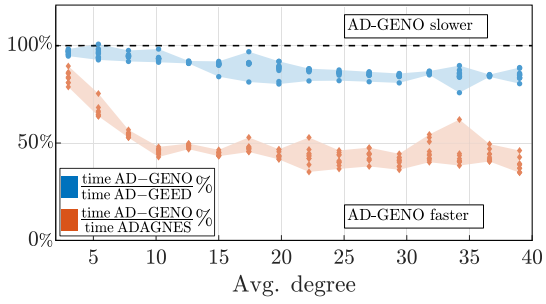


Fig. 3. Comparison of the computation time of ADAGNES vs AD-GENO (orange diamond) and AD-GEED vs AD-GENO (blue dots), w.r.t. the variation of the communication network connectivity.

[1] The computation is performed on a single computer, thus the considered time is due to the local updates only and not the communications between the agents.

The results of the simulations are presented in Fig. 3. As expected AD-GENO always outperforms AD-GEED, since it achieves the same dynamics of $(\boldsymbol{x}, \boldsymbol{\lambda})$ with fewer auxiliary variables. As expected, the gap between the two algorithms shrinks for a sparse graph while it increases for a dense one, from $\sim$3% to $\sim$20%. A similar behavior arises when AD-GENO is compared to ADAGNES, due to the increment of auxiliary variables for highly connected graphs. In particular, the advantage in using AD-GENO starts from $\sim$20% when the graph has an average degree of 3 and becomes $\sim$60% when the graph is complete.

8. Conclusion

The AD-GENO algorithm developed in this work solves GNE seeking problems in strongly monotone games via an asynchronous update scheme. It adopts node variables only, and ensures resilience to delayed information. In our numerical experience, AD-GENO outperforms the available solutions in the literature, both in terms of computational time and number of variables required.

Unfortunately, the "ARock" framework does not ensure robustness to lossy communication. This is currently an open problem that is left to future research. Another interesting topic is the generalization of the algorithm to the case of time-varying communication networks, as the independence from the edge variables makes the structure of AD-GENO more suitable to address this problem.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

The work of Cenedese, and Cao was supported in part by the European Research Council (ERC-CoG-771687) and the Netherlands Organization for Scientific Research (NWO-vidi-14134). The work of Grammatico was partially supported by NWO under research project OMEGA (613.001.702) and P2P-TALES (647.003.003) and by the European Research Council under research project COSMOS (802348).

Appendix A. Proofs of Section 4

A1. Proof of Lemma 1

The proof follows similar steps to that of [31, Lem. 5], hence we omit it.

A2. Proof of Theorem 1

Consider the equilibrium point col($\boldsymbol{x}^*$, $\boldsymbol{\lambda}^*$, $\boldsymbol{z}^*$) with $\boldsymbol{z}^* = \boldsymbol{V}^\top \boldsymbol{\sigma}^*$. Then, (16) at the equilibrium reduces to $\mathbf{0} = -\rho \boldsymbol{L} \boldsymbol{\lambda}^*$, and thus $\boldsymbol{\lambda}^* = \lambda^* \otimes \mathbf{1}$. Moreover, manipulating (17) and evaluating it in the equilibrium lead to

$$\mathbf{0} \in N_{\mathbb{R}^{mN}_{\geq 0}}(\boldsymbol{\lambda}^*) + \bar{b} - \Lambda \boldsymbol{x}^* + \rho \boldsymbol{z}^* + (2\rho\delta + 1)\boldsymbol{L}\boldsymbol{\lambda}^* . \tag{A.1}$$

Recalling that $\boldsymbol{L}\boldsymbol{\lambda}^* = \mathbf{0}$ and multiplying both sides of (A.1) by $(\mathbf{1}^\top \otimes I_m)$ leads to

$$\mathbf{0} \in (\mathbf{1}^\top \otimes I_m)(N_{\mathbb{R}^{mN}_{\geq 0}}(\boldsymbol{\lambda}^*) + \bar{b} - \Lambda \boldsymbol{x}^* + \rho \boldsymbol{z}^*) . \tag{A.2}$$

Using the fact that $\sum_{i\in\mathcal{N}} N_{\mathbb{R}^m_{\geq 0}}(\lambda^*) = N_{\cap_{i\in\mathcal{N}} \mathbb{R}^m_{\geq 0}}(\lambda^*) = N_{\mathbb{R}^m_{\geq 0}}(\lambda^*)$ and the assumption $\mathbf{1}^\top \boldsymbol{z}^* = 0$, (A.2) becomes

$$\mathbf{0} \in N_{\mathbb{R}^m_{\geq 0}}(\lambda^*) + \bar{b} - A\boldsymbol{x}^* . \tag{A.3}$$

Finally, (15) evaluated in the equilibrium is

$$\mathbf{0} \in F(\boldsymbol{x}^*) + N_{\boldsymbol{\Omega}}(\boldsymbol{x}^*) + \Lambda^\top \boldsymbol{\lambda}^*, \tag{A.4}$$

or equivalently

$$\mathbf{0} \in \nabla f_i(x_i^*, \boldsymbol{x}_{-i}^*) + N_{\Omega_i}(x_i^*) + A_i^\top \lambda^* , \quad \forall i \in \mathcal{N} . \tag{A.5}$$

Inclusions (A.5) and (A.3) are the KKT conditions in (6), and hence from [17, Th. 3.1] we conclude that col($\boldsymbol{x}^*$, $\boldsymbol{\lambda}^*$, $\boldsymbol{z}^*$) is a vGNE of the game.

A3. Proof of Theorem 2

From [18, Th. 2], the choice of ϑ, ε_i, δ and τ_i in (19) implies that $\Phi - \vartheta I \succ 0$. Then, from [31, Lem. 2], $\Phi^{-1}\mathcal{B}$ and $\Phi^{-1}\mathcal{A}$ are respectively $\chi\vartheta$-cocoercive and maximally monotone in the Φ induced norm. Furthermore, it also shows that $(\mathrm{Id} - \Phi^{-1}\mathcal{B})$ is $\frac{1}{2\chi\vartheta}$-AVG and $\mathrm{J}_{\Phi^{-1}\mathcal{A}} := (\mathrm{Id} + \Phi^{-1}\mathcal{A})$ is FNE. Applying [1, Prop. 4.44], we conclude that T is $\frac{2\chi\vartheta}{4\chi\vartheta-1}$-AVG. So, the iteration in (11) converges to $\varpi^* \in \mathrm{fix}(T)$ if $\eta \in \left(0, \frac{4\chi\vartheta-1}{2\chi\vartheta}\right)$, [1, Th. 5.14]. The above argument establishes that $\lim_{k\to+\infty} \boldsymbol{\sigma}^k = \boldsymbol{\sigma}^*$, and hence $\lim_{k\to+\infty} \boldsymbol{V}^\top \boldsymbol{\sigma}^k = \boldsymbol{V}^\top \boldsymbol{\sigma}^* =: \boldsymbol{z}^*$. So, $\boldsymbol{z}$ converges and consequently we conclude that Algorithm 1 converges to col($\boldsymbol{x}^*$, $\boldsymbol{\lambda}^*$, $\boldsymbol{z}^*$). The choice of $\boldsymbol{z}_0 = \mathbf{0}$, implies that $\mathbf{1}^\top \boldsymbol{z}^k = 0$, for all $k \in \mathbb{N}$ since its values will be in the range of L. Finally, applying Theorem 1 we prove that the equilibrium is the vGNE of the original game.

Appendix B. Proof of Theorem 3

Since T is $\frac{2\chi\vartheta}{4\chi\vartheta-1}$-AVG, then it can be rewritten as $T = (1 - \frac{2\chi\vartheta}{4\chi\vartheta-1})\mathrm{Id} + \frac{2\chi\vartheta}{4\chi\vartheta-1}P$, where P is nonexpansive, [1, Prop. 4.35]. The proof can be completed following similar steps to the ones in [31, Th. 2].

Appendix C. Proof of Theorem 4

If we show that the modified update of $\boldsymbol{\lambda}$, with $\boldsymbol{z}$ instead of $\boldsymbol{\sigma}$, is equivalent to the one in Algorithm 2, we can infer the convergence from Theorem 3.

We prove by induction that, given an agent i, the update of λ_i at time k in Algorithms 2 and 3 are equivalent. Note that the two update rules are equivalent if it holds that

$$\tilde{z}_i(k) = ([V^\top]_i \otimes I_m)\hat{\boldsymbol{\sigma}}(k), \tag{C.1}$$

for every $k > 0$ and $i \in \mathcal{N}$.

Base case: Iteration k is the first in which agent i updates its variables. If $k = 0$, then in AD-GENO $\mu_i = \mathbf{0}_m$ for every i and $\hat{\boldsymbol{\sigma}}(0) = \mathbf{0}_{mM}$ in AD-GEED, hence (C.1) is trivially verified.

If instead $k > 0$, it holds that $z_i(k) = z_i(0) = \mathbf{0}_m$, while $\hat{\boldsymbol{\sigma}}(k) \neq \mathbf{0}_{mM}$, since the neighbors of i can update more than once before the first update of i (as shown in Example 2). We define for each $j \in \mathcal{N}_i$ the set $\mathcal{S}_j(k)$, a $t \in \mathbb{N}$ where $t < k$ belongs to $\mathcal{S}_j(k)$ if, at the iteration t, the agent j completes an update. The maximum time in $\mathcal{S}_j(k)$ is denoted as $m_j(k) := \max\{\mathcal{S}_j(k)\}$ and $\check{\mathcal{S}}_j(k) := \mathcal{S}_j(k) \setminus m_j(k)$. From this definitions, we obtain that

$$([V^\top]_i \otimes I_m)\hat{\boldsymbol{\sigma}}(k) = \sum_{l\in\mathcal{E}_i^{\mathrm{out}}} \sigma_l(0) - \sum_{d\in\mathcal{E}_i^{\mathrm{in}}} \sigma_d(m_j(k)), \tag{C.2}$$

where j is the element of e_d different from i. Furthermore, from the update rule of σ_d in Algorithm 2, we derive

$$\sigma_d(m_j(k)) = \sigma_d\big(\max\{\check{S}_j(k)\}\big) + \eta\delta\rho\Big(\lambda_j\big(\max\{\check{S}_j(k)\}\big) - \lambda_i(0)\Big) = \eta\delta\rho \sum_{h\in\check{S}_j(k)} \big(\lambda_j(h) - \lambda_i(0)\big) \tag{C.3}$$

Substituting (C.3) into (C.2) leads to

$$([V^\top]_i \otimes I_m)\hat{\boldsymbol{\sigma}}(k) = \rho\eta\delta\left(\sum_{j\in\mathcal{N}_i\setminus\{i\}} \big|\check{S}_j(k)\big|\lambda_i(0) - \sum_{j\in\mathcal{N}_i\setminus\{i\}}\sum_{h\in\check{S}_j(k)} \lambda_j(h)\right). \tag{C.4}$$

From the definition given in Algorithm 3 of μ_i, we attain that $([V^\top]_i \otimes I_m)\hat{\boldsymbol{\sigma}}(k) = \eta\delta\mu_i = \tilde{z}_i(k)$, therefore (C.1) hold.

Induction step: Suppose that (C.1) holds for some $\bar{k} > 0$ that corresponds to the latest iteration in which agent i performed the update, i.e. $z_i(\bar{k}) \neq \mathbf{0}$.

Consider the next iteration k in which agent i updates, $k > \bar{k}$. Here, $\mathcal{S}_j(k)$ is defined as above, but for time indexes $(\bar{k}, k]$. Following similar reasoning in the previous case, we obtain

$$\begin{aligned}([V^\top]_i \otimes I_m)\hat{\boldsymbol{\sigma}}(k) = {} & ([V^\top]_i \otimes I_m)\hat{\boldsymbol{\sigma}}(\bar{k}) \\ & + \eta\delta\rho \sum_{l\in\mathcal{E}_i^{\text{out}}} ([V]_l \otimes I_m)\hat{\boldsymbol{\lambda}}(\bar{k}) \\ & + \eta\delta\rho\left(\sum_{j\in\mathcal{N}_i\setminus\{i\}} \big|\check{S}_j(k)\big|\lambda_i(0) \right. \\ & \left. - \sum_{j\in\mathcal{N}_i\setminus\{i\}}\sum_{h\in\check{S}_j(k)} \lambda_j(h)\right)\end{aligned} \tag{C.5}$$

where we used the fact that $l \in \mathcal{E}_i^{\text{out}}$ is updated at the same time of i. Furthermore, from the induction assumption,

$$\begin{aligned}([V^\top]_i \otimes I_m)\hat{\boldsymbol{\sigma}}(k) &= z_i(\bar{k}) + \eta\delta\rho\left(\sum_{j\in\mathcal{N}_i\setminus\{i\}} \big|\check{S}_j(k)\big|\lambda_i(0) \right. \\ &\quad \left. - \sum_{j\in\mathcal{N}_i\setminus\{i\}}\sum_{h\in\check{S}_j(k)} \lambda_j(h)\right) \\ &= z_i(\bar{k}) + \eta\delta\rho\mu_i = \tilde{z}_i(k),\end{aligned} \tag{C.6}$$

where the last step holds because in the reading phase of Algorithm 3, we reset to zero the values of μ_i, every time that i starts an update. Therefore, (C.1) holds for k.

Finally, the convergence of $(\boldsymbol{x}(k))_{k\in\mathbb{N}}$ to the vGNE of the game (2) follows from Theorem 3.

References

[1] H.H. Bauschke, P.L. Combettes, et al., Convex Analysis and Monotone Operator Theory in Hilbert Spaces, 408, Springer, 2011.

[2] G. Belgioioso, S. Grammatico, On convexity and monotonicity in generalized aggregative games, IFAC-PapersOnLine 50 (1) (2017) 14338–14343. 20th IFAC World Congress

[3] G. Belgioioso, S. Grammatico, Semi-decentralized Nash equilibrium seeking in aggregative games with coupling constraints and non-differentiable cost functions, IEEE Control Syst. Lett. 1 (2) (2017) 400–405.

[4] D.P. Bertsekas, J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, 23, Prentice hall Englewood, Cliffs, NJ, 1989.

[5] D.P. Bertsekas, J.N. Tsitsiklis, Some aspects of parallel and distributed iterative algorithms survey, Automatica 27 (1) (1991) 3–21.

[6] C. Cenedese, Y. Kawano, S. Grammatico, M. Cao, Towards time-varying proximal dynamics in multi-agent network games, in: 2018 IEEE Conference on Decision and Control (CDC), 2018, pp. 4378–4383.

[7] C. Cenedese, G. Belgioioso, S. Grammatico, M. Cao, An asynchronous, forward-backward, distributed generalized Nash equilibrium seeking algorithm, in: 2019 18th European Control Conference (ECC), 2019, pp. 3508–3513.

[8] C. Cenedese, F. Fabiani, M. Cucuzzella, J.M.A. Scherpen, M. Cao, S. Grammatico, Charging plug-in electric vehicles as a mixed-integer aggregative game, in: 2019 IEEE 58th Conference on Decision and Control (CDC), 2019, pp. 4904–4909.

[9] C. Cenedese, G. Belgioioso, S. Grammatico, M. Cao, Time-varying constrained proximal type dynamics in multi-agent network games (2020).

[10] C. Cenedese, G. Belgioioso, Y. Kawano, S. Grammatico, M. Cao, Asynchronous and time-varying proximal type dynamics multi-agent network games, IEEE Trans. Autom. Control (2020). (in press)

[11] P. Combettes, J. Pesquet, Stochastic quasi-fejér block-coordinate fixed point iterations with random sweeping, SIAM J. Optim. 25 (2) (2015) 1221–1248.

[12] F. Dörfler, J. Simpson-Porco, F. Bullo, Breaking the hierarchy: Distributed control and economic optimality in microgrids, IEEE Trans. Control Netw. Syst. 3 (3) (2016) 241–253.

[13] A. Dreves, F. Facchinei, C. Kanzow, S. Sagratella, On the solution of the KKT conditions of generalized Nash equilibrium problems, SIAM J. Optim. 21 (3) (2011) 1082–1108.

[14] J. Eckstein, Splitting Methods for Monotone Operators With Applications to Parallel Optimization, Massachusetts Institute of Technology, 1989 Ph.D. thesis.

[15] F. Facchinei, J. Pang, Finite-dimensional Variational Inequalities and Complementarity Problems, Springer Verlag, 2003.

[16] F. Facchinei, C. Kanzow, Generalized Nash equilibrium problems, 4or 5 (3) (2007) 173–210.

[17] F. Facchinei, A. Fischer, V. Piccialli, On generalized Nash games and variational inequalities, Oper. Res. Lett. 35 (2007) 159–164.

[18] D.G. Feingold, R.S. Varga, et al., Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem., Pac. J. Math. 12 (4) (1962) 1241–1250.

[19] A. Govaert, C. Cenedese, S. Grammatico, M. Cao, Relative best response dynamics in finite and convex network games, in: 2019 IEEE 58th Conference on Decision and Control (CDC), 2019, pp. 3134–3139.

[20] S. Grammatico, Proximal dynamics in multi-agent network games, IEEE Trans. Control Netw. Syst. (2018).

[21] A.A. Kulkarni, U. Shanbhag, On the variational equilibrium as a refinement of the generalized Nash equilibrium, Automatica 48 (2012a) 45–55.

[22] A.A. Kulkarni, U.V. Shanbhag, Revisiting generalized Nash games and variational inequalities, J. Optim. Theory Appl. 154 (1) (2012b) 175–186.

[23] A. Nedic, Asynchronous broadcast-based convex optimization over a network, IEEE Trans. Autom. Control 56 (6) (2011) 1337–1351.

[24] D. Paccagnan, B. Gentile, F. Parise, M. Kamgarpour, J. Lygeros, Distributed computation of generalized Nash equilibria in quadratic aggregative games with affine coupling constraints, in: 2016 IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 6123–6128.

[25] D. Paccagnan, B. Gentile, F. Parise, M. Kamgarpour, J. Lygeros, Nash and Wardrop equilibria in aggregative games with coupling constraints, IEEE Trans. Autom. Control 64 (4) (2019) 1373–1388.

[26] Z. Peng, Y. Xu, M. Yan, W. Yin, Arock: an algorithmic framework for asynchronous parallel coordinate updates, SIAM J. Sci. Comput. 38 (5) (2016) A2851–A2879.

[27] B. Recht, C. Re, S. Wright, F. Niu, Hogwild: a lock-free approach to parallelizing stochastic gradient descent, in: Advances in Neural Information Processing Systems, 2011, pp. 693–701.

[28] J. Rosen, Existence and uniqueness of equilibrium points for concave n-person games, Econometrica 33 (1965) 520–534.

[29] G. Scutari, D.P. Palomar, F. Facchinei, J. Pang, Convex optimization, game theory, and variational inequality theory, IEEE Signal Processing Magazine 27 (3) (2010) 35–49.

[30] P. Yi, L. Pavel, An operator splitting approach for distributed generalized Nash equilibria computation, Automatica 102 (2019) 111–121.

[31] P. Yi, L. Pavel, Asynchronous distributed algorithms for seeking generalized Nash equilibria under full and partial-decision information, IEEE Trans. Cybern. (2019) 1–13.