Detection and Mitigation of Corrupted Information in Distributed Model Predictive Control Based on Resource Allocation

R. A. Nogueira R. Bourdais H. Guéguen
{rafael-accacio.nogueira, romain.bourdais, herve.gueguen} at centralesupelec.fr

AUT Department
IETR — CentraleSupélec

5th International Conference on Control and Fault-Tolerant Systems, 2021

https://git.io/JEFGW

Context

Smart City

Context

Smart City

- Energy Distribution System
- Traffic management
- Heat distribution
- Water distribution

. . .

CentraleSupélec

Context

Smart City

- Energy Distribution System
- Traffic management
- Heat distribution
- Water distribution

. . .

Context

Smart City

- Energy Distribution System
- Traffic management
- Heat distribution
- Water distribution

. . .

Context

Smart City

- Energy Distribution System
- Traffic management
- Heat distribution
- Water distribution

. . .

Context

Smart City

- Geographically distributed
- Coupled by constraints (energy)
- Optimization objectives
    - Energy
    - User satisfaction
    - ...
- Solution → Model Predictive Control

Context

Smart City

- Geographically distributed
- Coupled by constraints (energy)
- Optimization objectives
  - Energy
  - User satisfaction
  - ...
- Solution → Model Predictive Control

CentraleSupélec

Context

Smart City

- Geographically distributed
- Coupled by constraints (energy)
- Optimization objectives
  - Energy
  - User satisfaction
  - ...
- Solution → Model Predictive Control

CentraleSupélec

Context

Smart City

- Geographically distributed
- Coupled by constraints (energy)
- Optimization objectives
  - Energy
  - User satisfaction
  - ...
- Solution → Model Predictive Control

Model Predictive Control

Objective: Find control input sequence that optimizes an objective function

$$\begin{array}{cc} \underset{\boldsymbol{u}(k:k+N_p-1|k)}{\text{optimize}} & J(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\ \text{subject to} & \left.\begin{array}{c} \boldsymbol{x}(\xi+1) = f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \\ g_i(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \leq 0 \\ h_j(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) = 0 \end{array}\right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array} \end{array}$$

Model Predictive Control

Objective: Find control input sequence that optimizes an objective function

$$
\begin{aligned}
\underset{\boldsymbol{u}(k:k+N_p-1|k)}{\text{optimize}} \quad & J(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\
\text{subject to} \quad & \left.\begin{array}{c} \boldsymbol{x}(\xi+1) = f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \\ g_i(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \leq 0 \\ h_j(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) = 0 \end{array}\right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array}
\end{aligned}
$$

CentraleSupélec

Model Predictive Control

Objective: Find control input sequence that optimizes an objective function

$$
\begin{array}{cc}
\underset{\boldsymbol{u}(k:k+N_p-1|k)}{\text{optimize}} & J(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\
\text{subject to} & \left.\begin{array}{c} \boldsymbol{x}(\xi+1) = f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \\ g_i(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \leq 0 \\ h_j(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) = 0 \end{array}\right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array}
\end{array}
$$

Model Predictive Control

Objective: Find control input sequence that optimizes an objective function

$$\begin{aligned}
\underset{\boldsymbol{u}(k:k+N_p-1|k)}{\text{optimize}} \quad & J(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\
\text{subject to} \quad & \left.\begin{aligned} \boldsymbol{x}(\xi+1) = f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \\ g_i(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \le 0 \\ h_j(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) = 0 \end{aligned}\right\} \begin{aligned} & \forall \xi \in \{1, \dots, N_p\} \\ & \forall i \in \{1, \dots, m\} \\ & \forall j \in \{1, \dots, p\} \end{aligned}
\end{aligned}$$

Model Predictive Control

Objective: Find control input sequence that optimizes an objective function

$$
\begin{aligned}
&\underset{\boldsymbol{u}(k:k+N_p-1|k)}{\text{optimize}} && J(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\
&\text{subject to} && \left.\begin{array}{c} \boldsymbol{x}(\xi+1) = f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \\ \color{red}{g_i(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \leq 0} \\ \color{red}{h_j(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) = 0} \end{array}\right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array}
\end{aligned}
$$

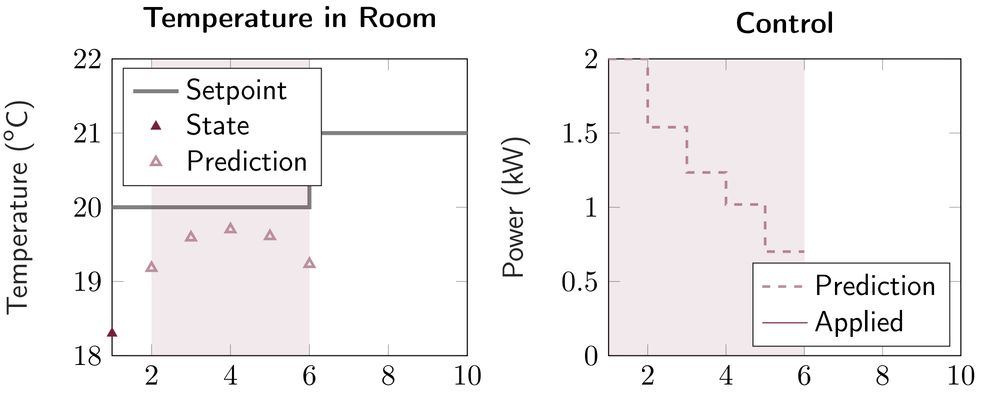
Model Predictive Control

Objective: Find control input sequence that optimizes an objective function

$$
\begin{aligned}
\underset{\boldsymbol{u}(k:k+N_p-1|k)}{\text{minimize}} \quad & \textstyle\sum_{j=1}^{N_p} \|\boldsymbol{v}(k+j|k)\|_Q^2 + \|\boldsymbol{u}(k+j-1|k)\|_R^2 \\
\text{subject to} \quad & \left.\begin{array}{c} \boldsymbol{x}(\xi+1) = f(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \\ g_i(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) \leq 0 \\ h_j(\boldsymbol{x}(\xi), \boldsymbol{u}(\xi)) = 0 \end{array}\right\} \begin{array}{l} \forall \xi \in \{1, \dots, N_p\} \\ \forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, p\} \end{array}
\end{aligned}
$$

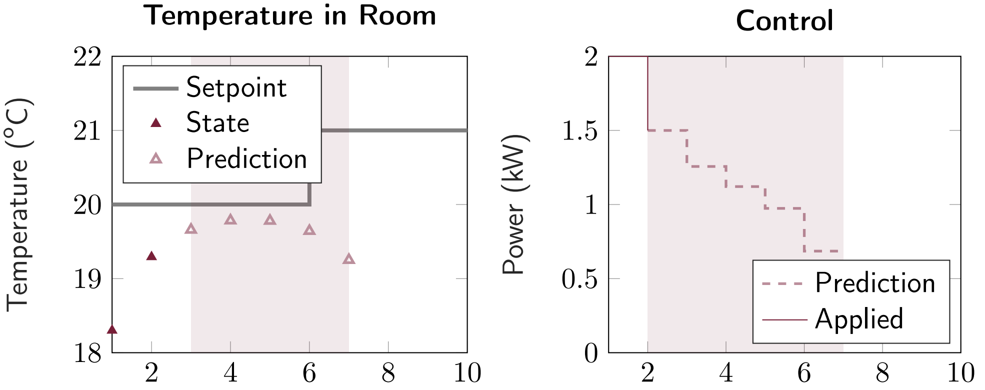
Model Predictive Control

Find optimal control sequence



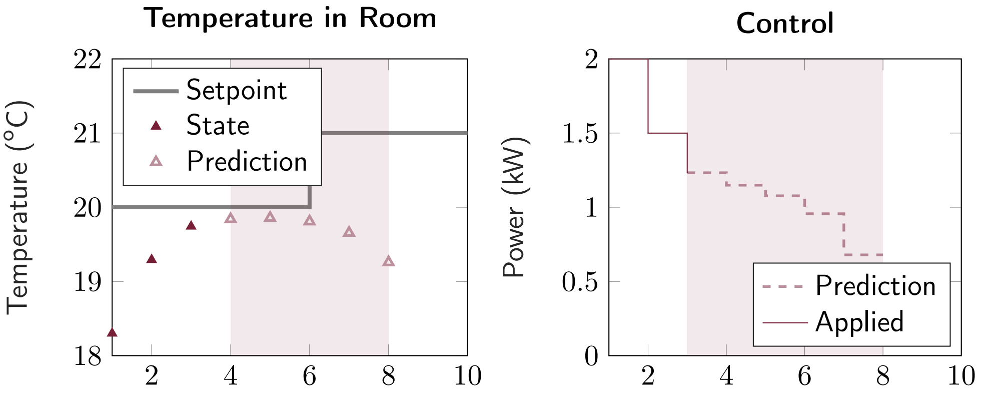
Model Predictive Control

Find optimal control sequence, apply first element



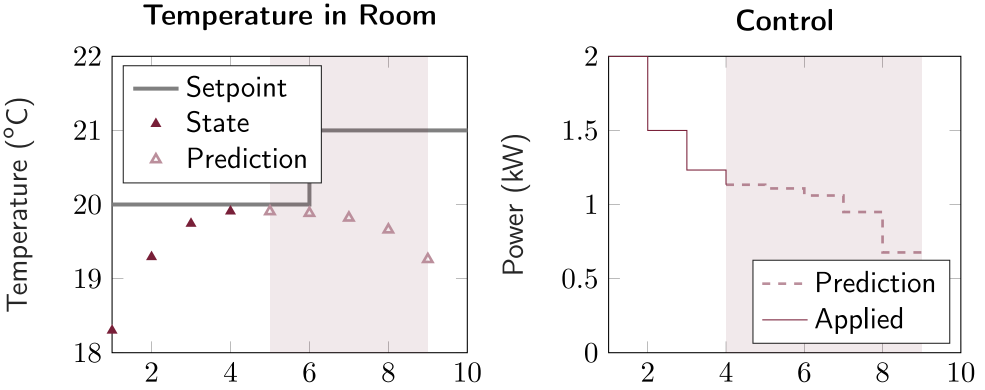
Model Predictive Control

Find optimal control sequence, apply first element, rinse repeat



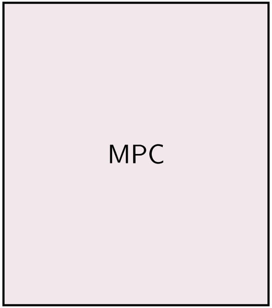
Model Predictive Control

Find optimal control sequence, apply first element, rinse repeat → Receding Horizon



Distributed Model Predictive Control

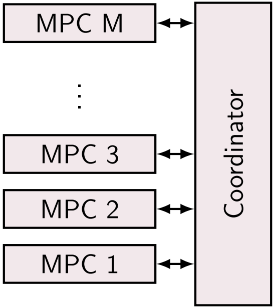
- Problem: Complexity depends on N_p, m, p and sizes of $\boldsymbol{x}$ and $\boldsymbol{u}$
- Solution: Divide and Conquer



MPC

Distributed Model Predictive Control

- Problem: Complexity depends on N_p, m, p and sizes of x and u
- Solution: Divide and Conquer



Distributed Model Predictive Control

$$\begin{array}{ll}\underset{\boldsymbol{u}_i(k:k+N_p-1|k)}{\text{minimize}} & \overbrace{\sum_{i=1}^{M}\overbrace{\sum_{j=1}^{N_p}\|\boldsymbol{v}_i(k+j|k)\|_{Q_i}^2+\|\boldsymbol{u}_i(k+j-1|k)\|_{R_i}^2}^{J_i(k)}}^{J_G(k)} \\ \text{subject to} & \left.\begin{array}{c} \boldsymbol{x}_i(k+1)=A_i\boldsymbol{x}_i(k)+B_i\boldsymbol{u}_i(k)\\ \sum_{i=1}^{M} \Gamma_i\boldsymbol{u}_i(k)=\boldsymbol{u}_{\max}\end{array}\right\} \begin{array}{l}\forall i\in\{1,\dots,M\}\\ \forall j\in\{1,\dots,N_p\}\end{array}\end{array}$$

Distributed Model Predictive Control

Quantity Decomposition | Resource Allocation

$$\left.\begin{aligned} J_i^\star(\boldsymbol{\theta}_i(k)) = \underset{\boldsymbol{u}_i(k:k+N_p-1|k)}{\text{minimize}} \; & J_i(k) \\ \text{s.t.} \quad & \boldsymbol{x}_i(k+1) = A_i\boldsymbol{x}_i(k) + B_i\boldsymbol{u}_i(k) \\ & \Gamma_i\boldsymbol{u}_i(k) = \boldsymbol{\theta}_i(k) : \boldsymbol{\lambda}_i(k) \end{aligned}\right\} \begin{aligned} &\forall i \in \{1,\dots,M\} \\ &\forall j \in \{1,\dots,N_p\} \end{aligned}$$

$$\begin{aligned} J^\star = \underset{\boldsymbol{\theta}(k:k+N_p-1|k)}{\text{minimize}} \; & \sum_{i=1}^{M} J_i^\star(\boldsymbol{\theta}_i(k)) \\ \text{s.t.} \quad & \sum_{i=1}^{M} \boldsymbol{\theta}_i(k) = \boldsymbol{u}_{\max} \end{aligned}$$

Distributed Model Predictive Control

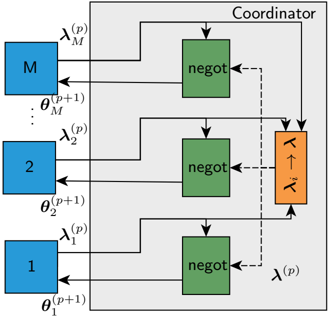
Quantity Decomposition | Resource Allocation

$$\left.\begin{aligned} J_i^\star(\boldsymbol{\theta}_i(k)) = &\underset{\boldsymbol{u}_i(k:k+N_p-1|k)}{\text{minimize}} \; J_i(k) \\ \text{s.t.} \quad & \boldsymbol{x}_i(k+1) = A_i\boldsymbol{x}_i(k) + B_i\boldsymbol{u}_i(k) \\ & \Gamma_i\boldsymbol{u}_i(k) = \boldsymbol{\theta}_i(k) : \boldsymbol{\lambda}_i(k) \end{aligned}\right\} \begin{array}{l} \forall i \in \{1,\dots,M\} \\ \forall j \in \{1,\dots,N_p\} \end{array}$$

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho\left(\boldsymbol{\lambda}_i^\star(\boldsymbol{\theta}_i^{(p)}) - \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{\lambda}_j^\star(\boldsymbol{\theta}_j^{(p)})\right)$$

Distributed Model Predictive Control

Quantity Decomposition | Resource Allocation



Coordinator
M
$\boldsymbol{\lambda}_M^{(p)}$
$\boldsymbol{\theta}_M^{(p+1)}$
⋮
2
$\boldsymbol{\lambda}_2^{(p)}$
$\boldsymbol{\theta}_2^{(p+1)}$
1
$\boldsymbol{\lambda}_1^{(p)}$
$\boldsymbol{\theta}_1^{(p+1)}$
negot
negot
negot
$\boldsymbol{\lambda}_i \rightarrow \boldsymbol{\lambda}$
$\boldsymbol{\lambda}^{(p)}$

$$\left.\begin{aligned} J_i^\star(\boldsymbol{\theta}_i(k)) = \underset{\boldsymbol{u}_i(k:k+N_p-1|k)}{\text{minimize}} \quad & J_i(k) \\ \text{s.t.} \quad & \boldsymbol{x}_i(k+1) = A_i\boldsymbol{x}_i(k) + B_i\boldsymbol{u}_i(k) \\ & \Gamma_i\boldsymbol{u}_i(k) = \boldsymbol{\theta}_i(k) : \boldsymbol{\lambda}_i(k) \end{aligned}\right\} \begin{array}{l} \forall i \in \{1,\ldots,M\} \\ \forall j \in \{1,\ldots,N_p\} \end{array}$$

$$\boldsymbol{\theta}_i^{(p+1)} = \boldsymbol{\theta}_i^{(p)} + \rho\left(\boldsymbol{\lambda}_i^\star(\boldsymbol{\theta}_i^{(p)}) - \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{\lambda}_j^\star(\boldsymbol{\theta}_j^{(p)})\right)$$

CentraleSupélec

Quantity Decomposition | Resource Allocation

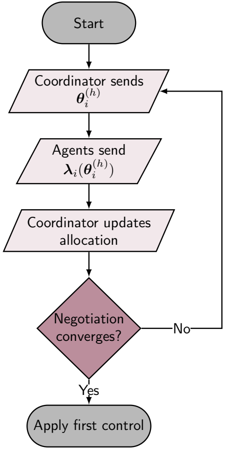

Start
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation
Negotiation converges?
No
Yes
Apply first control

Figure 1: Quantity decomposition based DMPC

Quantity Decomposition | Resource Allocation

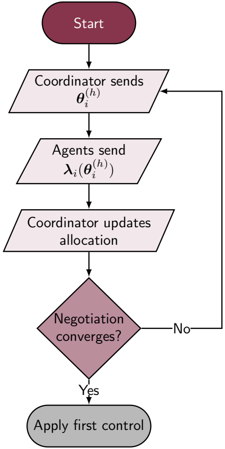

Start
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation
Negotiation converges?
No
Yes
Apply first control

Figure 1: Quantity decomposition based DMPC

Quantity Decomposition | Resource Allocation

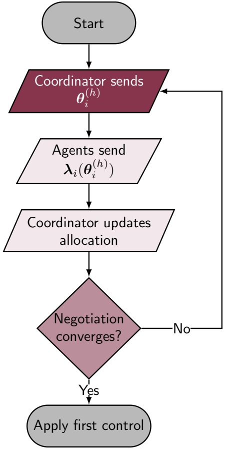

Start
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation
Negotiation converges?
No
Yes
Apply first control

Figure 1: Quantity decomposition based DMPC

Quantity Decomposition | Resource Allocation

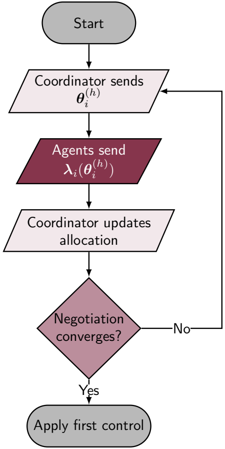

Start
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation
Negotiation converges?
No
Yes
Apply first control

Figure 1: Quantity decomposition based DMPC

Quantity Decomposition | Resource Allocation

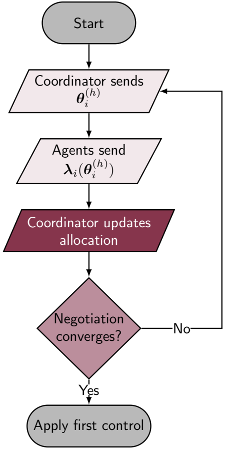

Start
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation
Negotiation converges?
No
Yes
Apply first control

Figure 1: Quantity decomposition based DMPC

Quantity Decomposition | Resource Allocation

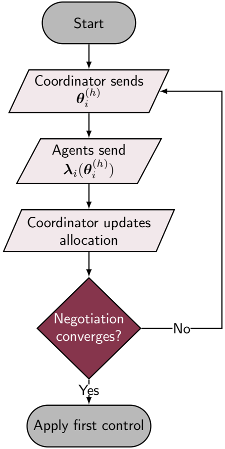

Start
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation
Negotiation converges?
No
Yes
Apply first control

Figure 1: Quantity decomposition based DMPC

Quantity Decomposition | Resource Allocation

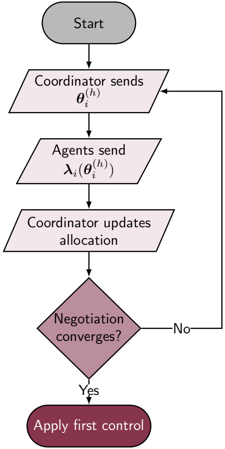

Start
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation
Negotiation converges?
No
Yes
Apply first control

Figure 1: Quantity decomposition based DMPC

What if agents send a non-agreed $\boldsymbol{\lambda}_i$?

Outline

1. Vulnerabilities in distributed MPC based on Resource Allocation
   - Attacks
   - Consequences

2. Securing the DMPC
   - Analysis of Subproblems
   - Detection Mechanism
   - Mitigation Mechanism
   - Complete Mechanism

3. Results

Outline

1. Vulnerabilities in distributed MPC based on Resource Allocation
   - Attacks
   - Consequences
2. Securing the DMPC
   - Analysis of Subproblems
   - Detection Mechanism
   - Mitigation Mechanism
   - Complete Mechanism
3. Results

How can a non-cooperative agent attack?

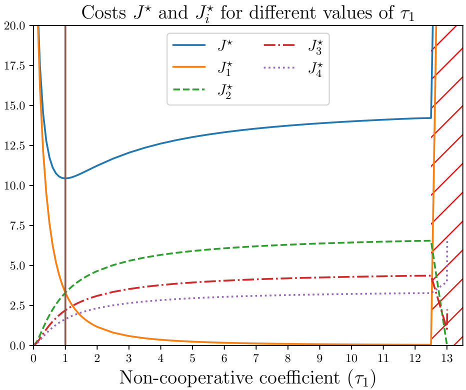
- $\boldsymbol{\lambda}_i$ is the only interface with coordination
- Non-cooperative agent sends $\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i)$

CentraleSupélec

How can a non-cooperative agent attack?

- $\boldsymbol{\lambda}_i$ is the only interface with coordination
- Non-cooperative agent sends $\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i)$

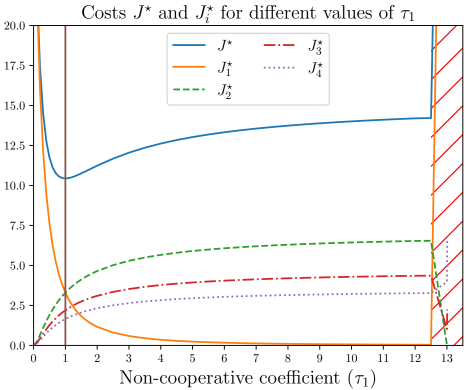
Example



4 distinct agents

- Agent 1 is non-cooperative
- It uses $\tilde{\boldsymbol{\lambda}}_1 = \gamma_1(\boldsymbol{\lambda}_1) = \tau_1 I \boldsymbol{\lambda}_1$

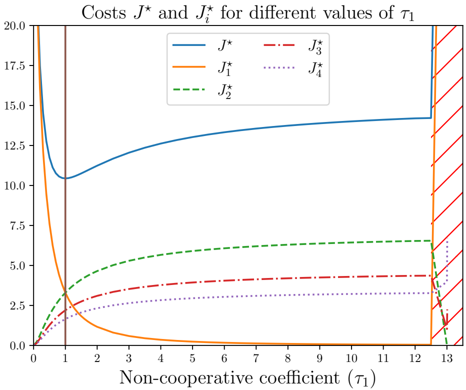
Example



4 distinct agents

- Agent 1 is non-cooperative
- It uses $\tilde{\boldsymbol{\lambda}}_1 = \gamma_1(\boldsymbol{\lambda}_1) = \tau_1 I \boldsymbol{\lambda}_1$

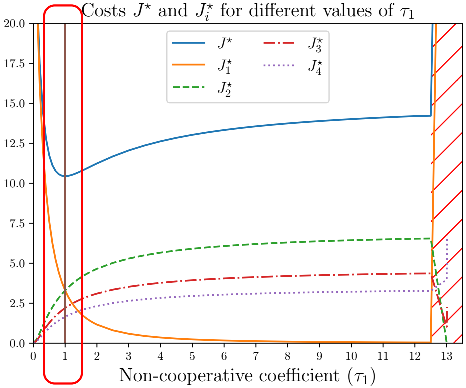
Example



4 distinct agents

- Agent 1 is non-cooperative
- It uses $\tilde{\boldsymbol{\lambda}}_1 = \gamma_1(\boldsymbol{\lambda}_1) = \tau_1 I \boldsymbol{\lambda}_1$

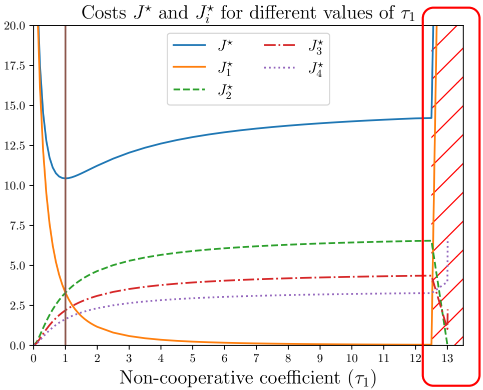
Example



4 distinct agents

- Agent 1 is non-cooperative
- It uses $\tilde{\boldsymbol{\lambda}}_1 = \gamma_1(\boldsymbol{\lambda}_1) = \tau_1 I \boldsymbol{\lambda}_1$

Example



4 distinct agents

- Agent 1 is non-cooperative
- It uses $\tilde{\boldsymbol{\lambda}}_1 = \gamma_1(\boldsymbol{\lambda}_1) = \tau_1 I \boldsymbol{\lambda}_1$

Make Titles Informative.

Outline

1. Vulnerabilities in distributed MPC based on Resource Allocation
   - Attacks
   - Consequences
2. Securing the DMPC
   - Analysis of Subproblems
   - Detection Mechanism
   - Mitigation Mechanism
   - Complete Mechanism
3. Results

CentraleSupélec

Quadratic Case

$$\begin{aligned} \underset{\boldsymbol{u}_i(k:k+N_p-1|k)}{\text{minimize}} \quad & \overbrace{\sum_{j=1}^{N_p} \|\boldsymbol{v}_i(k+j|k)\|_{Q_i}^2 + \|\boldsymbol{u}_i(k+j-1|k)\|_{R_i}^2}^{J_i(k)} \\ \text{s.t.} \quad & \left.\begin{aligned} \boldsymbol{x}_i(\xi+1) = A_i\boldsymbol{x}_i(\xi) + B_i\boldsymbol{u}_i(\xi) \\ \Gamma_i\boldsymbol{u}_i(\xi) = \boldsymbol{\theta}_i(\xi) : \boldsymbol{\lambda}_i(\xi) \end{aligned}\right\} \forall \xi \in \{1,\ldots,N_p\} \end{aligned}$$

Quadratic Case

$$\begin{array}{cc} \underset{\boldsymbol{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2}\boldsymbol{U}_i(k)^T H_i \boldsymbol{U}_i(k) + \boldsymbol{f}_i(k)^T \boldsymbol{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \Theta_i \boldsymbol{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{array}$$

Quadratic Case

$$\begin{aligned} \underset{\boldsymbol{U}_i(k)}{\text{minimize}} \quad & \overbrace{\frac{1}{2}\boldsymbol{U}_i(k)^T {\color{red}H_i} \boldsymbol{U}_i(k) + \boldsymbol{f}_i(k)^T \boldsymbol{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} \quad & \Theta_i \boldsymbol{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{aligned}$$

Quadratic Case

$$\begin{aligned} \underset{\boldsymbol{U}_i(k)}{\text{minimize}} \quad & \overbrace{\frac{1}{2}\boldsymbol{U}_i(k)^T H_i \boldsymbol{U}_i(k) + \boldsymbol{f}_i(k)^T \boldsymbol{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} \quad & \Theta_i \boldsymbol{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{aligned}$$

Quadratic Case

$$\begin{array}{cc} \underset{\boldsymbol{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2}\boldsymbol{U}_i(k)^T H_i \boldsymbol{U}_i(k) + \boldsymbol{f}_i(k)^T \boldsymbol{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \Theta_i \boldsymbol{U}_i(k) = \boldsymbol{\theta}_i : \color{red}{\boldsymbol{\lambda}_i} \end{array}$$

Quadratic Case

$$\begin{array}{cc} \underset{\boldsymbol{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2}\boldsymbol{U}_i(k)^T H_i \boldsymbol{U}_i(k) + \boldsymbol{f}_i(k)^T \boldsymbol{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \Theta_i \boldsymbol{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{array}$$

$$\boldsymbol{\lambda}_i = -P_i \boldsymbol{\theta}_i - \boldsymbol{s}_i(k)$$

where $P_i = \left(\Theta_i H_i^{-1} \Theta_i^{\mathrm{T}}\right)^{-1}$ and $\boldsymbol{s}_i(k) = P_i \Theta_i H_i^{-1} \boldsymbol{f}_i(k)$

Quadratic Case

$$\begin{array}{cc} \underset{\boldsymbol{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2}\boldsymbol{U}_i(k)^T H_i \boldsymbol{U}_i(k) + \boldsymbol{f}_i(k)^T \boldsymbol{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \Theta_i \boldsymbol{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{array}$$

$$\boldsymbol{\lambda}_i = -P_i \boldsymbol{\theta}_i - \boldsymbol{s}_i(k)$$

where $P_i = \left(\Theta_i H_i^{-1} \Theta_i^{\mathrm{T}}\right)^{-1}$ and $\boldsymbol{s}_i(k) = P_i \Theta_i H_i^{-1} \boldsymbol{f}_i(k)$

Quadratic Case

$$\begin{array}{cc} \underset{\boldsymbol{U}_i(k)}{\text{minimize}} & \overbrace{\frac{1}{2}\boldsymbol{U}_i(k)^T H_i \boldsymbol{U}_i(k) + \boldsymbol{f}_i(k)^T \boldsymbol{U}_i(k)}^{J_i(\boldsymbol{\theta}_i)} \\ \text{s.t.} & \Theta_i \boldsymbol{U}_i(k) = \boldsymbol{\theta}_i : \boldsymbol{\lambda}_i \end{array}$$

$$\boldsymbol{\lambda}_i = -P_i \boldsymbol{\theta}_i - {\color{red}\boldsymbol{s}_i(k)}$$

where $P_i = (\Theta_i H_i^{-1} \Theta_i^{\mathrm{T}})^{-1}$ and ${\color{red}\boldsymbol{s}_i(k)} = P_i \Theta_i H_i^{-1} \boldsymbol{f}_i(k)$

Detection

Assumption

We know nominal $\bar{P}_i$

Assumption

Attacker chooses $\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i) = T_i(k)\boldsymbol{\lambda}_i \to -T_i(k)P_i\boldsymbol{\theta}_i - T_i(k)\boldsymbol{s}_i(k)$

- We can estimate[1] $\hat{P}_i$ and $\widehat{\tilde{\boldsymbol{s}}}_i(k)$ such as:

$$\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i)) = -\widehat{\tilde{P}}_i(k)\boldsymbol{\theta}_i - \widehat{\tilde{\boldsymbol{s}}}_i(k)$$

- If $\widehat{\tilde{P}}_i(k) \neq \bar{P}_i \to$ Attack

[1] Using Recursive Least Squares

Detection

Assumption

We know nominal $\bar{P}_i$

Assumption

Attacker chooses $\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i) = T_i(k)\boldsymbol{\lambda}_i \rightarrow -T_i(k)P_i\boldsymbol{\theta}_i - T_i(k)\boldsymbol{s}_i(k)$

- We can estimate[1] $\hat{P}_i$ and $\widehat{\tilde{\boldsymbol{s}}}_i(k)$ such as:

$$\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i)) = -\widehat{\tilde{P}}_i(k)\boldsymbol{\theta}_i - \widehat{\tilde{\boldsymbol{s}}}_i(k)$$

- If $\widehat{\tilde{P}}_i(k) \neq \bar{P}_i \rightarrow$ Attack

[1] Using Recursive Least Squares

CentraleSupélec

Detection

Assumption

We know nominal $\bar{P}_i$

Assumption

Attacker chooses $\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i) = T_i(k)\boldsymbol{\lambda}_i \to -T_i(k)P_i\boldsymbol{\theta}_i - T_i(k)\boldsymbol{s}_i(k)$

- We can estimate[1] $\hat{P}_i$ and $\widehat{\tilde{\boldsymbol{s}}}_i(k)$ such as:

$$\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i)) = -\widehat{\tilde{P}}_i(k)\boldsymbol{\theta}_i - \widehat{\tilde{\boldsymbol{s}}}_i(k)$$

- If $\widehat{\tilde{P}}_i(k) \neq \bar{P}_i \to$ Attack

[1] Using Recursive Least Squares

Detection

Assumption

We know nominal $\bar{P}_i$

Assumption

Attacker chooses $\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i) = T_i(k)\boldsymbol{\lambda}_i \to -T_i(k)P_i\boldsymbol{\theta}_i - T_i(k)\boldsymbol{s}_i(k)$

- We can estimate[1] $\hat{P}_i$ and $\widehat{\tilde{\boldsymbol{s}}}_i(k)$ such as:

$$\tilde{\boldsymbol{\lambda}}_i = \gamma_i(\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i)) = -\widehat{\tilde{P}}_i(k)\boldsymbol{\theta}_i - \widehat{\tilde{\boldsymbol{s}}}_i(k)$$

- If $\widehat{\tilde{P}}_i(k) \neq \bar{P}_i \to$ Attack

[1] Using Recursive Least Squares

About Estimation

- We estimate $\hat{P}_i$ and $\widehat{\tilde{\boldsymbol{s}}}_i(k)$ simultaneously using Recursive Least Squares
- Problem: Estimation during negotiation fails
  - Consecutive $\boldsymbol{\lambda}_i^p$ and $\boldsymbol{\theta}_i^p$ are linearly dependent → low input excitation
- Solution: Send sequence of random values of $\boldsymbol{\theta}_i$ until estimates converge

About Estimation

- We estimate $\hat{P}_i$ and $\widehat{\tilde{\boldsymbol{s}}}_i(k)$ simultaneously using Recursive Least Squares
- Problem: Estimation during negotiation fails
  - Consecutive $\boldsymbol{\lambda}_i^p$ and $\boldsymbol{\theta}_i^p$ are linearly dependent → low input excitation
- Solution: Send sequence of random values of $\boldsymbol{\theta}_i$ until estimates converge

About Estimation

- We estimate $\hat{P}_i$ and $\widehat{\tilde{\boldsymbol{s}}}_i(k)$ simultaneously using Recursive Least Squares
- Problem: Estimation during negotiation fails
  - Consecutive $\boldsymbol{\lambda}_i^p$ and $\boldsymbol{\theta}_i^p$ are linearly dependent → low input excitation
- Solution: Send sequence of random values of $\boldsymbol{\theta}_i$ until estimates converge

CentraleSupélec

About Estimation

- We estimate $\hat{P}_i$ and $\widehat{\tilde{\boldsymbol{s}}}_i(k)$ simultaneously using Recursive Least Squares
- Problem: Estimation during negotiation fails
  - Consecutive $\boldsymbol{\lambda}_i^p$ and $\boldsymbol{\theta}_i^p$ are linearly dependent $\rightarrow$ low input excitation
- Solution: Send sequence of random values of $\boldsymbol{\theta}_i$ until estimates converge

Detection

In detail

- Error $E_i(k) = \|\widehat{\tilde{P}}_i(k) - \bar{P}_i\|_F$
- Create threshold ϵ_P
- Indicator $d_i \in \{0,1\}$ detects the attack in agent i.
- $d_i = 1$ if $E_i(k) > \epsilon_P$, 0 otherwise

Detection

In detail

- Error $E_i(k) = \|\widehat{\tilde{P}}_i(k) - \bar{P}_i\|_F$
- Create threshold ϵ_P
- Indicator $d_i \in \{0, 1\}$ detects the attack in agent i.
- $d_i = 1$ if $E_i(k) > \epsilon_P$, 0 otherwise

Detection

In detail

- Error $E_i(k) = \|\widehat{\tilde{P}}_i(k) - \bar{P}_i\|_F$
- Create threshold ϵ_P
- Indicator $d_i \in \{0, 1\}$ detects the attack in agent i.
- $d_i = 1$ if $E_i(k) > \epsilon_P$, 0 otherwise

Detection

In detail

- Error $E_i(k) = \|\widehat{\tilde{P}}_i(k) - \bar{P}_i\|_F$
- Create threshold ϵ_P
- Indicator $d_i \in \{0, 1\}$ detects the attack in agent i.
- $d_i = 1$ if $E_i(k) > \epsilon_P$, 0 otherwise

Mitigation

- Main idea: Reconstruct $\boldsymbol{\lambda}_i$ and use in negotiation

Assumption

We suppose $\tilde{\boldsymbol{\lambda}}_i = \mathbf{0}$ only if $\boldsymbol{\lambda}_i = \mathbf{0}$, which implies $T_i(k)$ invertible.

- Estimate the inverse of $T_i(k)$

$$\widehat{T_i(k)^{-1}} = \bar{P}_i \widehat{\tilde{P}}_i(k)^{-1}$$

- Reconstruct $\boldsymbol{\lambda}_i$

$$\boldsymbol{\lambda}_{i\text{rec}} = \widehat{T_i(k)^{-1}} \tilde{\boldsymbol{\lambda}}_i = -\bar{P}_i \boldsymbol{\theta}_i - \widehat{T_i(k)^{-1}} \widehat{\tilde{\boldsymbol{s}}}_i(k)$$

Mitigation

- Main idea: Reconstruct $\boldsymbol{\lambda}_i$ and use in negotiation

Assumption

We suppose $\tilde{\boldsymbol{\lambda}}_i = \mathbf{0}$ only if $\boldsymbol{\lambda}_i = \mathbf{0}$, which implies $T_i(k)$ invertible.

- Estimate the inverse of $T_i(k)$

$$\widehat{T_i(k)^{-1}} = \bar{P}_i \widehat{\tilde{P}}_i(k)^{-1}$$

- Reconstruct $\boldsymbol{\lambda}_i$

$$\boldsymbol{\lambda}_{i\text{rec}} = \widehat{T_i(k)^{-1}} \tilde{\boldsymbol{\lambda}}_i = -\bar{P}_i \boldsymbol{\theta}_i - \widehat{T_i(k)^{-1}} \hat{\boldsymbol{s}}_i(k)$$

Mitigation

- Main idea: Reconstruct $\boldsymbol{\lambda}_i$ and use in negotiation

Assumption

We suppose $\tilde{\boldsymbol{\lambda}}_i = \mathbf{0}$ only if $\boldsymbol{\lambda}_i = \mathbf{0}$, which implies $T_i(k)$ invertible.

- Estimate the inverse of $T_i(k)$

$$\widehat{T_i(k)^{-1}} = \bar{P}_i \widehat{\tilde{P}}_i(k)^{-1}$$

- Reconstruct $\boldsymbol{\lambda}_i$

$$\boldsymbol{\lambda}_{i\text{rec}} = \widehat{T_i(k)^{-1}} \tilde{\boldsymbol{\lambda}}_i = -\bar{P}_i \boldsymbol{\theta}_i - \widehat{T_i(k)^{-1}} \hat{\boldsymbol{s}}_i(k)$$

CentraleSupélec

Mitigation

- Main idea: Reconstruct $\boldsymbol{\lambda}_i$ and use in negotiation

Assumption

We suppose $\tilde{\boldsymbol{\lambda}}_i = \mathbf{0}$ only if $\boldsymbol{\lambda}_i = \mathbf{0}$, which implies $T_i(k)$ invertible.

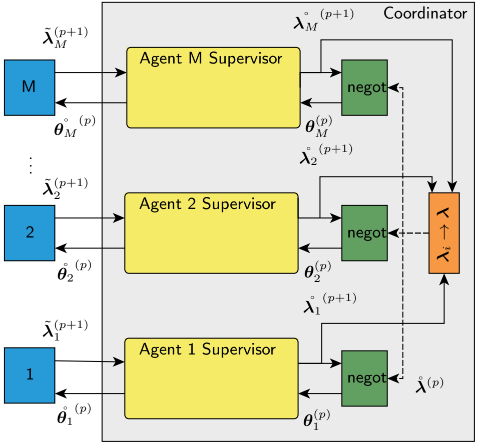
- Estimate the inverse of $T_i(k)$

$$\widehat{T_i(k)^{-1}} = \bar{P}_i \widehat{\tilde{P}}_i(k)^{-1}$$

- Reconstruct $\boldsymbol{\lambda}_i$

$$\boldsymbol{\lambda}_{i\mathrm{rec}} = \widehat{T_i(k)^{-1}} \tilde{\boldsymbol{\lambda}}_i = -\bar{P}_i \boldsymbol{\theta}_i - \widehat{T_i(k)^{-1}} \widehat{\tilde{\boldsymbol{s}}}_i(k)$$

Complete Mechanism


Coordinator
$\overset{\circ}{\boldsymbol{\lambda}}_M^{(p+1)}$
$\tilde{\boldsymbol{\lambda}}_M^{(p+1)}$
M
Agent M Supervisor
negot
$\overset{\circ}{\boldsymbol{\theta}}_M^{(p)}$
$\boldsymbol{\theta}_M^{(p)}$
$\overset{\circ}{\boldsymbol{\lambda}}_2^{(p+1)}$
⋮
$\tilde{\boldsymbol{\lambda}}_2^{(p+1)}$
2
Agent 2 Supervisor
negot
$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}_i$
$\overset{\circ}{\boldsymbol{\theta}}_2^{(p)}$
$\boldsymbol{\theta}_2^{(p)}$
$\overset{\circ}{\boldsymbol{\lambda}}_1^{(p+1)}$
$\tilde{\boldsymbol{\lambda}}_1^{(p+1)}$
1
Agent 1 Supervisor
negot
$\overset{\circ}{\boldsymbol{\lambda}}^{(p)}$
$\overset{\circ}{\boldsymbol{\theta}}_1^{(p)}$
$\boldsymbol{\theta}_1^{(p)}$

Two phases:

1. Detect which agents are non-cooperative
2. Reconstruct $\boldsymbol{\lambda}_i$ and use in negotiation

Secure DMPC

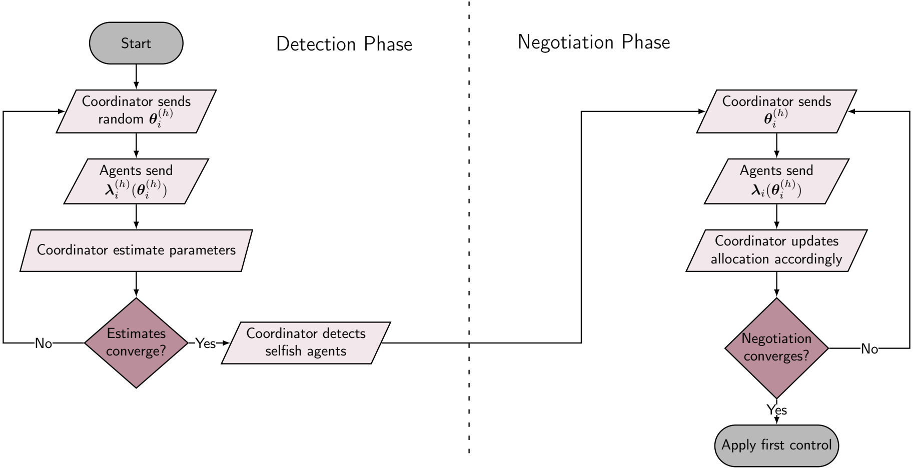

Detection Phase
Negotiation Phase
Start
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

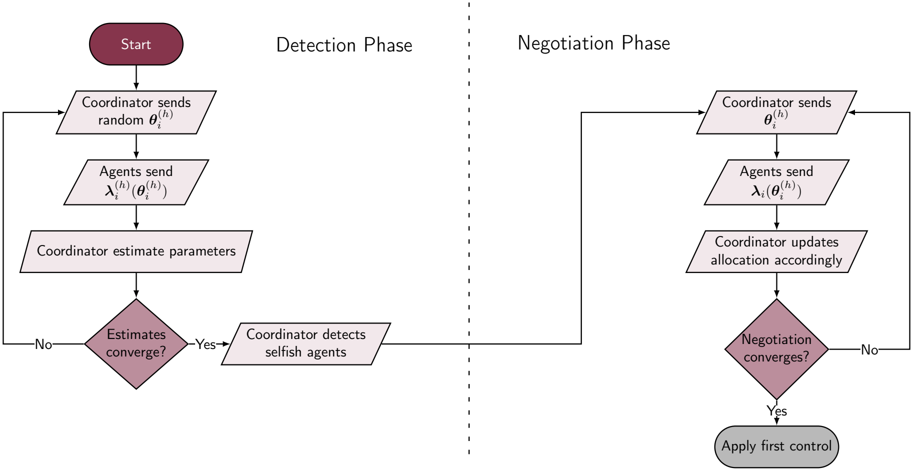

Detection Phase
Negotiation Phase
Start
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

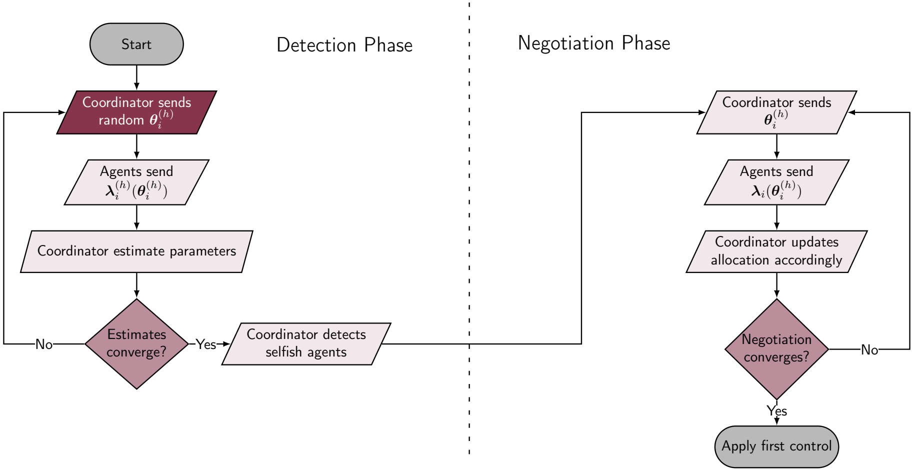

Detection Phase
Negotiation Phase
Start
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

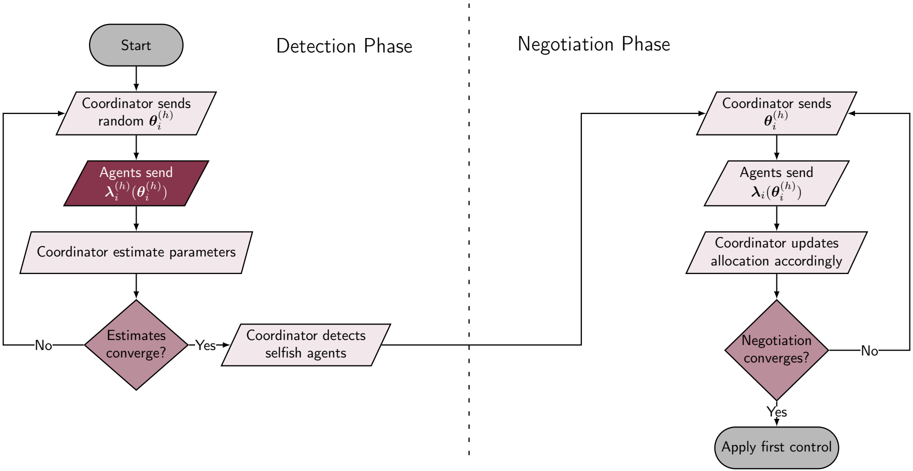

Detection Phase
Negotiation Phase
Start
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

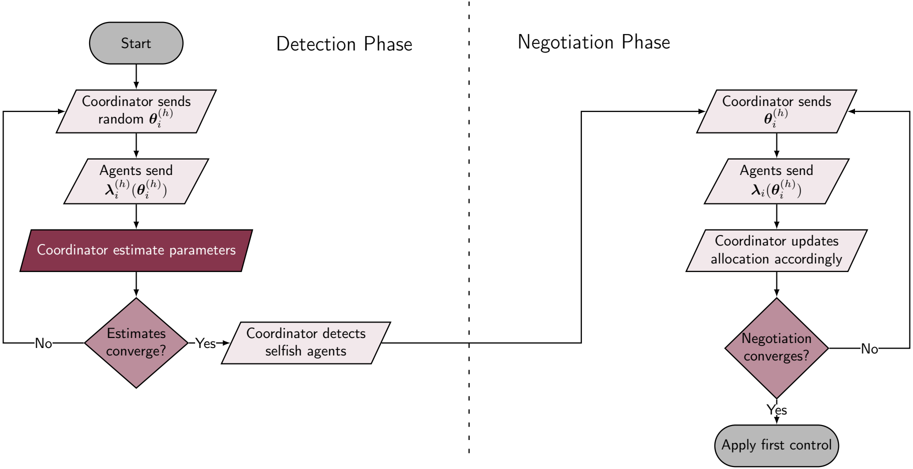

Start
Detection Phase
Negotiation Phase
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

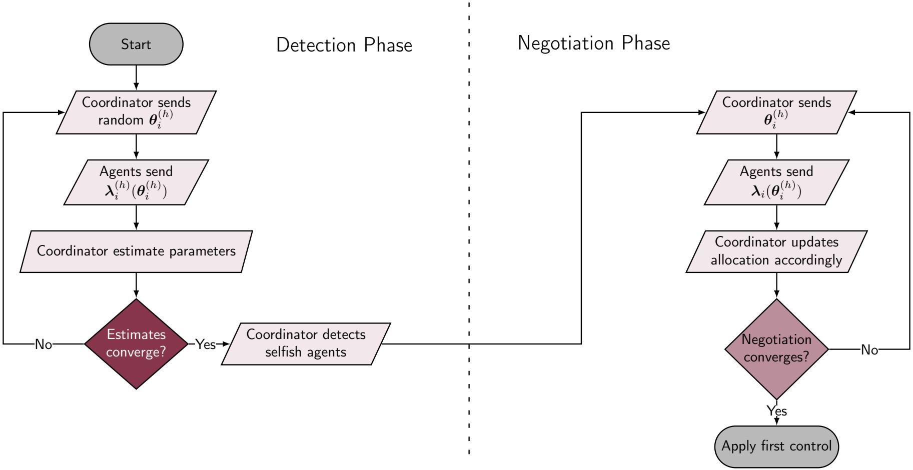

Detection Phase
Negotiation Phase
Start
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

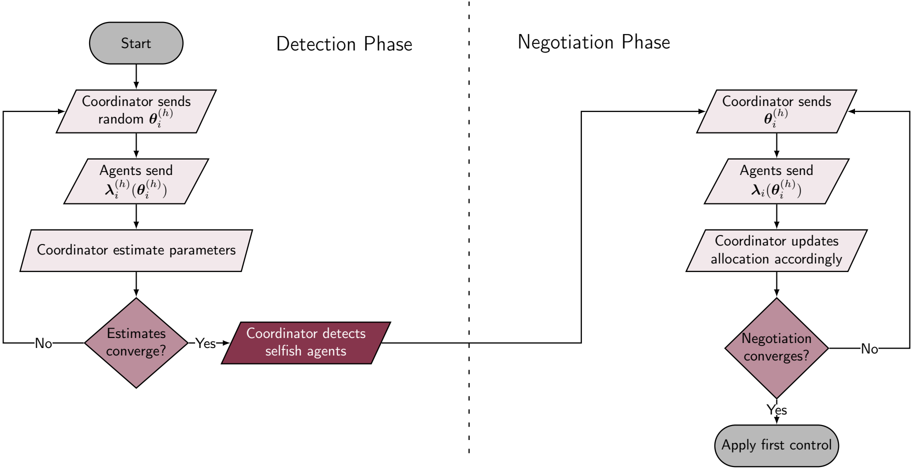

Detection Phase
Negotiation Phase
Start
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

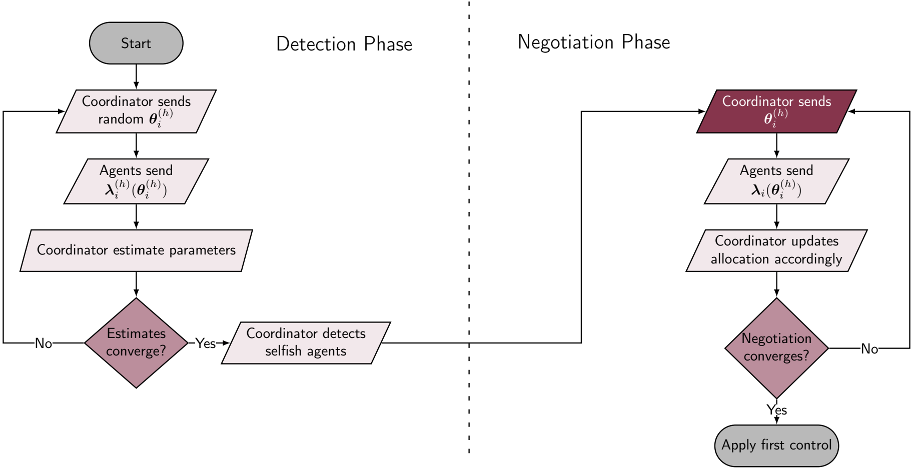

Start
Detection Phase
Negotiation Phase
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

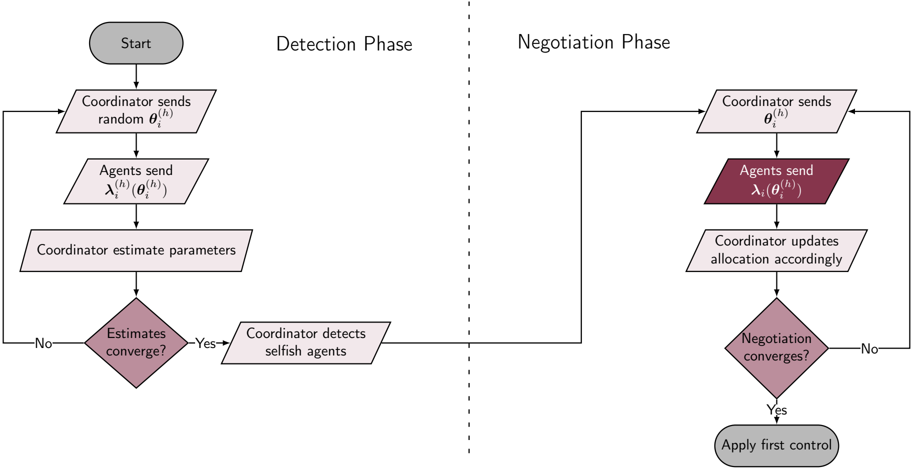

Start
Detection Phase
Negotiation Phase
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

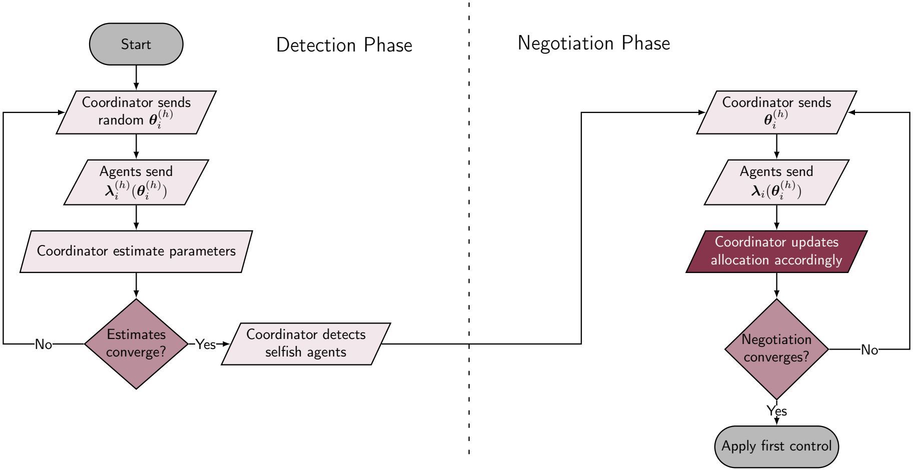

Detection Phase
Negotiation Phase
Start
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

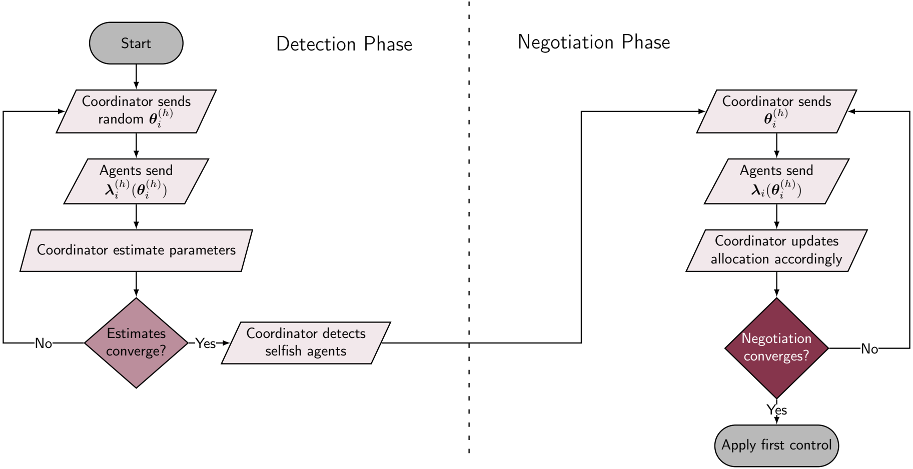

Detection Phase
Negotiation Phase
Start
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Secure DMPC

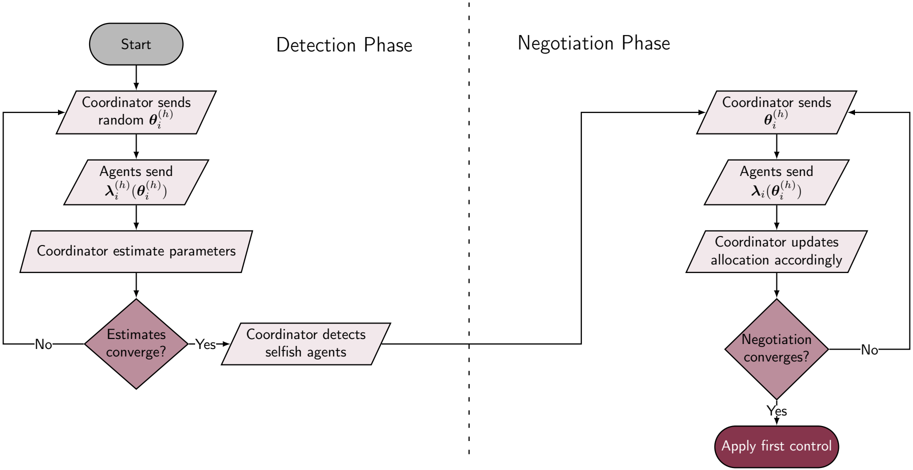

Start
Detection Phase
Negotiation Phase
Coordinator sends random $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i^{(h)}(\boldsymbol{\theta}_i^{(h)})$
Coordinator estimate parameters
Estimates converge?
No
Yes
Coordinator detects selfish agents
Coordinator sends $\boldsymbol{\theta}_i^{(h)}$
Agents send $\boldsymbol{\lambda}_i(\boldsymbol{\theta}_i^{(h)})$
Coordinator updates allocation accordingly
Negotiation converges?
No
Yes
Apply first control

Figure 2: Secure DMPC

Outline

1. Vulnerabilities in distributed MPC based on Resource Allocation
   - Attacks
   - Consequences
2. Securing the DMPC
   - Analysis of Subproblems
   - Detection Mechanism
   - Mitigation Mechanism
   - Complete Mechanism
3. Results

Example

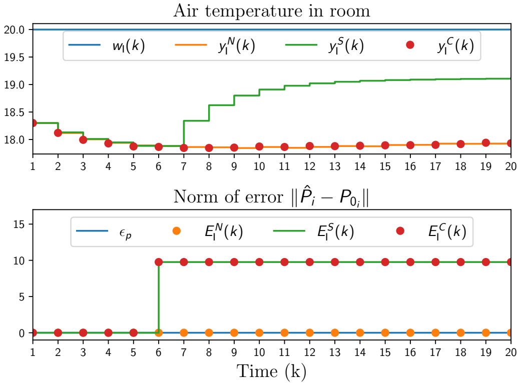
Temperature Control of 4 Distinct Rooms Under Power Scarcity

- 4 distinct rooms modeled using 3R-2C
- Initial temperature under $20^o\mathrm{C}$
- Not enough power to achieve setpoint $\left(\sum_{i=1}^{4} \boldsymbol{u}_i(k) \leq 4\mathrm{kW}\right)$
- Simulated for a period of $5\mathrm{h}$
- ZOH $T_s = 0.25\mathrm{h}$

CentraleSupélec

Results

Temporal



N Nominal

S Selfish behavior

C selfish behavior with Correction

Results

Table 1: Comparison of costs J_i^N and J_G^N

Agent	Nominal	Selfish	Selfish + correction
I	103	64	104
II	73	91	73
III	100	123	101
IV	132	154	131
Global	408	442	409

Summary

1. Resource allocation based DMPC is vulnerable to attacks.
2. Sub-problems' structure has time invariant parameters.
3. Attacks can be detected using these parameters.
4. Effects can be mitigated.

- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model

Summary

1. Resource allocation based DMPC is vulnerable to attacks.
2. Sub-problems' structure has time invariant parameters.
3. Attacks can be detected using these parameters.
4. Effects can be mitigated.

- Outlook
  - Inequality Constraints yield Hybrid behavior
  - Non-linear attack model

Summary

1. Resource allocation based DMPC is vulnerable to attacks.
2. Sub-problems' structure has time invariant parameters.
3. Attacks can be detected using these parameters.
4. Effects can be mitigated.

- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model

CentraleSupélec

Summary

1. Resource allocation based DMPC is vulnerable to attacks.
2. Sub-problems' structure has time invariant parameters.
3. Attacks can be detected using these parameters.
4. Effects can be mitigated.

- Outlook
  - Inequality Constraints yield Hybrid behavior
  - Non-linear attack model

CentraleSupélec

Summary

1. Resource allocation based DMPC is vulnerable to attacks.
2. Sub-problems' structure has time invariant parameters.
3. Attacks can be detected using these parameters.
4. Effects can be mitigated.

- Outlook
  - Inequality Constraints yield Hybrid behavior
  - Non-linear attack model

Summary

1. Resource allocation based DMPC is vulnerable to attacks.
2. Sub-problems' structure has time invariant parameters.
3. Attacks can be detected using these parameters.
4. Effects can be mitigated.

- Outlook
  - Inequality Constraints yield Hybrid behavior
  - Non-linear attack model

Summary

1. Resource allocation based DMPC is vulnerable to attacks.
2. Sub-problems' structure has time invariant parameters.
3. Attacks can be detected using these parameters.
4. Effects can be mitigated.

- Outlook
  - Inequality Constraints yield Hybrid behavior
  - Non-linear attack model

CentraleSupélec

Summary

1. Resource allocation based DMPC is vulnerable to attacks.
2. Sub-problems' structure has time invariant parameters.
3. Attacks can be detected using these parameters.
4. Effects can be mitigated.

- Outlook
    - Inequality Constraints yield Hybrid behavior
    - Non-linear attack model

For Further Reading I

J. M. Maestre, R. R. Negenborn *et al.*
Distributed Model Predictive Control made easy.
Springer, 2014, vol. 69.

P. Velarde, J. M. Maestre, H. Ishii, and R. R. Negenborn,
"Scenario-based defense mechanism for distributed model predictive control,"
2017 IEEE 56th Annual Conference on Decision and Control (CDC). IEEE, Dec 2017, pp. 6171–6176.

Questions?

Repository
https://github.com/Accacio/SysTol-21

Contact
rafael-accacio.nogueira@centralesupelec.fr