Distributed Algorithms for Aggregative Games on Graphs

Angelia Nedić
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign

Joint work with Jayash Koshal and Uday V. Shanbhag

Friday 1st April, 2016
Outline

1. Aggregative Nash Games
   - Aggregative Game
   - Aggregative Game on Network

2. Distributed synchronous algorithm

3. Distributed asynchronous algorithm

4. Numerical Results
An Aggregative Game

We have a system with \( N \) agents, where each agent \( i \) solves the following problem:

\[
\text{minimize} \quad f_i(x_i, s(x)), \quad s(x) = \sum_{j=1}^{N} x_j
\]

subject to \( x_i \in K_i \subseteq \mathbb{R}^n \).  

\( f_i(\cdot, \cdot) \) and \( K_i \) are known only to agent \( i \)

Agent \( i \) decides on \( x_i \), and does not have access to the decisions \( x_j \) of the other agents

Unlike [Jensen06, MartimortStole2010], the emphasis here is on computation of an equilibrium in the absence of instantaneous access to the aggregate value \( \bar{x} \)

This can be achieved when the agents are able to communicate locally over a connected network
Motivating Example

Example (Networked Nash-Cournot Oligopoly)

- $N$ firms competing at $M$ locations
- Optimization problem for firm $i$

\[
\begin{align*}
\text{minimize} & \quad \sum_{\ell=1}^{M} \left( c_{i\ell}(x_{i\ell}) - p_{\ell}(\bar{s}_{\ell})s_{i\ell} \right) + t_{i}(x_{i1}, \ldots, x_{iM}) \\
\text{subject to} & \quad \sum_{\ell=1}^{M} x_{i\ell} = \sum_{\ell=1}^{M} s_{i\ell}, \\
& \quad x_{i\ell}, s_{i\ell} \geq 0, \quad x_{i\ell} \leq \text{cap}_{i\ell}, \quad \ell = 1, \ldots, M.
\end{align*}
\]

- $x_{i\ell}$ and $s_{i\ell}$ are production and sales for firm $i$ at location $\ell$
- $p_{\ell}$ denotes the price function and $\bar{s}_{\ell} = \sum_{j=1}^{N} s_{j\ell}$
- $p_{\ell}(\bar{s}_{\ell})s_{i\ell}$ is the revenue at location $\ell$ for firm $i$
- $t_{i}$ is the transportation cost
Flow control games

- Multitude of applications on the internet
- A large number of users - inherently noncooperative in nature

Demands for bandwidth lead to congestion
Solution concept - a Nash equilibrium in flow rates, where each agent maximizes its utility less cost of sending flow

User $i$ has utility given by $U_i(x_i) := \ln(1 + x_i) + d_i, \ x_i \geq 0$,
Cost of sending flow can be modeled by $P_i(x_i, x_{-i}) := k_i x_i^2 + M p (C - \sum_{i=1}^{N} x_i)$

**Nash Equilibrium in flow decisions** $x^* = (x_1^*, \ldots, x_N^*)$

Users maximize the benefit of sending flow by solving

Player $i$ maximize $U_i(x_i) - P_i(x_i; x_{-i})$
subject to $x_i \geq 0$,

Nash equilibrium in flow generation levels: $(x_1^*, \ldots, x_N^*)$
Computational algorithm

minimize \quad f_i(x_i, s(x)), \quad s(x) = \sum_{j=1}^{N} x_j

subject to \quad x_i \in K_i \subseteq \mathbb{R}^n.

Are there "distributed" algorithms? 1

Under ideal conditions and suitable assumption, each player updates:

\[ x_i^{k+1} = \Pi_{K_i}[x_i^k - \alpha \nabla x_i f_i(x_i^k, s(x^k))] \]

What is the challenge then?

- No agent knows \( s(x^k) = \sum_{i=1}^{N} x_i^k \)
- No central entity exists to provide it.

---

1Projection based [Facchinei-Pang03, Alpcan03] and their regularized variants [Kannan-Shanbhag10]
minimize \( f_i(x_i, s(x)) \), \( s(x) = \sum_{j=1}^{N} x_j \)

subject to \( x_i \in K_i \subseteq \mathbb{R}^n \).

Are there "distributed" algorithms? \(^1\)

Under ideal conditions and suitable assumption, each player updates:

\[
x_i^{k+1} = \Pi_{K_i} [x_i^k - \alpha \nabla x_i f_i(x_i^k, s(x^k))]\]

What is the challenge then?

- No agent knows \( s(x^k) = \sum_{i=1}^{N} x_i^k \)
- No central entity exists to provide it.

\(^1\)Projection based [Facchinei-Pang03, Alpcan03] and their regularized variants [Kannan-Shanbhag10]
Computational algorithm

\[
\text{minimize} \quad f_i(x_i, s(x)), \quad s(x) = \sum_{j=1}^{N} x_j
\]

subject to \( x_i \in K_i \subseteq \mathbb{R}^n \).

Are there "distributed" algorithms? 1

Under ideal conditions and suitable assumption, each player updates:

\[
x_i^{k+1} = \prod_{K_i} [x_i^k - \alpha \nabla_{x_i} f_i(x_i^k, s(x^k))] \]

What is the challenge then?

- No agent knows \( s(x^k) = \sum_{i=1}^{N} x_i^k \)
- No central entity exists to provide it.

---

1Projection based [Facchinei-Pang03, Alpcan03] and their regularized variants [Kannan-Shanbhag10]
minimize $f_i(x_i, s(x))$, \quad s(x) = \sum_{j=1}^{N} x_j$

subject to $x_i \in K_i \subseteq \mathbb{R}^n$.

Are there "distributed" algorithms?  

Under ideal conditions and suitable assumption, each player updates:

$x_i^{k+1} = \Pi_{K_i}[x_i^k - \alpha \nabla_{x_i} f_i(x_i^k, s(x^k))]$

What is the challenge then?

- No agent knows $s(x^k) = \sum_{i=1}^{N} x_i^k$
- No central entity exists to provide it.

---

1Projection based [Facchinei-Pang03, Alpcan03] and their regularized variants [Kannan-Shanbhag10]
Computational algorithm

\[
\text{minimize} \quad f_i(x_i, s(x)), \quad s(x) = \sum_{j=1}^{N} x_j
\]

subject to \( x_i \in K_i \subseteq \mathbb{R}^n \) \( (\text{AggGame}) \)

- Are there "distributed" algorithms? \(^1\)
- Under ideal conditions and suitable assumption, each player updates:

\[
x_i^{k+1} = \Pi_{K_i} [x_i^k - \alpha \nabla x_i f_i(x_i^k, s(x^k))]
\]

- What is the challenge then?
  - No agent knows \( s(x^k) = \sum_{i=1}^{N} x_i^k \)
  - No central entity exists to provide it.

---

\(^1\)Projection based [Facchinei-Pang03, Alpcan03] and their regularized variants [Kannan-Shanbhag10]
The central theme

Goal:
- Develop distributed algorithms that can aid agents to learn the aggregate and lead to an equilibrium point
- And still satisfy the informational constraints

How?
- By allowing agents to communicate their beliefs of the aggregate: \( \sum_{i=1}^{N} x_i \)
- But not their decisions: \( x_i \)
- This is in sharp contrast with "distributed optimization" setting (since here agents are competing)
Basic Assumption 1

For each \( i = 1, \ldots, N \),

- The set \( K_i \subset \mathbb{R}^n \) is compact and convex
- Function \( f_i(x_i, y) \) is continuously differentiable in \((x_i, y)\) over some open set containing the set \( K_i \times \bar{K} \), where \( \bar{K} = \sum_{i=1}^{N} K_i \)
- The function \( x_i \mapsto f_i(x_i, s(x)) \), \( s(x) = \sum_{j=1}^{N} x_j \), is convex over the set \( K_i \).

- These assumptions are sufficient for the existence of an equilibrium.
Additional Assumption: Uniqueness

- Define mapping $\Phi(x)$ by

$$
\Phi(x) \triangleq \left( \begin{array}{c}
F_1(x_1, s(x)) \\
\vdots \\
F_N(x_N, s(x))
\end{array} \right)
$$

$$
F_i(x_i, s(x)) = \nabla_{x_i} f_i(x_i, s(x)), \quad s(x) = \sum_{j=1}^{N} x_j
$$

Basic Assumption 2 [Strict Monotonicity]

The mapping $\Phi(x)$ is strictly monotone over $K_1 \times \cdots \times K_N$, i.e.,

$$(\Phi(x) - \Phi(x'))^T (x - x') > 0, \quad \forall x, x' \in K_1 \times \cdots \times K_N.$$
Consider the aggregative Nash game defined in (AggGame). Suppose *Basic Assumptions* hold. Then, the game admits a unique Nash equilibrium.
Further Assumptions

Basic Assumption 3 [Computational]

The mapping $F_i(x_i, u)$ is uniformly Lipschitz continuous in $u$ over $ar{K} = K_1 + \cdots + K_N$, for every fixed $x_i \in K_i$ i.e., for some $L_{-i} > 0$

$$\|F_i(x_i, u) - F_i(x_i, z)\| \leq L_{-i}\|u - z\|.$$
Outline

1. Aggregative Nash Games
   - Aggregative Game
   - Aggregative Game on Network

2. Distributed synchronous algorithm

3. Distributed asynchronous algorithm

4. Numerical Results
Synchronous setting

- A time varying undirected connectivity graph $G_k = (\mathcal{N}, \mathcal{E}_k)$
- $\mathcal{N} = \{1, \ldots, N\}$ be a set of $N$ agents
- $\mathcal{E}_k$ is the set of edges at time $k$
- $\mathcal{N}_i(k)$: immediate neighbors of agent $i$ at time $k$

Connectivity Assumption

There exists an integer $Q \geq 1$ such that the graph $\bigcup_{l=1}^{Q} G_{l+k}$ is connected for all $k$. 
Limited connectivity of agents
Limited connectivity of agents
Limited connectivity of agents
Variational Inequality Formulation

We want to determine the point \( x^* = (x_1^*, \ldots, x_N^*) \in K \) such that

\[
\langle \Phi(x^*), x - x^* \rangle \geq 0 \quad \text{for all } x \in K,
\]

where

\[
\Phi(x) \triangleq \begin{pmatrix}
F_1(x_1, s(x)) \\
\vdots \\
F_N(x_N, s(x))
\end{pmatrix}
\]

\[
F_i(x_i, s(x)) = \nabla x_i f_i(x_i, s(x)), \quad s(x) = \sum_{j=1}^{N} x_j
\]

We use a suitable analog of

\[
x_i^{k+1} = \Pi_{K_i}[x_i^k - \alpha \nabla x_i f_i(x_i^k, s(x^k))]
\]

where we distribute learning of \( s(x^k) \) in the network
Outline of the synchronous algorithm: the three steps

1. **Learn aggregate:**²
   - Each player maintains and updates an estimate $v^k_i$ of $s(x^k) = \sum_{i=1}^{N} x^k_i$ (tricky since $x^k_j$’s are players’ private variables & are changing in time)
   - Every player communicates its estimate $v^k_i$ to the neighbors

2. **Update decision:**
   - Using aligned estimate agents update their decision ($x^k_i$)

3. **Update estimate of aggregate:**
   - Agents update their estimate to reflect their current decision.

² Inspiration: Consensus based algorithm [?, ?], Distributed optimization problem [?, ?]
**Synchronous algorithm**

1. **Learn aggregate:**

   \[
   \hat{v}_i^k = \sum_{j \in \mathcal{N}_i(k)} w_{ij}(k) v_j^k, \quad v_i^0 = x_i^0,
   \]

   \(w_{ij}(k)\): agent \(i\)'s weight to agent \(j\)'s information at step \(k\)

2. **Update decision:**

   \[
   x_i^{k+1} = \Pi_{K_i}[x_i^k - \alpha_k F_i(x_i^k, \mathcal{N}\hat{v}_i^k)].
   \]

   where agent \(i\) uses its estimate \(\mathcal{N}\hat{v}_i^k\) instead of \(\sum_{j=1}^N x_j^k\)

3. **Update estimate of aggregate:**

   \[
   v_i^{k+1} = \hat{v}_i^k + x_i^{k+1} - x_i^k
   \]

   Step suggested by Ram in [Ram-Nedić-Veeravali 2012]
Weight assumptions

**Synchronous Weight**

For all $i \in \mathcal{N}$ and all $k \geq 0$, the following hold:

(i) $w_{ij}(k) \geq \delta$ for all $j \in \mathcal{N}_i(k)$ and $w_{ij}(k) = 0$ for $j \notin \mathcal{N}_i(k)$;

(ii) $\sum_{j=1}^{\mathcal{N}} w_{ij}(k) = 1$ for all $i$;

(iii) $\sum_{i=1}^{\mathcal{N}} w_{ij}(k) = 1$ for all $j$. 
Aggregative Nash Games
Distributed synchronous algorithm
Distributed asynchronous algorithm
Numerical Results

Stepsizes assumptions

Synchronous Stepsize

The stepsizes $\alpha_k$ is chosen such that the following hold:

(i) The sequence $\{\alpha_k\}$ is monotonically non-increasing i.e.,
$\alpha_{k+1} \leq \alpha_k$ for all $k$;

(ii) $\sum_{k=0}^{\infty} \alpha_k = \infty$;

(iii) $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$. 
Convergence of synchronous algorithm

Proposition (Koshal-AN-Shanbhag 2016)

Let Basic Assumptions hold. Further let the Synchronous Weight and Synchronous Stepsize assumption hold. Then, the sequence \( \{x^k\} \) generated by the Synchronous Algorithm converges to the (unique) solution \( x^* \) of the game.
Proof sketch

1. Gradient projection algorithm technique \( (\|x^k - x^*\|) \)
   - Utilizing the Lipschitz continuity of the map leads to a bound on \( \|x_i^{k+1} - x_i^*\| \) in terms of \( \|Nv_i^k - \bar{x}^*\| \)

2. Estimate the errors for the beliefs \( \|Nv_i^k - \sum_{i=1}^{N} x_i^*\| \)
   - Introduce \( y^k = \sum_{\ell=1}^{\ell} v^{k}_{\ell} / N \), the average estimate under perfect information
   - \( y^k \) also tracks the average of the aggregate i.e., \( y^k = \sum_{\ell=1}^{\ell} x^{k}_{\ell} / N \)
   - Construct bounds for \( \|v_i^k - y^k\| \) and \( \|Ny^k - \sum_{i=1}^{N} x_i^*\| \)
Convergence established for synchronous regime but

**Key Drawback:**
- Synchronization is a challenge in large networks
- Requires coordination in terms of stepsize

**Remedy:** Desynchronization
- No explicit dependency
- Allows independent solution of stepsizes
- Drawback: cannot accommodate *time-varying graphs*
Outline

1. Aggregative Nash Games
   - Aggregative Game
   - Aggregative Game on Network

2. Distributed synchronous algorithm

3. Distributed asynchronous algorithm

4. Numerical Results
Asynchronous setting

Connected Graph

The undirected graph $G = (\mathcal{N}, \mathcal{E})$ is connected.

Figure: A depiction of a gossip communication.
Asynchronous setting

Connected Graph

The undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is connected.

Figure: A depiction of a gossip communication.
Asynchronous setting

Connected Graph

The undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is connected.

Figure: A depiction of a gossip communication.
Outline of the asynchronous algorithm: the four steps

1. Agents have local clocks ticking at rate 1
2. Wake Up:
   - Agent $I^k$ wakes up at time $k$ and then contacts $J^k$
3. Learn aggregate:
   - Only agents $\{I^k, J^k\}$ communicate their estimate ($v^{i_k}$) and perform mixing
4. Update decision:
   - Using aligned estimate agents update their decision ($x^{i_k}$)
5. Update estimate of aggregate:
   - Agents update their estimate to reflect their current decision.
Asynchronous algorithm

- **Wake Up:**
  - Agent $I^k$ wakes up at time $k$ and then contacts $J^k$

1. **Learn aggregate:**

   $$
   \hat{v}_i^k = \frac{v_{I^k}^k + v_{J^k}^k}{2} \quad \text{for } i \in \{I^k, J^k\}, \quad v_i^0 = x_i^0,
   $$

2. **Update decision:**

   $$
   x_i^{k+1} = \left( \prod_{K_i} [x_i^k - \alpha_{k,i} F_i(x_i^k, N\hat{v}_i^k)] - x_i^k \right) \mathbb{1}_{\{i \in \{I^k, J^k\}\}} + x_i^k
   $$

3. **Update estimate of aggregate:**

   $$
   v_i^{k+1} = \hat{v}_i^k + x_i^{k+1} - x_i^k.
   $$
Asynchronous Algorithm

Stepsize Assumptions

Asynchronous Stepsize

The stepsize $\alpha_{k,i}$ is updated as follows:

$$\alpha_{k,i} = \frac{1}{\Gamma_k(i)},$$

where

- $\Gamma_k(i)$ denotes the number of updates that agent $i$ has executed up to time $k$ inclusively.
Proposition (Koshal-AN-Shanbhag 2016)

Let Basic Assumption, Connected Graph and the Asynchronous Stepsize assumption hold. Then, the sequence \( \{x^k\} \) generated by the Asynchronous Algorithm converges to the (unique) solution \( x^* \) of the game with probability 1.
Key property: almost surely we have for all \( k \) large enough

\[
\alpha_{k,i} \leq \frac{2}{kp_i}, \quad \left| \alpha_{k,i} - \frac{1}{kp_i} \right| \leq \frac{2}{k^{3/2-q}p_{\min}^2}.
\]

where \( p_i \) is the probability that agent \( i \) updates, \( p_{\min} = \min_i p_i \) and \( q \in (0, 1/2) \).

1. The stepsize

\[
\alpha_{k,i} = \frac{1}{\Gamma_k(i)}
\]

behaves almost surely as \( 1/k \) for all agents \( i \) and for all \( k \) large enough

2. Stochastic gradient technique \((\mathbb{E} [\|x^k - x^*\|^2 | \mathcal{F}_{k-1}])\)

3. Estimate the expected errors for the beliefs \((\mathbb{E} [\|Nv_i^k - \bar{x}^*\| | \mathcal{F}_{k-1}])\)
Constant stepsize asynchronous algorithm

Agents employ a constant deterministic yet uncoordinated stepsize

- **Wake Up:**
  - Agent $I^k$ wakes up at time $k$ and then contacts $J^k$

1. **Learn aggregate:**

   \[
   \hat{v}_i^k = \frac{v_{I_k}^k + v_{J_k}^k}{2} \quad \text{for } i \in \{I_k, J_k\}, \quad v_i^0 = x_i^0,
   \]

2. **Update decision:**

   \[
   x_i^{k+1} = \left(\prod_{K_i} [x_i^k - \alpha F_i(x_i^k, N\hat{v}_i^k)] - x_i^k\right) 1_{\{i \in \{I_k, J_k\}\}} + x_i^k
   \]

3. **Update estimate of aggregate:**

   \[
   v_i^{k+1} = \hat{v}_i^k + x_i^{k+1} - x_i^k.
   \]
Basic Assumption 4

The mapping $F_i(x_i, u)$ is uniformly Lipschitz continuous in $x_i$ over $K_i$, for every fixed $u \in \bar{K}$ i.e., for some $L_i > 0$

$$\|F_i(x_i, u) - F_i(y_i, u)\| \leq L_i \|x_i - y_i\|;$$
Error bound: constant stepsize asynchronous algorithm

Proposition

Let Basic Assumptions hold with the mapping $\Phi$ be strongly monotone over the set $K$. Further let Connected Graph assumption hold. Then, the following holds for the sequence $\{x^k\}$ generated by the Asynchronous Algorithm with stepsize $\alpha_{k,i} = \alpha_i$

$$\limsup_{k \to \infty} \mathbb{E}[\|x^{k+1} - x^*\|^2] \leq \text{Err}(N, n, \mathcal{G}),$$

where $x^*$ is the unique equilibrium of the game.
Expression for error

\[
Err(N, n, G) = \frac{4p_{\max}\alpha_{\max}^2 C^2 N + 2p_{\max}\alpha_{\max}^2 B \sqrt{2nN} C}{2\mu p_{\min}\alpha_{\min} - 2p_{\max}(\max_i L_i)(\alpha_{\max} - \alpha_{\min})}
\]

- where \( \mu \) is the strong monotonicity constant of the gradient map \( \Phi \) and \( C \) is a bound on \( \|\Phi_i(x^k)\| \)
- \( \alpha_{\max} = \max_{i=1,\ldots,N} \{\alpha_i\}, \quad \alpha_{\min} = \min_{i=1,\ldots,N} \{\alpha_i\} \)
- \( p_{\max} = \max_{i=1,\ldots,N} \{p_i\} \) and \( p_{\min} = \min_{i=1,\ldots,N} \{p_i\} \)
- \( p_i \) is the probability of agent \( i \) updating
- \( B = (\max_i L_i)NM \)
- \( M \geq \max_{x_i, z_i \in K_i} \|x_i - z_i\| \) for all \( i \)
Expression for error - equal step sizes

\[
Err(N, n, G) = \frac{\alpha p_{\text{max}}}{\mu p_{\text{min}}} \left(2C^2N + B \frac{\sqrt{2nN} C}{1 - \sqrt{\lambda}}\right)
\]

- where \(\alpha_{\text{max}} = \max_{i=1,\ldots,N} \{\alpha_i\}\), \(\alpha_{\text{min}} = \min_{i=1,\ldots,N} \{\alpha_i\}\),

- \(p_{\text{max}} = \max_{i=1,\ldots,N} \{p_i\}\) and \(p_{\text{min}} = \min_{i=1,\ldots,N} \{p_i\}\).

- \(p_i\) is the probability of agent \(i\) updating

- \(B = (\max_i L_{-i})NM\)

- \(M \geq \max_{x_i, z_i \in K_i} \|x_i - z_i\|\) for all \(i\)
Outline

1. Aggregative Nash Games
   - Aggregative Game
   - Aggregative Game on Network

2. Distributed synchronous algorithm

3. Distributed asynchronous algorithm

4. Numerical Results
A Nash-Cournot game: a classic aggregative game

Firm $i$ solves the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad c_i(x_i) - x_i g(\bar{x}) \\
\text{subject to} & \quad x_i \in K_i,
\end{align*}
\]

- $c_i(x_i) = u_i x_i + v_i x_i^2$
- $g(\bar{x}) = 100 - \bar{x}$
- $K_i = [0, 500]$
- Nash equilibrium exists and is unique

\[
x_i^* = \frac{100 - u_i}{2v_i + 1} - \frac{1}{(2v_i + 1) \left(1 + \sum_{j=1}^{N} \frac{1}{2v_j + 1}\right)} \sum_{j=1}^{N} \frac{100 - u_j}{(2v_j + 1)}
\]
A Nash-Cournot game: a classic aggregative game

Firm $i$ solves the following optimization problem:

minimize $c_i(x_i) - x_i g(\bar{x})$

subject to $x_i \in K_i$,

- $c_i(x_i) = u_i x_i + v_i x_i^2$
- $g(\bar{x}) = 100 - \bar{x}$
- $K_i = [0, 500]$
- Nash equilibrium exists and is unique

$$x_i^* = \frac{100 - u_i}{2v_i + 1} - \frac{1}{(2v_i + 1)} \left( \sum_{j=1}^{N} \frac{100 - u_j}{2v_j + 1} \right)$$
A Nash-Cournot game: a classic aggregative game

Firm $i$ solves the following optimization problem:

$$\begin{align*}
\text{minimize} & \quad c_i(x_i) - x_ig(\bar{x}) \\
\text{subject to} & \quad x_i \in K_i,
\end{align*}$$

- $c_i(x_i) = u_ix_i + v_ix_i^2$
- $g(\bar{x}) = 100 - \bar{x}$
- $K_i = [0, 500]$
- Nash equilibrium exists and is unique

$$x_i^* = \frac{100 - u_i}{2v_i + 1} - \frac{1}{(2v_i + 1)} \left( \sum_{j=1}^{N} \frac{100-u_j}{2v_j+1} \right)$$
A Nash-Cournot game: a classic aggregative game

Firm $i$ solves the following optimization problem:

\[
\text{minimize} \quad c_i(x_i) - x_i g(\bar{x}) \\
\text{subject to} \quad x_i \in K_i,
\]

- $c_i(x_i) = u_i x_i + v_i x_i^2$
- $g(\bar{x}) = 100 - \bar{x}$
- $K_i = [0, 500]$
- Nash equilibrium exists and is unique

\[
x_i^* = \frac{100 - u_i}{2v_i + 1} - \frac{1}{(2v_i + 1)} \left( \frac{\sum_{j=1}^{N} \frac{100 - u_j}{2v_j + 1}}{1 + \sum_{j=1}^{N} \frac{1}{2v_j + 1}} \right)
\]
Simulation setup

- $u_i$ and $v_i$ are once chosen and remain fixed
  
  $$u_i \sim U(2, 10) \quad v_i \sim U(2, 4) \quad \forall i$$

- Network size: $N = 20, 50, \text{ and } 100$

- Terminate after $\tilde{k} = 5e3 \text{ and } 1e4$ iterations

- Report mean error and confidence interval for 50 sample paths

  $$\text{err}_\ell := \left\| x^{\tilde{k}} - x^* \right\|_\infty = \max_{1 \leq i \leq n} \{|[x^{\tilde{k}}]_i - [x^*]_i|\}. $$
Synchronous algorithm: Setting

- Generate an adjacency matrix $A(k)$ such that $G$ is connected.
- Given $A(k)$ generate the weight matrix $W(k)$ as

$$[W]_{ij} = \begin{cases} 
0 & \text{if } A_{ij} = 0 \\
\delta & \text{if } A_{ij} = 1 \\
1 - \delta d(i) & \text{if } i = j,
\end{cases}$$

$d(i)$: number of neighbors of player $i$, and

$$\delta = \frac{0.5}{\max_i\{d(i)\}}.$$ 

- Stepsize update rule

$$\alpha_{k,i} = \frac{\alpha}{k}, \quad \text{for all } i = 1, \ldots, N, \quad \text{where } \alpha = 0.5.$$
Synchronous algorithm: Mean error

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\tilde{k} = 5e3$</th>
<th>$\tilde{k} = 1e4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>6.49e-5</td>
<td>6.03e-6</td>
</tr>
<tr>
<td>50</td>
<td>1.13e-2</td>
<td>1.70e-3</td>
</tr>
<tr>
<td>100</td>
<td>2.29e-1</td>
<td>3.18e-2</td>
</tr>
</tbody>
</table>
### Synchronous algorithm: Mean error

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\tilde{k} = 5\times 10^3$</th>
<th>$\tilde{k} = 1\times 10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>6.49e-5</td>
<td>6.03e-6</td>
</tr>
<tr>
<td>50</td>
<td>1.13e-2</td>
<td>1.70e-3</td>
</tr>
<tr>
<td>100</td>
<td>2.29e-1</td>
<td>3.18e-2</td>
</tr>
</tbody>
</table>

**Table: Dynamic network**

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\tilde{k} = 5\times 10^3$</th>
<th>$\tilde{k} = 1\times 10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.11e-8</td>
<td>1.47e-9</td>
</tr>
<tr>
<td>50</td>
<td>5.07e-7</td>
<td>8.06e-8</td>
</tr>
<tr>
<td>100</td>
<td>3.26e-6</td>
<td>5.04e-7</td>
</tr>
</tbody>
</table>

**Table: Complete network**
Synchronous algorithm: Confidence interval width

Table: Dynamic network

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\tilde{k} = 5e3$</th>
<th>$\tilde{k} = 1e4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5.91e-9</td>
<td>5.81e-10</td>
</tr>
<tr>
<td>50</td>
<td>5.03e-7</td>
<td>9.77e-8</td>
</tr>
<tr>
<td>100</td>
<td>1.32e-5</td>
<td>1.43e-6</td>
</tr>
</tbody>
</table>

Table: Complete network

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\tilde{k} = 5e3$</th>
<th>$\tilde{k} = 1e4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.61e-12</td>
<td>1.09e-13</td>
</tr>
<tr>
<td>50</td>
<td>5.28e-11</td>
<td>4.39e-12</td>
</tr>
<tr>
<td>100</td>
<td>2.54e-10</td>
<td>2.14e-11</td>
</tr>
</tbody>
</table>
Asynchronous algorithm: Types of network

(a) Cycle
(b) Wheel
(c) Grid
(d) Complete
Asynchronous algorithm: Mean error

**Table:** Diminishing stepsize: $\tilde{k} = 5e3$

<table>
<thead>
<tr>
<th>N</th>
<th>Cycle</th>
<th>Wheel</th>
<th>Grid</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5.97e-1</td>
<td>2.45e-4</td>
<td>9.74e-3</td>
<td>7.41e-5</td>
</tr>
<tr>
<td>50</td>
<td>21.17</td>
<td>1.67</td>
<td>4.78</td>
<td>8.99</td>
</tr>
<tr>
<td>100</td>
<td>107.76</td>
<td>163.71</td>
<td>84.9</td>
<td>16.86</td>
</tr>
</tbody>
</table>
Asynchronous algorithm: Mean error

**Table:** Diminishing stepsize: $\tilde{k} = 5e3$

<table>
<thead>
<tr>
<th>N</th>
<th>Cycle</th>
<th>Wheel</th>
<th>Grid</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5.97e-1</td>
<td>2.45e-4</td>
<td>9.74e-3</td>
<td>7.41e-5</td>
</tr>
<tr>
<td>50</td>
<td>21.17</td>
<td>1.67</td>
<td>4.78</td>
<td>8.99</td>
</tr>
<tr>
<td>100</td>
<td>107.76</td>
<td>163.71</td>
<td>84.9</td>
<td>16.86</td>
</tr>
</tbody>
</table>

**Table:** Diminishing stepsize: $\tilde{k} = 1e4$

<table>
<thead>
<tr>
<th>N</th>
<th>Cycle</th>
<th>Wheel</th>
<th>Grid</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>7.69e-2</td>
<td>7.16e-6</td>
<td>2.85e-4</td>
<td>4.82e-7</td>
</tr>
<tr>
<td>50</td>
<td>11.0</td>
<td>1.05e-1</td>
<td>4.78</td>
<td>8.98e-3</td>
</tr>
<tr>
<td>100</td>
<td>54.2</td>
<td>13.31</td>
<td>34.5</td>
<td>1.72</td>
</tr>
</tbody>
</table>
Asynchronous algorithm: Mean error

Complete ≻ Wheel ≻ Grid ≻ Cycle

Table: Diminishing stepsize: $\tilde{k} = 5e3$

<table>
<thead>
<tr>
<th>N</th>
<th>Cycle</th>
<th>Wheel</th>
<th>Grid</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5.97e-1</td>
<td>2.45e-4</td>
<td>9.74e-3</td>
<td>7.41e-5</td>
</tr>
<tr>
<td>50</td>
<td>21.17</td>
<td>1.67</td>
<td>4.78</td>
<td>8.99</td>
</tr>
<tr>
<td>100</td>
<td>107.76</td>
<td>163.71</td>
<td>84.9</td>
<td>16.86</td>
</tr>
</tbody>
</table>

Table: Diminishing stepsize: $\tilde{k} = 1e4$

<table>
<thead>
<tr>
<th>N</th>
<th>Cycle</th>
<th>Wheel</th>
<th>Grid</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>7.69e-2</td>
<td>7.16e-6</td>
<td>2.85e-4</td>
<td>4.82e-7</td>
</tr>
<tr>
<td>50</td>
<td>11.0</td>
<td>1.05e-1</td>
<td>4.78</td>
<td>8.98e-3</td>
</tr>
<tr>
<td>100</td>
<td>54.2</td>
<td>13.31</td>
<td>34.5</td>
<td>1.72</td>
</tr>
</tbody>
</table>
Asynchronous algorithm: Confidence interval width

Complete $\succ$ Wheel $\succ$ Grid $\succ$ Cycle

**Table:** Diminishing stepsize: $\tilde{k} = 5e3$

<table>
<thead>
<tr>
<th>N</th>
<th>Cycle</th>
<th>Wheel</th>
<th>Grid</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>2.42e-4</td>
<td>2.10e-7</td>
<td>9.51e-6</td>
<td>2.29e-8</td>
</tr>
<tr>
<td>50</td>
<td>6.34e-3</td>
<td>9.25e-4</td>
<td>2.77e-3</td>
<td>1.12e-4</td>
</tr>
<tr>
<td>100</td>
<td>5.88e-2</td>
<td>8.38e-1</td>
<td>7.46e-2</td>
<td>1.65e-2</td>
</tr>
</tbody>
</table>

**Table:** Diminishing stepsize: $\tilde{k} = 1e4$

<table>
<thead>
<tr>
<th>N</th>
<th>Cycle</th>
<th>Wheel</th>
<th>Grid</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.46e-5</td>
<td>2.54e-8</td>
<td>1.21e-7</td>
<td>7.64e-10</td>
</tr>
<tr>
<td>50</td>
<td>1.13e-3</td>
<td>5.55e-5</td>
<td>8.21e-4</td>
<td>3.67e-6</td>
</tr>
<tr>
<td>100</td>
<td>9.90e-3</td>
<td>4.09e-3</td>
<td>4.94e-3</td>
<td>1.78e-3</td>
</tr>
</tbody>
</table>
Asynchronous algorithm: Confidence interval width

Complete $>$ Wheel $>$ Grid $>$ Cycle

- $p_{\text{min}}/p_{\text{max}}$ : probability of agents performing update
- $\lambda$ : second largest eigenvalue of the expected weight matrix
Asynchronous algorithm: Confidence interval width

Complete $\succ$ Wheel $\succ$ Grid $\succ$ Cycle

- $p_{\text{min}}/p_{\text{max}}$: probability of agents performing update
- $\lambda$: second largest eigenvalue of the expected weight matrix

Table: Number of iteration for concurrence of player’s aggregate within an error of $1e^{-2}$

<table>
<thead>
<tr>
<th>Network</th>
<th>$p_{\text{min}}/p_{\text{max}}$</th>
<th>$\lambda$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle</td>
<td>1</td>
<td>0.9994</td>
<td>177</td>
</tr>
<tr>
<td>Grid</td>
<td>$5/7$</td>
<td>0.3151</td>
<td>66</td>
</tr>
<tr>
<td>Wheel</td>
<td>$1/19$</td>
<td>0.1622</td>
<td>31</td>
</tr>
<tr>
<td>Complete</td>
<td>1</td>
<td>$1.0888e-08$</td>
<td>17</td>
</tr>
</tbody>
</table>
Insights from numerics

- Synchronous algorithm outperforms asynchronous algorithm
- Connectivity graph plays an important role
- Asynchronous algorithm is slow
- Complete $\succeq$ Wheel $\succeq$ Grid $\succeq$ Cycle
  - Depends on the probabilities of agents performing update
  - And the second largest eigenvalue of the expected weight matrix
References I
Further extensions

- More general case can be solved

  \[
  \begin{align*}
  \text{minimize} & \quad f_i(x_i, \bar{x})), \quad \bar{x} = \sum_{j=1}^{N} x_j \\
  \text{subject to} & \quad x_i \in K_i \subseteq \mathbb{R}^n, \\
  & \quad g_1(x) \leq 0, \ldots, g_m(x) \leq 0 \quad \text{(AggGame)}
  \end{align*}
  \]
  
  under suitable assumptions on \( g_i \).

- Push-sum can be used for mixing (instead of convex combinations)