Model-Free Optimization and Learning for Multi-Agent Systems

Yujie Tang

Department of Industrial Engineering & Management

College of Engineering

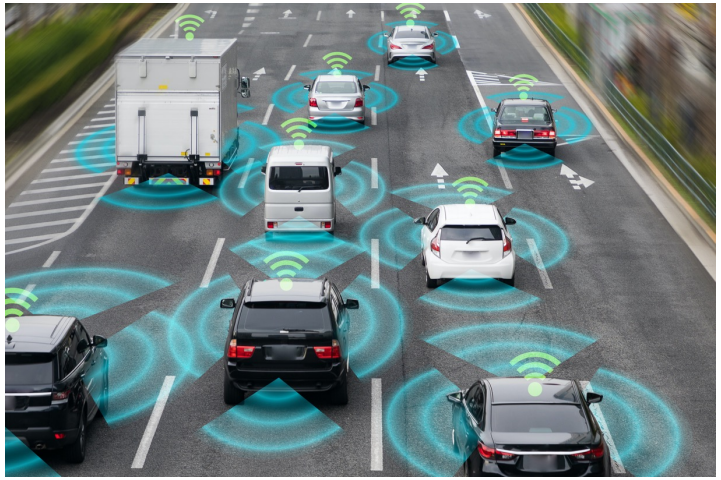
PEKING UNIVERSITY

Acknowledgement:

Na Li, Zhaolin Ren, Yingying Li, Runyu Zhang

Jie Song, Ruiyang Jin

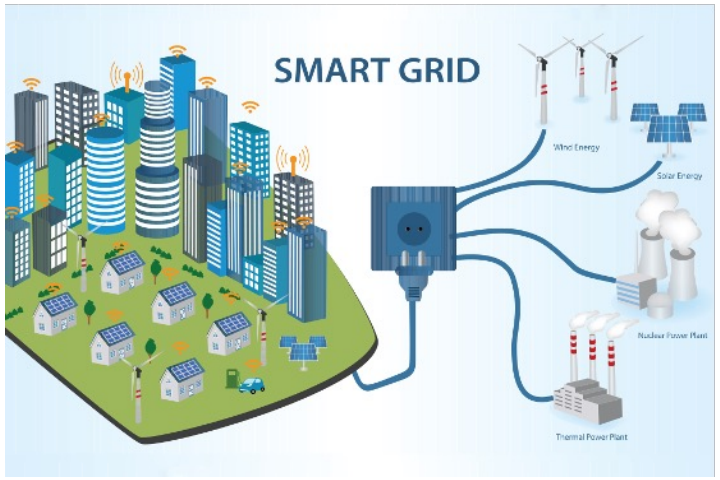
Optimization and Control of Multi-Agent Systems



Transportation

Swarm robotics

Smart manufacturing



Smart grid

Opportunities and Challenges

Smart Grid

- Smart meters/sensors, mobile devices, communication networks
- Large system, complicated mechanism, hard to model
- Large numbers of heterogenous devices

Opportunities:

- Abundant, real-time data
- Computing power, fast communication

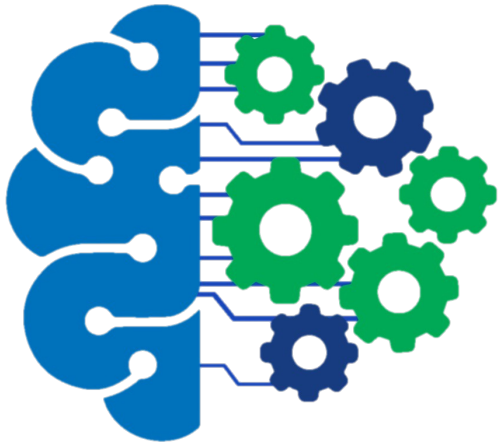
Challenges:

- Unknown system model
- Coordination of agents
- Rigorous performance guarantees
- …

Opportunities and Challenges

Data, computing power, and communication

How?



Smarter decision-making for industrial & societal systems

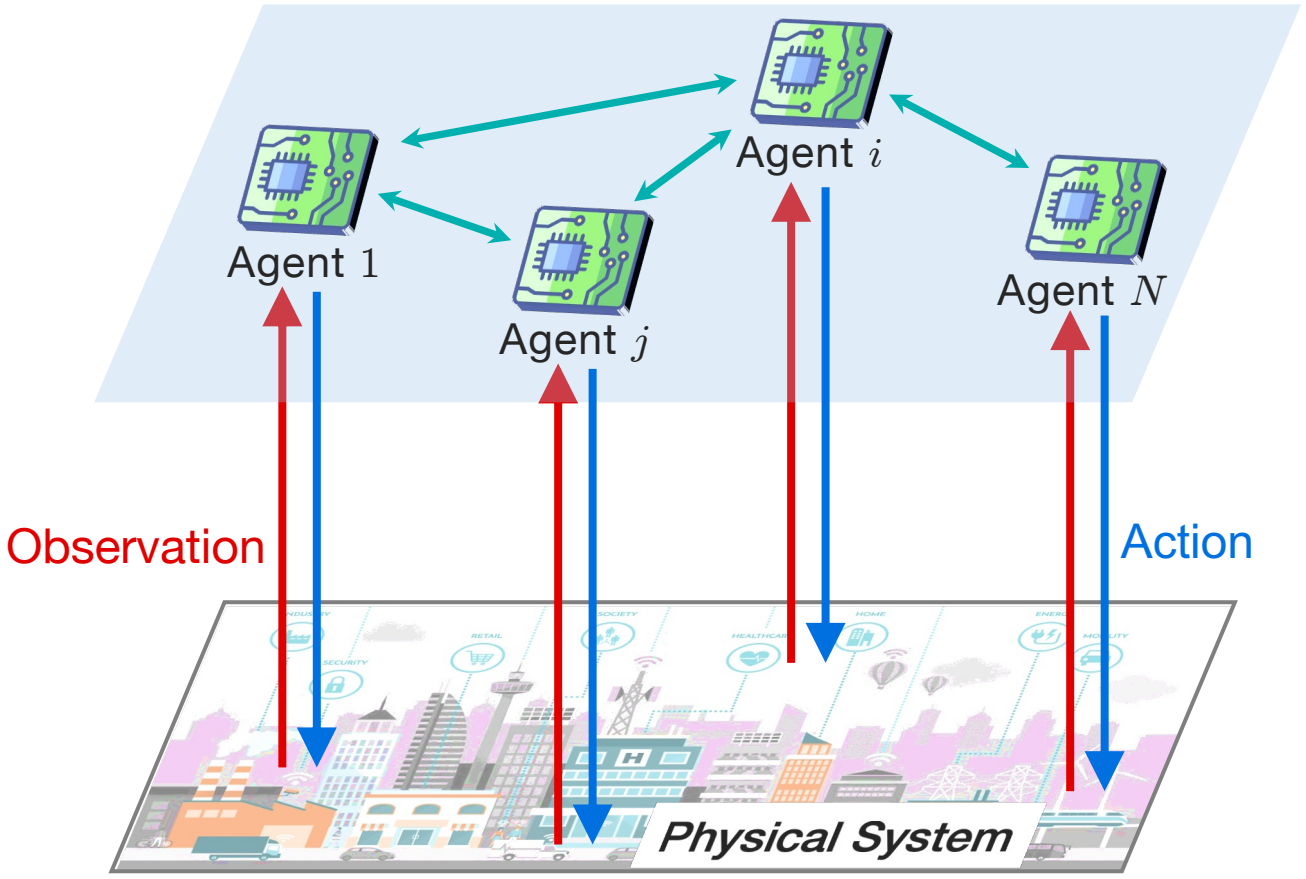
Opportunities:

- Abundant, real-time data
- Computing power, fast communication

Challenges:

- Unknown system model
- Coordination of agents
- Rigorous performance guarantees
- …

This Talk



Distributed optimization/RL algorithms

- **Model-free & data-driven**
- **Coordinate** a large number of agents
- **Theoretical performance guarantees**
  - Sample complexity
  - Scalability
  - Stability

Zeroth-order optimization + **Decentralized coordination**

Preliminaries on Zeroth-Order Optimization

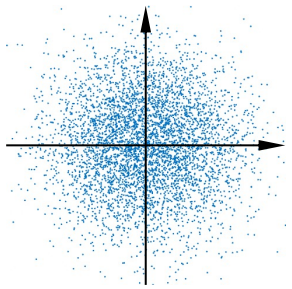
$$\min_{x \in \mathbb{R}^d} \; f(x)$$

- Gradient unknown
- Can only query its values (zeroth-order info)

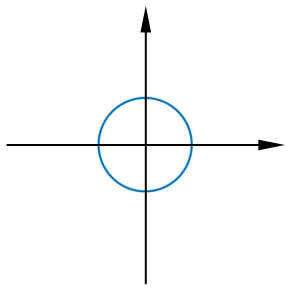
One approach: Estimate gradient from zeroth-order information

$$\mathsf{G}_f(x; r, z) = \frac{d}{2r}(f(x + rz) - f(x - rz))z$$

- z : random perturbation
- r : smoothing radius



$\mathcal{N}(0, I/d)$



$\mathrm{Unif}(\mathbb{S}_{d-1})$

Lemma. If f is L-smooth, then

$$\mathbb{E}[\mathsf{G}_f(x; r, z)] = \nabla f(x) + O(r)$$

✓ A stochastic gradient with nonzero (but controllable) bias

[Flaxman 2005] [Nesterov 2017]

Preliminaries on Zeroth-Order Optimization

$$\min_{x \in \mathbb{R}^d} \; f(x)$$

- Gradient unknown
- Can only query its values (zeroth-order info)

One approach: Estimate gradient from zeroth-order information

$$\mathsf{G}_f(x; r, z) = \frac{d}{2r}(f(x + rz) - f(x - rz))z$$

- Two-point gradient estimator
- Other commonly used zeroth-order gradient estimators:

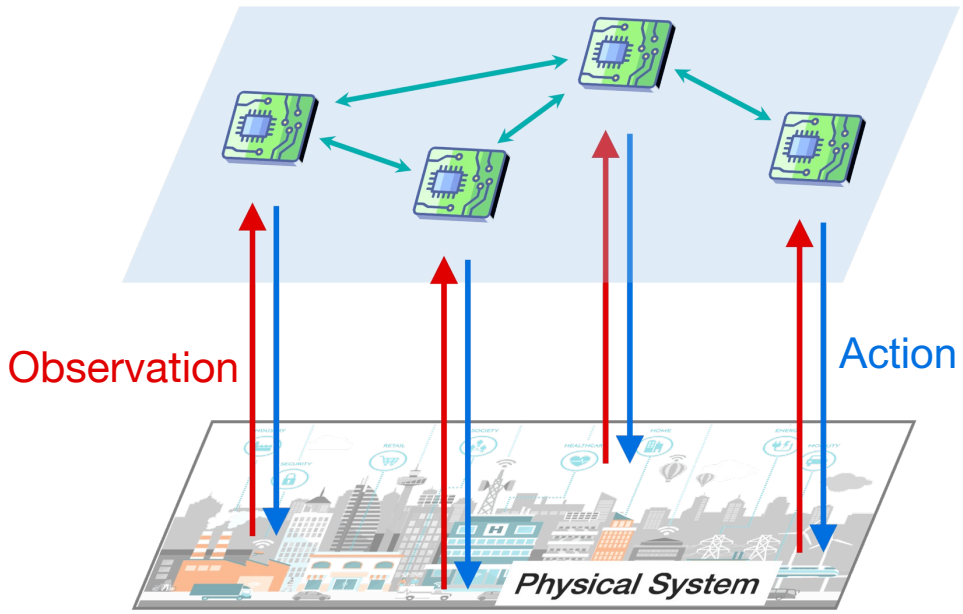
$$\frac{d}{r}\left(f(x+rz) - f(x)\right)z \quad \text{(two-point)} \qquad \frac{d}{r} f(x + rz)z \quad \text{(single-point)}$$

- Stochastic gradient descent + zeroth-order gradient estimation

$$x(k+1) = x(k) - \eta \cdot \mathsf{G}_f(x(k); r, z(k))$$

[Flaxman 2005] [Nesterov 2017]

Model-Free Optimization and Learning for Multi-Agent Systems

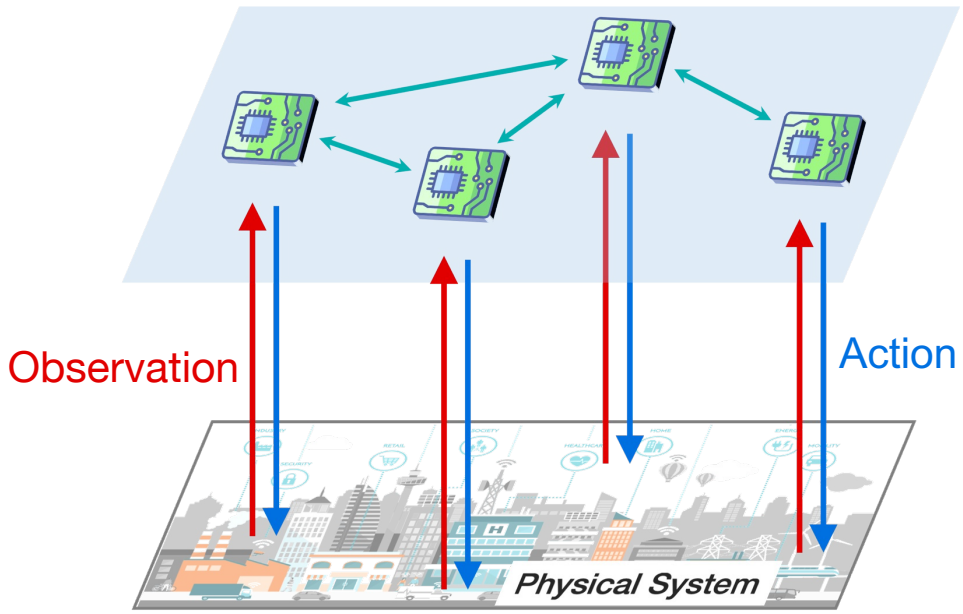


Zeroth-order optimization + Decentralized coordination

I. Multi-agent feedback optimization

II. Distributed reinforcement learning

Model-Free Optimization and Learning for Multi-Agent Systems

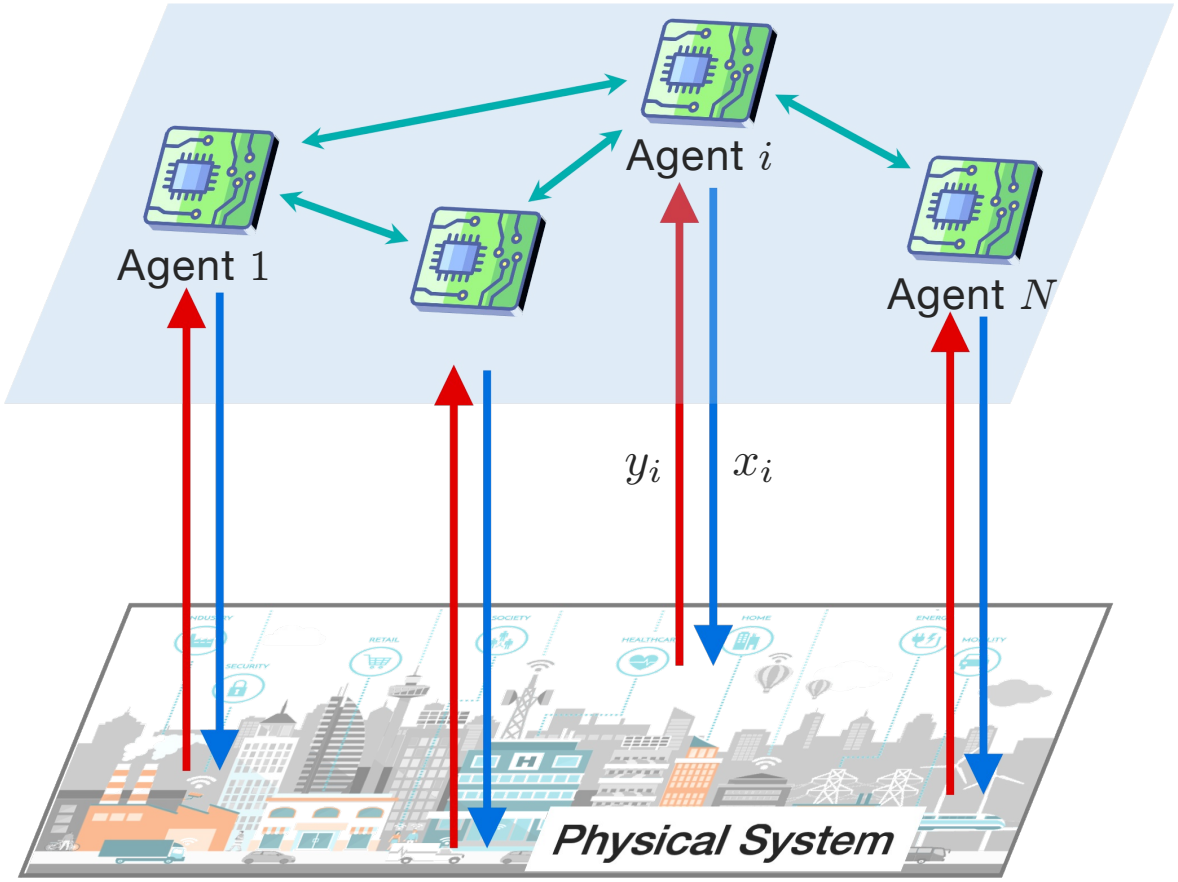


Zeroth-order optimization + Decentralized coordination

I. Multi-agent feedback optimization

II. Distributed reinforcement learning

Zeroth-Order Feedback Optimization for Cooperative Multi-Agent Systems



Static system: $y_i = f_i(x_1, \ldots, x_N)$

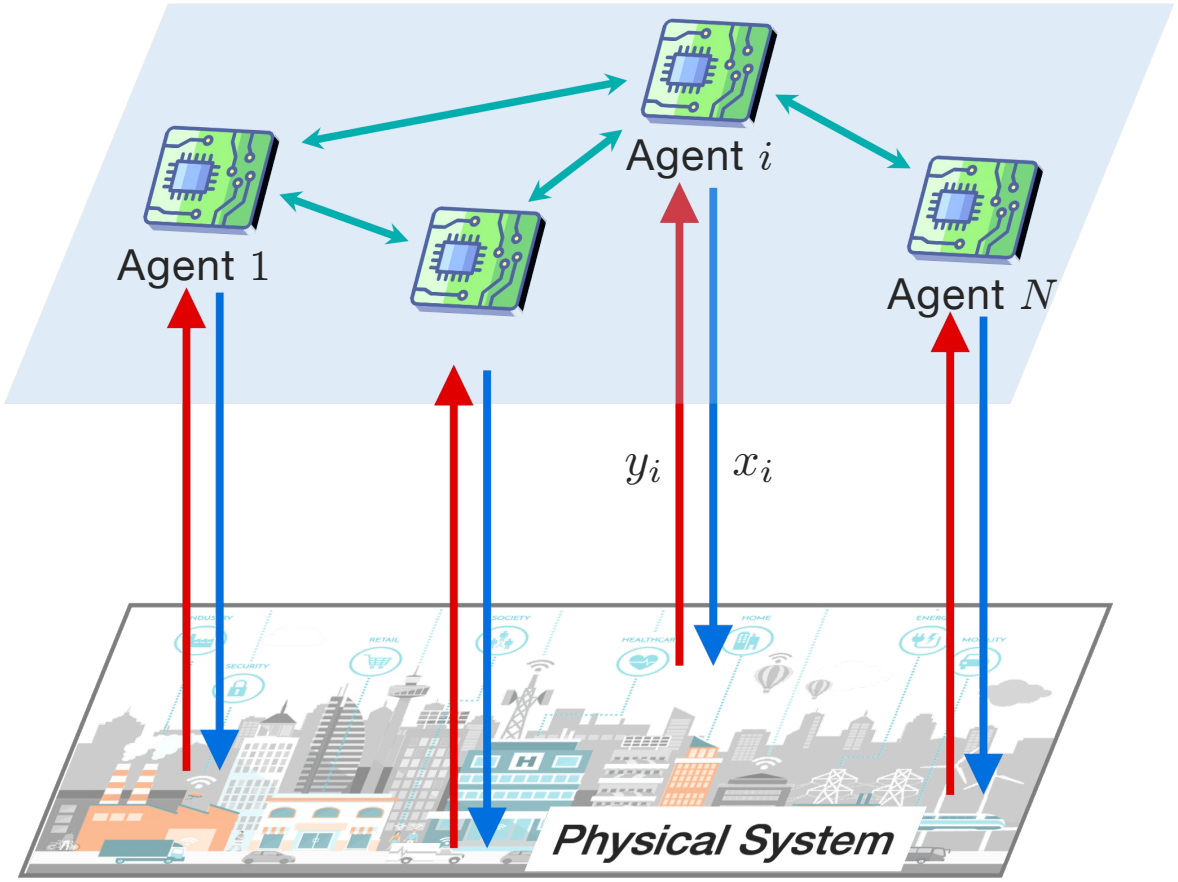
- **Local action:** $x_i \in \mathcal{X}_i$
- **Local observation:** $y_i = f_i(x_1, \ldots, x_N)$
- **Goal:**

  minimize $f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x_1, \ldots, x_N)$
- **Gradient unknown**
- **Communication network**

Agent i's action $x_i(k)$ purely based on

- Historical observations of local costs
- Information exchanged with neighbors in the network

Zeroth-Order Feedback Optimization for Cooperative Multi-Agent Systems



Static system: $y_i = f_i(x_1, \ldots, x_N)$

Comparison with consensus optimization

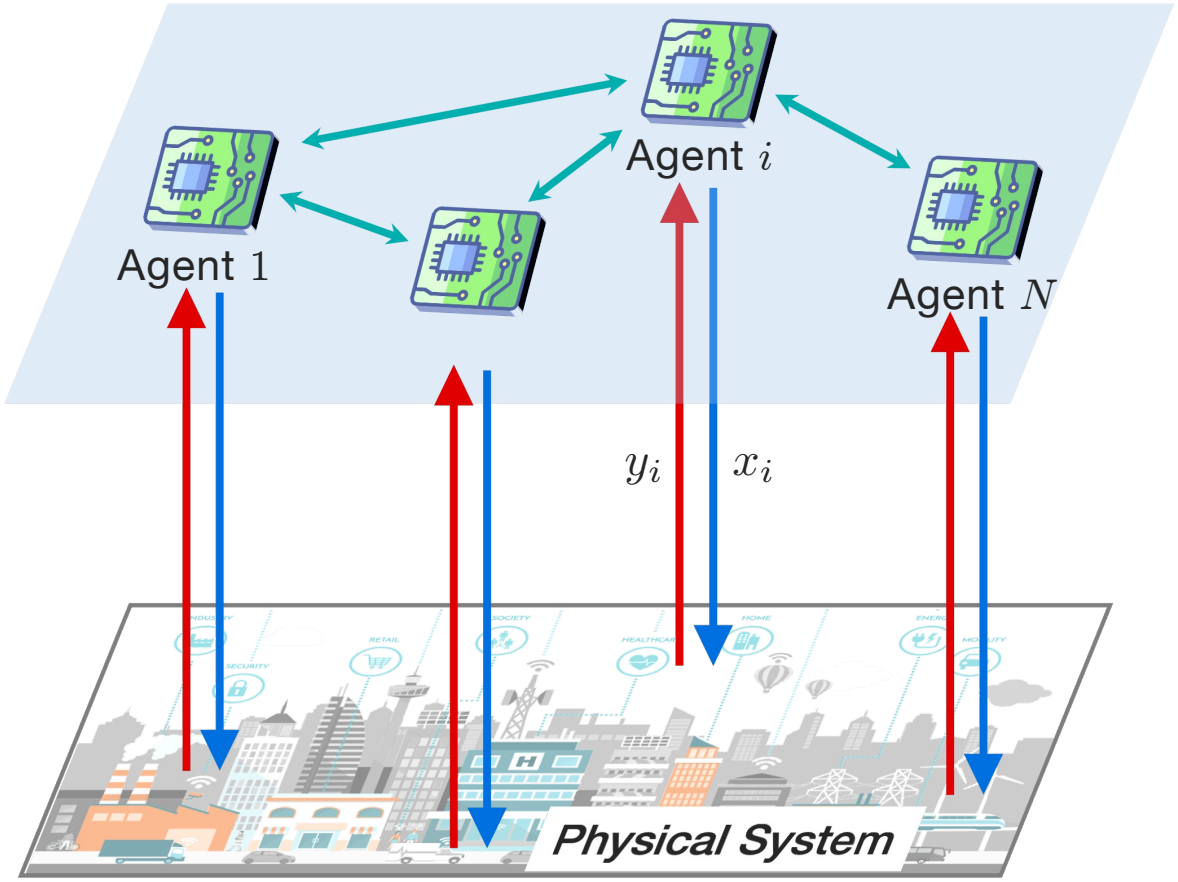
Consensus optimization:

$$\min_{x_1,\ldots,x_N} \quad \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$$
$$\text{s.t.} \quad x_i = x_j, \quad \forall i, j$$

Multi-agent feedback optimization:

$$\min_{x_1,\ldots,x_N} \quad \frac{1}{N}\sum_{i=1}^{N} f_i(x_1, \ldots, x_N)$$
$$\text{s.t.} \quad x_i \in \mathcal{X}_i$$

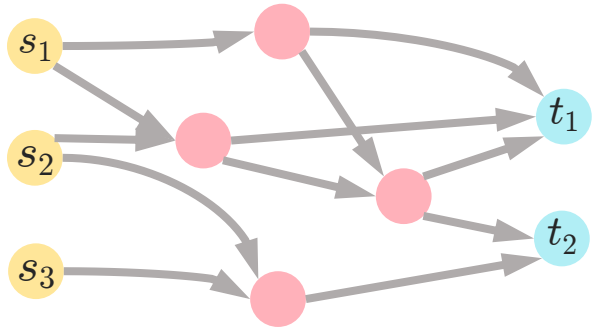
Zeroth-Order Feedback Optimization for Cooperative Multi-Agent Systems



Static system: $y_i = f_i(x_1, \ldots, x_N)$

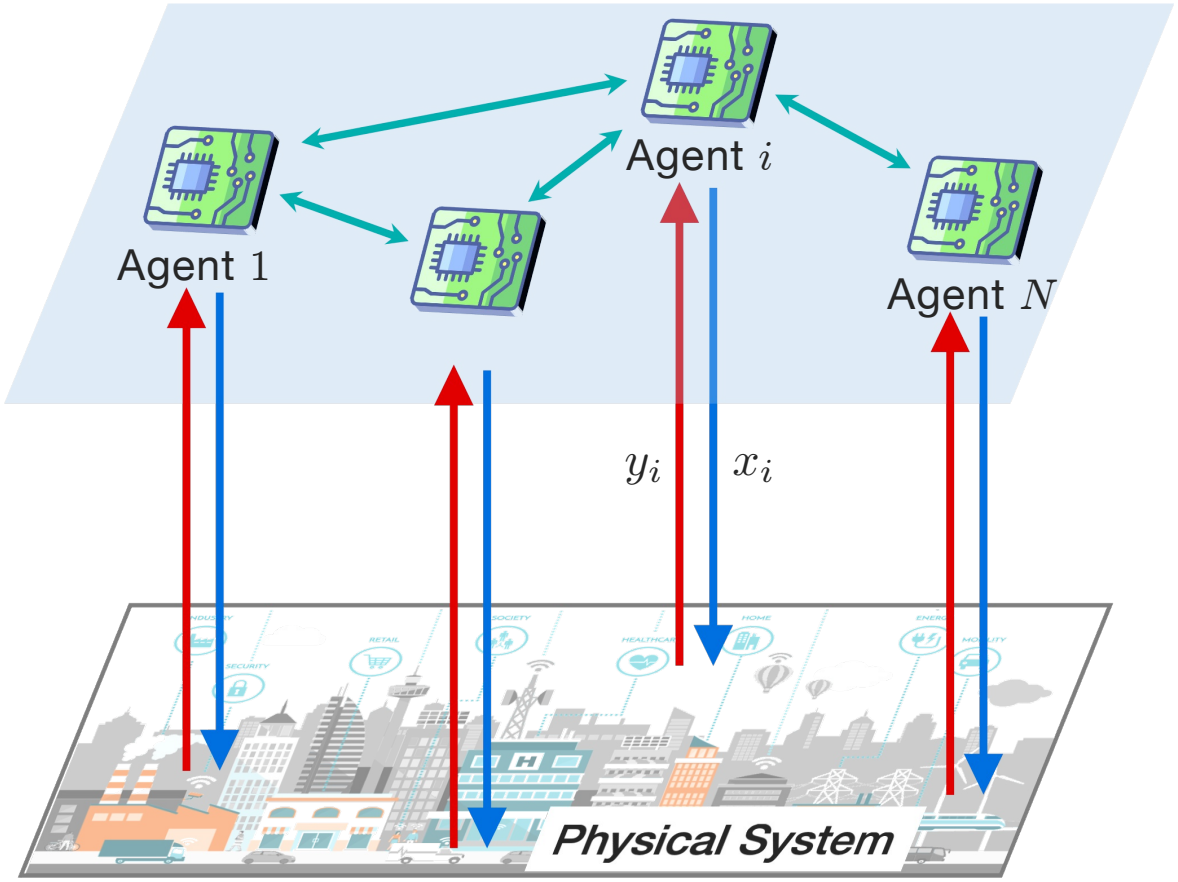
Examples:

- Distributed routing control



- Wind farm power maximization

Zeroth-Order Feedback Optimization for Cooperative Multi-Agent Systems



Static system: $y_i = f_i(x_1, \ldots, x_N)$

Related works

- [Marden 2013] [Menon 2013] [Menon 2014] [Dougherty 2016]
- Discrete action space/continuous-time algorithm
- Weak guarantees, no sample complexity result

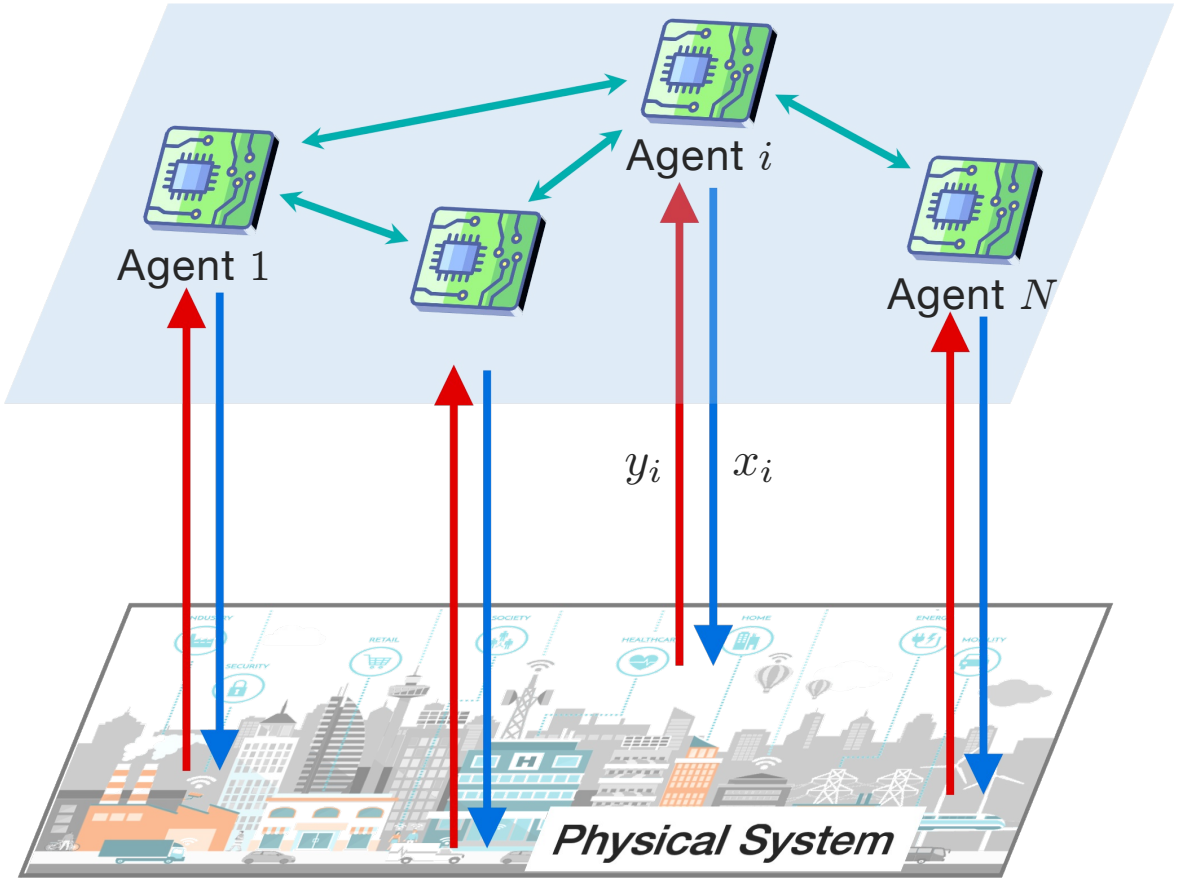
Our algorithm

✓ Continuous action space, discrete-time algorithm

✓ Detailed complexity analysis, stronger guarantees

Tang, Ren & Li. *Zeroth-order feedback optimization for cooperative multi-agent systems*. Automatica 2022

Algorithm Design



Static system: $y_i = f_i(x_1, \ldots, x_N)$

Basic ingredients:

- Zeroth-order gradient estimation:

$$x_i(k+1) = x_i(k) - \eta\, G_i(k)$$

$$G_i(k) = \frac{d}{r}\left(\frac{1}{N}\sum_{j=1}^{N} D_j(k)\right) z_i(k)$$

Partial gradient estimator

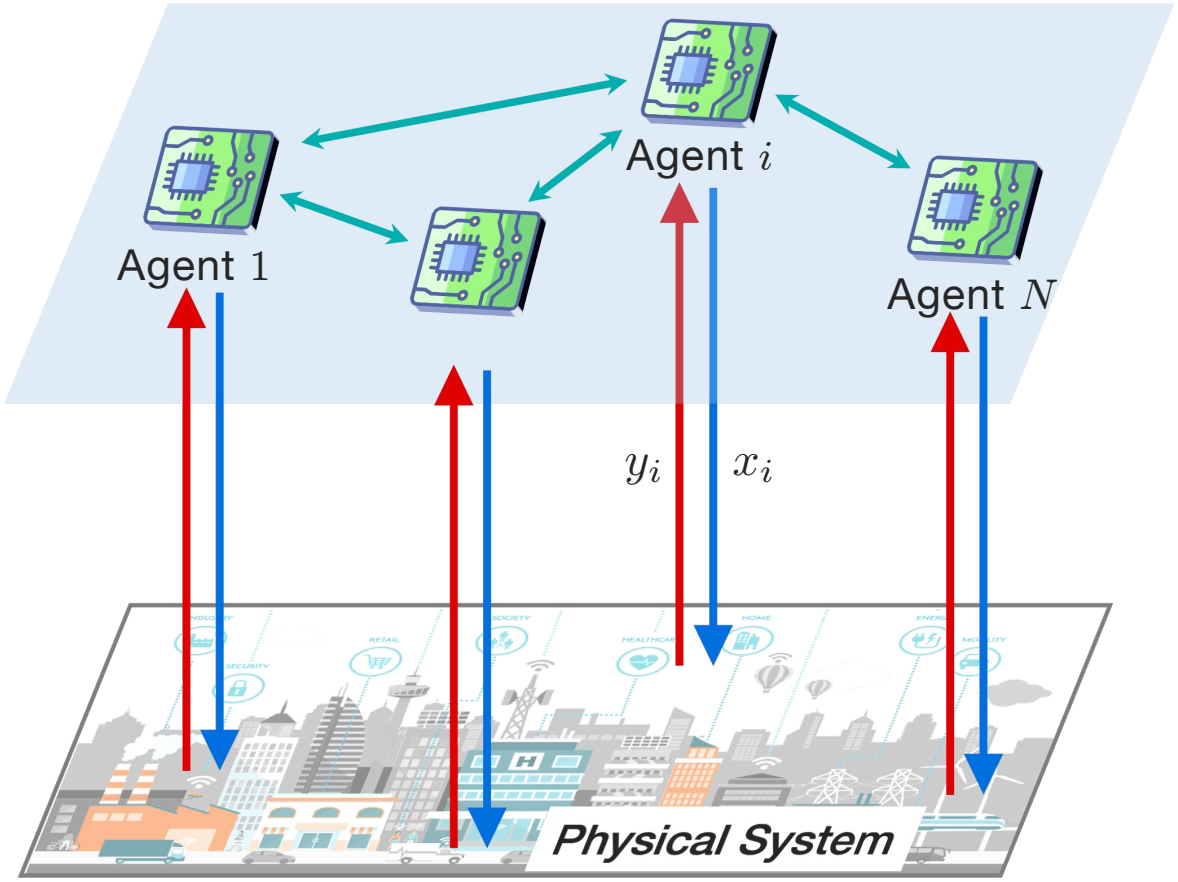
$$D_j(k) = f_j(x(k)+rz(k)) - f_j(x(k))$$

Agent j's cost difference

- Exchange info with neighbors so that agent i can get the most up-to-date value of D_j for $j \neq i$
  - Note: Delay is inevitable!

Tang, Ren & Li. ***Zeroth-order feedback optimization for cooperative multi-agent systems***. Automatica 2022

Algorithm Design



Static system: $y_i = f_i(x_1, \ldots, x_N)$

Basic ingredients:

- Zeroth-order gradient estimation:

$$x_i(k+1) = x_i(k) - \eta\, G_i(k)$$

Partial gradient estimator with **delayed** info

$$G_i(k) = \frac{d}{r}\left(\frac{1}{N}\sum\nolimits_{j=1}^{N} D_j(k-b_{ij})\, z_i(k-b_{ij})\right)$$

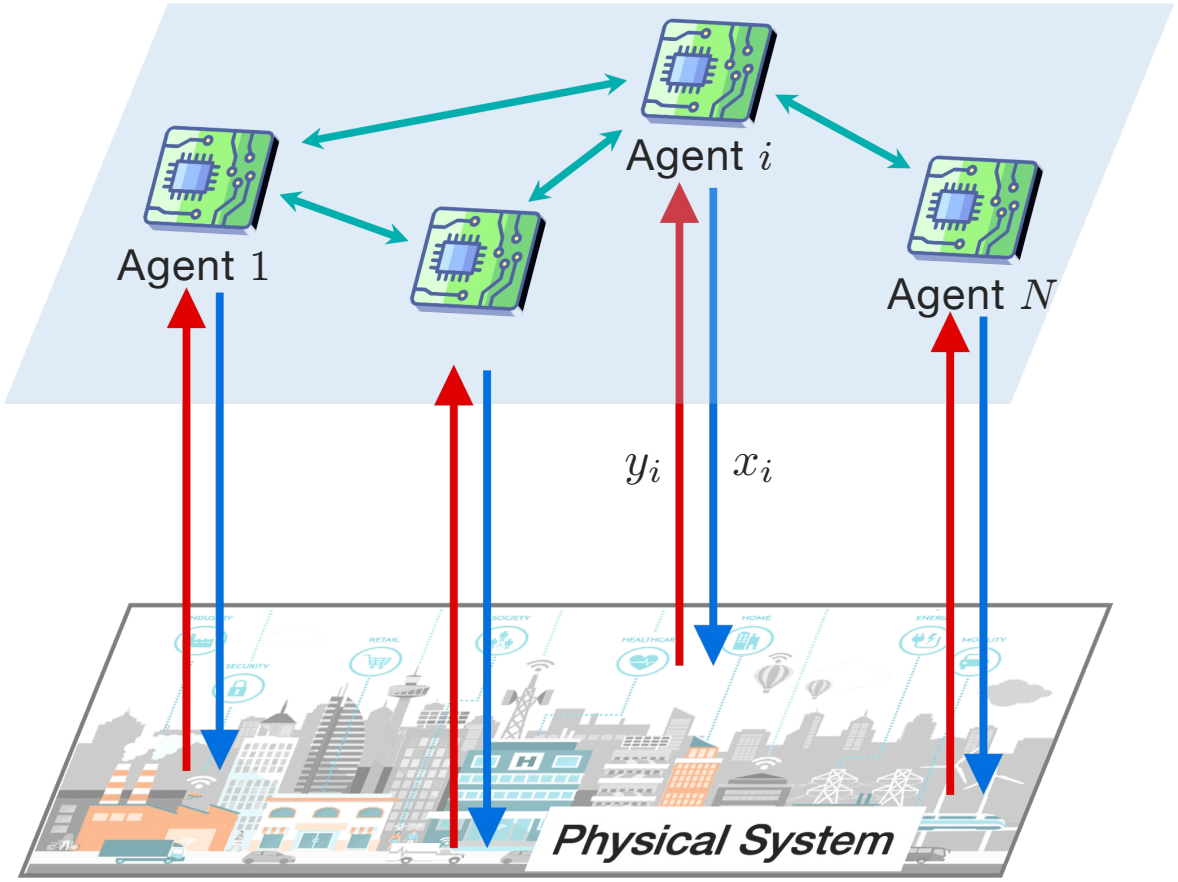
$$D_j(k) = f_j(x(k)+rz(k)) - f_j(x(k))$$

b_{ij} : distance from j to i

- Exchange info with neighbors so that agent i can get the most up-to-date value of D_j for $j \neq i$
  - Note: Delay is inevitable!

Tang, Ren & Li. *Zeroth-order feedback optimization for cooperative multi-agent systems*. Automatica 2022

Algorithm Design



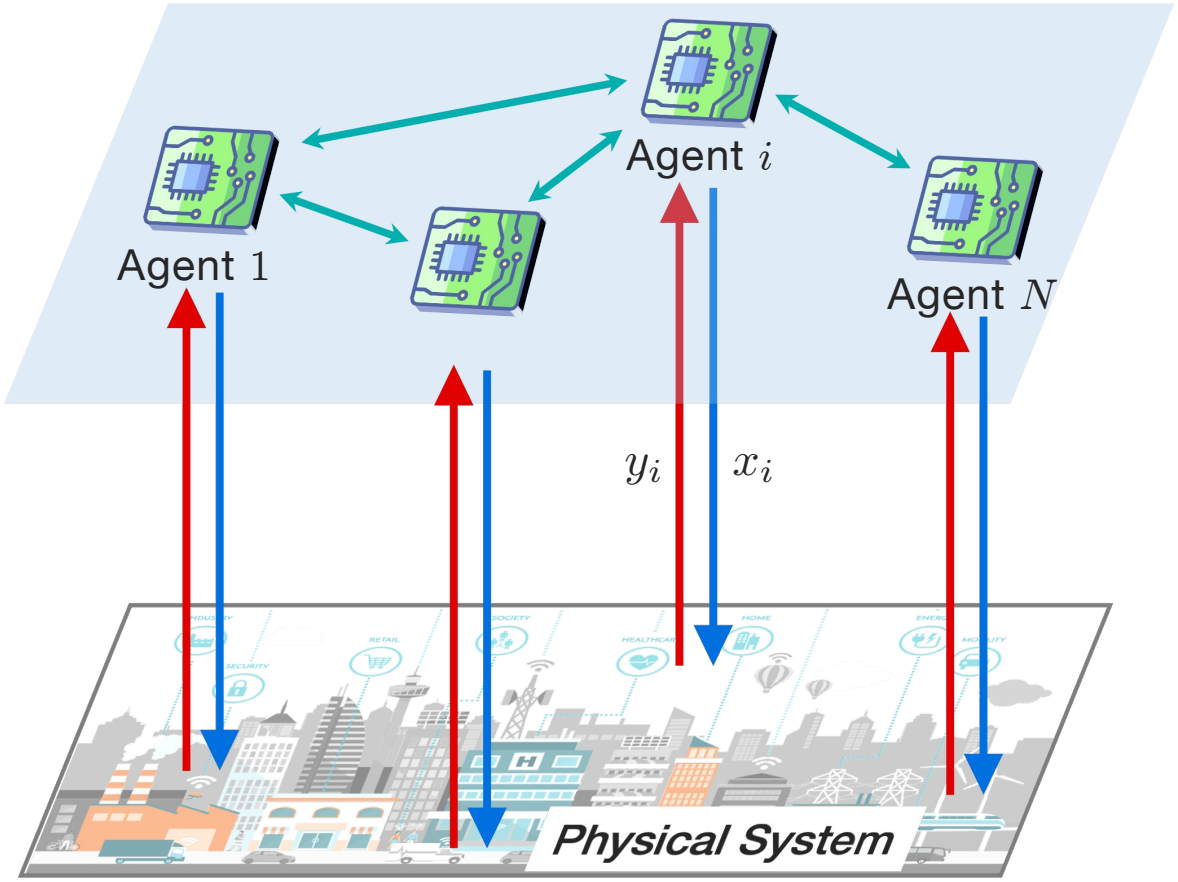
Static system: $y_i = f_i(x_1, \ldots, x_N)$

- Each agent maintains a table:

Agent #	Value	Timestamp
i	D_i	…
j	D_j	…
…	…	…

Tang, Ren & Li. *Zeroth-order feedback optimization for cooperative multi-agent systems*. Automatica 2022

Algorithm Design



Static system: $y_i = f_i(x_1, \ldots, x_N)$

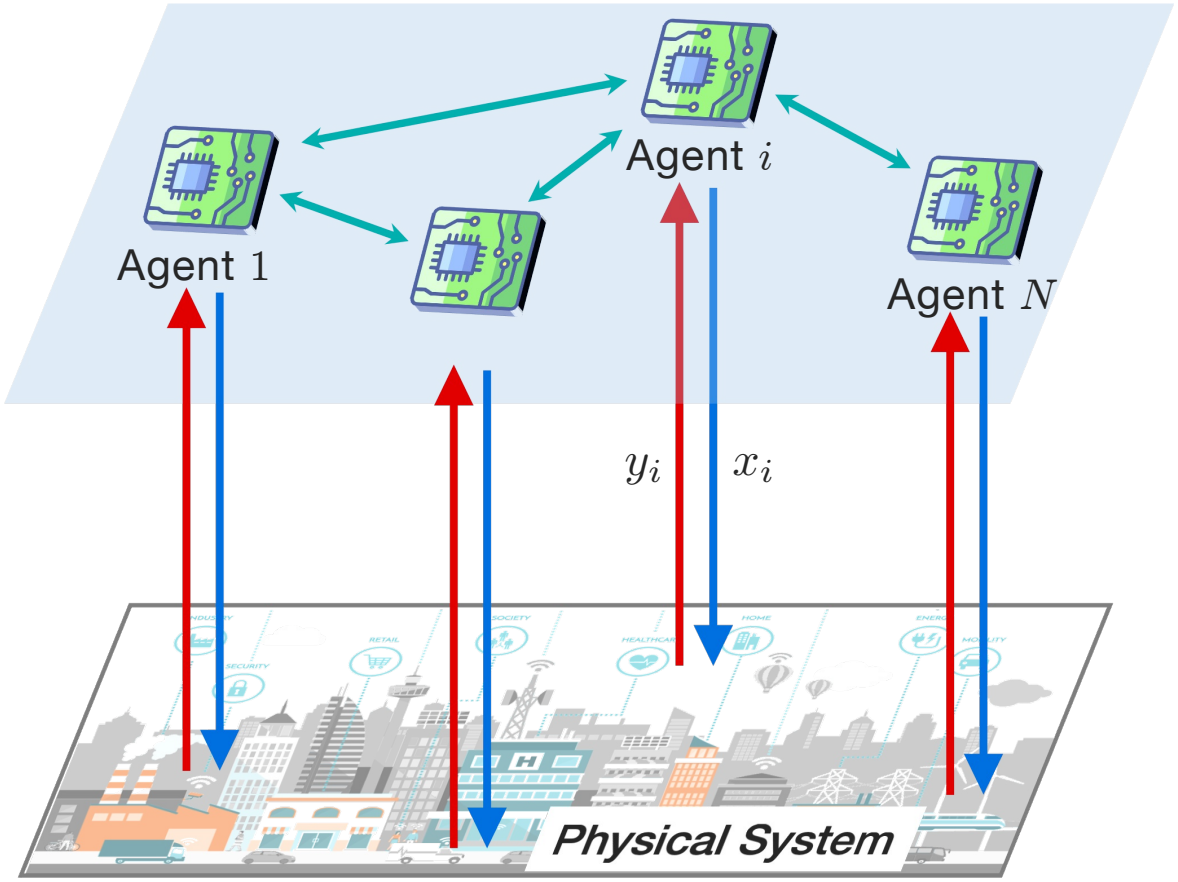
- Each agent maintains a table:

Agent #	Value	Timestamp
i	D_i	current time instant
j	D_j	…
…	…	…

- Each agent i first records and updates its own D_i and timestamp.

Tang, Ren & Li. ***Zeroth-order feedback optimization for cooperative multi-agent systems***. Automatica 2022

Algorithm Design



Static system: $y_i = f_i(x_1, \ldots, x_N)$

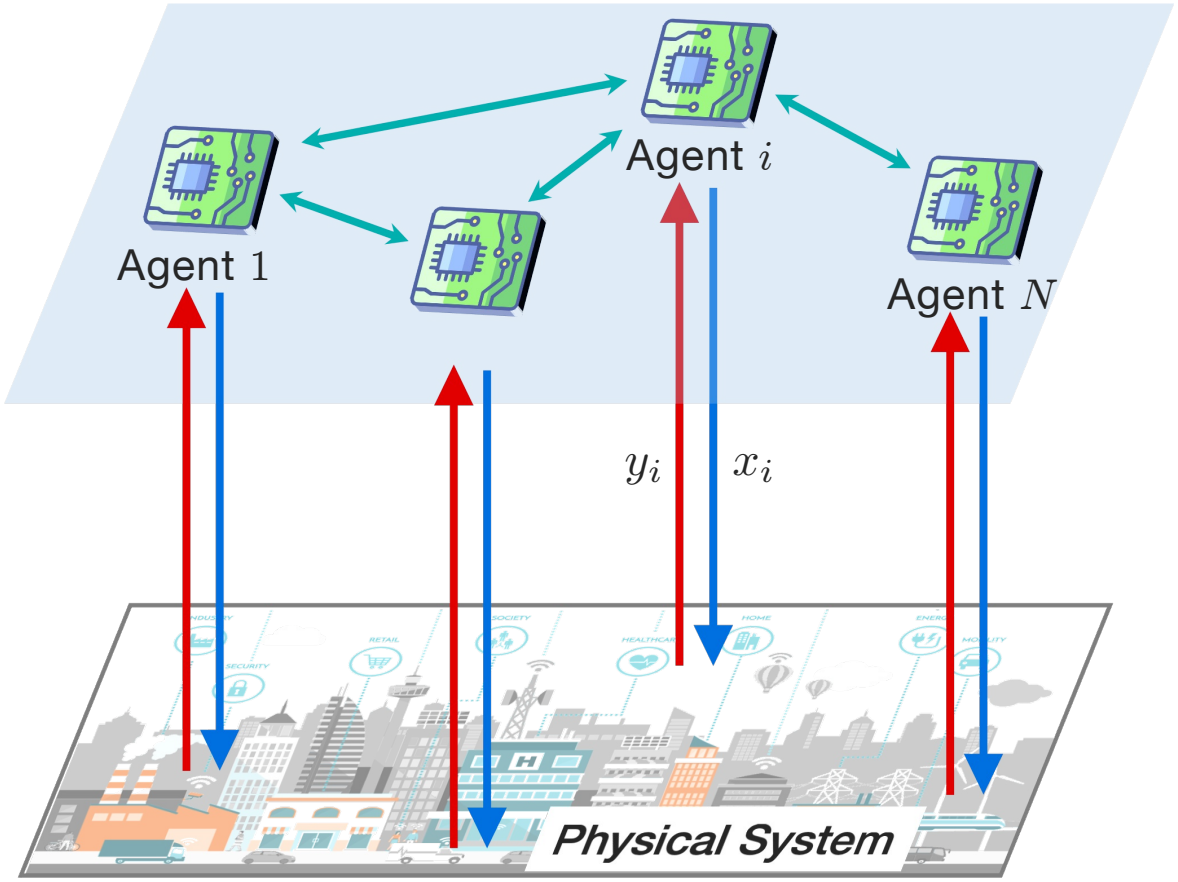
- Each agent maintains a table:

Agent #	Value	Timestamp
i	D_i	current time instant
j	D_j	...
...	...	...

- Each agent i first records and updates its own D_i and timestamp.
- Agent i collects its neighbors' tables via the network.
- For each $j \neq i$, agent i picks out the newest D_j among all collected tables by comparing their timestamps.

Tang, Ren & Li. ***Zeroth-order feedback optimization for cooperative multi-agent systems***. Automatica 2022

Algorithm Design



Static system: $y_i = f_i(x_1, \ldots, x_N)$

Some other issues:

- How to handle constraints?
  - Projected gradient descent
- How to ensure $x_i(k) + rz_i(k) \in \mathcal{X}_i$?
  - The distribution of $z_i(k)$ needs to be redesigned
- Is it possible to reduce the size of the table?

Tang, Ren & Li. ***Zeroth-order feedback optimization for cooperative multi-agent systems***. Automatica 2022

Performance Guarantees – Sample Complexity

Iteration complexity:

Given $\epsilon>0$, the number of iterations K to achieve

$$\mathbb{E}[f(\bar{x}(K)) - f^*] \leq \epsilon \quad \text{(convex)}$$

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1} \|\nabla f(x(k))\|^2\right] \leq \epsilon \quad \text{(nonconvex)}$$

Sample complexity: iteration complexity $\times$ 2

	constrained & convex		unconstrained & nonconvex
	$\nabla f(x^*)=0$ known *a priori*	$\nabla f(x^*)=0$ not assumed	
noiseless	$\Theta\left(\frac{\bar{\mathfrak{b}}d}{\epsilon^2}\right)$		$\Theta\left(\frac{\bar{b}\sqrt{N}d}{\epsilon^2}\right)$
noisy	$\Theta\left(\frac{\bar{\mathfrak{b}}(d^2+d\ln(1/\epsilon)}{\epsilon^3}\right)$	$\Theta\left(\frac{\bar{\mathfrak{b}}(d^2+d\ln(1/\epsilon)}{\epsilon^4}\right)$	$\Theta\left(\frac{\bar{b}\sqrt{N}d^2}{\epsilon^3}\right)$

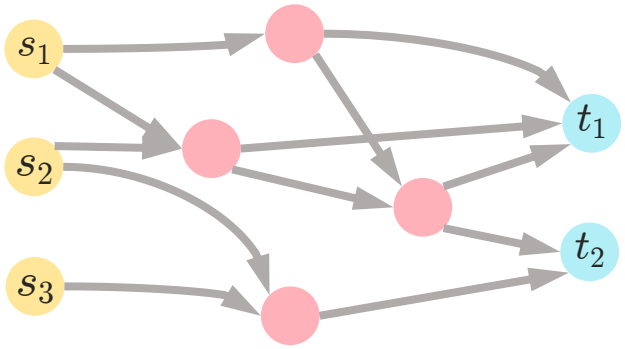
Guaranteed Sample Complexity

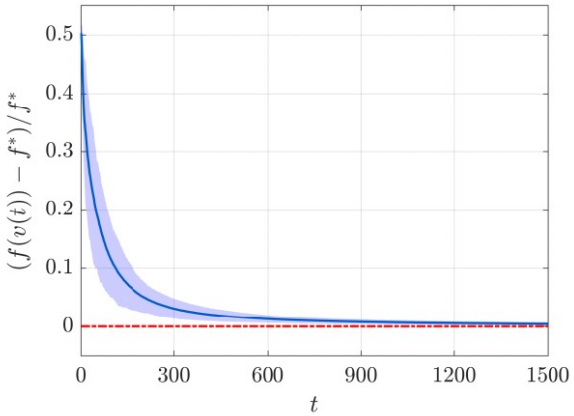
$\bar{b}, \bar{\mathfrak{b}}$: weighted average of pairwise distances of the network

Tang, Ren & Li. ***Zeroth-order feedback optimization for cooperative multi-agent systems***. CDC 2020; Automatica 2022

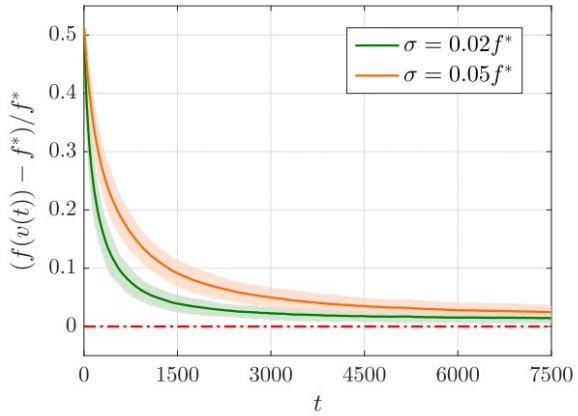
Numerical Example

Distributed routing control



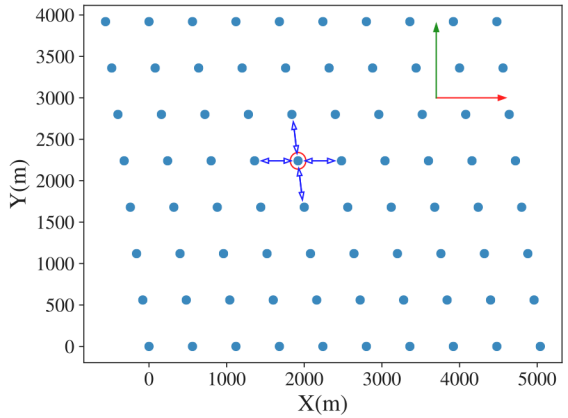


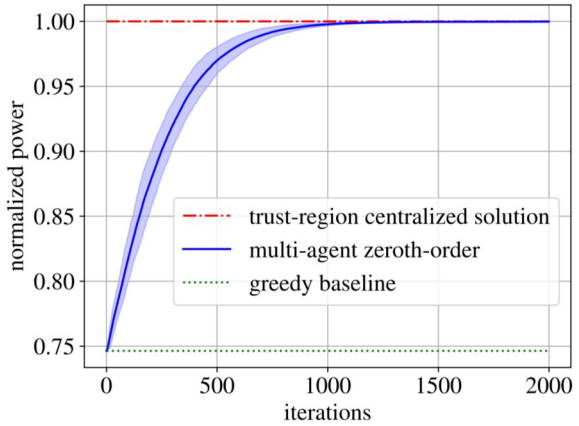
Noiseless



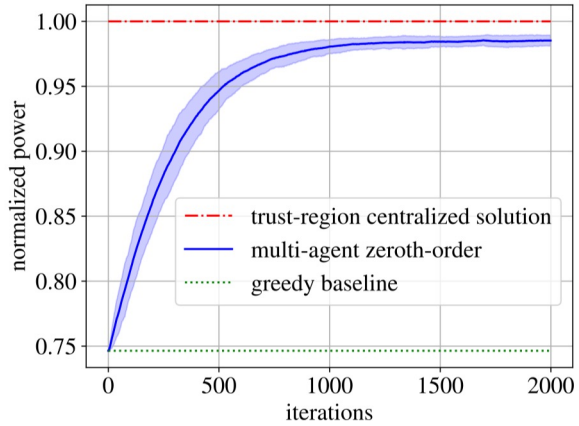
Noisy

Wind farm power maximization





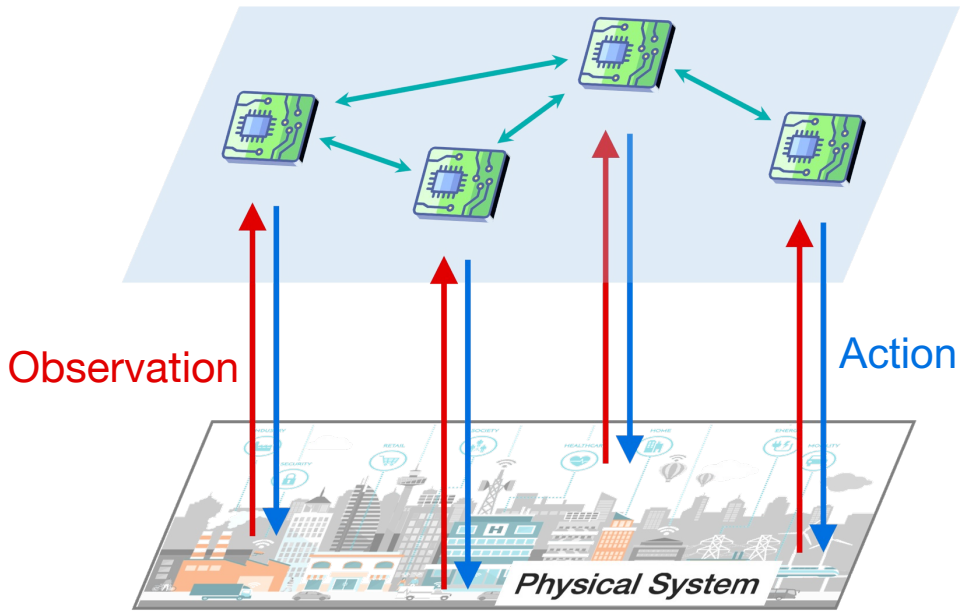
Noiseless



Noisy

Tang, Ren & Li. *Zeroth-order feedback optimization for cooperative multi-agent systems*. CDC 2020; Automatica 2022

Model-Free Optimization and Learning for Multi-Agent Systems

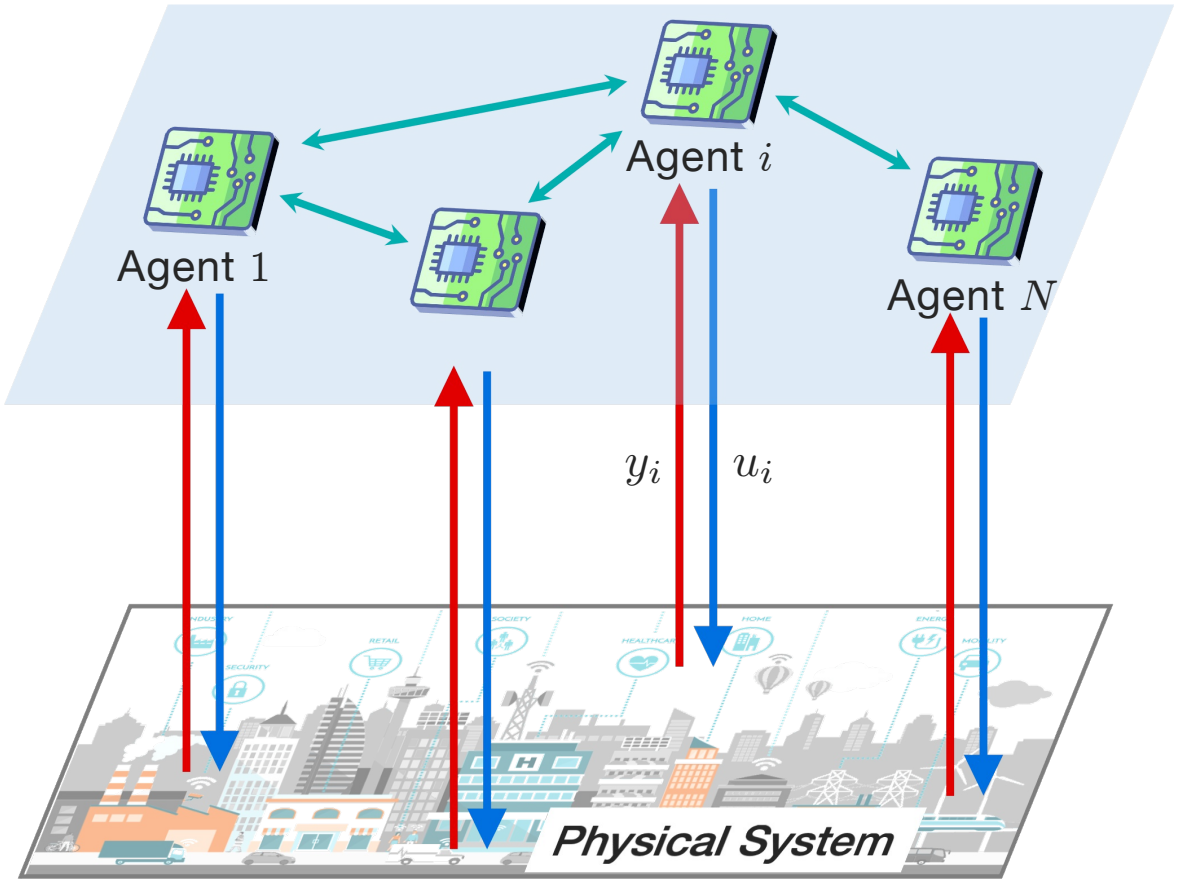


Zeroth-order optimization + Decentralized coordination

I. Multi-agent feedback optimization

II. Distributed reinforcement learning

Decentralized Linear Quadratic Control



$$x(t+1) = Ax(t) + \sum_{i=1}^{N} B_i\, u_i(t) + w(t)$$

LTI dynamics

- **Local observation:** $y_i(t) = C_i\, x(t)$
- **Local control input:** $u_i(t) = K_i\, y_i(t)$
- **Local stage cost:**

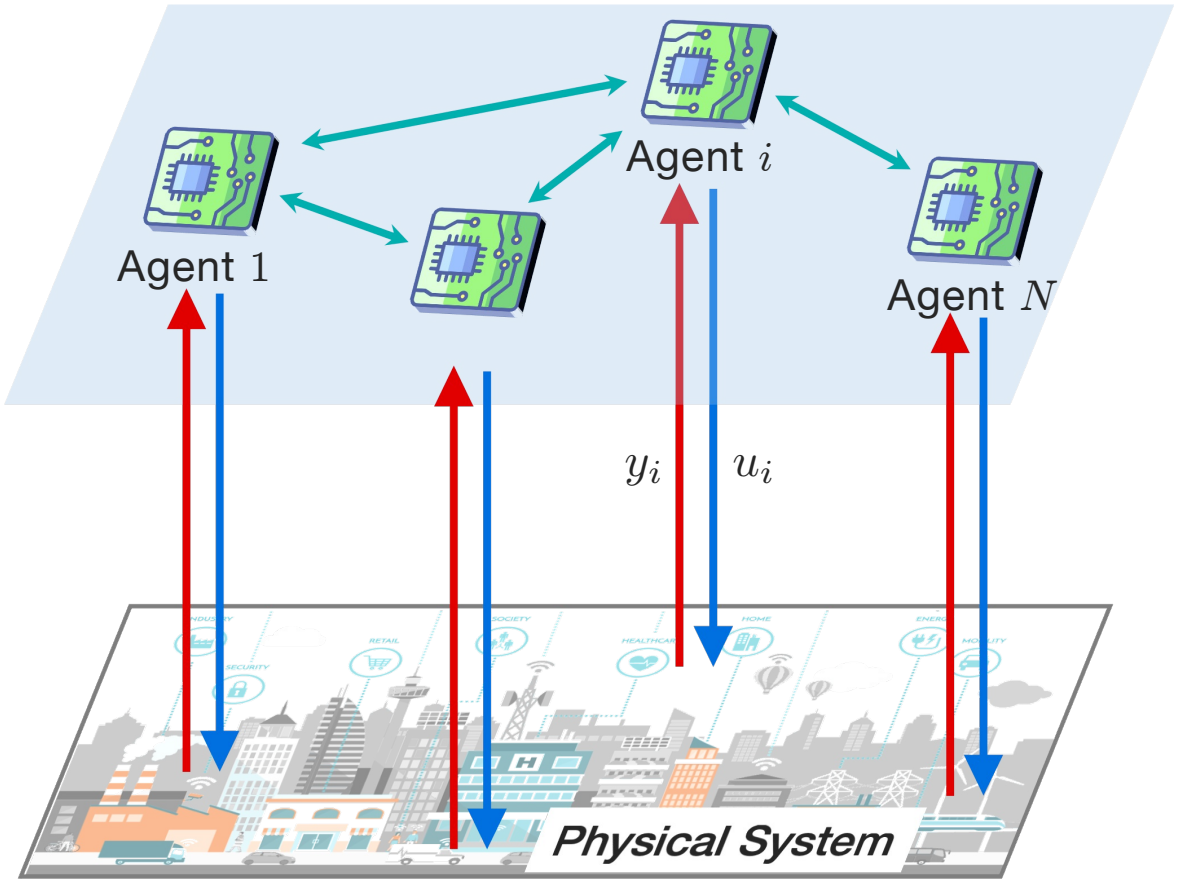
$$c_i(t) = x(t)^\top Q_i\, x(t) + u_i(t)^\top R_i\, u_i(t)$$

- **Goal:**

$$\underset{K_1,\ldots,K_N}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[c_i(t)]$$

- **Communication network**
- **Distributed reinforcement learning**
  - Unknown matrices A, B_i, C_i, Q_i, R_i

Decentralized Linear Quadratic Control



$$x(t+1) = Ax(t) + \sum_{i=1}^{N} B_i\, u_i(t) + w(t)$$

LTI dynamics

An optimization viewpoint:

$$\min_{K=(K_1,\ldots,K_N)} \quad J(K)$$
$$\text{s.t.} \quad K \text{ stabilizes the system}$$

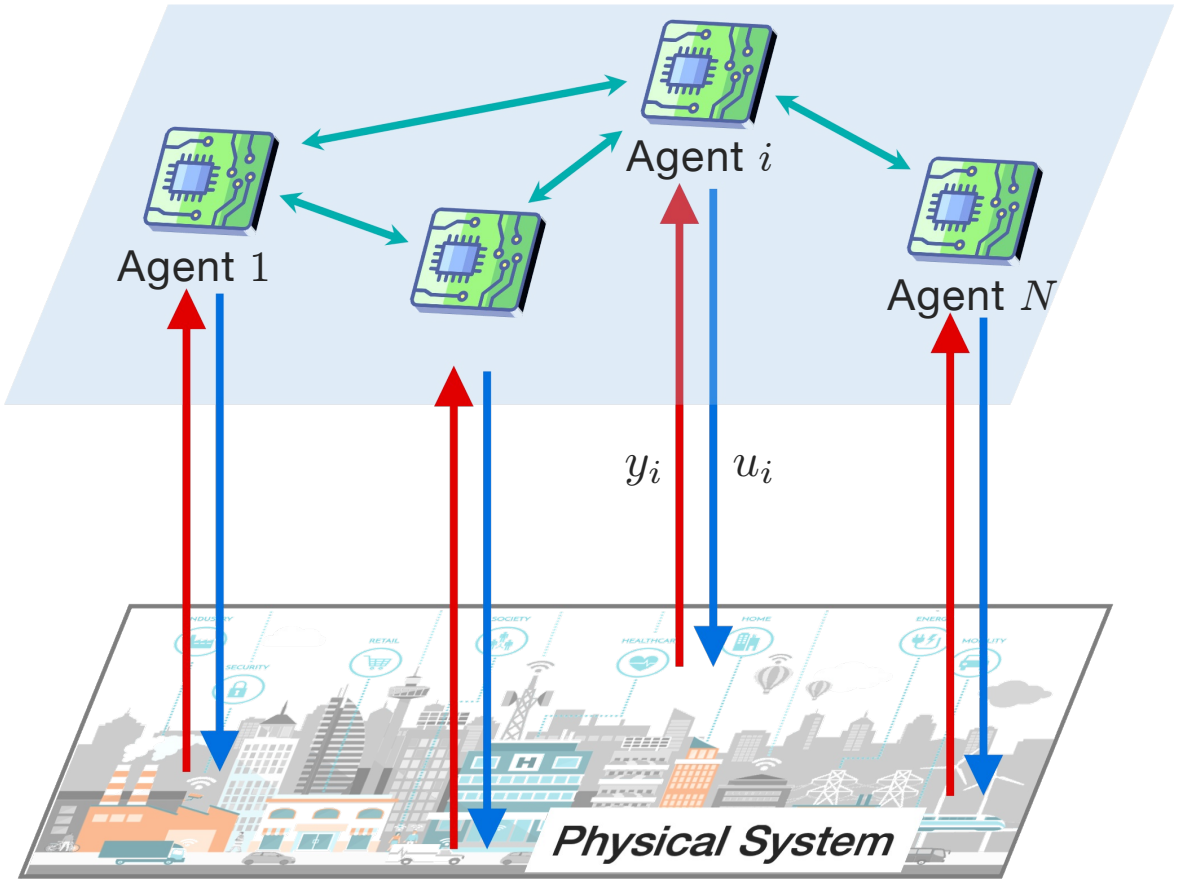
$$J(K) = \frac{1}{N}\sum_{i=1}^{N} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[c_i(t)]$$

Zeroth-order method:

$$K(s+1) = K(s) - \eta \cdot \frac{d}{r} J(K(s) + rz(s))z(s)$$

[Fazel 2018] [Malik 2020] [Mohammadi 2022] [Zhang 2021]

Decentralized Linear Quadratic Control



$$x(t+1) = Ax(t) + \sum_{i=1}^{N} B_i\, u_i(t) + w(t)$$

LTI dynamics

Zeroth-order method (decentralized):

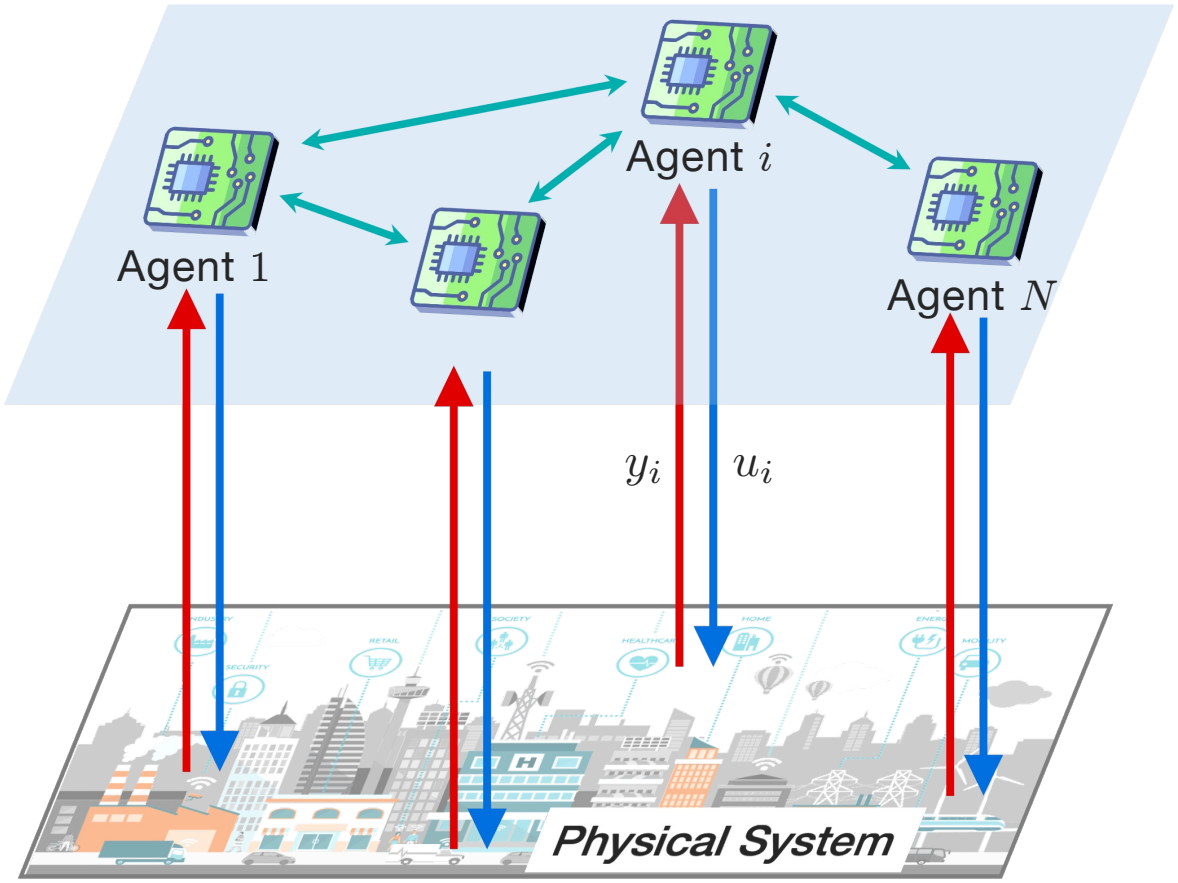
$$K_i(s+1) = K_i(s) - \eta \cdot \hat{G}_i(s)$$

Zeroth-order **partial** gradient estimator

$$= K_i(s) - \eta \cdot \frac{d}{r}\, J(K(s)+rz(s))\, z_i(s)$$

Challenges:

Decentralized Linear Quadratic Control



$$x(t+1) = Ax(t) + \sum_{i=1}^{N} B_i\, u_i(t) + w(t)$$

LTI dynamics

Zeroth-order method (decentralized):

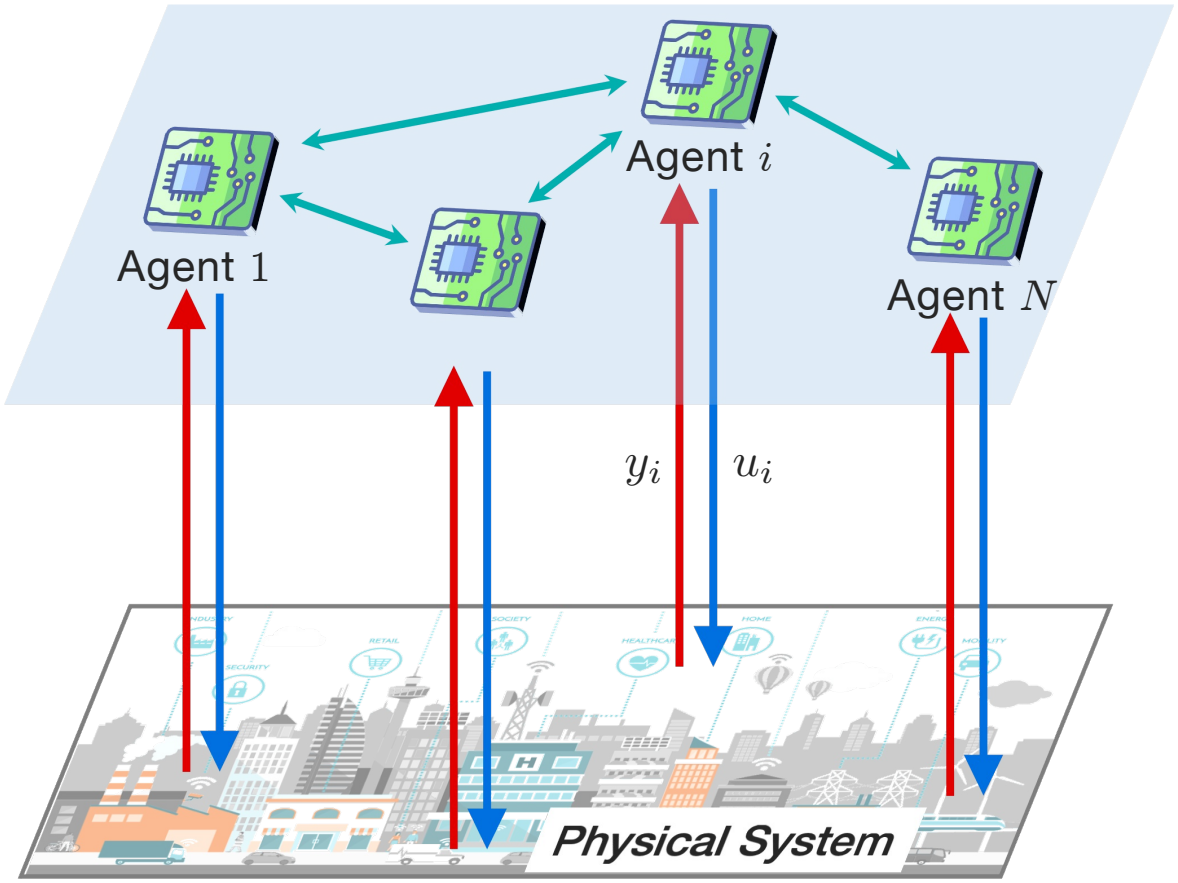
$$K_i(s+1) = K_i(s) - \eta \cdot \hat{G}_i(s)$$

$$= K_i(s) - \eta \cdot \frac{d}{r} J(K(s)+rz(s)) z_i(s)$$

Challenges:

- How to locally estimate the **global** objective value?

Decentralized Linear Quadratic Control



$$x(t+1) = Ax(t) + \sum_{i=1}^{N} B_i\, u_i(t) + w(t)$$

LTI dynamics

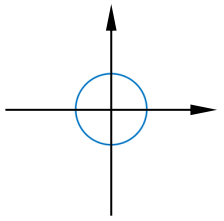
Zeroth-order method (decentralized):

$$K_i(s+1) = K_i(s) - \eta \cdot \hat{G}_i(s)$$
$$= K_i(s) - \eta \cdot \frac{d}{r}\, J(K(s)+rz(s))\, z_i(s)$$

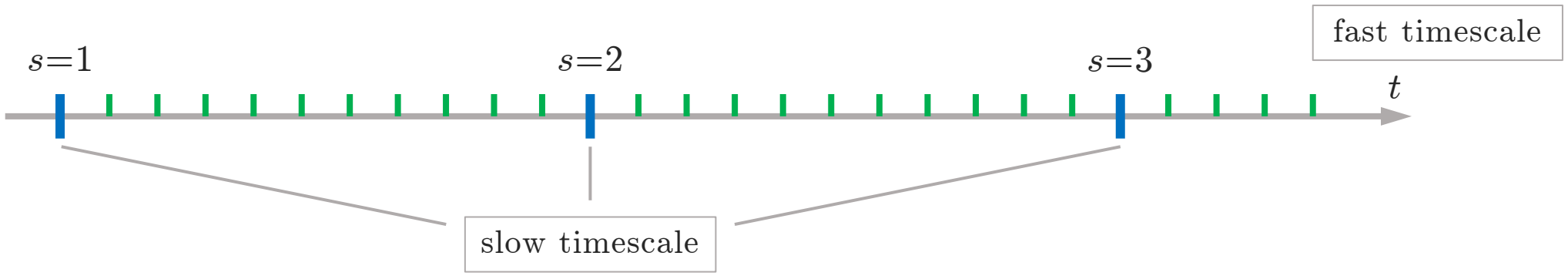
Challenges:

- How to locally estimate the global objective value?
- How to ensure $K(s)+rz(s)$ stabilizes the system?
  - At least the perturbation $z(s)$ should be bounded
  - In a decentralized setup: How?

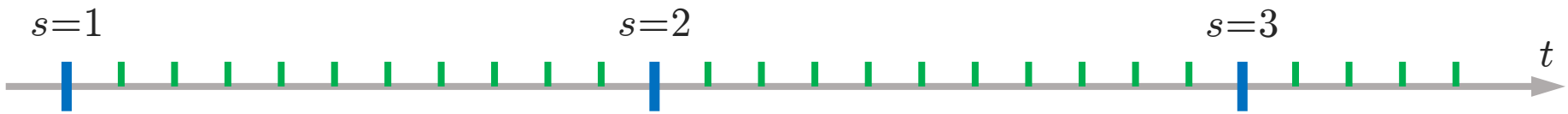
$$(z_1(s), \ldots, z_N(s)) \sim \mathrm{Unif}(\mathbb{S}_{d-1})$$



Two-Timescale Algorithm Design



Two-Timescale Algorithm Design



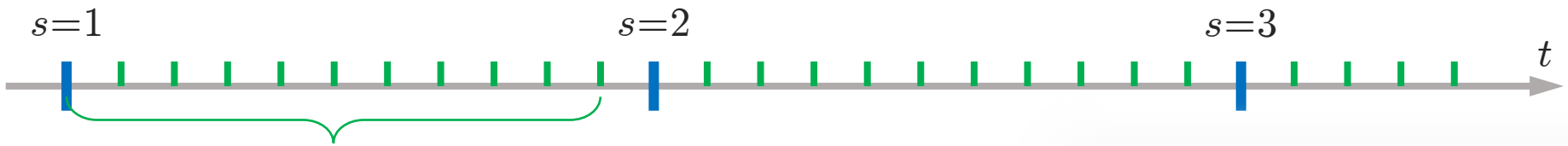
1. Generate random perturbation $z_i(s)$
2. Apply control policy $K_i(s) + rz_i(s)$ to the system
3. Accumulate costs $c_i(t)$ & exchange info with neighbors
4. Produce an estimated global obj. $\hat{J}_i(s) \approx J(K(s)+rz(s))$
5. Construct zeroth-order partial gradient estimator

$$\hat{G}_i(s) = \frac{d}{r}\,\hat{J}_i(s)\,z_i(s)$$

6. Update by stochastic gradient descent $K_i(s+1) = K_i(s) - \eta\hat{G}_i(s)$

Li*, **Tang***, Zhang & Li. *Distributed reinforcement learning for decentralized linear quadratic control: A derivative-free policy optimization approach*. TAC 2021 (* Equal contribution)

Two-Timescale Algorithm Design



1. Generate random perturbation $z_i(s)$
2. Apply control policy $K_i(s) + rz_i(s)$ to the system
3. Accumulate costs $c_i(t)$ & exchange info with neighbors
4. Produce an estimated global obj. $\hat{J}_i(s) \approx J(K(s)+rz(s))$
5. Construct zeroth-order partial gradient estimator

$$\hat{G}_i(s) = \frac{d}{r}\hat{J}_i(s)\, z_i(s)$$

6. Update by stochastic gradient descent $K_i(s+1) = K_i(s) - \eta \hat{G}_i(s)$

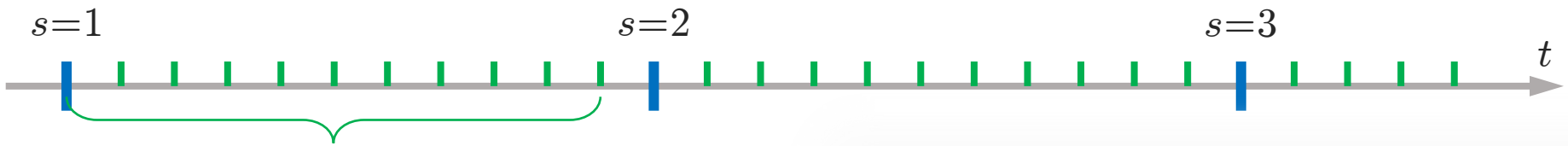
Consensus method

$$\mu_i(t) = \frac{t-1}{t}\sum_j W_{ij}\,\mu_i(t-1) + \frac{1}{t}c_i(t)$$

- W: communication weight matrix
  - $N\times N$ doubly stochastic
  - $W_{ij} = 0$ if (i,j) not connected

Li*, **Tang***, Zhang & Li. *Distributed reinforcement learning for decentralized linear quadratic control: A derivative-free policy optimization approach*. TAC 2021 (* Equal contribution)

Two-Timescale Algorithm Design



The closed-loop system evolve for T_J steps

1. Generate random perturbation $z_i(s)$
2. Apply control policy $K_i(s) + rz_i(s)$ to the syst
3. Accumulate costs $c_i(t)$ & exchange info with
4. Produce an estimated global obj. $\hat{J}_i(s) \approx J(K$
5. Construct zeroth-order partial gradient estim

$$\hat{G}_i(s) = \frac{d}{r}\hat{J}_i(s)\, z_i(s)$$

6. Update by stochastic gradient descent $K_i(s-$

Lemma: If $\tilde{z} \sim \mathcal{N}(0, I_d)$, then $\tilde{z}/\|\tilde{z}\| \sim \mathrm{Unif}(\mathbb{S}_{d-1})$

Sample $\tilde{z}_i$ from standard Gaussian

$q_i(0) = \|\tilde{z}_i\|_F^2$

For $t = 1$ **to** T_J

$$q_i(t) = \sum\nolimits_j W_{ij}\, q_j(t-1)$$

Consensus again

$$q_i(T_J) \approx \frac{1}{N}\sum\nolimits_j \|\tilde{z}_i\|_F^2$$

end for

$z_i(s) = \tilde{z}_i / \sqrt{N q_i(T_J)}$

Concurrent with $\tilde{J}_i(s)$ estimation

Li*, **Tang***, Zhang & Li. *Distributed reinforcement learning for decentralized linear quadratic control: A derivative-free policy optimization approach*. TAC 2021 (* Equal contribution)

Performance Guarantees

Theorem (informal)

Let $\epsilon > 0$ be arbitrary. By choosing the parameters of the algorithm to satisfy

$$r \sim O(\sqrt{\epsilon}) \qquad \eta \sim O\left(\epsilon r^2\right) \qquad T_J \sim \Omega\left(\frac{1}{r\sqrt{\epsilon}}\right) \qquad T_G \sim \Theta\left(\frac{1}{\eta\epsilon}\right)$$

we can achieve the following with high probability:

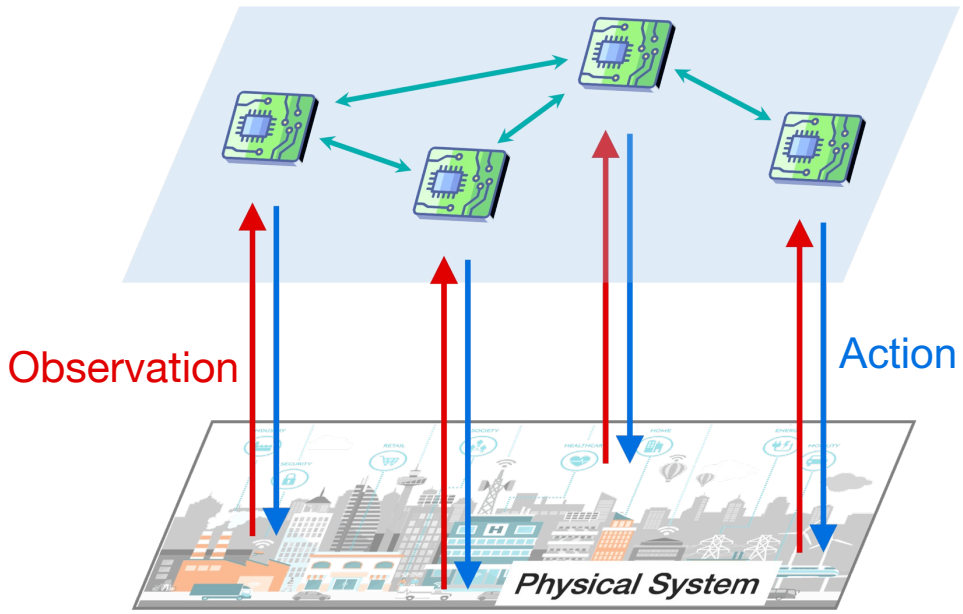
- The closed-loop system remain **stable** during the learning procedure
- Optimality guarantee given by

$$\frac{1}{T_G}\sum_{s=1}^{T_G} \|\nabla J(K(s))\|^2 \leq \epsilon$$

Corollary: Sample complexity bound given by $T_G T_J \sim \Theta\left(\frac{1}{\epsilon^4}\right)$

Li*, **Tang***, Zhang & Li. *Distributed reinforcement learning for decentralized linear quadratic control: A derivative-free policy optimization approach*. TAC 2021 (* Equal contribution)

Model-Free Optimization and Learning for Multi-Agent Systems



Zeroth-order optimization + Decentralized coordination

I. Multi-agent feedback optimization

II. Distributed reinforcement learning

- Different coordination schemes designed for different setups
- Theoretical convergence rate/complexity analysis results

Revisiting the Results

- Complexity for multi-agent zeroth-order feedback optimization

	Constrained & convex	
	$\nabla f(x^*)=0$ known	$\nabla f(x^*)=0$ unknown
Noiseless	$O\left(\frac{\bar{\mathfrak{b}}d}{\epsilon^2}\right)$	

- Complexity for centralized unconstrained smooth zeroth-order optimization

$O\left(\frac{d}{\epsilon}\right)$ [Nesterov 2017]

A *gap* between **constrained** and **unconstrained** zeroth-order optimization

Revisiting the Results

$$\min_x \quad f(x)$$
$$\text{s.t.} \quad x \in C$$

deterministic
convex & smooth

Existing Results

- 2-point zeroth-order methods

$$O\left(\frac{d}{\epsilon^2}\right)$$ [Duchi 2015]

- $2d$-point zeroth-order methods

$$O\left(\frac{d}{\epsilon}\right)$$ [Sahu 2020]

Why is there a gap?

$$\lim_{r\to 0} \mathrm{Var}(\mathsf{G}_f(x; r, z)) \approx (d-1)\|\nabla f(x)\|^2$$

- Unconstrained: $\nabla f(x(t)) \to 0$

$$\implies \quad \mathrm{Var}(\mathsf{G}_f(x(t); r_t, z(t))) \to 0$$

- Constrained: $\nabla f(x(t)) \nrightarrow 0$

$$\implies \quad \mathrm{Var}(\mathsf{G}_f(x(t); r_t, z(t))) \geq C > 0$$

Revisiting the Results

$$\min_{x} \quad f(x)$$
$$\text{s.t.} \quad x \in C$$

deterministic
convex & smooth

Existing Results

- 2-point zeroth-order methods

$$O\left(\frac{d}{\epsilon^2}\right)$$ [Duchi 2015]

- $2d$-point zeroth-order methods

$$O\left(\frac{d}{\epsilon}\right)$$ [Sahu 2020]

Our Findings

- 2-point zeroth-order for **box** constraints

$$O\left(\frac{d}{\epsilon}\right)$$

Algorithm: Adopting the framework of **randomized coordinate descent** rather than gradient descent

arXiv:2311.00372

RZFGD for Distributed Demand Response

Aggregator samples $\alpha(k) \sim \mathcal{U}\{1, \ldots, d\}$

Aggregator sends $\phi(x(k))$ to agent $\alpha(k)$

Agent $\alpha(k)$ samples $z_{\alpha(k)}(k) \in \{-1, +1\}$

Agent $\alpha(k)$ applies $x_{\alpha(k)}(k) + r(k) z_{\alpha(k)}(k)$

Aggregator observes $\phi(x(k) + r(k) z(k)) = \phi(x^+(k))$

and sends it to agent $\alpha(k)$

Agent $\alpha(k)$ calculates

$$g_{\alpha(k)}(k) = \frac{\partial f_{\alpha(k)}(x(k))}{\partial x_{\alpha(k)}} + \frac{\phi(x^+(k)) - \phi(x(k))}{r(k)} z_{\alpha(k)}(k)$$

Agent $\alpha(k)$ updates

$$x_{\alpha(k+1)} = \left[x_{\alpha(k)}(k) - \eta g_{\alpha(k)}(k) \right]_{l_{\alpha(k)}}^{u_{\alpha(k)}}$$

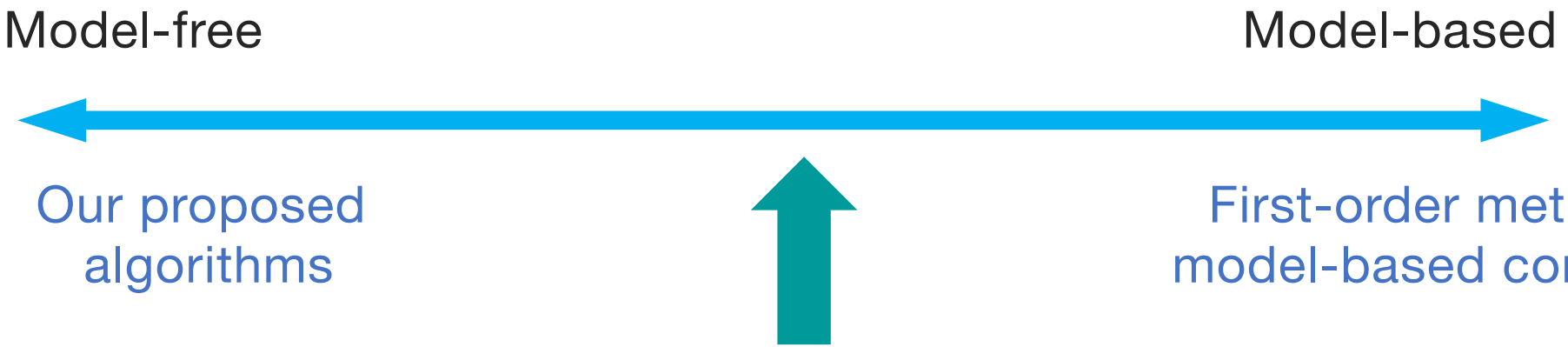
$$\min_{x_1, \ldots, x_N} \quad \phi(x_1, \ldots, x_N) + \sum_{i=1}^{N} f_i(x_i)$$

Complexity guarantees:

Algorithm	Feasible region	Complexity (convex)	Complexity (nonconvex)
2-ZFGD	Convex & compact	$\mathcal{O}\left(\frac{d}{\epsilon^2}\right)$	$\mathcal{O}\left(\frac{d}{\epsilon^2}\right)$
RZFCD	Box	$\mathcal{O}\left(\frac{d}{\epsilon}\right)$	$\mathcal{O}\left(\frac{d}{\epsilon}\right)$

arXiv:2311.00372

Future Directions



What about in between?

- We have some prior structural information.
- How to incorporate them in algorithm design?
- Better convergence rate/scalability?