Asynchronous Constrained Convex Optimization in Blocks

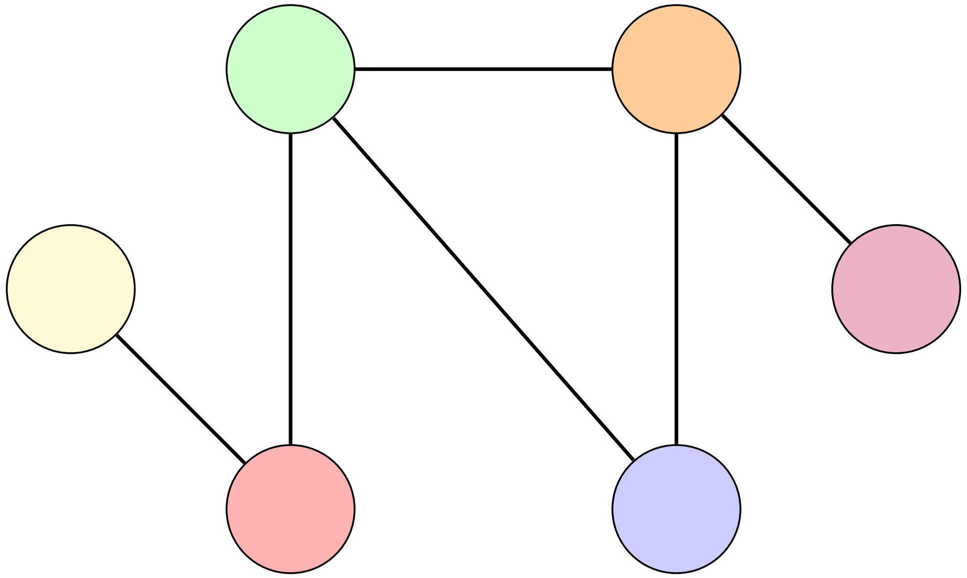
Katherine Hendrickson and Matthew Hale

Department of Mechanical and Aerospace Engineering
University of Florida

AFOSR Center of Excellence Review
October 30th, 2020

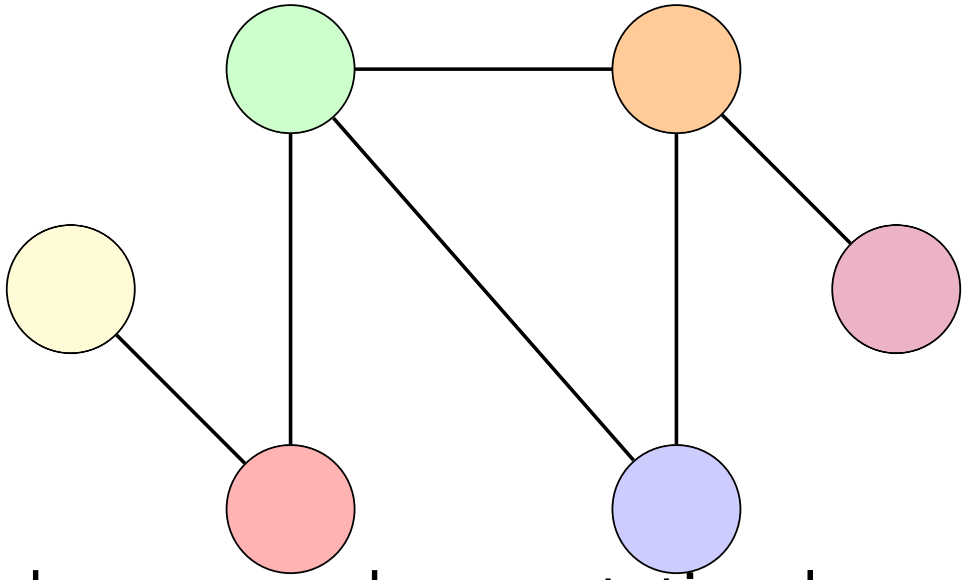
Computational problems must be decentralized

- Agents must interact to collectively solve problems



Computational problems must be decentralized

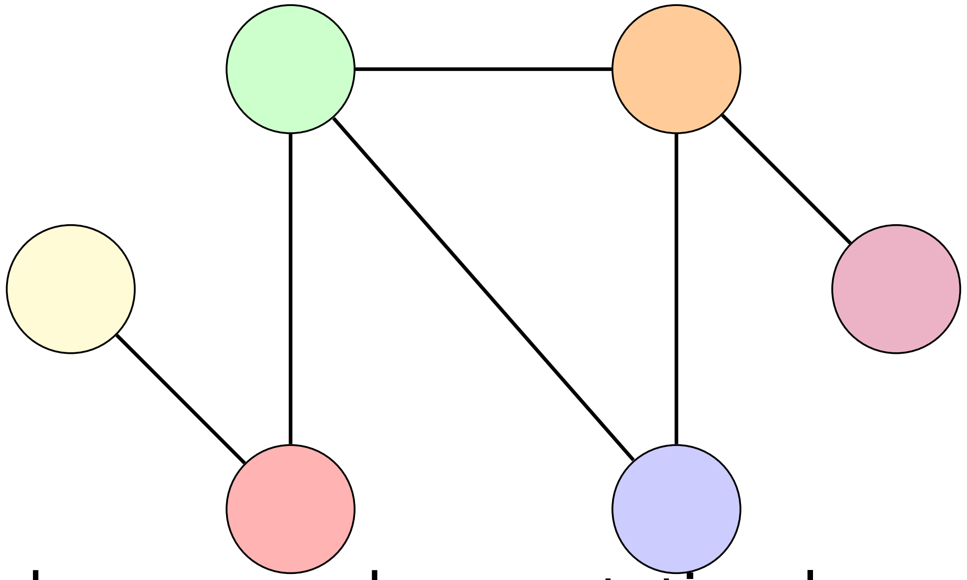
- Agents must interact to collectively solve problems



- Agents have limited energy and computational power
  $\implies$ Challenge #1: Algorithms must be lightweight, simple to implement

Computational problems must be decentralized

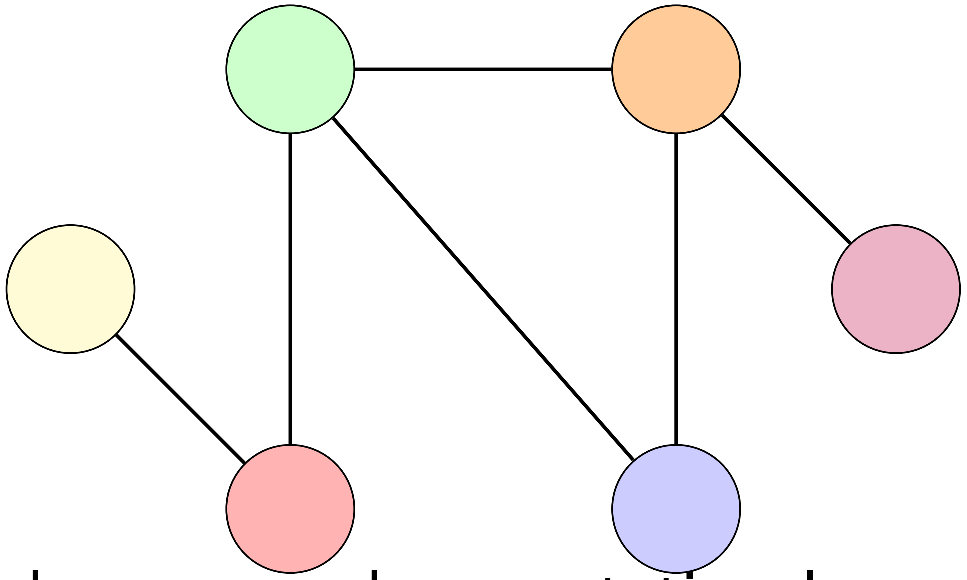
- Agents must interact to collectively solve problems



- Agents have limited energy and computational power
  $\Longrightarrow$ Challenge #1: Algorithms must be lightweight, simple to implement
- Agents can generate and share information with unpredictable timing
  $\Longrightarrow$ Challenge #2: Algorithms must be robust to asynchrony

Computational problems must be decentralized

- Agents must interact to collectively solve problems



- Agents have limited energy and computational power
  $\implies$ Challenge #1: Algorithms must be lightweight, simple to implement
- Agents can generate and share information with unpredictable timing
  $\implies$ Challenge #2: Algorithms must be robust to asynchrony

Problems of interest

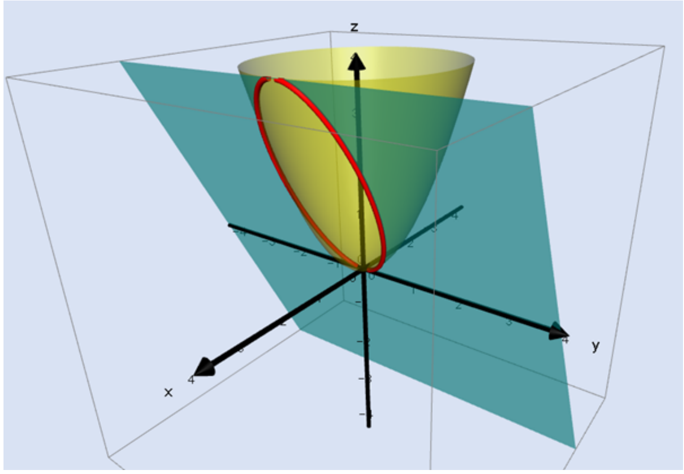
We are interested in problems from trajectory planning, machine learning, estimation, and others arising in autonomy.

We will solve constrained optimization problems

General convex programs

The problems of interest (convex for now) are formalized as

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

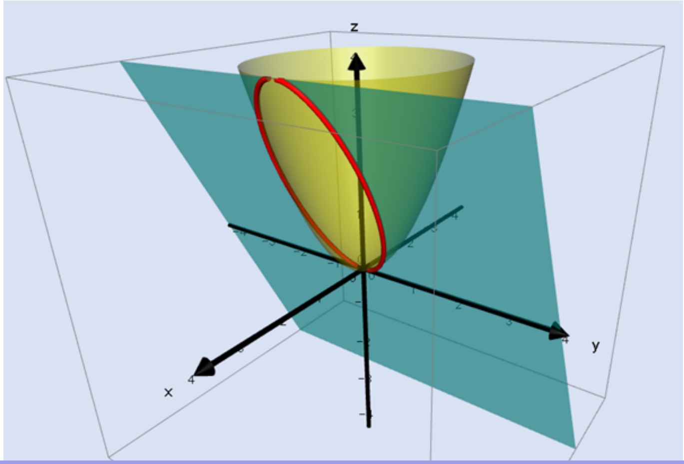


We will solve constrained optimization problems

General convex programs

The problems of interest (convex for now) are formalized as

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



In this talk

- Optimize in a distributed way that is robust to information delays
- Avoid averaging-based update laws:
  1. Promotes scalability for computationally constrained agents
  2. Respects division of responsibility in autonomy

We use a Lagrangian saddle point formulation

Saddle point formulation

- We write problems as

$$\underset{x \in X}{\text{minimize}} \ \underset{\mu \in \mathbb{R}^m_+}{\text{maximize}} \ L_{\alpha,\beta}(x,\mu) = \underbrace{f(x) + \mu^T g(x)}_{\text{Usual Lagrangian } L(x,\mu)} + \frac{\alpha}{2}\|x\|^2 - \frac{\beta}{2}\|\mu\|^2$$

- Regularizing makes $L_{\alpha,\beta}$ strongly convex-strongly concave

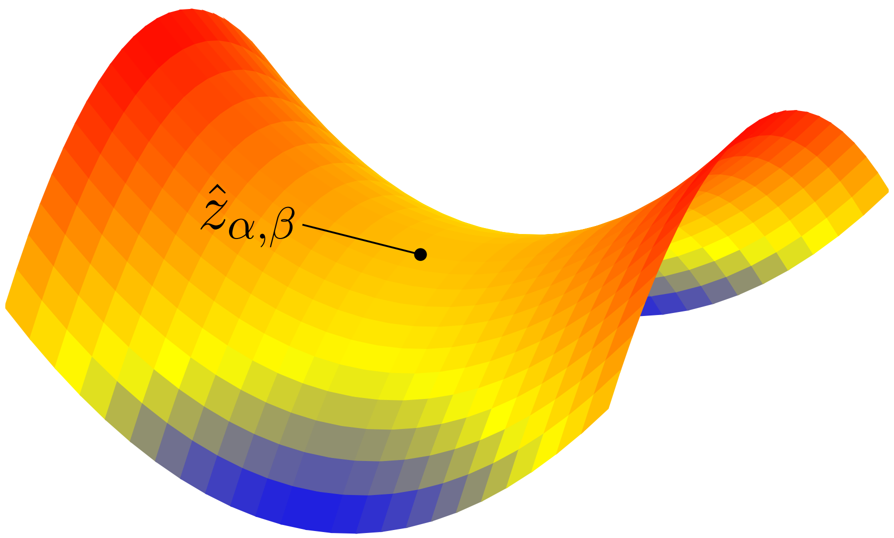
We use a Lagrangian saddle point formulation

Saddle point formulation

- We write problems as

$$\underset{x \in X}{\text{minimize}} \ \underset{\mu \in \mathbb{R}^m_+}{\text{maximize}} \ L_{\alpha,\beta}(x,\mu) = \underbrace{f(x) + \mu^T g(x)}_{\text{Usual Lagrangian } L(x,\mu)} + \frac{\alpha}{2}\|x\|^2 - \frac{\beta}{2}\|\mu\|^2$$

- Regularizing makes $L_{\alpha,\beta}$ strongly convex-strongly concave
- We now want a saddle point $\hat{z}_{\alpha,\beta} = \left(\hat{x}_{\alpha,\beta}, \hat{\mu}_{\alpha,\beta}\right)$



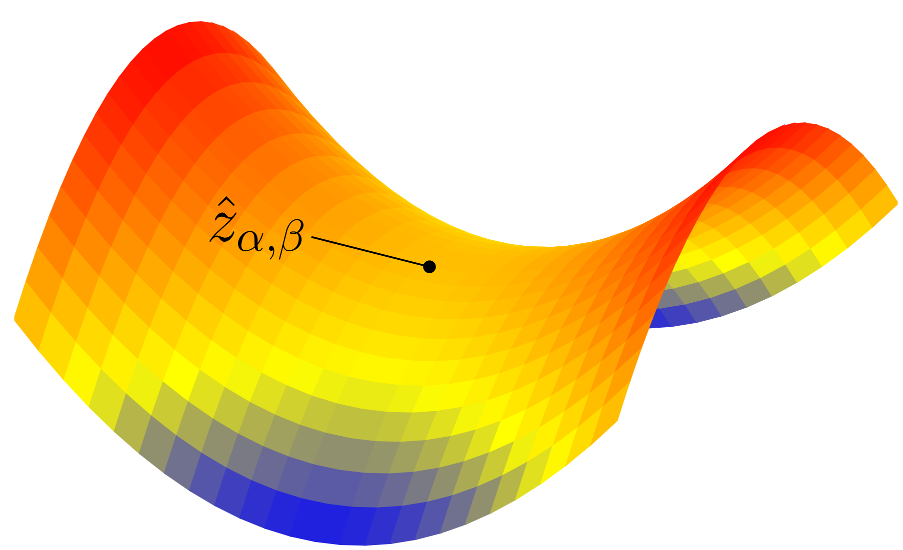
We use a Lagrangian saddle point formulation

Saddle point formulation

- We write problems as

$$\underset{x \in X}{\text{minimize}} \; \underset{\mu \in \mathbb{R}^m_+}{\text{maximize}} \; L_{\alpha,\beta}(x,\mu) = \underbrace{f(x) + \mu^T g(x)}_{\text{Usual Lagrangian } L(x,\mu)} + \frac{\alpha}{2}\|x\|^2 - \frac{\beta}{2}\|\mu\|^2$$

- Regularizing makes $L_{\alpha,\beta}$ strongly convex-strongly concave
- We now want a saddle point $\hat{z}_{\alpha,\beta} = \left(\hat{x}_{\alpha,\beta}, \hat{\mu}_{\alpha,\beta}\right)$



- Small $(\alpha, \beta) \implies$ small $\|\hat{z} - \hat{z}_{\alpha,\beta}\|$

We must accommodate asynchronous interactions

- Agents' computations are asynchronous due to clock mismatches and heterogeneous hardware
- Comms. are asynchronous due to environmental hazards and jamming

We must accommodate asynchronous interactions

- Agents' computations are asynchronous due to clock mismatches and heterogeneous hardware
- Comms. are asynchronous due to environmental hazards and jamming
- Agents disagree and we track their knowledge at each time:

$$\underset{(x^i(k), \mu^i(k))}{\textcircled{i}} \neq \underset{(x^j(k), \mu^j(k))}{\textcircled{j}}$$

We must accommodate asynchronous interactions

- Agents' computations are asynchronous due to clock mismatches and heterogeneous hardware
- Comms. are asynchronous due to environmental hazards and jamming
- Agents disagree and we track their knowledge at each time:

i $\qquad$ j

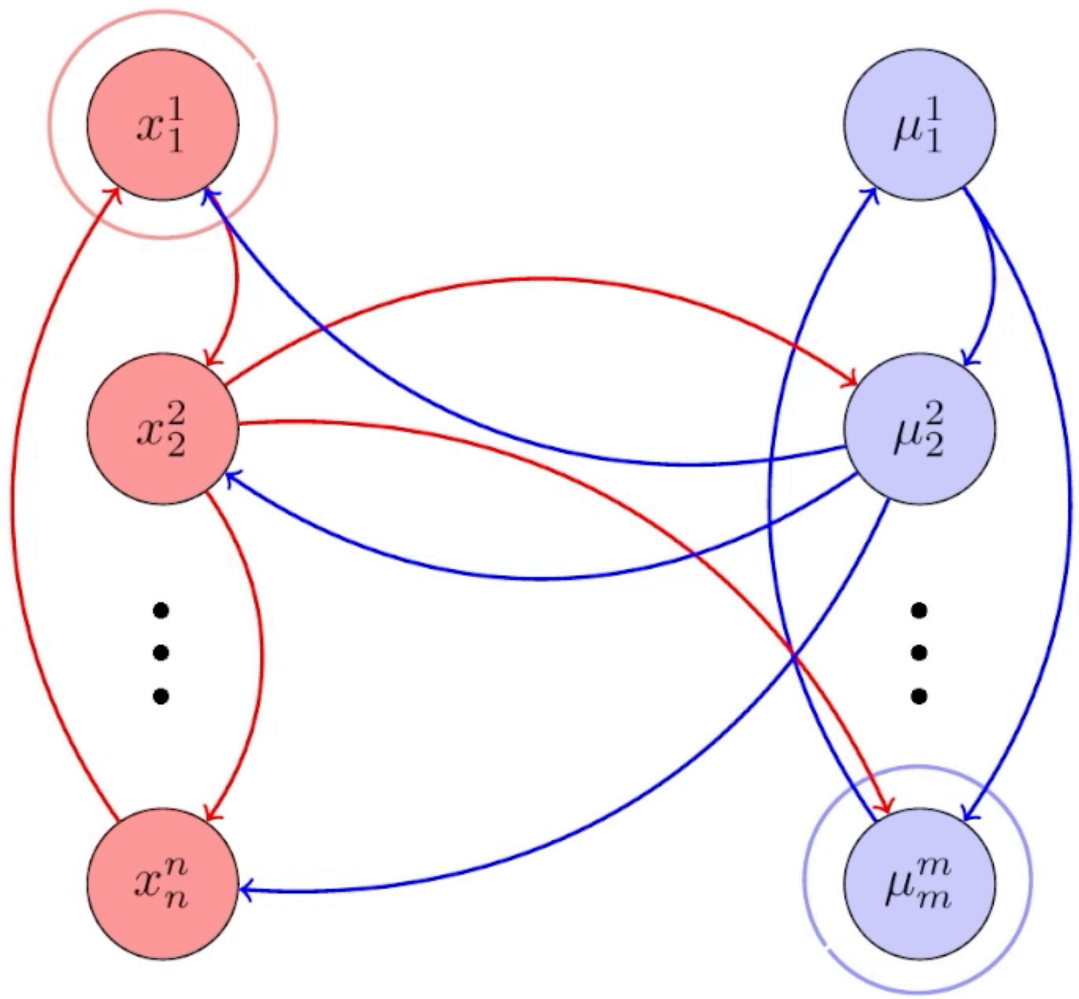
$$(x^{i}(k), \mu^{i}(k)) \neq (x^{j}(k), \mu^{j}(k))$$

Only one agent updates each decision variable

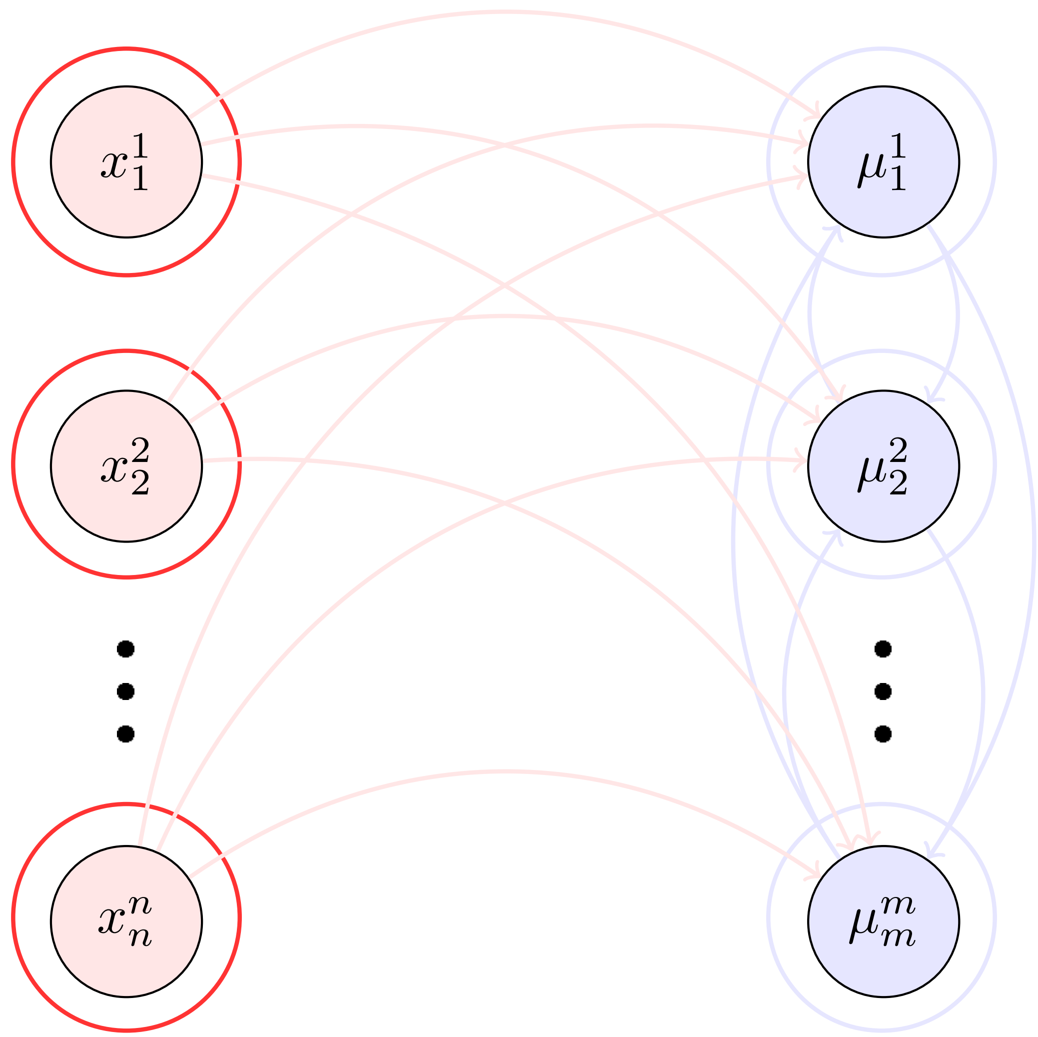
Updated & shared by agent i

$$(x^{i}(k), \mu^{i}(k)) = \left(\begin{bmatrix} x_1^i(k) \\ \vdots \\ x_i^i(k) \\ \vdots \\ x_n^i(k) \end{bmatrix}, \begin{bmatrix} \mu_1^i(k) \\ \vdots \\ \vdots \\ \vdots \\ \mu_n^i(k) \end{bmatrix} \right)$$

- Interactions look like this:

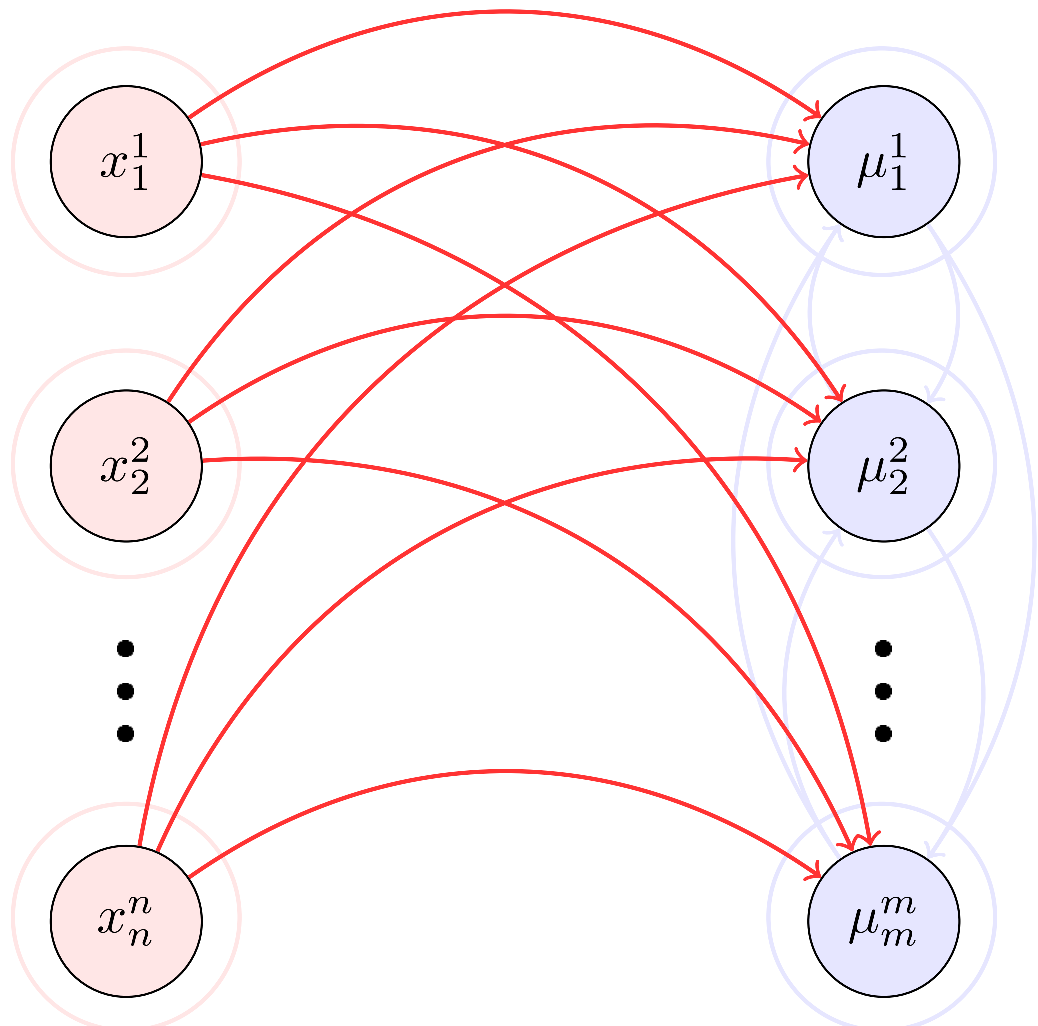


- Interactions look like this:



- The 4 types of asynchrony are:
  1. Computations of primal variables

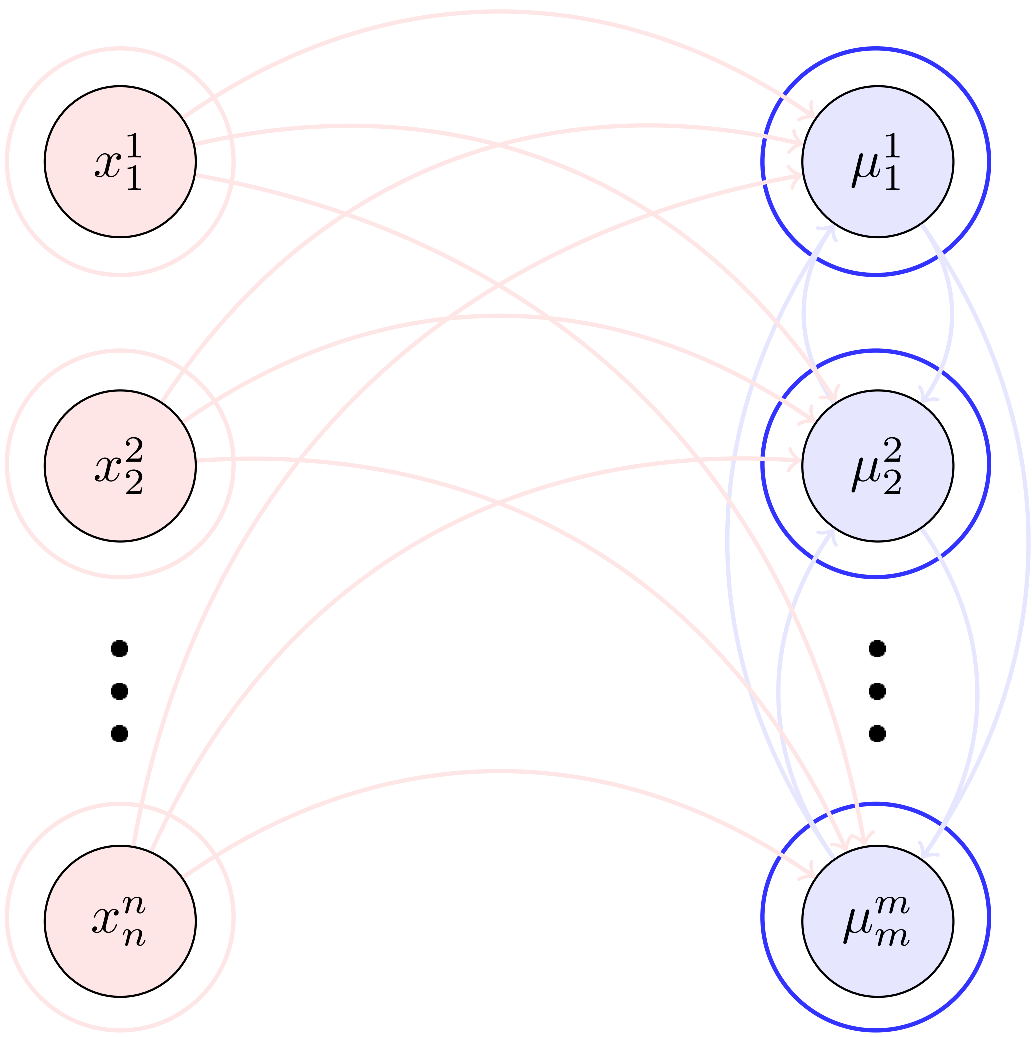
- Interactions look like this:



- The 4 types of asynchrony are:
  1. Computations of primal variables
  2. Communication of primal variables

Asynchrony appears in 4 forms

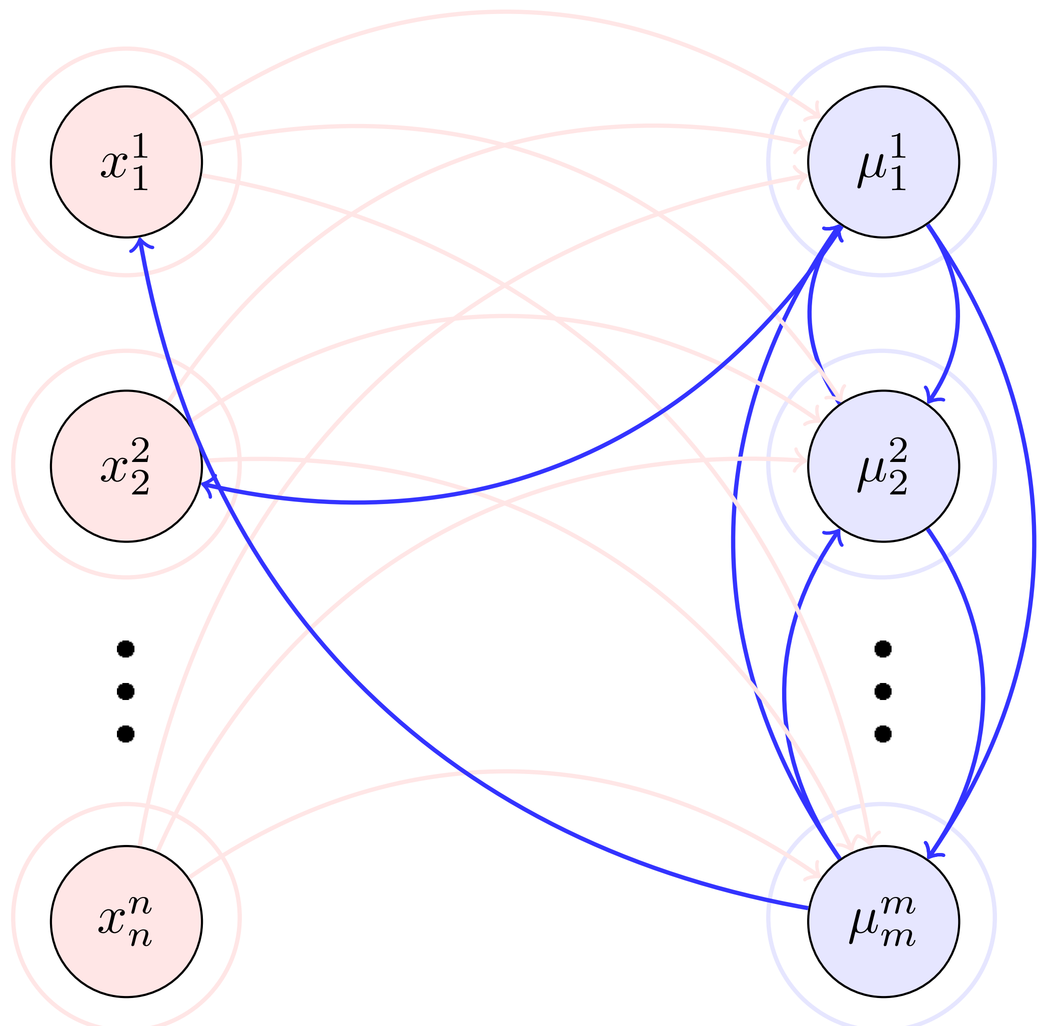
- Interactions look like this:



- The 4 types of asynchrony are:
  1. Computations of primal variables
  2. Communication of primal variables
  3. Computations of dual variables

Asynchrony appears in 4 forms

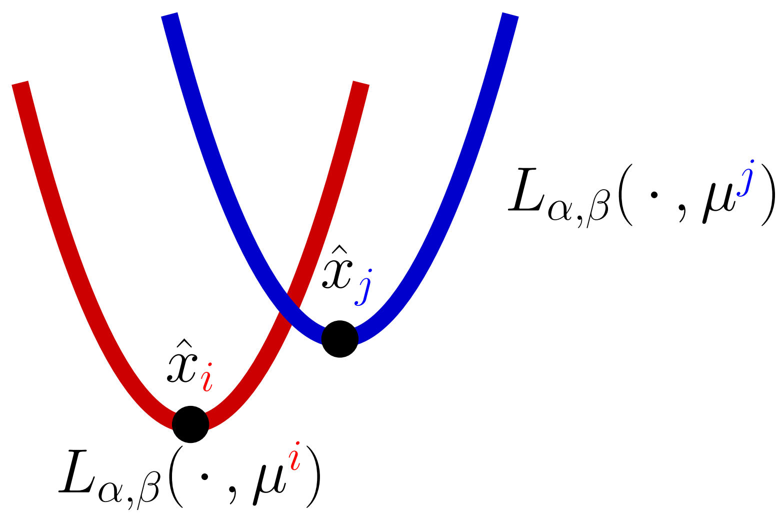
- Interactions look like this:



- The 4 types of asynchrony are:
  1. Computations of primal variables
  2. Communication of primal variables
  3. Computations of dual variables
  4. Communication of dual variables

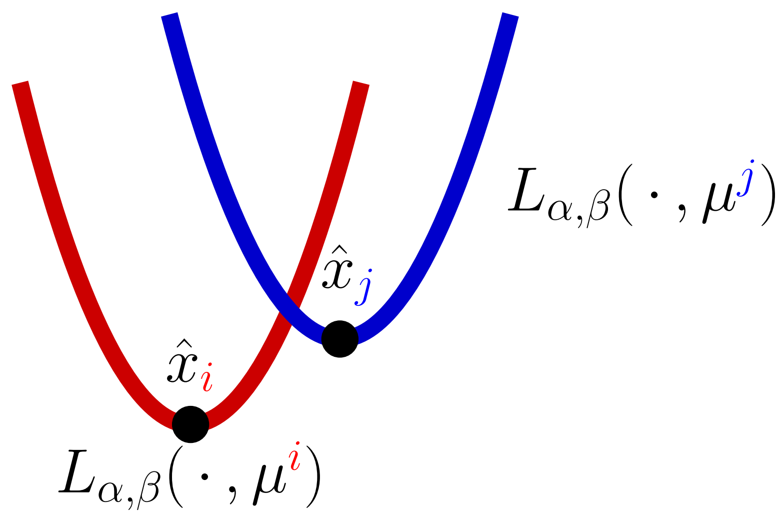
Asynchronous dual communications are problematic

- For $\mu^j \neq \mu^i$, agent i minimizes $L_{\alpha,\beta}(\,\cdot\,,\mu^i)$ but agent j minimizes $L_{\alpha,\beta}(\,\cdot\,,\mu^j)$



Asynchronous dual communications are problematic

- For $\mu^{j} \neq \mu^{i}$, agent i minimizes $L_{\alpha,\beta}(\cdot, \mu^{i})$ but agent j minimizes $L_{\alpha,\beta}(\cdot, \mu^{j})$



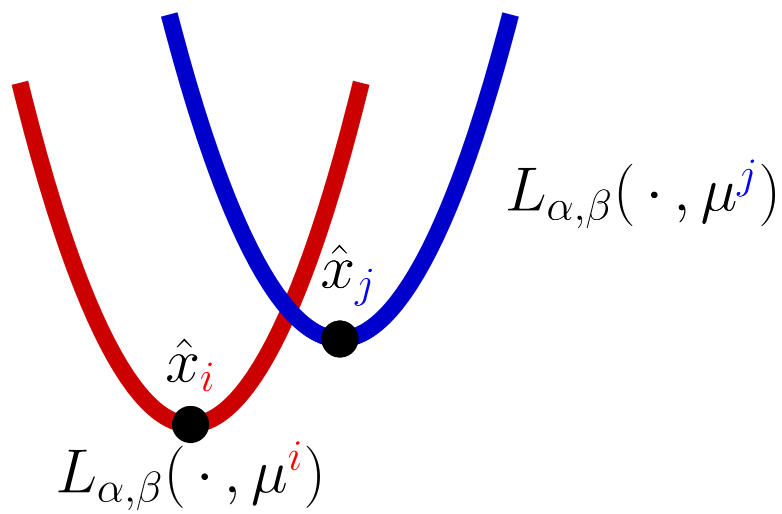
Theorem 1: Dual asynchrony stops convergence (Hendrickson&Hale, CDC2020)

Choose any $L > 0, \epsilon > 0$. Then there is a problem under our assumptions s.t.

1. $\|\mu^{i} - \mu^{j}\| < \epsilon$
2. $\|\hat{x}_i - \hat{x}_j\| > L$

Asynchronous dual communications are problematic

- For $\mu^j \neq \mu^i$, agent i minimizes $L_{\alpha,\beta}(\cdot, \mu^i)$ but agent j minimizes $L_{\alpha,\beta}(\cdot, \mu^j)$



Theorem 1: Dual asynchrony stops convergence (Hendrickson&Hale, CDC2020)

Choose any $L > 0, \epsilon > 0$. Then there is a problem under our assumptions s.t.

1. $\|\mu^i - \mu^j\| < \epsilon$
2. $\|\hat{x}_i - \hat{x}_j\| > L$

- This holds for a perfectly conditioned QP (with $\frac{\lambda_1(Q)}{\lambda_n(Q)} = 1$):

$$\text{minimize } \frac{1}{2}x^T Q x + r^T x \qquad \text{subject to } Ax \leq b$$

- **Takeaway: Agents *must* agree on μ.** Call it μ^p

Other asynchrony can be mitigated with update laws

- **Takeaway: Agents *must* agree on μ.** Call it μ^p

Primal update law

- For primal agent i, do

$$x_i^i(k+1) = \Pi_{X_i}\left[x_i^i(k) - \gamma \frac{\partial L_{\alpha,\beta}}{\partial x_i}\left(x^i(k), \mu^p(k)\right)\right]$$

- **Takeaway: Agents *must* agree on μ.** Call it μ^p

Primal update law

- For primal agent i, do

$$x_i^i(k+1) = \Pi_{X_i}\left[x_i^i(k) - \gamma \frac{\partial L_{\alpha,\beta}}{\partial x_i}\left(x^i(k), \mu^p(k)\right)\right]$$

$$x_j^i(k+1) = \begin{cases} x_j^j & x_j^j \text{ just received} \\ x_j^i(k) & \text{no message from agent } j \text{ received} \end{cases}$$

Other asynchrony can be mitigated with update laws

- **Takeaway: Agents *must* agree on μ.** Call it μ^p

Primal update law

- For primal agent i, do

$$x_i^i(k+1) = \Pi_{X_i}\left[x_i^i(k) - \gamma \frac{\partial L_{\alpha,\beta}}{\partial x_i}\left(x^i(k), \mu^p(k)\right)\right]$$

$$x_j^i(k+1) = \begin{cases} x_j^j & x_j^j \text{ just received} \\ x_j^i(k) & \text{no message from agent } j \text{ received} \end{cases}$$

$$\mu_\ell^p(k+1) = \begin{cases} \mu_\ell^\ell & \mu_\ell^\ell \text{ just received} \\ \mu_\ell^p(k) & \text{no message from dual agent } \ell \text{ just received} \end{cases}$$

Other asynchrony can be mitigated with update laws

- **Takeaway: Agents *must* agree on μ.** Call it μ^p

Primal update law

- For primal agent i, do

$$x_i^i(k+1) = \Pi_{X_i}\left[x_i^i(k) - \gamma \frac{\partial L_{\alpha,\beta}}{\partial x_i}\left(x^i(k), \mu^p(k)\right)\right]$$

$$x_j^i(k+1) = \begin{cases} x_j^j & x_j^j \text{ just received} \\ x_j^i(k) & \text{no message from agent } j \text{ received} \end{cases}$$

$$\mu_\ell^p(k+1) = \begin{cases} \mu_\ell^\ell & \mu_\ell^\ell \text{ just received} \\ \mu_\ell^p(k) & \text{no message from dual agent } \ell \text{ just received} \end{cases}$$

"Do gradient descent when you can with what you have"

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

Other asynchrony can be mitigated with update laws

- **Takeaway: Agents *must* agree on μ.** Call it μ^p

Primal update law

- For primal agent i, do

$$x_i^i(k+1) = \Pi_{X_i}\left[x_i^i(k) - \gamma \frac{\partial L_{\alpha,\beta}}{\partial x_i}\left(x^i(k), \mu^p(k)\right)\right]$$

$$x_j^i(k+1) = \begin{cases} x_j^j & x_j^j \text{ just received} \\ x_j^i(k) & \text{no message from agent } j \text{ received} \end{cases}$$

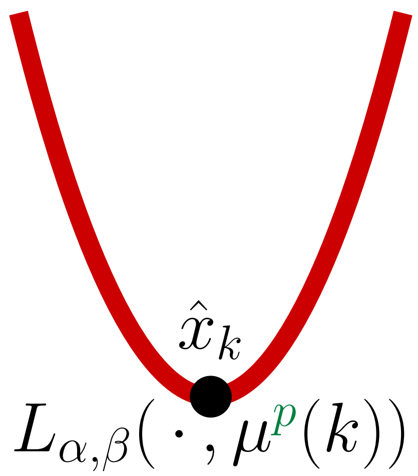
$$\mu_\ell^p(k+1) = \begin{cases} \mu_\ell^\ell & \mu_\ell^\ell \text{ just received} \\ \mu_\ell^p(k) & \text{no message from dual agent } \ell \text{ just received} \end{cases}$$

"Do gradient descent when you can with what you have"

- Dual agent ℓ is analogous, but with gradient *ascent* law

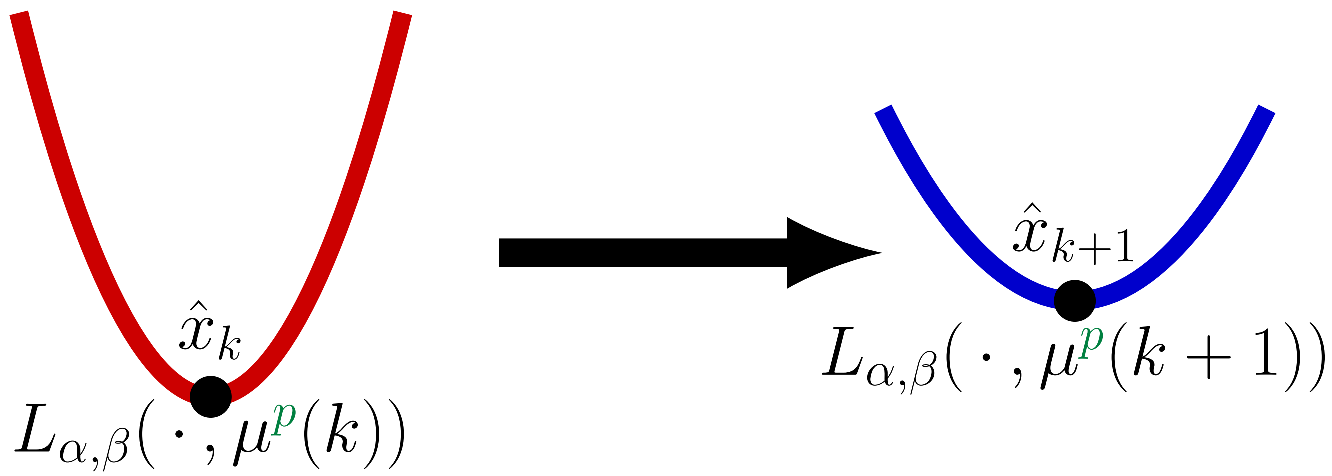
$$\mu_\ell^\ell(k+1) = \Pi_{\mathbb{R}_+^{m_i}}\left[\mu_\ell^\ell(k) + \gamma \frac{\partial L}{\partial \mu_\ell}\left(\mu^\ell(k), x^\ell(k)\right)\right]$$

- Given $\mu^{p}(k)$, all primal agents minimize $L_{\alpha,\beta}\big(\cdot\,,\mu^{p}(k)\big)$



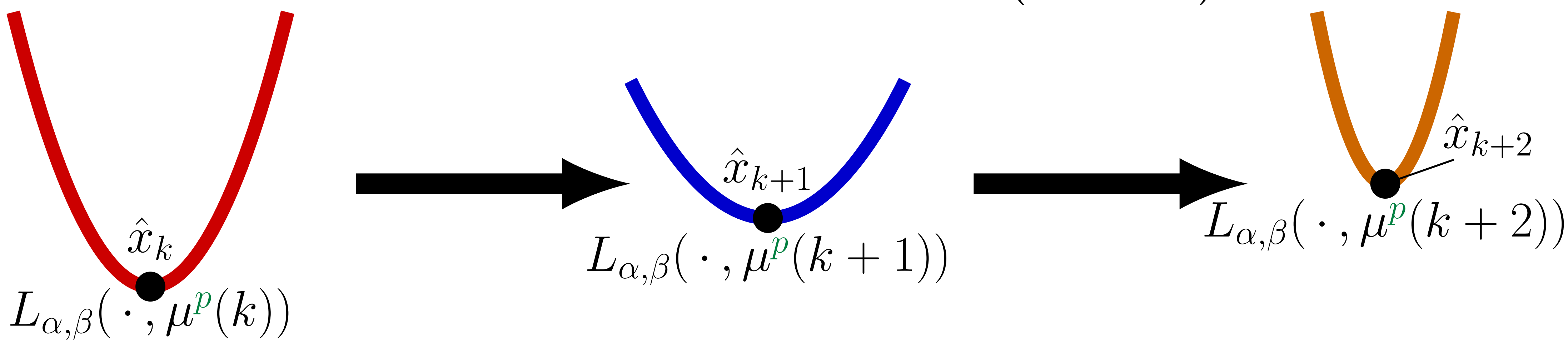
Regularized geometry helps ensure convergence

- Given $\mu^p(k)$, all primal agents minimize $L_{\alpha,\beta}\big(\,\cdot\,, \mu^p(k)\big)$



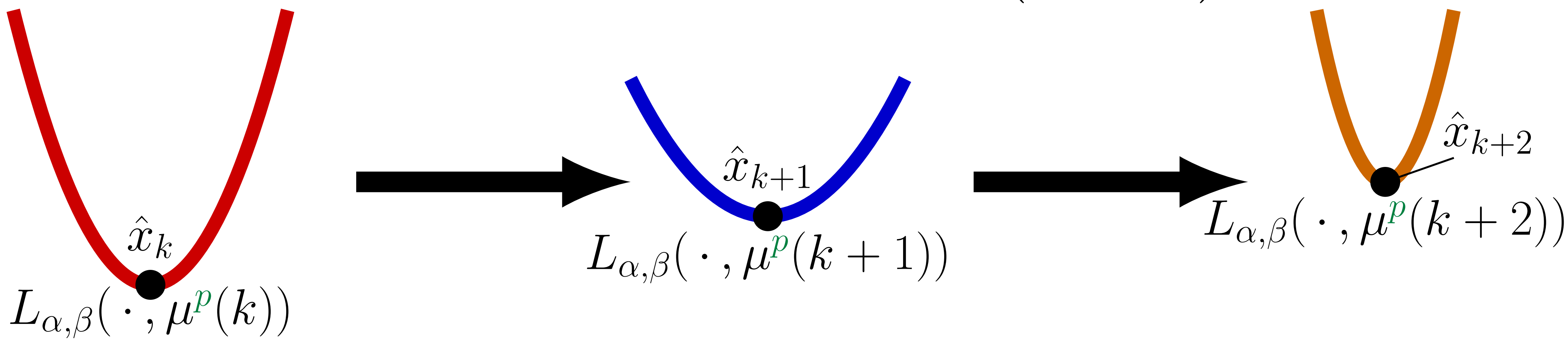
Regularized geometry helps ensure convergence

- Given $\mu^{p}(k)$, all primal agents minimize $L_{\alpha,\beta}\big(\,\cdot\,, \mu^{p}(k)\big)$



Regularized geometry helps ensure convergence

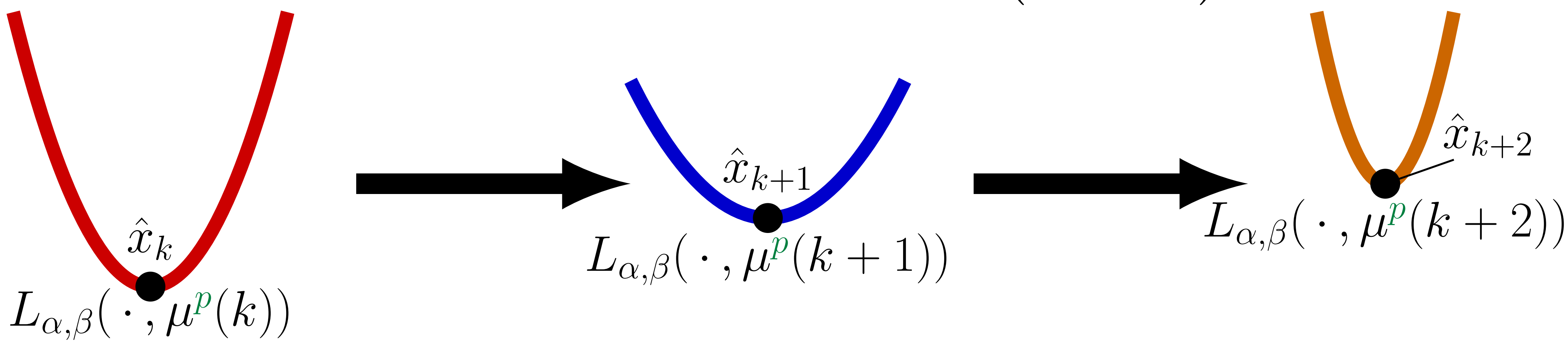
- Given $\mu^p(k)$, all primal agents minimize $L_{\alpha,\beta}\big(\cdot\,,\mu^p(k)\big)$



- Since $\alpha > 0$, agents at worst slide along level curves of $L_{\alpha,\beta}\big(\cdot\,,\mu^p(k)\big)$

Regularized geometry helps ensure convergence

- Given $\mu^p(k)$, all primal agents minimize $L_{\alpha,\beta}\big(\,\cdot\,, \mu^p(k)\big)$



- Since $\alpha > 0$, agents at worst slide along level curves of $L_{\alpha,\beta}\big(\,\cdot\,, \mu^p(k)\big)$

Theorem 2: Dual convergence (Hendrickson & Hale CDC2020)

Using $\beta > 0$ lets us stitch together the above progress:

$$\|\mu(k+t) - \hat{\mu}_{\alpha,\beta}\|^2 \leq \underbrace{\sum_{j=1}^{N_d} q^{c_j(t)} \|\mu_j(k) - \hat{\mu}_{\alpha,\beta,j}\|^2}_{\text{Convergence of each block}} + \underbrace{\sum_{i=1}^{\max_j c_j(t)} q^{i-1} K_i}_{\text{Penalty due to asynchrony}} \,,$$

where $q \in (0,1)$ and $c_j(t)$ counts updates to μ_j in last t timesteps.

We also have primal convergence

- Define the "omniscient iterate"

$$x(k) = \left(x_1^1(k)^T, x_2^2(k)^T, \dots, x_n^n(k)^T\right)^T$$

We also have primal convergence

- Define the "omniscient iterate"

$$x(k) = \left(x_1^1(k)^T, x_2^2(k)^T, \dots, x_n^n(k)^T\right)^T$$

Theorem 3: Primal Convergence (Hendrickson & Hale, In preparation)

The distributed asynchronous primal-dual algorithm converges according to

$$\|x(k) - \hat{x}_{\alpha,\beta}\|^2 \leq C_1 q^{\text{ops(k)}} + C_2 \underbrace{\|\mu(k) - \hat{\mu}_{\alpha,\beta\|}\|}_{\text{Rate from last slide}}$$

for $q \in (0,1)$ and ops(k) the # of operations completed with $\mu^p(k)$ onboard

- Define the "omniscient iterate"

$$x(k) = \left(x_1^1(k)^T, x_2^2(k)^T, \ldots, x_n^n(k)^T\right)^T$$

Theorem 3: Primal Convergence (Hendrickson & Hale, In preparation)

The distributed asynchronous primal-dual algorithm converges according to

$$\|x(k) - \hat{x}_{\alpha,\beta}\|^2 \leq C_1 q^{\text{ops(k)}} + C_2 \underbrace{\|\mu(k) - \hat{\mu}_{\alpha,\beta}\|}_{\text{Rate from last slide}}$$

for $q \in (0,1)$ and ops(k) the # of operations completed with $\mu^p(k)$ onboard

- There is a fundamental principle underlying these results

- Define the "omniscient iterate"

$$x(k) = \left(x_1^1(k)^T, x_2^2(k)^T, \ldots, x_n^n(k)^T\right)^T$$

Theorem 3: Primal Convergence (Hendrickson & Hale, In preparation)

The distributed asynchronous primal-dual algorithm converges according to

$$\|x(k) - \hat{x}_{\alpha,\beta}\|^2 \leq C_1 q^{\text{ops(k)}} + C_2 \underbrace{\|\mu(k) - \hat{\mu}_{\alpha,\beta\|}\|}_{\text{Rate from last slide}}$$

for $q \in (0,1)$ and ops(k) the # of operations completed with $\mu^p(k)$ onboard

- There is a fundamental principle underlying these results
- (1989) Without $g(x) \leq 0$: faster computations *always* converge faster (Bertsekas & Tsitsiklis, 1989)

gorithms send the same number of messages and the messages have the same delays. We may conclude that, in the case of monotone iterations, it is preferable to perform as many updates as possible even if they are based on outdated information and, therefore, asynchronous algorithms are advantageous.

We also have primal convergence

- Define the "omniscient iterate"

$$x(k) = \left(x_1^1(k)^T, x_2^2(k)^T, \ldots, x_n^n(k)^T\right)^T$$

Theorem 3: Primal Convergence (Hendrickson & Hale, In preparation)

The distributed asynchronous primal-dual algorithm converges according to

$$\|x(k) - \hat{x}_{\alpha,\beta}\|^2 \leq C_1 q^{\text{ops(k)}} + C_2 \underbrace{\|\mu(k) - \hat{\mu}_{\alpha,\beta\|}\|}_{\text{Rate from last slide}}$$

for $q \in (0,1)$ and ops(k) the # of operations completed with $\mu^p(k)$ onboard

- There is a fundamental principle underlying these results
- (1989) Without $g(x) \leq 0$: faster computations *always* converge faster (Bertsekas & Tsitsiklis, 1989)

> the messages have the same delays. We may conclude that, in the case of monotone iterations, it is preferable to perform as many updates as possible even if they are based on outdated information and, therefore, asynchronous algorithms are advantageous.

- (2020) With $g(x) \leq 0$: faster dual updates can slow convergence down!

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

- Consider $n = 10$ agents solving the problem

$$\text{minimize } f(x) = \sum_{i=1}^{10} x_i^4 + \frac{1}{20}\sum_{i=1}^{10}\sum_{\substack{j=1 \\ j\neq i}}^{n}(x_i - x_j)^2$$

$$\text{subject to } Ax \leq b \text{ and } x \in [1, 10]^{10}$$

Convergence is predictably non-monotone

- Consider $n = 10$ agents solving the problem

$$\text{minimize } f(x) = \sum_{i=1}^{10} x_i^4 + \frac{1}{20} \sum_{i=1}^{10} \sum_{\substack{j=1 \\ j \neq i}}^{n} (x_i - x_j)^2$$

$$\text{subject to } Ax \leq b \text{ and } x \in [1, 10]^{10}$$

- Agents have a 25% chance of communicating at each time
- Set $\alpha = \beta = 0.001$

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

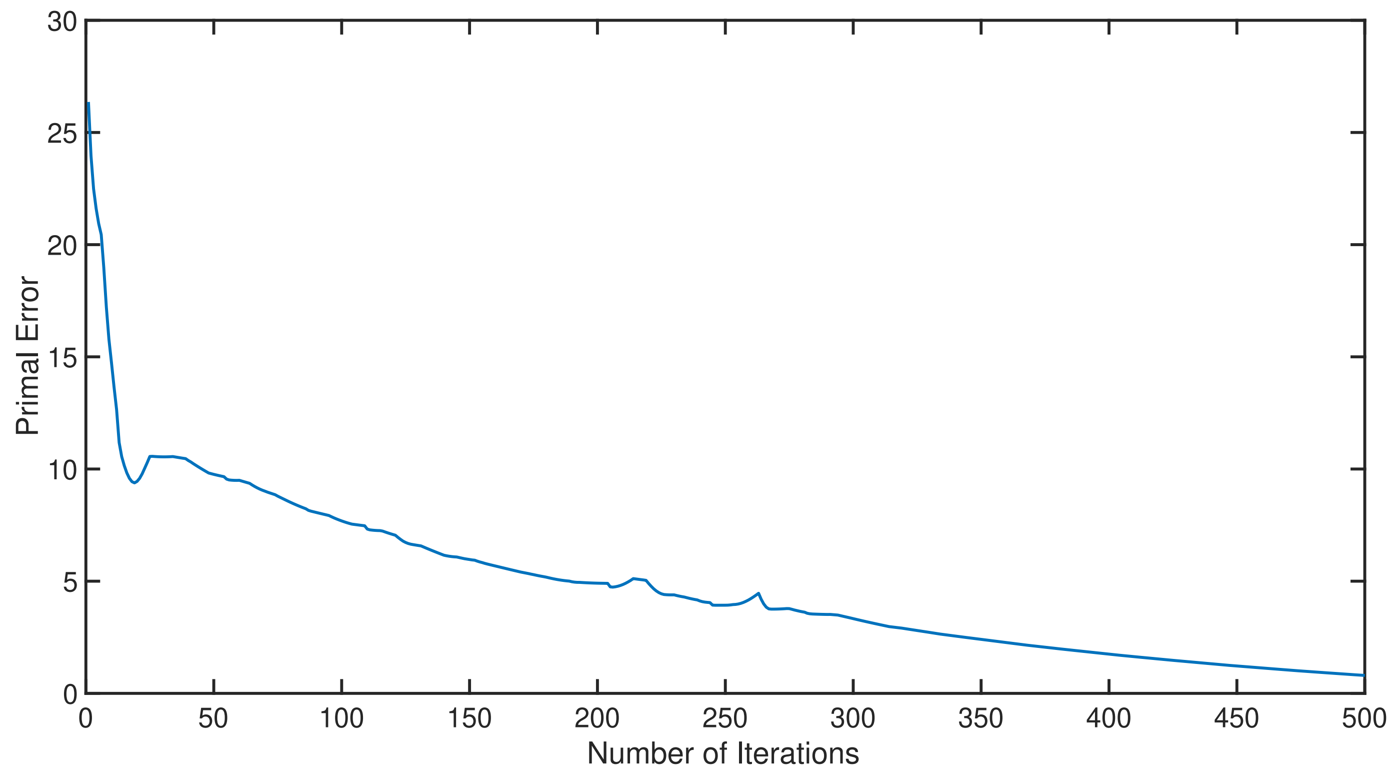
Convergence is predictably non-monotone

- Consider $n = 10$ agents solving the problem

$$\text{minimize } f(x) = \sum_{i=1}^{10} x_i^4 + \frac{1}{20} \sum_{i=1}^{10} \sum_{\substack{j=1 \\ j \neq i}}^{n} (x_i - x_j)^2$$

$$\text{subject to } Ax \leq b \text{ and } x \in [1, 10]^{10}$$

- Agents have a 25% chance of communicating at each time
- Set $\alpha = \beta = 0.001$



Thank you