Deploying Robust Security in IoT

Ruozhou Yu, **Guoliang Xue**, Vishnu Teja Kilari, Xiang Zhang

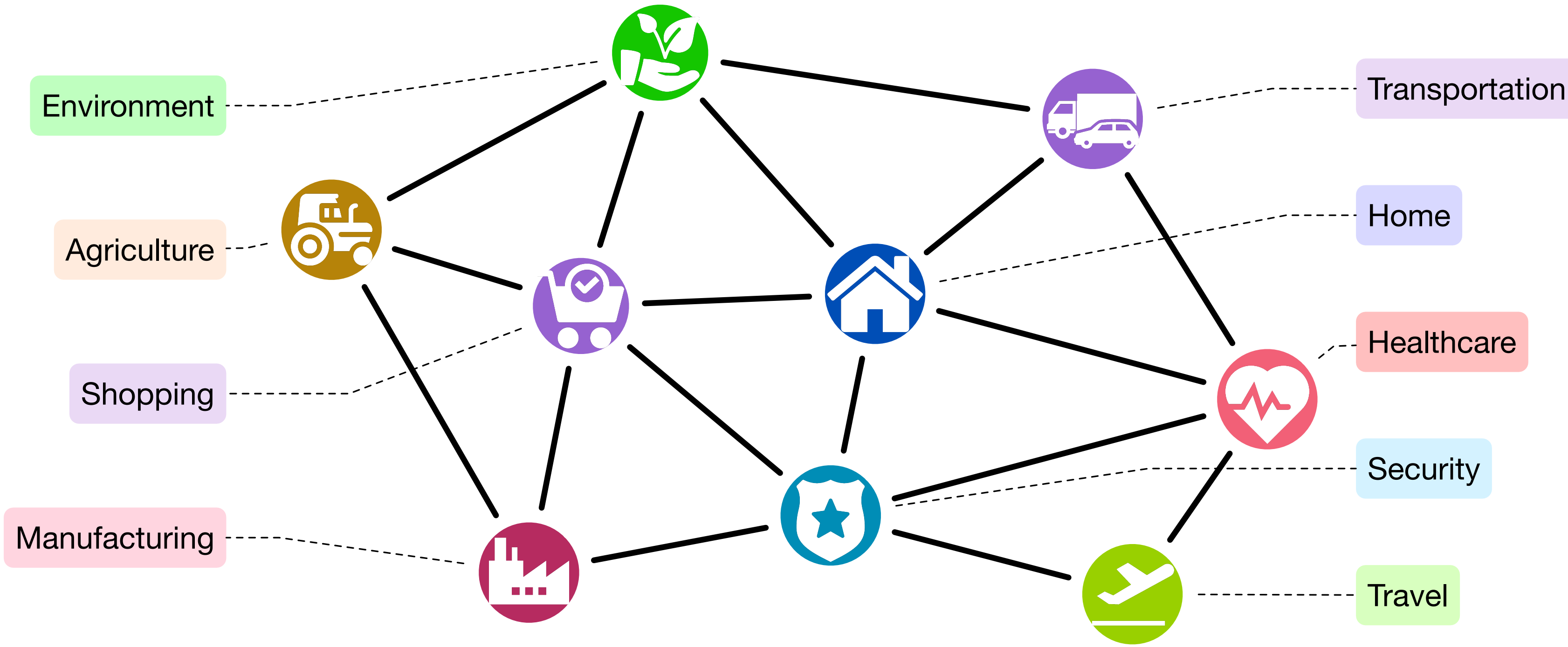
Arizona State University

Outlines

- Introduction and Methodology Overview
- System Model
- Optimization Framework
- Performance Evaluation
- Conclusions

IoT: The Future Internet

- IoT is the future Internet that connects every aspect of our work and life.



New Threats?

Top: https://www.techrepublic.com/article/ddos-attacks-increased-91-in-2017-thanks-to-iot/
Right: https://www.welivesecurity.com/2016/10/24/10-things-know-october-21-iot-ddos-attacks/
Left: https://securityintelligence.com/the-weaponization-of-iot-rise-of-the-thingbots/
Bottom: https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/

What's the problem?

- **Careless people**
  - Default / Weak username + password
    - Mirai Botnet: largest-ever DDoS attack on Dyn, Oct 21, 2016
  - Obsolete firmware / software
  - Misused security settings
    - Authorization, access control, network settings, …
  - Data security

- **Constrained and vulnerable devices**
  - Computing power
  - Energy
  - Memory
  - Hardware deficits
  - Unrevealed vulnerabilities

Current Progresses

- **Lightweight crypto** for constrained devices
  - Active on-going research efforts
  - *Not quite practical in major IoT scenarios…*
    - Difficult on small devices: RFID, light bulbs, smart switches, cameras, …
    - Cannot protect system from careless/malicious users

- **Security offloading**
  - Offload part of / all security functions to helper nodes in the network
    - Fog nodes, cloud, security providers, …
  - Can protect both users and the system
    - User-oriented security vs. system-oriented security
  - *Inevitable security risk of offloading*
    - Unprotected/unmonitored traffic before processing
    - Prolonged security procedure: more vulnerable to opportunistic attacks

Our Standing

- Operator as a central security enforcer
  - Monitors network-wide user traffic
    - Traffic classification based on access/exit, QoS, policy
    - Aggregate periodic network status and user demand reports
  - Security function deployment / adjustment
    - Minimize security risk of offloading
    - Based on overall cost budget, predicted user demands and network status
    - Can be periodically adjusted based on historical data
  - User traffic steering
    - Direct user traffic to nearest / selected security functions
    - Different steering techniques can be used here
    - In this work we assume nearest selection and shortest path routing

Methodology Overview

Inputs:

User Demands
- Traffic volumes at APs

Network Status
- Topology & availability

System-wide Optimization:

Abstract System Model
- System uncertainties
- Security risk model
- Robustness model

Optimization Framework
- Benders' (row) decomposition
- Efficient subproblem solving

Outputs:

Security Deployment
- Subject to cost budget

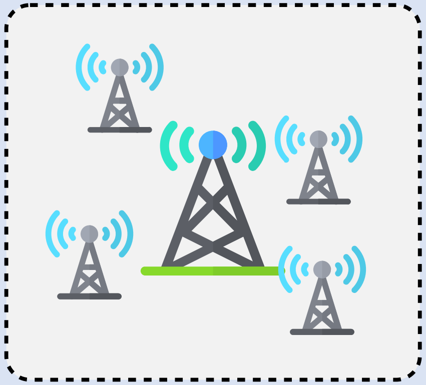
Traffic Steering
- Selected security func.

Outlines

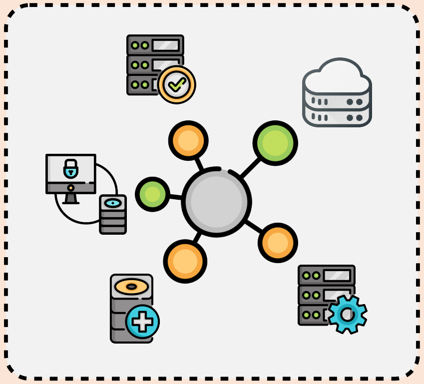
- Introduction and Methodology Overview
- System Model
- Optimization Framework
- Performance Evaluation
- Conclusions

IoT Network: A General Model

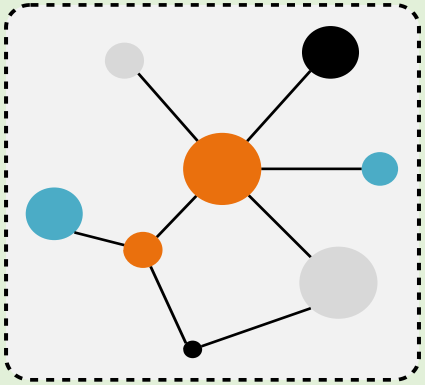
- Challenge: heterogeneous network environments



Wireless RANs:
- Geo-distributed
- Limited capacity
- Interference



Edge Network:
- Complex topo
- Distributed
- Dynamic load



Backbones:
- Large-scale
- High latency
- ISP policies

- Model: general directed graph G=(V, E), with fog nodes F and APs A
  - Weights: hop, delay, negative log safe probability, …

Measurement of Security Risk

- User demands: # devices at APs
  - Extensible to traffic volumes, different device types, etc.

- Security risk:
  - Average amount of unmonitored/unprotected traffic per unit demand.
  - Assuming shortest-path to nearest security functions:
    - Security risk of device = shortest path distance to nearest security function.
    - Security risk of system = $\sum$ distances / total demand
    - Extensible to maximum distance per demand, etc.

- What affect security risk:
  - Different user demands at APs
  - Different topology information
  - Deployment of security functions

Uncertainties in IoT

- IoT is dynamic: both user demands and topology
  - Fluctuating user demands, due to
    - New devices, device mobility, events, failures and maintenance, …
  - **Model**: random variables $D = \{ d_a \in \mathbb{R}^* \mid a \in A \}$

  - Volatile topology, due to
    - Device mobility, interference, congestion, failures and maintenance, …
  - **Model**: random variables $Y = \{ y_e \in \{0, 1\} \mid e \in E \}$

  - **Realization**: observed values of the random variables
  - $\Pi = (\bar{D}, \bar{Y})$: a realization of system state

- Security risk $R(X, D, Y)$: a function of random variables D and Y.
  - Depends on security deployment $X = \{ x_v \in \{0, 1\} \mid v \in F \}$.

SO and CVaR

- **Stochastic Optimization (SO)**: optimize a function in presence of randomness (random objective and/or random constraints)
  - Traditional approach: expectation optimization

$$\min_X \quad \mathbb{E}[\, R(X, D, Y) \,]$$

  - **Issue**: unbounded risk in rare but unfortunate scenarios
    - E.g., abnormal demands due to public events, rare large-scale failures, …

  - How to model these unfortunate scenarios?
  - Value-at-Risk (VaR) and Conditional-Value-at-Risk (CVaR):
    - Widely used in economics and finance
    - $\mathrm{VaR}_\alpha(R) = \min \{\, c \in \mathbb{R} \mid R$ does not exceed c with at least α prob. $\}$
    - $\mathrm{CVaR}_\alpha(R) = \mathbb{E}[\, R \mid R \geq \mathrm{VaR}_\alpha(R) \,]$
      - Expectation of R in the worst $(1-\alpha)$ scenarios

  - **Our approach**: optimize both expectation and CVaR

$$\min_X \quad \mathbb{E}[\, R(X, D, Y) \,] + \rho\, \mathrm{CVaR}_\alpha(\, R(X, D, Y) \,)$$

Rockafellar-Uryasev Theorem

- Computing CVaR requires the value of VaR?
- Rockafellar-Uryasev [RU2000]:
  - Computation of CVaR does not need VaR beforehand.

$$\mathrm{CVaR}_{\alpha}(R) = \min_c \{ c + \frac{1}{1-\alpha}\mathbb{E}[(R - c)^{+}] \}$$

  - $\mathrm{VaR}_{\alpha}(R) = \mathrm{argmin}_c \{ c + \frac{1}{1-\alpha}\mathbb{E}[(R - c)^{+}] \}$: jointly computed
  - $(z)^{+}$: $\max\{z, 0\}$

- A transformed formulation for our problem

$$\min_{X,c} \qquad \mathbb{E}[R(X, D, Y)] + \rho \left(c + \frac{1}{1-\alpha}\mathbb{E}[(R - c)^{+}] \right)$$

  - (because both problems are minimizations…)

ARIZONA STATE UNIVERSITY

[RU2000] R. T. Rockafellar and S. Uryasev, "Optimization of Conditional Value-at-Risk," *J. Risk*, vol. 2, pp. 21–41, 2000.

Sample Average Approximation

- How to optimize R(X, D, Y) in face of D and Y?
  - **Challenge 1**: hard to model underlying distribution.
  - **Challenge 2**: R(X, D, Y) hard to write in closed-form.

- Sample Average Approximation (SAA):
  - Approximate expectations as sample averages
  - How to sample D and Y: historical network measurement data
    - Regard historical data as samples from the real-world distributions

- Scenario-based optimization: generate N samples $\mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_N$

$$\min_{X,c} \quad \frac{1}{N}\sum_{i=1}^{N}\bar{R}_i + \rho\left(c + \frac{1}{1-\alpha}\frac{1}{N}\sum_{i=1}^{N}(\bar{R}_i - c)^+\right)$$

  - $\bar{R}_i = R(X, \bar{D}_i, \bar{Y}_i)$: security risk of scenario i, for i=1…N.

The Overall Problem

- Master Problem

$$\min_{X,c} \quad \frac{1}{N}\sum_{i=1}^{N}\bar{R}_i + \rho\left(c + \frac{1}{1-\alpha}\frac{1}{N}\sum_{i=1}^{N}(\bar{R}_i - c)^+\right)$$

$$\text{s.t.} \quad \sum_{v} c_v x_v \le b$$

- Slave Problem ($\bar{R}_i$)

$$R(X, \overline{D}_i, \overline{Y}_i) =$$

$$\min_{t} \quad \frac{1}{d^i_{\text{sum}}}\sum_{a\in A} d^i_a \sum_{v\in F} \text{dist}^i_a(v) t^i_a(v) \tag{1a}$$

$$\text{s.t.} \quad \sum_{v} t^i_a(v) = 1, \quad \forall a; \tag{1b}$$

$$t^i_a(v) \le x_v, \quad \forall a, v; \tag{1c}$$

$$t^i_a(v) \in [0,1], \quad \forall a, v. \tag{1d}$$

Outlines

- Introduction and Methodology Overview
- System Model
- Optimization Framework
- Performance Evaluation
- Conclusions

Decomposition Framework

- Two-stage SO:
  - **Master Problem**: integer programming, size linear to |F| (# fog nodes)
  - **Slave Problem**: linear programming, size linear to N·|A|·|F| (N: # samples)
    - Decomposable to N independent per-scenario LPs of sizes |A|·|F|
  - In practice, N >> |F|:
    - # fog nodes: let's say 10-100
    - # samples: at least 1000 to get a good approximation

  - **Benders' decomposition**: (Row Generation) In each iteration, add new constraints (cuts) to the problem that push the master towards the optimal:
    - INIT: feasible master solution; then proceed in iterations:
      - Solve slave dual problem based on master solution (UB).
      - If dual slave unbounded, add feasibility cut to master;
        if dual slave optimal, add optimality cut to master.
      - Solve updated master (LB).
    - Until UB – LB < ϵ.

Speeding-up Slave Dual Solving

- How to solve the slave dual?
  1. Solve the whole linear program.
     - Cubic time complexity to entire program size N·|A|·|F|.
  2. Solve for each independent scenario, then aggregate.
     - Cubic time complexity to per-scenario program size |A|·|F|.
  3. Closed-form solution for each scenario, then aggregate.
     - **Linear** time to program size!

$$\lambda_i = \begin{cases} \dfrac{\rho}{1-\alpha} & \text{if } \sum_a \delta_a^i \mathrm{dist}_a^i[1] \geq c \\ 0 & \text{otherwise} \end{cases} \tag{14a}$$

$$\phi_i(a) = \delta_a^i \mathrm{dist}_a^i[2](1+\lambda_i) \tag{14b}$$

$$\mu_i(a,v) = \begin{cases} \delta_a^i(\mathrm{dist}_a^i[2] - \mathrm{dist}_a^i(v))(1+\lambda_i) & \text{if } v = v_a^i[1] \text{ or } x_v = 0 \\ 0 & \text{otherwise} \end{cases} \tag{14c}$$

Outlines

- Introduction and Methodology Overview
- System Model
- Optimization Framework
- Performance Evaluation
- Conclusions

Simulation Settings

- Three different experiment settings.

Expectation vs. CVaR

- Social Organization Framework (SoF) [Ning2011]-based Topology
- Uniform 99% network link reliabilities
- Time varying Gamma distribution user demands

Benders' vs. Exhaustive Search

- Random Waxman graphs with α=β=0.3, varying # nodes
- Uniform 99% network link reliabilities
- Erlang(1, 2) distribution user demands

Benders' vs. Random vs. Greedy

- Synthesized Dartmouth College topology from AP map
- Uniform 99% network link reliabilities
- 1-yr real user data: 4-mon for optz., 8-mon for validation

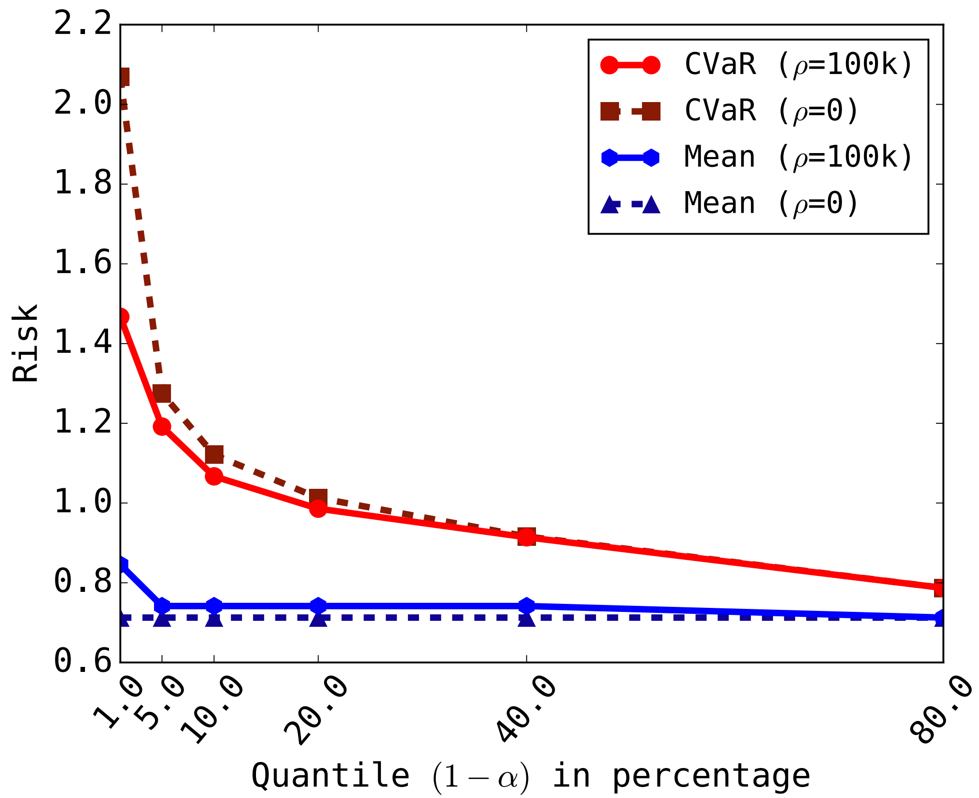
Parameters:

- α=95%
- ρ=100k (CVaR only except noted)

[Ning2011] H. Ning and Z. Wang, "Future Internet of Things Architecture: Like Mankind Neural System or Social Organization Framework?," *IEEE Commun. Lett.*, vol. 15, no. 4, pp. 461–463, Apr. 2011.

Dartmouth Dataset: https://crawdad.org/ dartmouth/campus/20090909

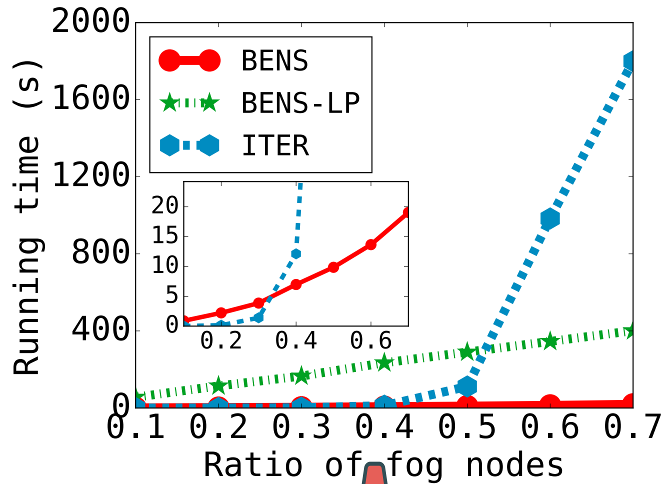
Result: Expectation vs. CVaR



Expectation vs. CVaR

- CVaR approaches mean when $\alpha \to 0$.
- There is a trade-off between expectation and CVaR.
- CVaR can be 1.5x larger if optimizing expectation alone.

Result: Optimality & Overhead

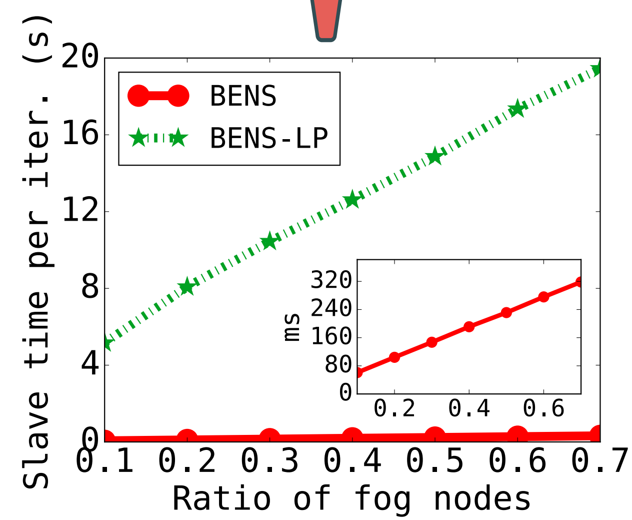


Running Time

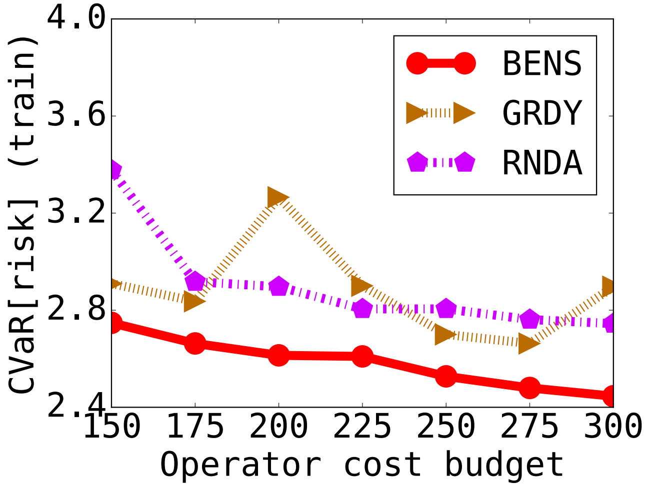
- Benders' much more efficient than exhaustive search.
- Our closed-form solution achieves great speed-up over solving slave duals by LP.

Slave Solving Time

- Speed-up is indeed due to our slave dual solving.



Result: Synthesized Data Simulation

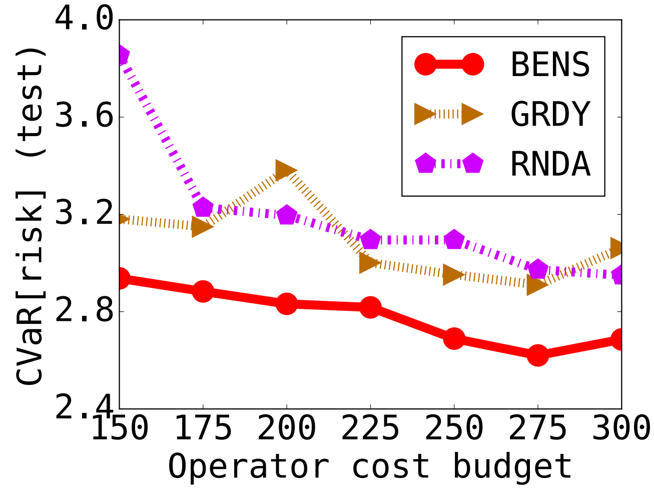


Training CVaR

- Benders' much better than greedy and random.

Testing CVaR

- Optimal for training may not be optimal for testing
  - Both network and user demands are evolving…



Outlines

- Introduction and Methodology Overview
- System Model
- Optimization Framework
- Performance Evaluation
- Conclusions

Conclusions

- The IoT security challenge
  - Lightweight crypto has a long way to go
  - Security offloading brings inevitable risk
- Modeling IoT security with offloading
  - Uncertainty model
  - Expectation vs. CVaR
  - Scenario-based optimization
- Robust security deployment algorithm
  - Benders' decomposition
  - Speed-up per-iteration solving
- Simulations: outperforming and efficient solution!

Thank you very much!

Q&A?