# Asynchronous Distributed Optimization

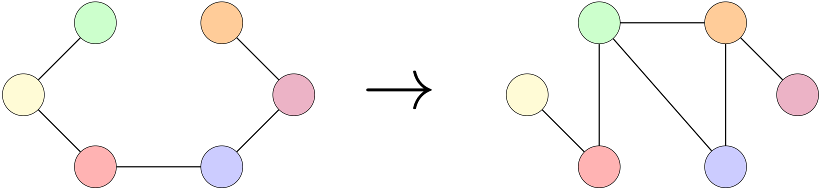
**Matthew Hale**

**Department of Mechanical and Aerospace Engineering**
**University of Florida**

**AFOSR Center of Excellence Kickoff**
**May 14, 2019**

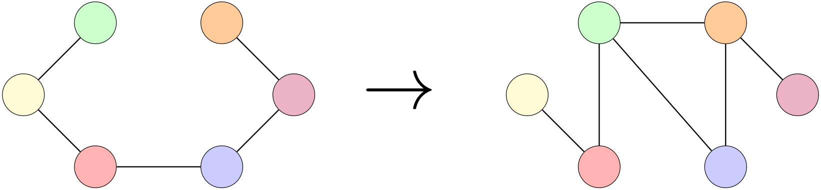
# Agents collaborate on several types of tasks

- Modifying the network itself to improve communications



# Agents collaborate on several types of tasks

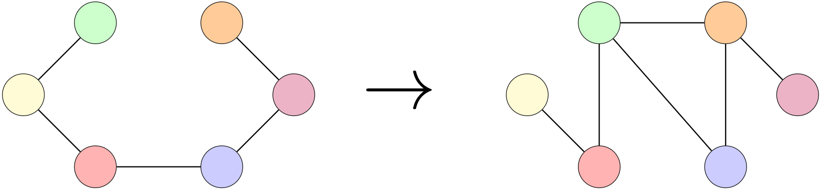
- Modifying the network itself to improve communications



- Exchanging positions to jointly plan trajectories

# Agents collaborate on several types of tasks

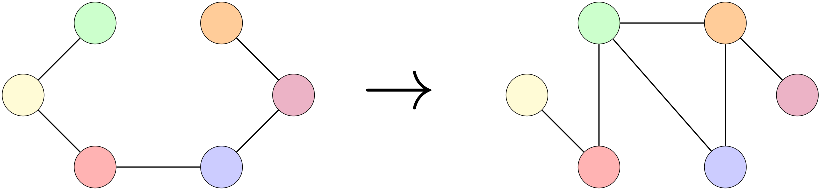
- Modifying the network itself to improve communications



- Exchanging positions to jointly plan trajectories
- Exchanging sensor data to process data streams

# Agents collaborate on several types of tasks

- Modifying the network itself to improve communications



- Exchanging positions to jointly plan trajectories
- Exchanging sensor data to process data streams

**Common Threads**

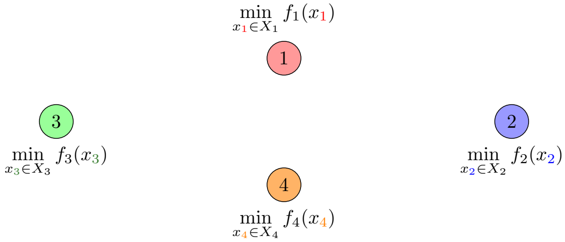
These tasks require (i) information sharing and (ii) optimizing some quantity.

# What problems do we want to solve?

- Convex programs formalize several classes of these tasks

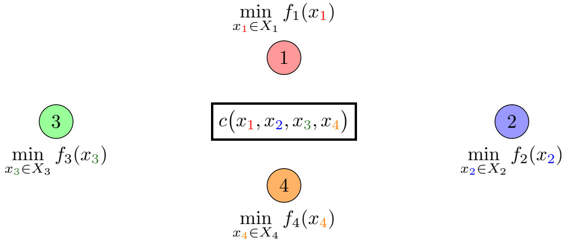
# What problems do we want to solve?

- Convex programs formalize several classes of these tasks
- Consider a general multi-agent optimization setup:



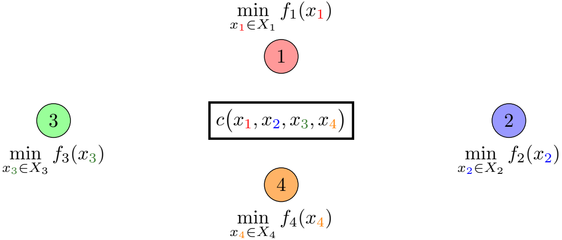
# What problems do we want to solve?

- Convex programs formalize several classes of these tasks
- Consider a general multi-agent optimization setup:



# What problems do we want to solve?

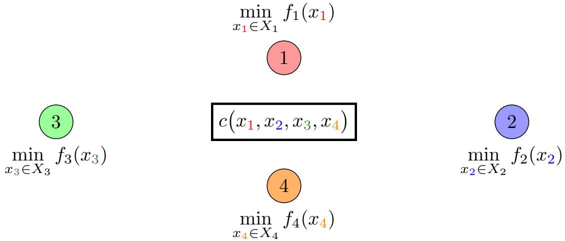
- Convex programs formalize several classes of these tasks
- Consider a general multi-agent optimization setup:




- Define $x = (x_1, x_2, x_3, x_4)$ and $X = X_1 \times X_2 \times X_3 \times X_4$

# What problems do we want to solve?

- Convex programs formalize several classes of these tasks
- Consider a general multi-agent optimization setup:

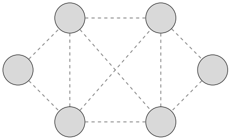



- Define $x = (x_1, x_2, x_3, x_4)$ and $X = X_1 \times X_2 \times X_3 \times X_4$
- Want to solve

$$\underset{x \in X}{\text{minimize}} \; f(x) := c(x) + \sum_{i=1}^{N} f_i(x_i)$$

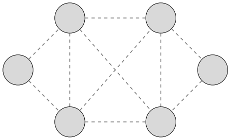
# Agents must optimize asynchronously

- Communications are unavoidably asynchronous



# Agents must optimize asynchronously

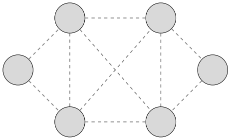
- Communications are unavoidably asynchronous



- Agents receive different information at different times and agents disagree as they work together

# Agents must optimize asynchronously

- Communications are unavoidably asynchronous



- Agents receive different information at different times and agents disagree as they work together

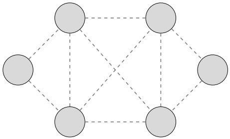
**Overall goals**

Design an asynchronous optimization framework that:

- Makes progress toward an optimum when new information is shared

# Agents must optimize asynchronously

- Communications are unavoidably asynchronous



- Agents receive different information at different times and agents disagree as they work together

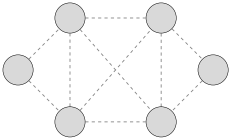
**Overall goals**

Design an asynchronous optimization framework that:

- Makes progress toward an optimum when new information is shared
- Does not undo progress when information is not shared

# Agents must optimize asynchronously

- Communications are unavoidably asynchronous



- Agents receive different information at different times and agents disagree as they work together

**Overall goals**

Design an asynchronous optimization framework that:

- Makes progress toward an optimum when new information is shared
- Does not undo progress when information is not shared

- This is a forward invariance condition!

# Accounting for Heterogeneity

- We track agents' different information:

$$x^i(k) = \begin{pmatrix} x_1^i(k) \\ \vdots \\ x_N^i(k) \end{pmatrix}$$

- Always contains most recent information held by agent $i$

# Accounting for Heterogeneity

- We track agents' different information:

$$x^i(k) = \begin{pmatrix} x_1^i(k) \\ \vdots \\ x_N^i(k) \end{pmatrix}$$

- Always contains most recent information held by agent $i$
- We expect $x^i(k) \neq x^j(k)$ at all times $k$

# Accounting for Heterogeneity

- We track agents' different information:

$$x^i(k) = \begin{pmatrix} x_1^i(k) \\ \vdots \\ x_N^i(k) \end{pmatrix}$$

- Always contains most recent information held by agent $i$
- We expect $x^i(k) \neq x^j(k)$ at all times $k$
- Let agents independently regularize

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| $f(x) + \alpha_1 \|x_1\|^2$ | $f(x) + \alpha_2 \|x_2\|^2$ | $f(x) + \alpha_3 \|x_3\|^2$ | $f(x) + \alpha_4 \|x_4\|^2$ |

# Block-Based Update Law

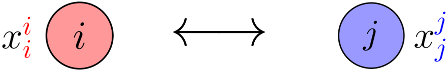
- Gradient descent is robust to many things

# Block-Based Update Law

- Gradient descent is robust to many things
- Only agent $i$ updates its own decision variable $x_i$

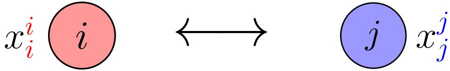
# Block-Based Update Law

- Gradient descent is robust to many things
- Only agent $i$ updates its own decision variable $x_i$
- Agent $i$ updates $x_i^i$ but not $x_j^i$ – it gets that from agent $j$



# Block-Based Update Law

- Gradient descent is robust to many things
- Only agent $i$ updates its own decision variable $x_i$
- Agent $i$ updates $x_i^i$ but not $x_j^i$ – it gets that from agent $j$

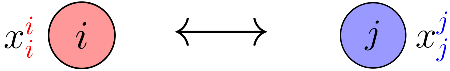



**Desired Update Law**

$$x_i^i(k+1) = \begin{cases} \Pi_{X_i}\left[x_i^i(k) - \gamma\left(\frac{\partial f}{\partial x_i}(x^i(k)) + \alpha_i x_i^i(k)\right)\right] & i \text{ updates at } k \\ \end{cases}$$

# Block-Based Update Law

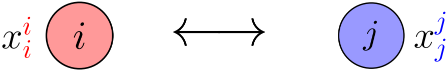
- Gradient descent is robust to many things
- Only agent $i$ updates its own decision variable $x_i$
- Agent $i$ updates $x_i^i$ but not $x_j^i$ – it gets that from agent $j$




**Desired Update Law**

$$x_i^i(k+1) = \begin{cases} \Pi_{X_i}\left[x_i^i(k) - \gamma\left(\frac{\partial f}{\partial x_i}(x^i(k)) + \alpha_i x_i^i(k)\right)\right] & i \text{ updates at } k \\ x_i^i(k) & \text{otherwise} \end{cases}$$

- Gradient descent is robust to many things
- Only agent $i$ updates its own decision variable $x_i$
- Agent $i$ updates $x_i^i$ but not $x_j^i$ – it gets that from agent $j$



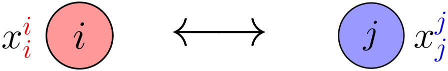

**Desired Update Law**

$$x_i^i(k+1) = \begin{cases} \Pi_{X_i}\left[x_i^i(k) - \gamma\left(\frac{\partial f}{\partial x_i}(x^i(k)) + \alpha_i x_i^i(k)\right)\right] & i \text{ updates at } k \\ x_i^i(k) & \text{otherwise} \end{cases}$$

$$x_j^i(k+1) = \begin{cases} x_j^j & x_j^j \text{ received at time } k \\ & \end{cases}$$

# Block-Based Update Law

- Gradient descent is robust to many things
- Only agent $i$ updates its own decision variable $x_i$
- Agent $i$ updates $x_i^i$ but not $x_j^i$ – it gets that from agent $j$



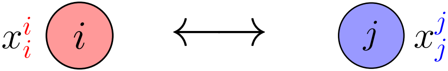

**Desired Update Law**

$$x_i^i(k+1) = \begin{cases} \Pi_{X_i}\left[x_i^i(k) - \gamma\left(\frac{\partial f}{\partial x_i}(x^i(k)) + \alpha_i x_i^i(k)\right)\right] & i \text{ updates at } k \\ x_i^i(k) & \text{otherwise} \end{cases}$$

$$x_j^i(k+1) = \begin{cases} x_j^j & x_j^j \text{ received at time } k \\ x_j^i(k) & \text{otherwise} \end{cases}$$

# Block-Based Update Law

- Gradient descent is robust to many things
- Only agent $i$ updates its own decision variable $x_i$
- Agent $i$ updates $x_i^i$ but not $x_j^i$ – it gets that from agent $j$




**Desired Update Law**

$$x_i^i(k+1) = \begin{cases} \Pi_{X_i}\left[x_i^i(k) - \gamma\left(\frac{\partial f}{\partial x_i}(x^i(k)) + \alpha_i x_i^i(k)\right)\right] & i \text{ updates at } k \\ x_i^i(k) & \text{otherwise} \end{cases}$$

$$x_j^i(k+1) = \begin{cases} x_j^j & x_j^j \text{ received at time } k \\ x_j^i(k) & \text{otherwise} \end{cases}$$

"Do gradient descent with whatever you have"

# Lyapunov Convergence Analysis

- Define

$$V(k) = \max_{i \in [N]} \|x^i(k) - \hat{x}\|_2$$

# Lyapunov Convergence Analysis

- Define

$$V(k) = \max_{i \in [N]} \|x^i(k) - \hat{x}\|_2$$

**Theorem 1: Asymptotic Convergence**

Suppose $\gamma$ is small enough and $\alpha_i > 0$ for all $i \in [N]$. Then $V(k) \to 0$ and

$$x_i^i(k+1) = \Pi_{X_i}\left[x_i^i(k) - \gamma\left(\frac{\partial f}{\partial x_i}(x^i(k)) + \alpha_i x_i^i(k)\right)\right]$$

converges to $\hat{x}$ asymptotically.

# Lyapunov Convergence Analysis

- Define

$$V(k) = \max_{i \in [N]} \|x^i(k) - \hat{x}\|_2$$

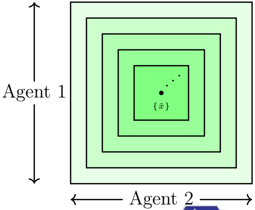
**Theorem 1: Asymptotic Convergence**

Suppose $\gamma$ is small enough and $\alpha_i > 0$ for all $i \in [N]$. Then $V(k) \to 0$ and

$$x_i^i(k+1) = \Pi_{X_i}\left[x_i^i(k) - \gamma\left(\frac{\partial f}{\partial x_i}(x^i(k)) + \alpha_i x_i^i(k)\right)\right]$$

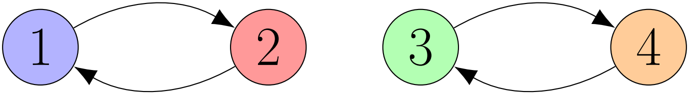
converges to $\hat{x}$ asymptotically.

- And each sub-level set is forward-invariant!



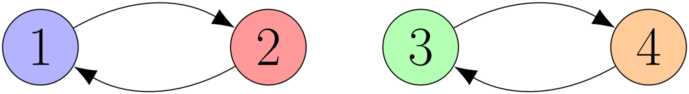

UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

- Definition: A **communication cycle** occurs after each agent has (i) computed a state update and (ii) shared it with others that need it



# Convergence Analysis

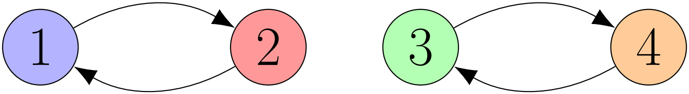
- Definition: A **communication cycle** occurs after each agent has (i) computed a state update and (ii) shared it with others that need it




- Use $c(k)$ to denote $\#$ of cycles completed by time $k$

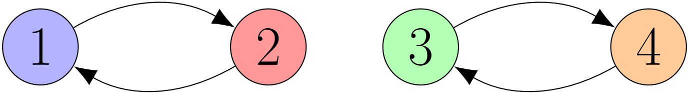
# Convergence Analysis

- Definition: A **communication cycle** occurs after each agent has (i) computed a state update and (ii) shared it with others that need it




- Use $c(k)$ to denote # of cycles completed by time $k$
- Use $L_i$ to denote the Lipschitz constant of $\nabla_i f$
- Define $q = \max_{i \in [N]} \left\{ |1 - \gamma \alpha_i|, |1 - \gamma L_i| \right\}$

- Definition: A **communication cycle** occurs after each agent has (i) computed a state update and (ii) shared it with others that need it




- Use $c(k)$ to denote # of cycles completed by time $k$
- Use $L_i$ to denote the Lipschitz constant of $\nabla_i f$
- Define $q = \max_{i \in [N]} \left\{ |1 - \gamma \alpha_i|, |1 - \gamma L_i| \right\}$

**Theorem 2: Convergence Rate**

We have $q \in (0, 1)$ and

$$V(k) \leq q^{c(k)} V(0)$$

- In this setting, best to interleave communications and computations onboard each agent

# Impacts and Interpretations

- In this setting, best to interleave communications and computations onboard each agent
- If there is a known delay bound $B$, then convergence rate is

$$V(k) \leq q^{\lfloor k/B \rfloor} V(0)$$

# Impacts and Interpretations

- In this setting, best to interleave communications and computations onboard each agent
- If there is a known delay bound $B$, then convergence rate is

$$V(k) \leq q^{\lfloor k/B \rfloor} V(0)$$

**Direct Generalization of Classic Result**

Make centralized, set all parameters equal. Then

$$\|x(k) - \hat{x}\|_2^2 \leq q^k \|x(0) - \hat{x}\|_2^2.$$

# Next Steps

- Add in constraints!

$$\begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \leq 0 \\ & x \in X \end{aligned}$$

# Next Steps

- Add in constraints!

$$\begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \leq 0 \\ & x \in X \end{aligned}$$

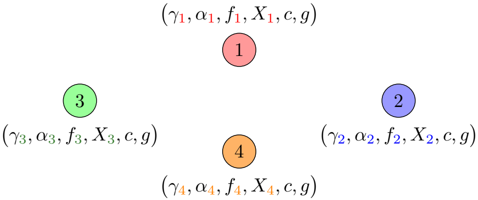
- How to set $\gamma_i \neq \gamma_j$ and still converge?

# Next Steps

- Add in constraints!

$$\begin{aligned} \text{minimize } & f(x) \\ \text{subject to } & g(x) \leq 0 \\ & x \in X \end{aligned}$$

- How to set $\gamma_i \neq \gamma_j$ and still converge?
- End goal:



# Thank you