

# Multi-Agent Optimization and Learning in Power Distribution Systems

Qianzhi Zhang

Supervisor: Zhaoyu Wang

Committee: Venkataramana Ajjarapu, James McCalley, Ian Dobson, Chao Hu

Electrical and Computer Engineering

PhD Preliminary Exam

# Outline

- Motivation of Multi-Agent Framework for Optimization and Learning
- Power Management Problem of Networked Microgrids
- Introduction of Reinforcement Learning
- Proposed Multi-agent Safe Learning for Power Management of Networked Microgrids
- Simulation Results
- Conclusions and Future Works

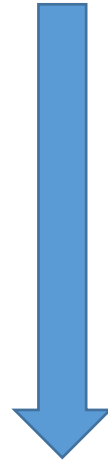
# Optimization and Decision-Making Problem in Active Distribution Systems

## Objectives:

- Economic dispatch
- Power/energy management
- Operational cost
- Voltage and reactive (var) control
- Active power losses
- Selling/purchase power
- Load restoration
- Generation curtailment
- Hosting capacity
- Social welfare
- ...



$$\begin{aligned} & \text{minimize or maximize } f(x, u) \\ & \text{subject to} \\ & \quad g(x, u) = 0 \\ & \quad h(x, u) \leq 0 \\ & \quad LB \leq x \leq UB \end{aligned}$$



## Solution algorithm:

- **Centralized approach**
- **Multi-agent (distributed) approach**

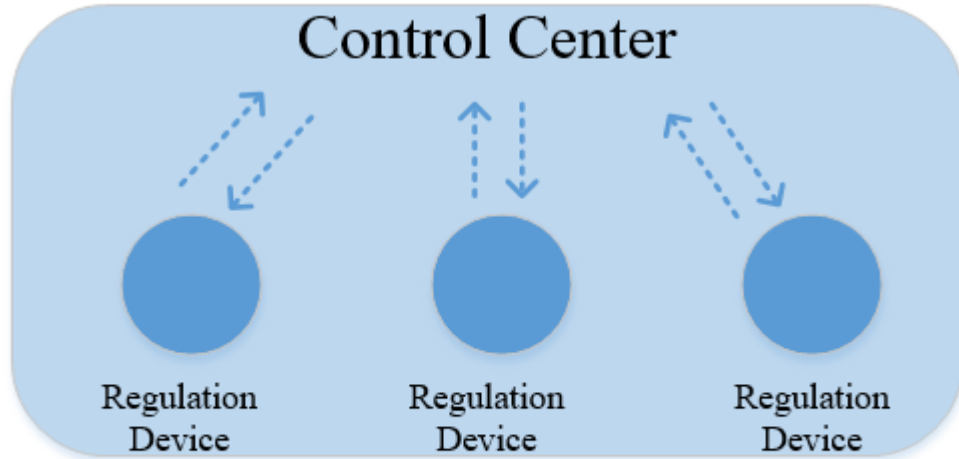
## Constraints

- Bus active/reactive power balance
- Bus voltage
- Line active/reactive power flow
- Flexible load
- Generation power output
- Network reconfiguration
- ...

## Variables

- Continuous variables
- Discrete variables (binary, integer)

# Centralized and Multi-Agent (Distributed)



## Centralized



## Multi-agent (distributed)

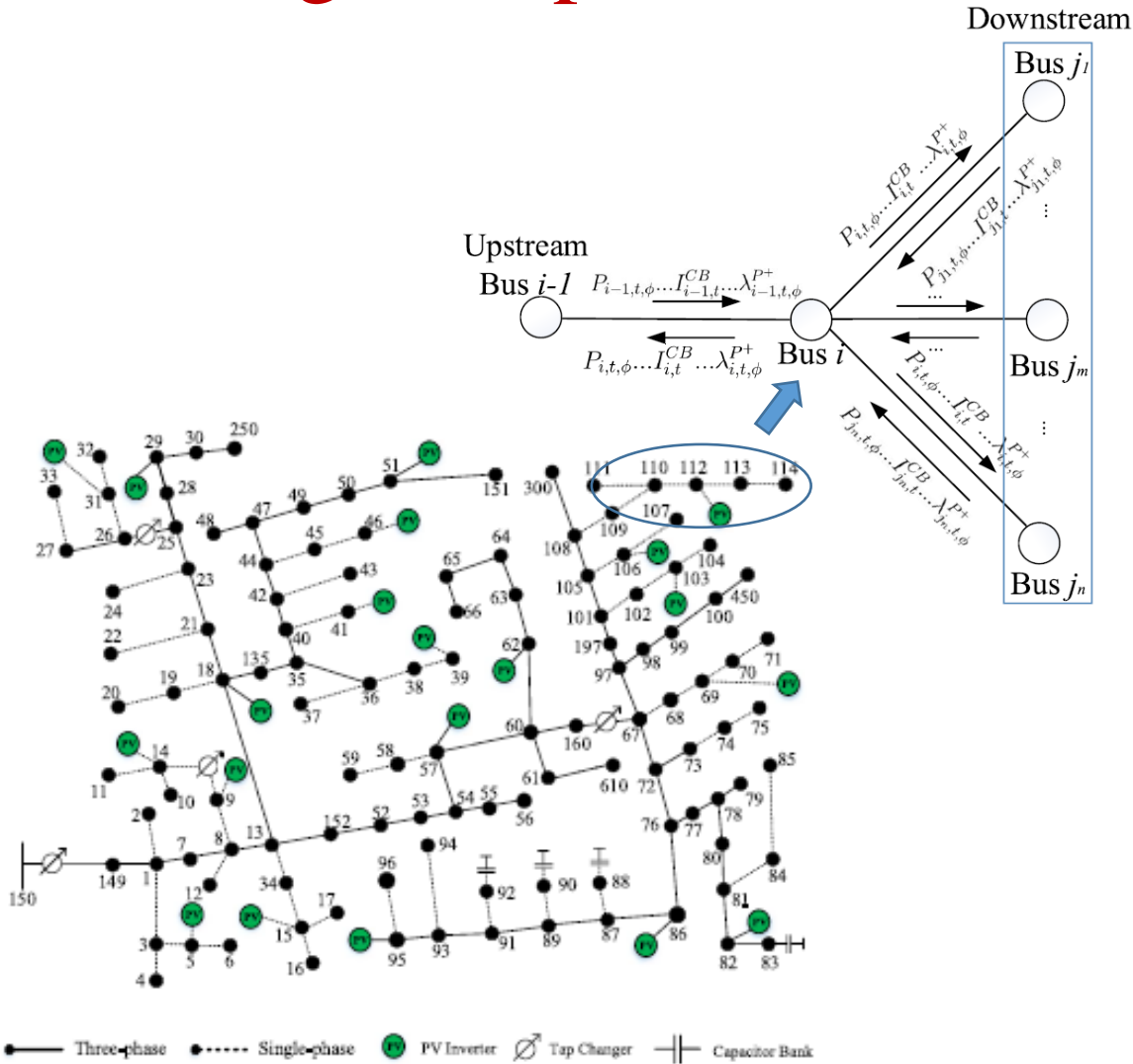
### Centralized approach:

- Solve a large-scale problem
- Require the system-wide collection of data
- Require reliable communications between a control center and regulation devices
- Susceptible to single point of failure

### Multi-agent (distributed) approach:

- Decompose the large-scale problem
- Solve multiple small-scale problems
- Maintain the data privacy and ownership of customers
- Resilient against agent communication failure or limited communication

# Multi-Agent Optimization and Learning

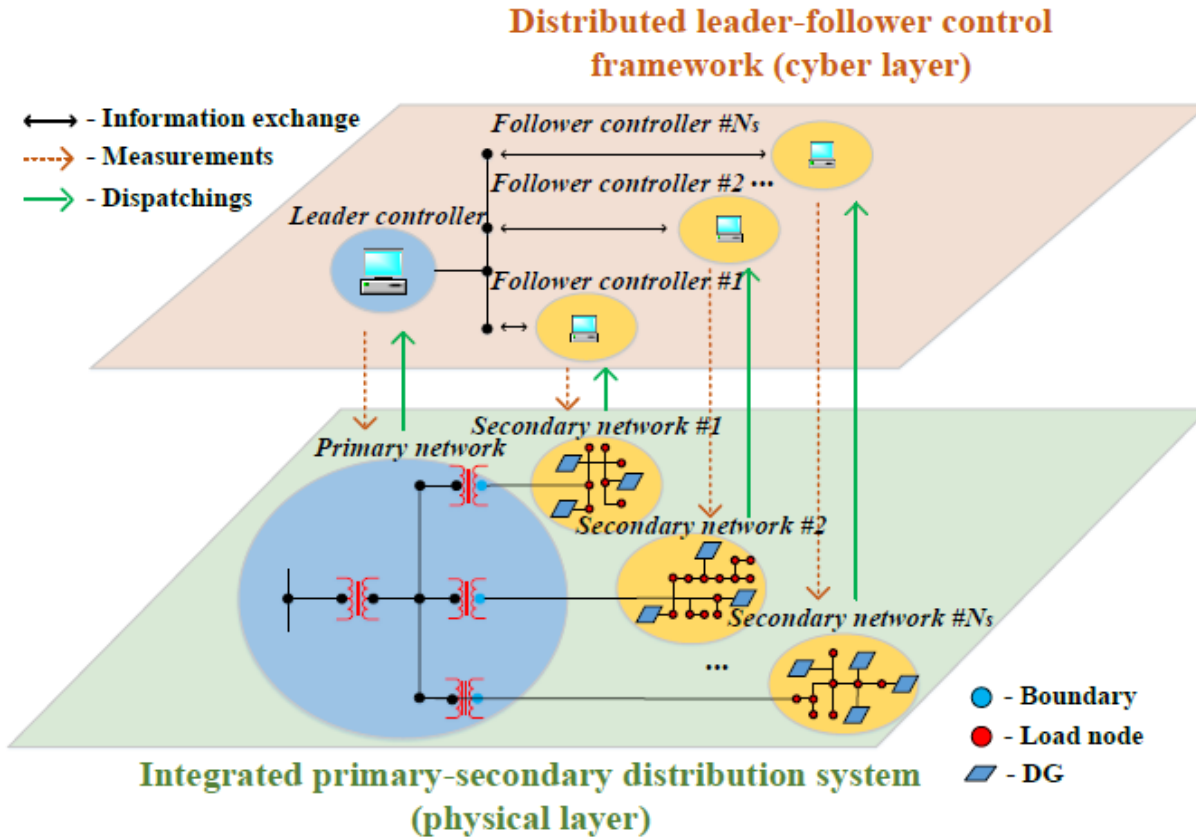


Distributed conservation voltage reduction (CVR) in unbalanced distribution systems with high PV penetration

- An optimization model is developed to facilitate voltage reduction and coordinate the fast-dispatch of photovoltaic (PV) inverters and the slow-dispatch of on-load tap changer (OLTC) and capacitor banks (CBs)
- In order to ensure the solution optimality and maintain customer data privacy and ownership, a distributed solution method is proposed to decompose the large-scale optimization problem into bus-level small-scale optimization problems.
- Bus-level control agents are in charge of managing the local controllable resources and communicating with neighboring bus-level control agents.
- A modified alternating direction method of multipliers (ADMM) algorithm is proposed to handle non-convex optimization problems with discrete switching and tap changing variables.

Q. Zhang, K. Dehghanpour and Z. Wang, "Distributed CVR in Unbalanced Distribution Systems With PV Penetration," in IEEE Transactions on Smart Grid, vol. 10, no. 5, pp. 5308-5319, Sept. 2019.

# Multi-Agent Optimization and Learning



Q. Zhang, Y. Guo, Z. Wang and F. Bu, "Distributed optimal conservation voltage reduction in integrated primary-secondary distribution systems," in IEEE Transactions on Smart Grid, under third round review.

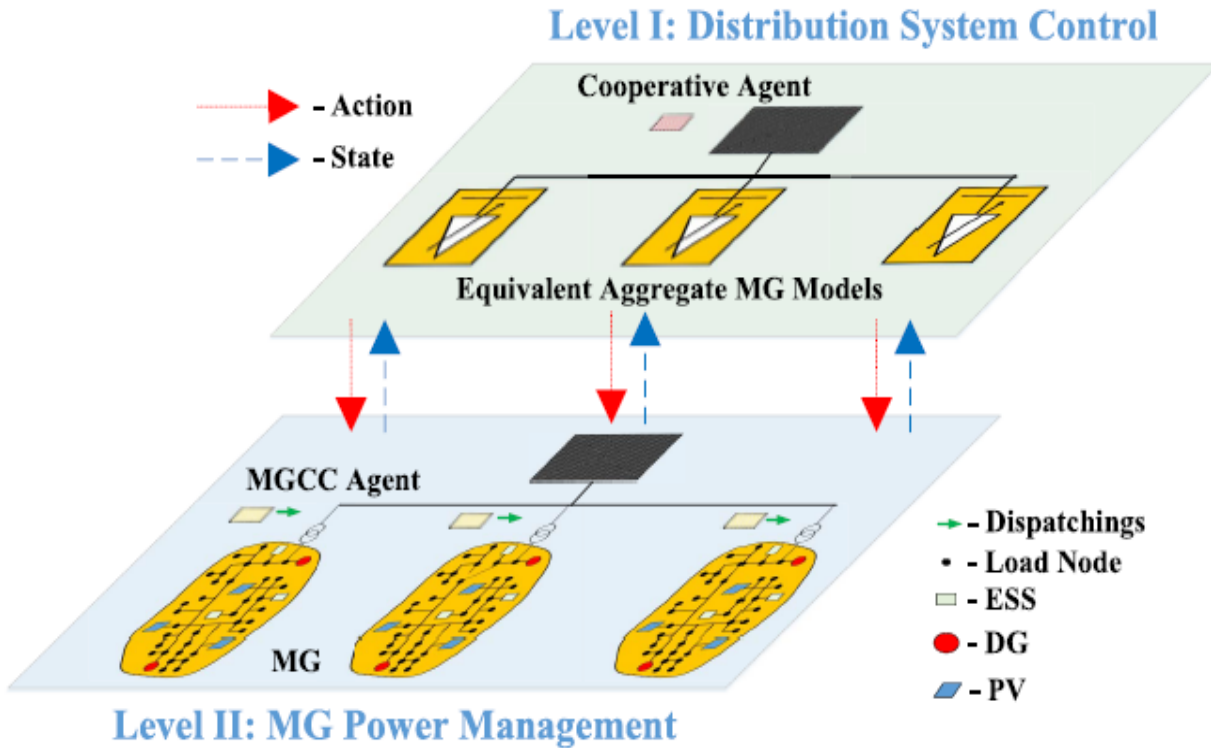
Distributed CVR in integrated primary-secondary distribution systems:

- A Volt/Var optimization based CVR (VVO-CVR) is modelled in an integrated medium-voltage (MV) primary distribution network and low-voltage (LV) secondary distribution networks.
- To solve the VVO-CVR problem in a distributed way, we first split the primary and secondary networks from modeling perspective, then introduce coupling constraints at boundary nodes, finally map the primary and secondary networks into leader and follower controllers in ADMM distributed framework.
- We propose an online feedback-based linear approximation method, where the instantaneous power and voltage measurements are used as system feedback in each iteration of ADMM to linearize the nonlinear terms of power calculation for both power flow and ZIP load models.

# Multi-Agent Optimization and Learning

A learning-based power management method for networked microgrids (MGs) under incomplete information:

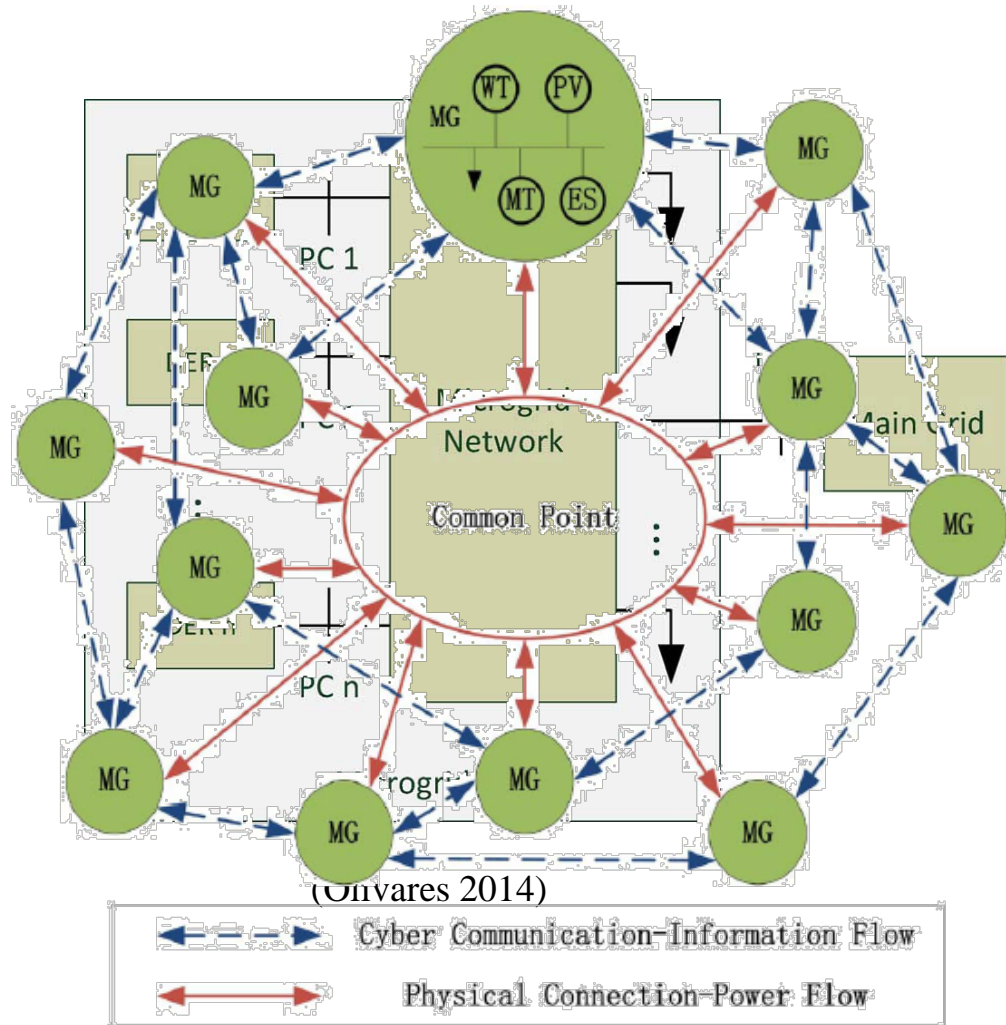
- A bi-level cooperative framework is proposed using a reinforcement (RL) based method for a distribution system consisting of multiple networked privately-owned MGs.
- At Level I, a non-profit cooperative RL agent maximizes the total MGs' revenue by setting retail power price signal. By considering the model-free nature of our RL-based method, the data privacy of MGs and the data confidentiality of costumers are maintained. The power management problem is solved with access to only minimal and aggregated data.
- At Level II, each MG agent receives the price signals from Level I and solves a constrained mixed integer nonlinear programming (MINLP) to dispatch their local generation and energy storage system units.



Q. Zhang, K. Dehghanpour, Z. Wang and Q. Huang, "A Learning-Based Power Management Method for Networked Microgrids Under Incomplete Information," in IEEE Transactions on Smart Grid, vol. 11, no. 2, pp. 1193-1204, March 2020.



# Networked MGs



MGs are active clusters of:

- Distributed energy resources (DERs), such as diesel generators (DGs), photo-voltaic generators (PVs), wind generators (WTs).
- Energy storage systems (ESSs)
- Other onsite electric components

Smart distribution systems may consist of multiple MGs and the coordinated control of the networked MGs can offer various benefits:

- Higher penetration of local DERs
- Improved controllability
- Enhancement of power system resilience and reliability.

Various ways to solving the power management problem of networked MGs :

- Centralized and multi-agent (distributed)
- Model-based and model-free



# Power Management Problem of Networked MGs

- This optimization problem is solved over a moving look-ahead decision window  $t' \in [t, t + T]$ .
- Objective + Global constraints (whole distribution network) + Local constraints (inside each MG)

$$\min_{x_p, x_q} \sum_{n=1}^N \sum_{t'=t}^{T+t} (-\lambda_n^R P_{n,t'}^{PCC} + \lambda_{i,n}^F F_{i,n,t'})$$

$$F_{i,t,n} = a_n^f (P_{i,n,t'}^{DG})^2 + b_n^f P_{i,n,t'}^{DG} + c_n^f$$

The objective minimize MGs' total cost of operation.

- Each MG is assumed to have local DGs, ESS, solar PV panels and several loads. The control action vector is  $[P^{DG}, P^{Ch}, P^{Dis}, Q^{DG}, Q^{PV}, Q^{ESS}] \in (x_p, x_q)$ .
- The fuel consumption of DGs can be expressed as a quadratic polynomial function.

s.t.

$$V_i^m \leq V_{i,t'} \leq V_i^M$$

$$-I_{ij}^M \leq I_{ij,t'} \leq I_{ij}^M$$

Global constraints are defined over variables that are impacted by control actions of all the MGs:

- Nodal voltage amplitude constraints for the entire nodes.
- Branch current flow constraints throughout the network

Our solution leverage AC power flow equations in an implicit way in the training process

- Calculate the gradient factor of objective and constraints w.r.t. learning parameters
- Ensure that the learning modules are generating feasible outcomes

# Power Management Problem of Networked MGs

$$0 \leq P_{i,n,t'}^{DG} \leq P_{i,n}^{DG,M}$$

$$0 \leq Q_{i,n,t'}^{DG} \leq Q_{i,n}^{DG,M}$$

$$|P_{i,n,t'}^{DG} - P_{i,n,t'-1}^{DG}| \leq P_{i,n}^{DG,R}$$

$$|Q_{i,n,t'}^{PV}| \leq Q_{i,n}^{PV,M}$$

$$|P_{t,n}^{PCC}| \leq P_{t,n}^{PCC,M}$$

$$|Q_{t,n}^{PCC}| \leq Q_{t,n}^{PCC,M}$$

$$SOC_{i,n,t'} = SOC_{i,n,t'-1} + \Delta t (P_{i,n,t'}^{Ch} \eta_{Ch} - P_{i,n,t'}^{Dis} / \eta_{Dis}) / E_{i,n}^{Cap}$$

$$SOC_{i,n}^m \leq SOC_{i,n,t'} \leq SOC_{i,n}^M$$

$$0 \leq P_{i,n,t'}^{Ch} \leq P_{i,n}^{Ch,M}$$

$$0 \leq P_{i,n,t'}^{Dis} \leq P_{i,n}^{Dis,M}$$

$$P_{i,n,t'}^{Ch} P_{i,n,t'}^{Dis} = 0$$

$$|Q_{i,n,t'}^{ESS}| \leq Q_{i,n}^{ESS,M}$$

Local constraints are defined over the local control actions of each MG:

- DG active and reactive power output constraints.
- DG ramp generation constraints.

- PV reactive power output constraints

- Active and reactive power transfer constraints at the PCCs

- ESS state of charge (SOC) constraints
- ESS maximum capacity constraints
- ESS active charging and discharging power constraints.
- ESS reactive power output constraints

# Literature Review

Ref.	Application	Solution	Model
Wang 2015	Centralized power management of networked MGs	Centralized	Model-based
Lu 2017	Centralized power management of networked MGs		
Farzin 2018	Enhancement of reliability performance of networked MGs		
Wang 2016	Consensus-based normal operation and self-healing of networked MGs	Multi-agent (distributed)	
Mojtaba 2017	A multi-agent framework for fault resiliency enhancement of networked MGs		
Shi 2018	A distributed cooperative control framework for synchronized reconnection of networked MGs		
Zhang 2020	A bi-level learning-based power management method for networked MGs	RL	Model-free

Limitations of model-based optimization methods:

- Solve a large-scale optimization problem with numerous linear and nonlinear constraints
- Rely heavily accurate knowledge of grid topology and parameters

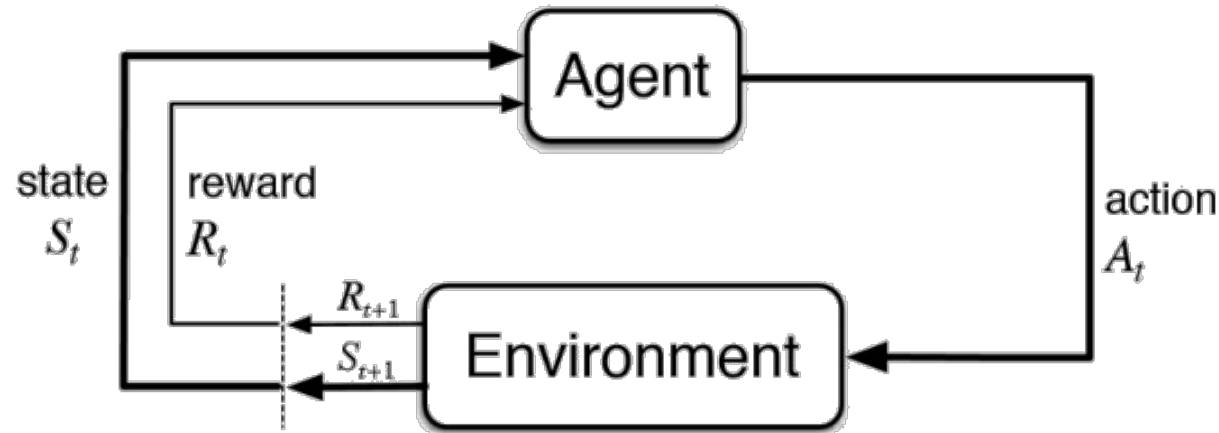
Potential infeasibility of model-free machine learning methods:

- Conventional RL methods train black-box functions to approximate the state-action through trial and error.
- The trained black-box functions can fail to satisfy critical operational constraints.
- This can lead to unsafe operational states and control action infeasibility.

# Introduction: Conventional RL

Conventional RL:

- Learn to make a good sequence of decisions and maximize the expected cumulative reward
- Markov decision process (MDP),  $S_t, A_t, P_t, R_t$
- Repeated interactions (environment, distribution system)
- Do not need reliable and complete distribution network model
- **Policy** is the agent's behavior and a map from state to action,  $a = \pi(s)$
- Offline centralized training
- Online centralized implementation (well trained)



(Sutton 2017)

Challenges of conventional RL:

- Incorporating constraints into the training process of conventional black-box RL methods
- A large-scale training problem, inefficient implementation and **potential** violation of customers' privacy

# Literature Review: Multi-Agent and Safe RL

Ref.	Application	Solution algorithm	Constraints
Zhang 2020	Power management of networked MGs	<ul style="list-style-type: none"><li>• Offline centralized training</li><li>• Online centralized implementation</li></ul>	Constraints are considered in a lower-level optimization problem
Ye 2020	Power management of residential house		Constraint violations are penalized in the reward function
Sun 2021	Volt/Var control	<ul style="list-style-type: none"><li>• Offline centralized training</li><li>• Online distributed implementation</li></ul>	
Gao 2021			

## Safe RL:

- Large number of constraints
- Manually design penalty coefficients for constraint violations, which either offers inadequate penalization of the constraint violations or excessive punishment for the constraints

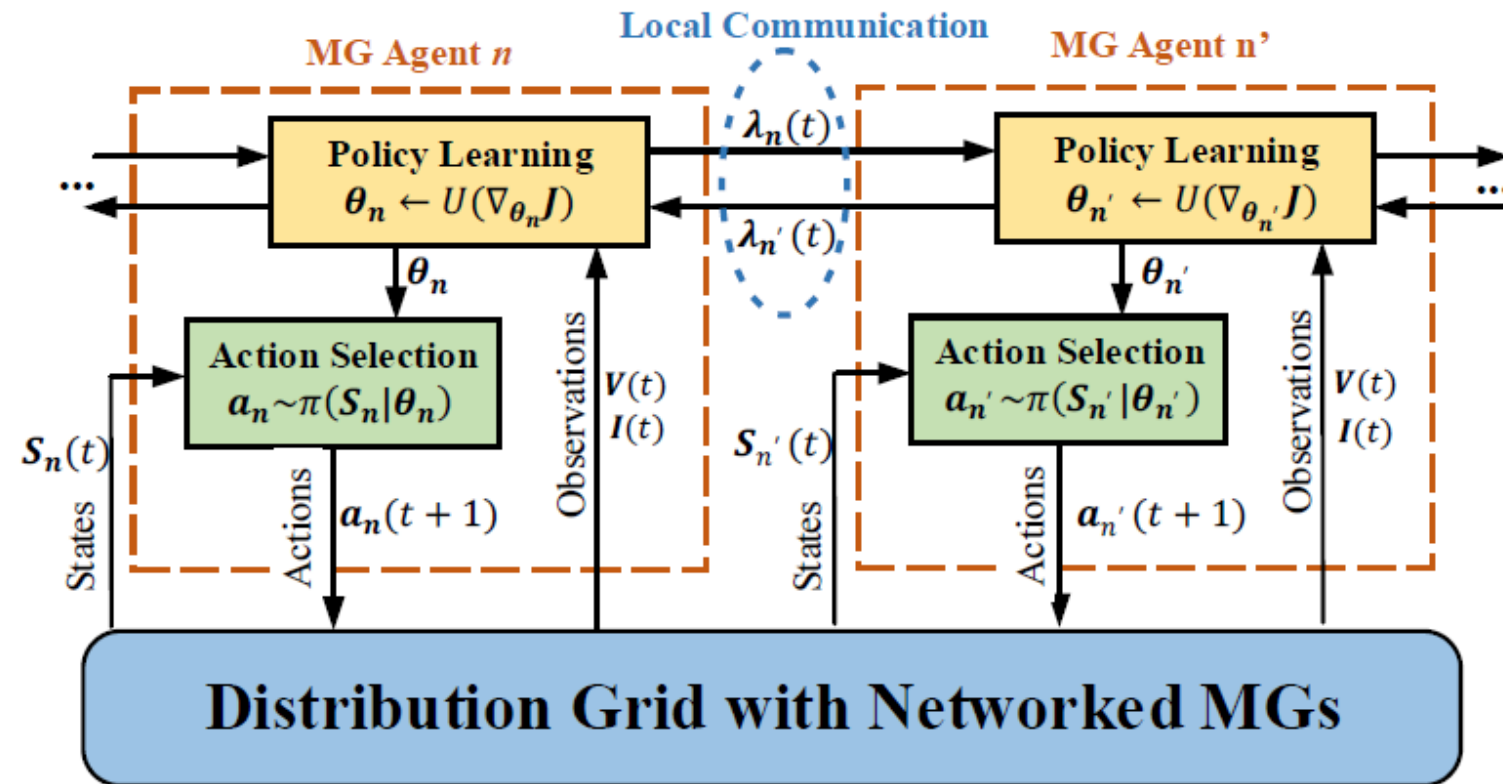
## Multi-agent RL (distributed training and implementation):

- Privacy of each control agent
- Efficiency of training process

# Problem Overview

The general framework of the proposed supervised multi-agent safe policy learning (SMAS-PL) method is shown as follows:

- To ensure safety of control policies, MG agents receive the observations from the distribution grid and determine gradient factors of the objective and constraints w.r.t. to learning parameters of policy functions.
- The multi-agent framework employs local communication between MGs agents and exchange the dual Lagrangian variables (not critical and private information).



- Each MG is controlled by an agent
- State  $S$
- Action  $a$
- Observation  $O$
- Policy function  $\pi$
- Update rule  $U$
- Learning parameter  $\theta$
- Gradient  $\nabla_{\theta} J$
- Lagrangian  $\lambda$



# Contributions

We propose SMAS-PL method for optimal power management of networked MGs in distribution system.

- Compared to conventional black-box RL, our proposed SMAS-PL:
  - Calculates gradients of all the operational constraints w.r.t. actions to promote the safety and feasibility of control policies.
  - Considers a backtracking mechanism into the PL framework to perform a final verification of feasibility before issuing control commands to the assets.
- Compared to conventional centralized RL, our proposed SMAS-PL:
  - Preserves the privacy of MG agents, including their control policies parameters and structures, operation cost functions, and local asset constraints;
  - Enhances computational efficiency and maintains scalability as the number of learning parameters grows into a humongous size.

Note that the proposed method introduces a trade-off between model-free and model-based methods and combines the benefits offered by both sides.

# Constrained MDP

To transform an optimal power management problem into a SMAS-PL problem, we introduce constrained MDP:

(1) **Agents:** Each control agent is dispatching the resources within an individual MG.

(2) **State set:** The state vector for the n'th MG agent at time t is defined as  $S_{n,t}$  over time window  $[t, t + T]$ :

$$S_{n,t} = [\hat{I}_{n,t'}^{PV}, \hat{P}_{n,t'}^D]_{t'=t}^{t+T}$$

- $\hat{I}_{n,t'}^{PV}$  and  $\hat{P}_{n,t'}^D$  are the vectors of predicted aggregate internal load power and solar irradiance of n'th MG at time t', respectively.
- The prediction errors follow random distributions with zero mean and the standard deviations selected from the beta and Gaussian distributions.

(3) **Action set:** The action vector for the n'th MG agent at time t is defined as  $a_{n,t}$  over time window  $[t, t + T]$ :

$$a_{n,t} = [P_{n,t'}^{DG}, P_{n,t'}^{Ch}, P_{n,t'}^{Dis}, Q_{n,t'}^{DG}, Q_{n,t'}^{PV}, Q_{n,t'}^{ESS}]_{t'=t}^{t+T}$$

(4) **Observation set:** The observation variable vector at time t is defined as  $O_t$ :

$$O_t = [V_t, I_t]$$

- $V_t$  and  $I_t$  are the vectors of grid's nodal voltages and current injections.
- The observation variables are implicitly determined by the agents' control actions

# Constrained MDP

(5) **Control policy:** the control policies are modelled as multivariate Gaussian distributions as follows:

$$a_n \sim \pi_n(a_n | \theta_n) = \frac{1}{\sqrt{|\Sigma_n| (2\pi)^{D_n}}} e^{-\frac{1}{2}(a_n - \mu_n)^T \Sigma_n^{-1} (a_n - \mu_n)}$$

$$\mu_n = DNN(S_n | \theta_{\mu_n})$$

$$\Sigma_n = DNN(S_n | \theta_{\Sigma_n})$$

- Gaussian distribution allow for explicit learning of both expectations and uncertainties of control policies: mean vector  $\mu_n$  and covariance matrix  $\Sigma_n$ .
- Control policy ( $\mu_n$  and  $\Sigma_n$ ) is parameterized by deep neural network (DNN) with weights and bias  $\theta_{\mu_n}$  and  $\theta_{\Sigma_n}$

(6) **Reward:** the reward function is defined as the discounted negative accumulated operational cost of individual MG over  $[t, t+T]$ :

$$J_{R_n}(\pi_n) = E_{\pi_n} \left[ \sum_{t'=t}^{t+T} \gamma^{t'} R_{n,t'} \right], \forall n \in \{1 \dots N\}$$

- Discount factor  $\gamma$  determines each MG agent's bias towards reward at different time instances
- Expectation operator  $E_{\pi_n}$  is used to calculate reward w.r.t. the future expected action-states, which is impacted by the uncertainties of states and observations.

(7) **Constraint return:** the constraint return is defined as the discounted accumulated constraint value over  $[t, t+T]$ :

$$J_{C_m}(\pi) = E_{\pi} \left[ \sum_{t'=t}^{t+T} \gamma^{t'} C_{m,t'} \right] = or \leq d_m, \forall m \in \{1 \dots M_c\}$$

- $C_{m,t'}$  is the return value of m'th constraint under the control policy
- $d_m$  is the upper-boundary of the m'th constraint

# Safe Policy Learning Formulation

- Given the definitions of constrained MDP, the power management problem of the networked MG is first transformed into an intractable and non-convex PL problem [Achiam 2017].
- The control policies of the agents are updated at time  $t$ , around their latest values, by maximizing a reward function, while satisfying constraint return criteria, as follows:

$$\begin{aligned} \pi^{t+1} &= \arg \max_{\pi_1, \dots, \pi_n} \sum_{n=1}^N J_{R_n}(\pi_n) \\ \text{s.t.} \quad & a_n \sim \pi_n(S_n) \\ & J_{C_m}(\pi) \leq d_m, \forall m \\ & \Delta(\pi_n, \pi_n^t) \leq \delta, \forall n \end{aligned}$$

- $\pi = \{\pi_1, \dots, \pi_n\}$  denotes the set of control policies of all agents.
- Agent's policy is a function of the state vector  $a_n \sim \pi_n(S_n)$ .
- The expected constraint return value are used to ensure the satisfaction of  $m$ 'th constraint based on control policies.
- $\Delta$  is the Kullback Leibler (KL) divergence function that serves as a distance measure between the previous  $\pi_n^t$  and the updated policy  $\pi_n^{t+1}$ .
- Step size  $\delta$

# Safe Policy Learning Formulation

- Then, the intractable non-convex PL problem can be solved by a trust region policy optimization method (TRPO), and further transformed into a tractable convex iteratively quadratically constrained linear program (QCLP), which enables learning the PL parameters  $\theta_1, \dots, \theta_n$ .
- Our solution leverages the linear approximations of the objective and constraint returns around the latest parameter value  $\theta^t$  [Achiam 2017]:

$$\begin{array}{l}
 \pi^{t+1} = \arg \max_{\pi_1, \dots, \pi_n} \sum_{n=1}^N J_{R_n}(\pi_n) \\
 \text{s.t.} \\
 a_n \sim \pi_n(S_n) \\
 J_{C_m}(\pi) \leq d_m, \forall m \\
 \Delta(\pi_n, \pi_n^t) \leq \delta, \forall n
 \end{array}
 \quad \longrightarrow \quad
 \begin{array}{l}
 \theta^{t+1} = \arg \max_{\theta_1, \dots, \theta_n} \sum_{n=1}^N g_n^T(\theta_n - \theta_n^t) \\
 \text{s.t.} \\
 J_{C_m}(\theta^t) + b_m^T(\theta - \theta^t) \leq d_m, \forall m \\
 \frac{1}{2}(\theta_n - \theta_n^t)^T H_n(\theta_n - \theta_n^t) \leq \delta, \forall n
 \end{array}
 \left. \vphantom{\begin{array}{l} \pi^{t+1} \\ \theta^{t+1} \end{array}} \right\}
 \begin{array}{l}
 \bullet \quad g_n = \nabla_{\theta} J_{R_n} \text{ and } b_m = \nabla_{\theta} J_{C_m} \text{ are the} \\
 \text{gradient factors of the reward and} \\
 \text{constraints returns with respect to} \\
 \text{the learning parameters } \theta. \\
 \bullet \quad \text{The KL divergence function is} \\
 \text{transformed by using the Fisher} \\
 \text{information matrix } H_n.
 \end{array}$$

- Before obtain  $\theta_n \in [\theta_{\mu_n}, \theta_{\Sigma_n}]$ , we must calculate the gradient factors  $g_n$  and  $b_m$  first based on power flow formulations.

# Gradient Factor Determination: Chain Role

To determine gradient factors  $g_n$  and  $b_m$ , the following information are used:

- Observation variable, including nodal voltages and current injections;
- Latest system state of each MG agent;
- Latest control actions of each MG agent;
- Latest learning parameters;
- Network parameters, including the nodal admittance matrix.


Using the above information and chain rule, the gradients  $g_n = [g_{\mu_n}, g_{\Sigma_n}]$  and  $b_m = [b_{m,\mu_n}, b_{m,\Sigma_n}]$  can be obtained as follows:

$$g_{\mu_n} = \frac{\partial J_{Rn}}{\partial a_n} \frac{\partial a_n}{\partial \pi_n} \frac{\partial \pi_n}{\partial \mu_n} \frac{\partial \mu_n}{\partial \theta_{\mu_n}}$$

$$g_{\Sigma_n} = \frac{\partial J_{Rn}}{\partial a_n} \frac{\partial a_n}{\partial \pi_n} \frac{\partial \pi_n}{\partial \Sigma_n} \frac{\partial \Sigma_n}{\partial \theta_{\Sigma_n}}$$

$$b_{m,\mu_n} = \frac{\partial J_{Cm}}{\partial a_n} \frac{\partial a_n}{\partial \pi_n} \frac{\partial \pi_n}{\partial \mu_n} \frac{\partial \mu_n}{\partial \theta_{\mu_n}}$$

$$b_{m,\Sigma_n} = \frac{\partial J_{Cm}}{\partial a_n} \frac{\partial a_n}{\partial \pi_n} \frac{\partial \pi_n}{\partial \Sigma_n} \frac{\partial \Sigma_n}{\partial \theta_{\Sigma_n}}$$

- $\frac{\partial J_{Rn}}{\partial a_n}$  and  $\frac{\partial J_{Cm}}{\partial a_n}$ : use current injection-based AC power flow equations. 
- $\frac{\partial a_n}{\partial \pi_n}$ : use the latest value of  $a_n$  and the probability density function of multivariate Gaussian distribution  $\pi_n$ .
- $\frac{\partial \pi_n}{\partial \mu_n}$  and  $\frac{\partial \pi_n}{\partial \Sigma_n}$ : use the latest values of  $\mu_n$ ,  $\Sigma_n$  and the probability density function of multivariate Gaussian distribution  $\pi_n$ .
- $\frac{\partial \mu_n}{\partial \theta_{\mu_n}}$  and  $\frac{\partial \Sigma_n}{\partial \theta_{\Sigma_n}}$ : use back-propagation process of two DNNs,  $\mu_n$  and  $\Sigma_n$  are the outputs of DNNs, and  $\theta_{\mu_n}$  and  $\theta_{\Sigma_n}$  are weights and bias of DNNs.



# Gradient Factor Determination

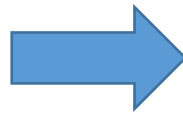
The gradients of the expected reward  $J_{R_n}$  and the expected constraint return  $J_{C_m}$  w.r.t. control action  $a_n$ ,  $\frac{\partial J_{R_n}}{\partial a_n}$  and  $\frac{\partial J_{C_m}}{\partial a_n}$  can be obtained by a **four-step process** and **current injection-based AC power flow equation**.

- Step 1: Obtain the partial derivations of real and imaginary parts of nodal current injection w.r.t. control actions  $\frac{\partial I^{Re}}{\partial a_n}$  and  $\frac{\partial I^{Im}}{\partial a_n}$

Nodal current injection:

$$I_{i,t'}^{Re} = \frac{p_{i,n,t'} V_{i,t'}^{Re} + q_{i,n,t'} V_{i,t'}^{Im}}{V_{i,t'}^2}$$

$$I_{i,t'}^{Im} = \frac{p_{i,n,t'} V_{i,t'}^{Im} - q_{i,n,t'} V_{i,t'}^{Re}}{V_{i,t'}^2}$$



Partial derivations of  $I_{i,t'}^{Re}$  and  $I_{i,t'}^{Im}$  w.r.t  $a_n$

$a_n$	$P_{n,t'}^{DG}$	$P_{n,t'}^{Ch}$	$P_{n,t'}^{Dis}$	$Q_{n,t'}^{DG}$	$Q_{n,t'}^{PV}$	$Q_{n,t'}^{ESS}$
-						
$I_{i,t'}^{Re}$	$-\frac{V_{i,t'}^{Re}}{V_{i,t'}^2}$	$\frac{V_{i,t'}^{Re}}{V_{i,t'}^2}$	$-\frac{V_{i,t'}^{Re}}{V_{i,t'}^2}$	$\frac{V_{i,t'}^{Im}}{V_{i,t'}^2}$	$\frac{V_{i,t'}^{Im}}{V_{i,t'}^2}$	$-\frac{V_{i,t'}^{Im}}{V_{i,t'}^2}$
$I_{i,t'}^{Im}$	$-\frac{V_{i,t'}^{Im}}{V_{i,t'}^2}$	$\frac{V_{i,t'}^{Im}}{V_{i,t'}^2}$	$-\frac{V_{i,t'}^{Im}}{V_{i,t'}^2}$	$-\frac{V_{i,t'}^{Re}}{V_{i,t'}^2}$	$-\frac{V_{i,t'}^{Re}}{V_{i,t'}^2}$	$\frac{V_{i,t'}^{Re}}{V_{i,t'}^2}$

Nodal power balance:

$$p_{i,n,t'} = P_{i,n,t'}^D - P_{i,n,t'}^{DG} - P_{i,n,t'}^{PV} + P_{i,n,t'}^{Ch} - P_{i,n,t'}^{Dis}$$

$$q_{i,n,t'} = Q_{i,n,t'}^D - Q_{i,n,t'}^{DG} - Q_{i,n,t'}^{PV} + Q_{i,n,t'}^{ESS}$$

# Gradient Factor Determination

- Step 2: Using  $\frac{\partial I^{Re}}{\partial a_n}$  and  $\frac{\partial I^{Im}}{\partial a_n}$  from Step 1,  $\frac{\partial V^{Re}}{\partial a_n}$  and  $\frac{\partial V^{Im}}{\partial a_n}$  are obtained by employing the network-wide relationship between nodal voltages and current injections:

$$\begin{bmatrix} \frac{\partial V^{Re}}{\partial a_n} \\ \frac{\partial V^{Im}}{\partial a_n} \end{bmatrix} = \begin{bmatrix} Y^{Re} - Y_D^{(Re,Re)} & -Y^{Im} - Y_D^{(Re,Im)} \\ Y^{Im} - Y_D^{(Im,Re)} & Y^{Re} - Y_D^{(Im,Im)} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial I^{Re}}{\partial a_n} \\ \frac{\partial I^{Im}}{\partial a_n} \end{bmatrix}$$

The elements in diagonal matrices  $Y_D^{(Re,Re)}$ ,  $Y_D^{(Re,Im)}$ ,  $Y_D^{(Im,Re)}$  and  $Y_D^{(Im,Im)}$

$$Y_D^{(Re,Re)}(i, i) = \frac{p_{i,n,t'}}{V_{i,t'}^2} - \frac{2V_{i,t'}^{Re}(p_{i,n,t'}V_{i,t'}^{Re} + q_{i,n,t'}V_{i,t'}^{Im})}{V_{i,t'}^4}$$

$$Y_D^{(Re,Im)}(i, i) = \frac{q_{i,n,t'}}{V_{i,t'}^2} - \frac{2V_{i,t'}^{Im}(p_{i,n,t'}V_{i,t'}^{Re} + q_{i,n,t'}V_{i,t'}^{Im})}{V_{i,t'}^4}$$

$$Y_D^{(Im,Re)}(i, i) = \frac{q_{i,n,t'}}{V_{i,t'}^2} - \frac{2V_{i,t'}^{Re}(p_{i,n,t'}V_{i,t'}^{Im} - q_{i,n,t'}V_{i,t'}^{Re})}{V_{i,t'}^4}$$

$$Y_D^{(Im,Im)}(i, i) = \frac{p_{i,n,t'}}{V_{i,t'}^2} - \frac{2V_{i,t'}^{Im}(p_{i,n,t'}V_{i,t'}^{Im} - q_{i,n,t'}V_{i,t'}^{Re})}{V_{i,t'}^4}$$

# Gradient Factor Determination

- Step 3: Using the branch current flow equations, the gradients of branch current flows are determined as a function of the derivatives of nodal voltages and current injections:

$$\frac{\partial I_{ij,t'}^{Re}}{\partial a_{n,t'}} = y_{ij}^{Im} \left( \frac{\partial V_{i,t'}^{Im}}{\partial a_{n,t'}} - \frac{\partial V_{j,t'}^{Im}}{\partial a_{n,t'}} \right) - y_{ij}^{Re} \left( \frac{\partial V_{i,t'}^{Re}}{\partial a_{n,t'}} - \frac{\partial V_{j,t'}^{Re}}{\partial a_{n,t'}} \right)$$

$$\frac{\partial I_{ij,t'}^{Im}}{\partial a_{n,t'}} = y_{ij}^{Im} \left( \frac{\partial V_{i,t'}^{Re}}{\partial a_{n,t'}} - \frac{\partial V_{j,t'}^{Re}}{\partial a_{n,t'}} \right) + y_{ij}^{Re} \left( \frac{\partial V_{i,t'}^{Im}}{\partial a_{n,t'}} - \frac{\partial V_{j,t'}^{Im}}{\partial a_{n,t'}} \right)$$

- Step 4: Using the gradients obtained from Steps 1, 2, and 3,  $\frac{\partial J_{R_n}}{\partial a_n}$  and  $\frac{\partial J_{C_m}}{\partial a_n}$  can be determined through straightforward algebraic manipulations.
- As an example, the gradient of reward function w.r.t. the action  $P_{n,t'}^{DG}$  is calculated as:

$$\frac{\partial J_{R_n}}{\partial P_{n,t'}^{DG}} = \sum_{t'=t}^{t+T} \left( \lambda_{i,n}^F (2a_f + b_f) - \lambda_n^R \left( \frac{\partial V_{i,t'}^{Re}}{\partial P_{n,t'}^{DG}} I_{ij,t'}^{Re} + V_{i,t'}^{Re} \frac{\partial I_{i,t'}^{Re}}{\partial P_{n,t'}^{DG}} + \frac{\partial V_{i,t'}^{Im}}{\partial P_{n,t'}^{DG}} I_{ij,t'}^{Im} + V_{i,t'}^{Im} \frac{\partial I_{i,t'}^{Im}}{\partial P_{n,t'}^{DG}} \right) \right)$$

# Consensus-based Multi-agent Learning

- Using the gradient factors, the QCLP is fully specified and can be solved at each policy update iteration for training the agents' policy learning framework.
- Two challenges when solving this problem:
  - (i) the size of the DNN parameters  $\theta$  can be extreme large, which results in high computational cost during training;
  - (ii) the control policy privacy of the MG agents might not have access to each other's control policies, cost functions, and local constraints on assets.
- We propose a multi-agent consensus-based constrained training algorithm:

## *Step I Initialize:*

$$\theta_n^t(0) \leftarrow \theta_n^{t-1}$$

- The previous values of learning parameters  $\theta_n^{t-1}$  are considered as initial values for  $\theta_n^t(0)$ .

## *Step II Weighted averaging operation (global constraints):*

$$\bar{\lambda}_n(k) = \sum_{n'=1}^{N_n} w_n(n') \lambda_{n'}(k)$$

- For global constraints, MG agent  $n$  receives the Lagrangian multiplier  $\lambda_{n'}(k)$  from its neighboring MG agents and combines the received estimates using weighted  $w_n(n')$  averaging  $\bar{\lambda}_n(k)$ .

# Consensus-based Multi-agent Learning

## *Step III Primal gradient update (global constraints):*

$$\bar{\theta}_n(k) = \theta_n^t(k) - \rho_1(g_n \theta_n^t(k) + b_{m'} \theta_n^t(k) \bar{\lambda}_n(k))$$

- MG agent  $n$  updates  $\theta_n^t$  by employing a gradient descent operation with  $g_n$ ,  $b_{m'}$ , and step size  $\rho_1$ .

## *Step IV Projection on local constraints:*

$$\theta_n^t(k+1) = \arg \min_{\theta} \|\bar{\theta}_n(k) - \theta\|$$

s.t.

$$J_{C_m}(\theta_n^t(0)) + b_m^T(\theta_n^t(0) - \theta) \leq d_m, \forall m$$

$$\frac{1}{2}(\theta_n^t(0) - \theta)^T H_n(\theta_n^t(0) - \theta) \leq \delta, \forall n$$

- MG agent  $n$  projects the local learning parameters  $\theta_n^t$  to the feasible region defined the gradients of the local constraints.

## *Step V Dual gradient update:*

$$\lambda_n(k) = [\bar{\lambda}_n(k) + \rho_1(b_{m'} \theta_n^t(k+1) - d_{m'})]^+$$

- MG agent  $n$ 's estimations of dual variable  $\lambda_n$  for the global constraints can be updated using a gradient ascent process over  $\bar{\lambda}_n$ .

Note that MG agent  $n$  only shares the Lagrangian multiplier with its neighboring MG agents, not critical control policy and actions.

# SMAS-PL Training and Implementation

Algorithm 1 SMAS-PL Training	
Initialization	1: Select $t^{max}, T, \delta, k^{max}, w_n(n'), \rho_1, \rho_2, \Delta\theta_n$
Update states	2: Initialize $\theta_n^{to}$
Update actions	3: for $t \leftarrow 1$ to $t^{max}$ do
	4: $\mathbf{S}_n \leftarrow [\mathbf{S}_n(t), \dots, \mathbf{S}_n(t+T)]$
	5: $\boldsymbol{\mu}_n \leftarrow (18)$ [Parameter insertion]
	6: $\boldsymbol{\Sigma}_n \leftarrow (19)$ [Parameter insertion]
	7: $\mathbf{a}_n \sim \pi_n(\mathbf{S}_n \boldsymbol{\theta}_n) \leftarrow (17)$ [Action selection]
	8: $\partial J_{R_n}/\partial \mathbf{a}_n \leftarrow (55)-(56)$
	9: $\partial J_{C_m}/\partial \mathbf{a}_n \leftarrow (59), (57)-(58)$
	10: $\partial \mathbf{a}_n/\partial \pi_n \leftarrow (32)$
	11: $\partial \pi_n/\partial \boldsymbol{\mu}_n \leftarrow (33)$
	12: $\partial \pi_n/\partial \boldsymbol{\Sigma}_n \leftarrow (34)$
Calculate gradient factors	13: $\partial \boldsymbol{\mu}_n/\partial \boldsymbol{\theta}_{\mu_n} \leftarrow DNN_{\mu_n}$ [Back-propagation]
	14: $\partial \boldsymbol{\Sigma}_n/\partial \boldsymbol{\theta}_{\Sigma_n} \leftarrow DNN_{\Sigma_n}$ [Back-propagation]
	15: $\mathbf{g}_{\mu_n}, \mathbf{b}_{m,\mu_n} \leftarrow (30)$ [Chain rule]
	16: $\mathbf{g}_{\Sigma_n}, \mathbf{b}_{m,\Sigma_n} \leftarrow (31)$ [Chain rule]
	17: $H_n \leftarrow (29)$ [FIM Construction]
	18: Initialize $\boldsymbol{\lambda}_n(k_0)$
	19: for $k \leftarrow 1$ to $k^{max}$ do
	20: $\bar{\boldsymbol{\lambda}}_n(k) \leftarrow (35)$ [Averaging operation]
	21: $\bar{\boldsymbol{\theta}}_n(k) \leftarrow (36)$ [Primal gradient update]
	22: $\boldsymbol{\theta}_n^t(k+1) \leftarrow (37)-(39)$ [Projection on $M^L$ ]
	23: $\boldsymbol{\lambda}_n(k+1) \leftarrow (40)$ [Dual gradient update]
	24: if $\ \boldsymbol{\theta}_n^t(k+1) - \boldsymbol{\theta}_n^t(k)\  \leq \Delta\theta_n$ then
	25: $\boldsymbol{\theta}_n^{t+1} \leftarrow \boldsymbol{\theta}_n^t(k+1)$ ; Break;
	26: end if
	27: end for
	28: if $\ \boldsymbol{\theta}_n^{t+1} - \boldsymbol{\theta}_n^t\  \leq \Delta\theta_n$ then
	29: Output $\boldsymbol{\theta}_n^* \leftarrow \boldsymbol{\theta}_n^{t+1}$ ; Break;
	30: end if
	31: end for
Output trained policy	32: Output well-trained parameterized policy $\pi_n(\boldsymbol{\theta}_n^*)$

## • Offline Training:

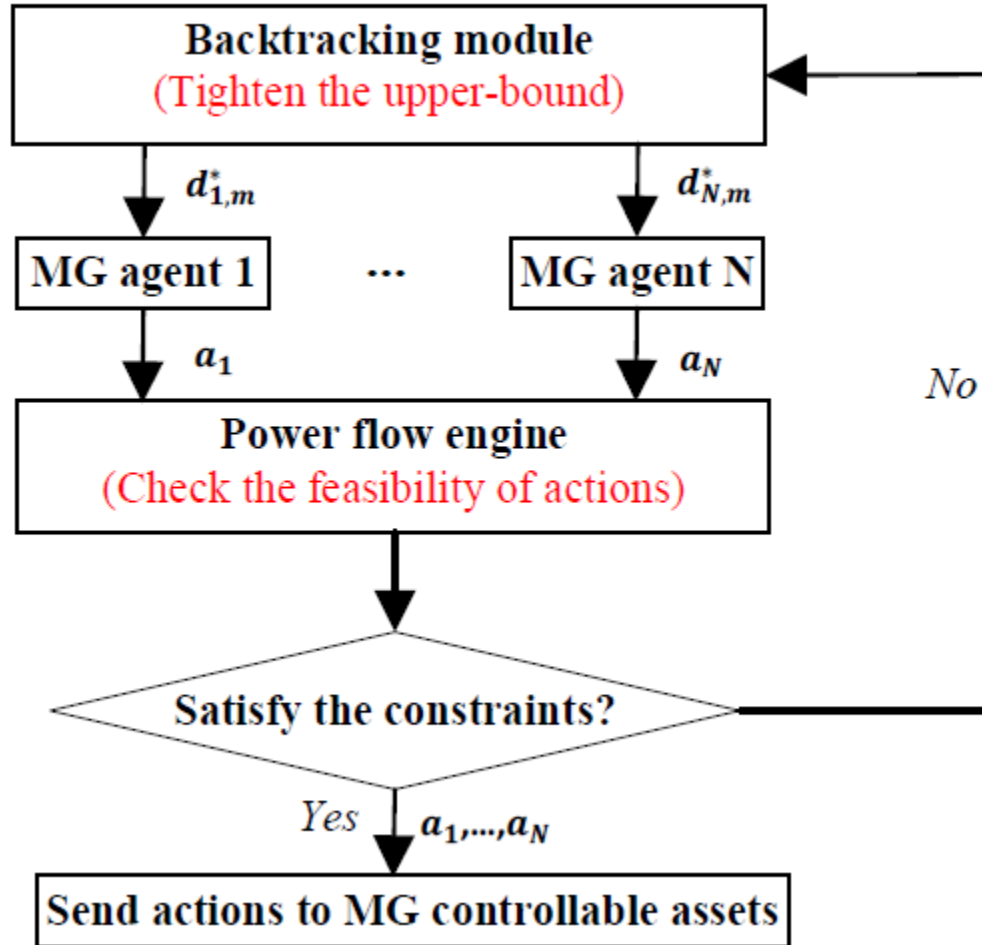
- A multi-agent framework is used to train the policy function of each MG agent
- The agents repeatedly estimate and communicate the Lagrangian multiplier of global constraints.
- The agents solve its own local constraints.

## • Online Implementation:

- First, the agents receive the latest states and input them into the trained DNNs to obtain the mean and covariance matrices of the policy functions.
- Then samples are generated from the multivariate Gaussian distributions.
- These samples are averaged and passed to the local controllers of each controllable asset as a reference signal.



# Backtracking Strategy



To ensure feasibility, we can add a backtracking strategy into the proposed solution:

## 1. Power flow engine (PFE):

- The PFE receives the control actions from the agents and runs a simple power flow program.
- If no constraint is violated, the control signals are passed to controllable assets.
- If some constraints are violated, then the PFE will engage the backtracking process

## 2. Backtracking module:

- The backtracking module tightens the upper-bound limits ( $d_m$ ) for the constraints that have been violated.
- The parameters of the trained DNNs will be re-updated and with the modified upper-bounds.
- The purpose of tightening the upper-bound is to provide a safety margin.
- In this paper the tightening process is performed using a user-defined coefficient multiplier,  $0 < \tau < 1$ , as follows.

$$d_m^* = \tau \times d_m$$

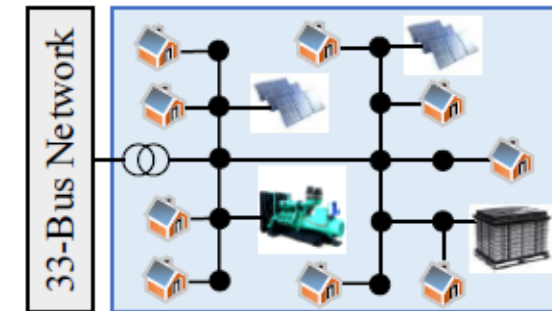
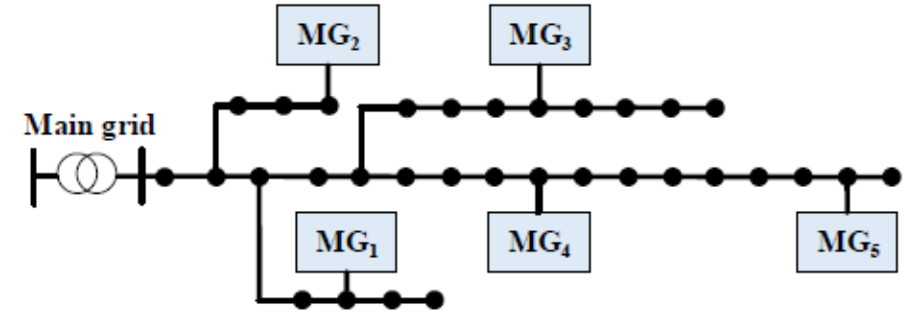
# Simulation: Setup

Test system:

- Modified 33-bus distribution system
- Modified 13-bus MG system
- The average capacities for DGs in MGs are 60 kWh.
- The average capacities for ESSs in MGs are 20 kWh, the maximum charging/discharging rate is 4 kW, and the charging/discharging efficiencies are 95% and 90%, respectively.

Taring and testing data:

- The input data for load have PV have 15 min time resolution are obtained from smart meter database to provide realistic numerical experiments.
- The training and testing datasets are selected through uniform randomization to ensure that the proposed solver functions reasonably.



Description	Notion	Value
Average electricity price (\$/kWh)	$\lambda^R$	0.046
Average DG fuel price (\$/L)	$\lambda^f$	0.57
Fuel cons. quadratic function parameter ( $L/kW^2$ )	$a^f$	0.0001773
Fuel cons. quadratic function parameter ( $L/kW$ )	$b^f$	0.1709
Fuel cons. quadratic function parameter ( $L$ )	$c^f$	14.67

# Simulation: Hyperparameters

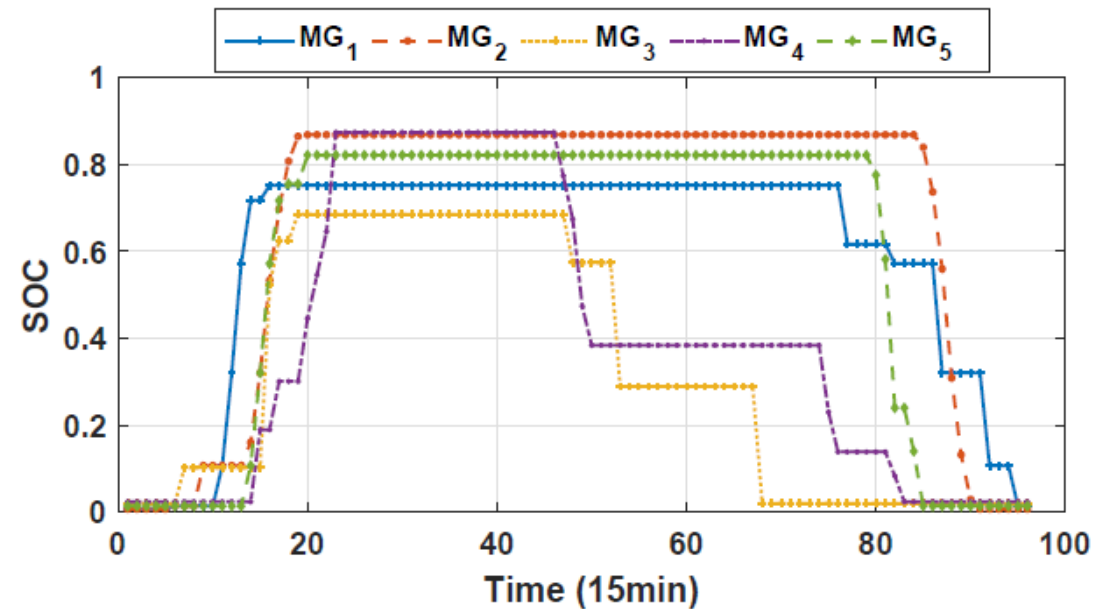
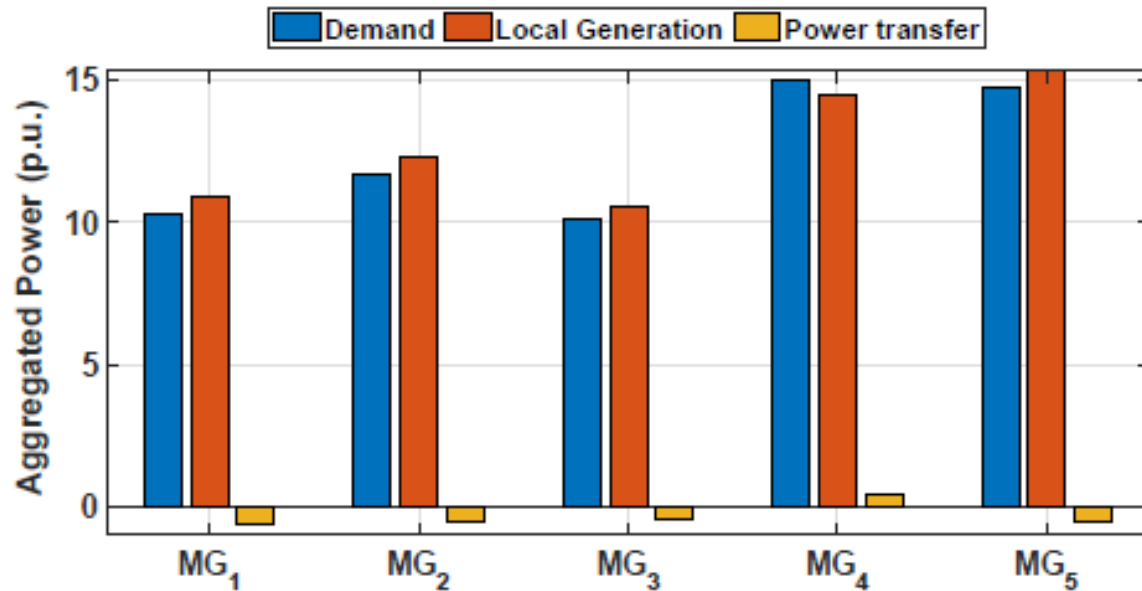
All selected DNN hyperparameters and other user-defined coefficients in simulations are summarized in the following table:

Description	Notion	Value
Length of the decision window in episode	$T$	4
Discount factor	$\gamma$	0.99
Step size for updating $\theta$	$\delta$	$1 \times 10^{-3}$
Maximum iteration	$k^{max}$	200
Weight assigned to received information	$w_n$	0.2
Step size for primal gradient update	$\rho_1$	0.01
Step size for dual gradient update	$\rho_2$	0.01
Threshold for parameter updating	$\Delta\theta$	$1 \times 10^{-4}$
Tightening multiplier	$\tau$	0.9
Number of hidden layer	-	3
Number of neurons per hidden layer	-	10
Size of minibatches	-	128
Activation function of DNNs	-	<i>tansig</i>

- All hyperparameters have been selected through cross-validations, including repeated try-outs.
- Each episode is a learning update iteration based on the data that comes from one moving decision window.
- The length of the moving window is 4 samples with a 15-minute time step, which gives us a 1-hour window.
- The DNNs have 3 hidden layers, 1 input layer, and 1 output layer, where each hidden layer consists of 10 neurons.
- The activation function of each layer of the DNN is hyperbolic tangent sigmoid transfer function (*tansig*).

# Simulation: Results

- In the case study, action selection is performed by sampling 100 times from the trained policy functions (distribution).
- A trade-off is involved in choosing the number of action samples:
  - If this number of samples is too large, then the selected action will converge to the policy mean, which implies that model uncertainties are ignored.
  - If this number of samples is too small, then the outcome can deviate from the learned mean value, which can also result in low-quality outcomes.
- Then the dispatch action is obtained by averaging the selected samples.



# Simulation: Benchmark Methods

To demonstrate the effectiveness of SMAS-PL, two benchmark RL methods have been considered:

- The benchmark unconstrained policy learning (U-PL) method leverages the same algorithm as the proposed SMAS-PL, however, certain constraints are removed during the training process of U-PL.
- The benchmark deep Q-network (DQN), which uses DNNs to approximate the Q-function and provide Q-value estimation for discretized control actions.
  - To consider constraints in DQN, penalty terms are added to the reward function of the benchmark DQN to discourage constraint violation

$$R_{t'} = - \left[ \sum_{n=1}^N \sum_{t'=t}^{t+T} (-\lambda_n^R P_{n,t'}^{PCC} + \lambda_n^F F_{n,t'}) \right] - \rho_{m'} \left[ \sum_{m'=1}^{M_C^G} \sum_{t'=t}^{t+T} (J_{C_{m',t'}-d_{m',t'}}) \right] - \rho_m \left[ \sum_{m=1}^{M_C^L} \sum_{t'=t}^{t+T} (J_{C_{m,t'}-d_{m,t'}}) \right]$$

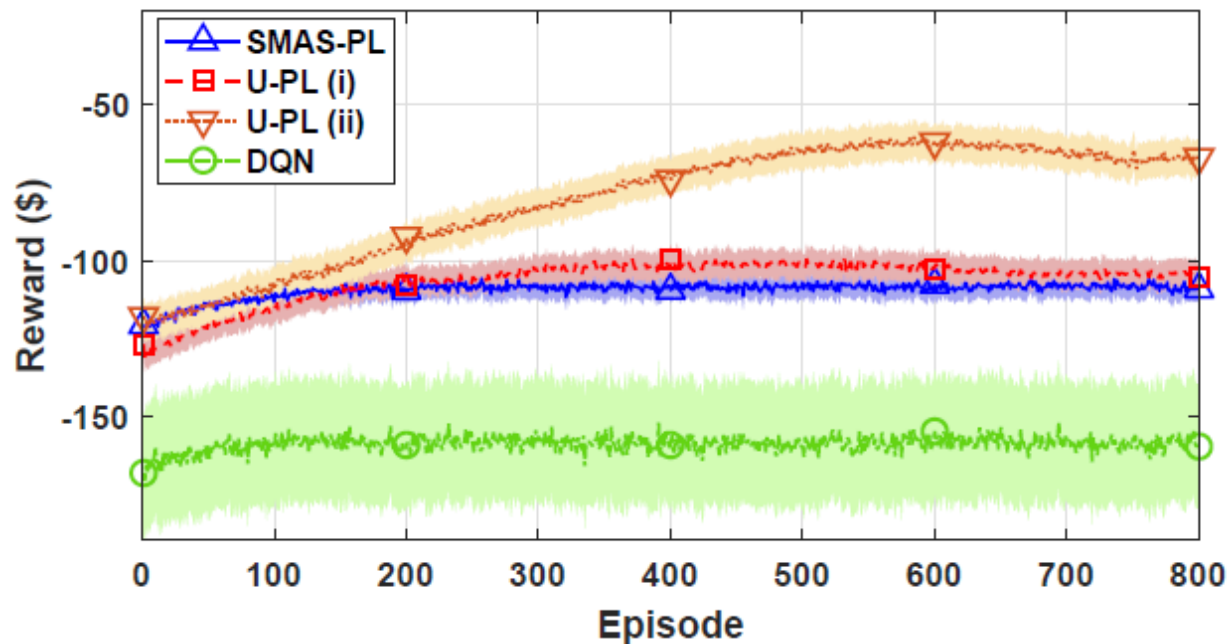
- The DNN is parameterized by  $\theta$  as a function approximator to represent the Q-value function. The temporal difference (TD) learning algorithm is used to train the DNN by minimizing the mean-squared TD error  $\mathcal{L}_\theta$

$$\mathcal{L}_\theta = E \left[ \left( R_{t'} + \gamma \max_{a_{t'+1}} Q_{\theta^*}(s_{t'+1}, a_{t'+1}) - Q_\theta(s_{t'}, a_{t'}) \right)^2 \right]$$

$$\theta \leftarrow \theta + \delta \frac{\partial \mathcal{L}_\theta}{\partial \theta}$$

# Simulation: Comparison

- The rewards under SMAS-PL, U-PL, and DQN are compared with each other.
- Note that here, the moving average rewards and the episode rewards of different methods are depicted by dark and light curves.
- Two cases are considered in implementing U-PL: (i) no DG capacity constraints for MG1 and MG2; (ii) no DG capacity constraints for MG1-MG5.



- Both SMAS-PL and U-PL both outperform DQN in term of the total reward.
- Both SMAS-PL and U-PL both leverage the proposed iterative and distributed technique to adaptively tune the Lagrangian multipliers through information exchange between MG agents.
- DQN needs to manually design penalty coefficients for constraint violations, which either offers inadequate penalization of the constraint violations or excessive punishment for the constraints.



# Simulation: Comparison

This table presents comparisons between the benchmark optimization-based method, the benchmark DQN and the proposed SMAS-PL, including the average daily cost of operation over numerous scenarios, average online decision time, and MG privacy maintenance.

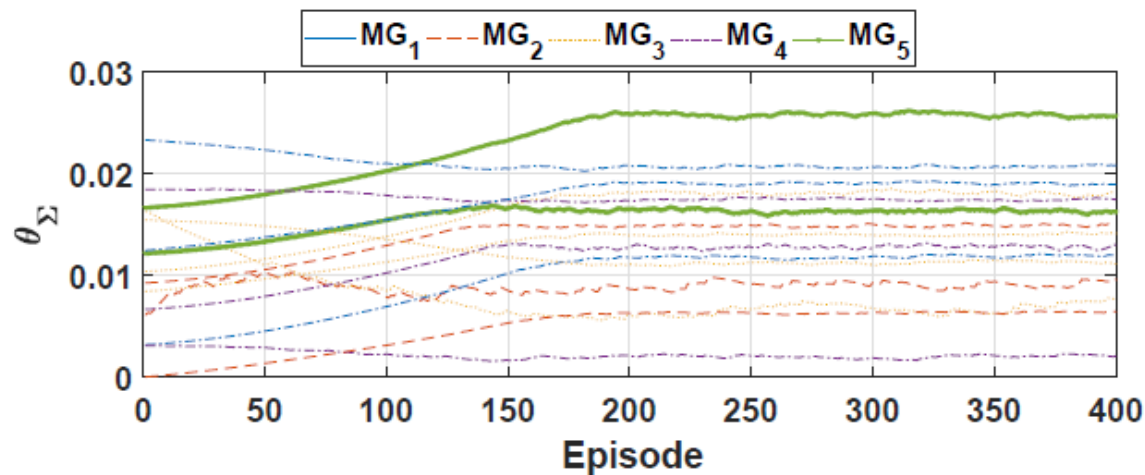
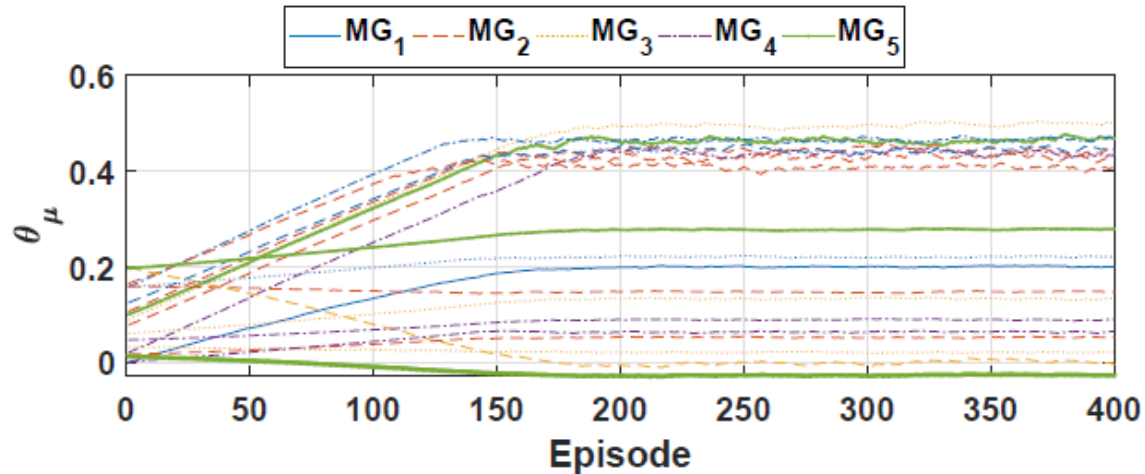
	Cen. solver	DQN	SMAS-PL
Average daily cost (\$)	1356.60	1928.4	1372.11
Average time (second)	145.50	10.30	1.40 (per agent)
MG privacy maintenance	No	No	Yes

- The daily cost for the proposed SMAS-PL is close to the centralized optimization solver.
- The online decision time for the proposed SMAS-PL is 1.4 seconds per agent (samples the actions from the learned multivariate Gaussian distributions), which is shorter than the 10.3 seconds for DQN (solve an optimization problem for optimal action) and 145.5 seconds for the centralized optimization solver (solve a large-scale optimization problem).
- Due to its distributed nature, the proposed SMAS-PL method maintains the privacy and data ownership boundaries of individual MG.



# Simulation: Convergence

The following figures show the convergence of a selected group of learning parameters, and during the training process, for each MG agent.

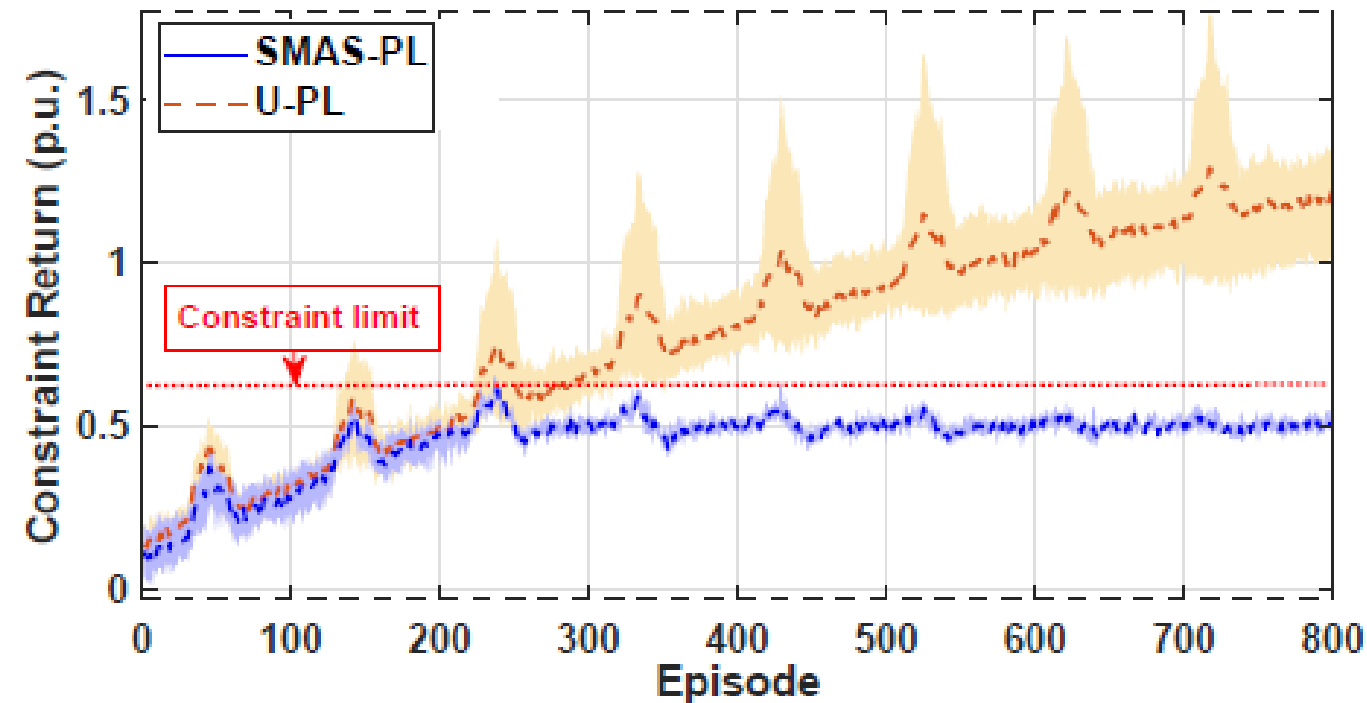


- The changes in  $\theta_\mu$  are relatively larger than that of  $\theta_\Sigma$ .
- This is due to the higher levels of sensitivity of MG agents' objective functions to the mean  $\mu$  of the control actions compared with their covariance  $\Sigma$ .

# Simulation: Check Local Constraints

This figure shows the constraint return values during the training iterations.

- Two cases are built with and without DG capacity constraints in  $MG_1$ .
- The dark blue and red curves represent averaged constraint returns and the light blue and red areas around the average curves represent variation ranges of the constraint return for the SMAS-PL and black-box RL, respectively.

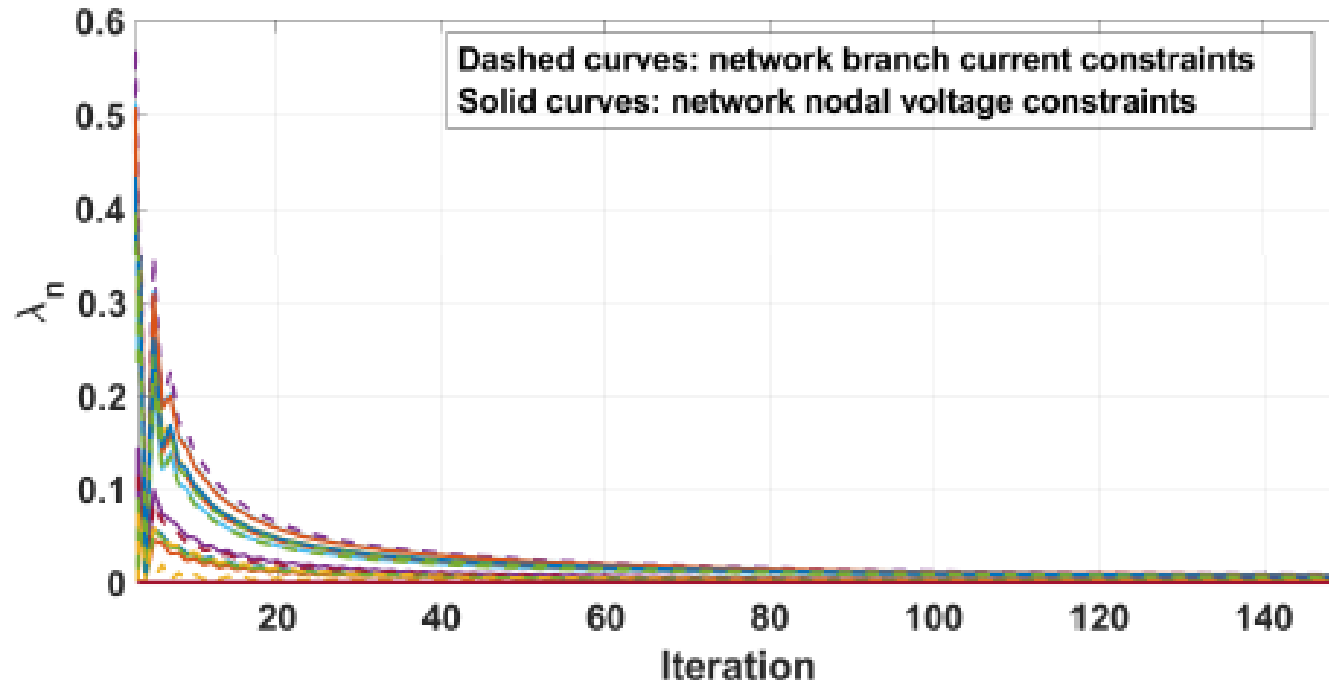


- U-PL violates the upper boundary for DG generation limit (i.e., local constraint case study).
- SMAS-PL solver satisfies the DG generation capacity constraints, which implies that the local constraints can be safely maintained

# Simulation: Check Global Constraints

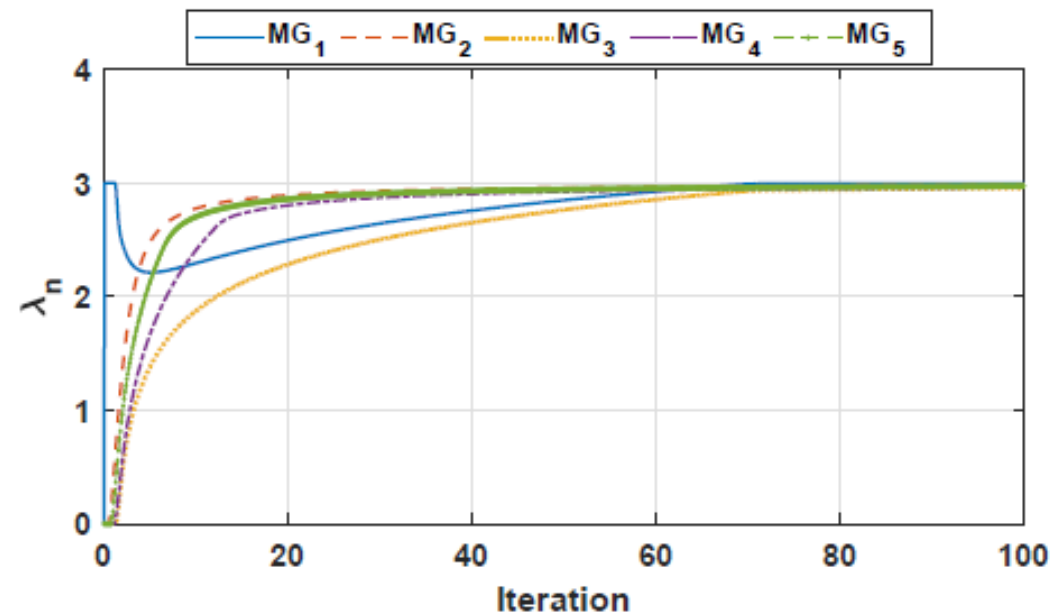
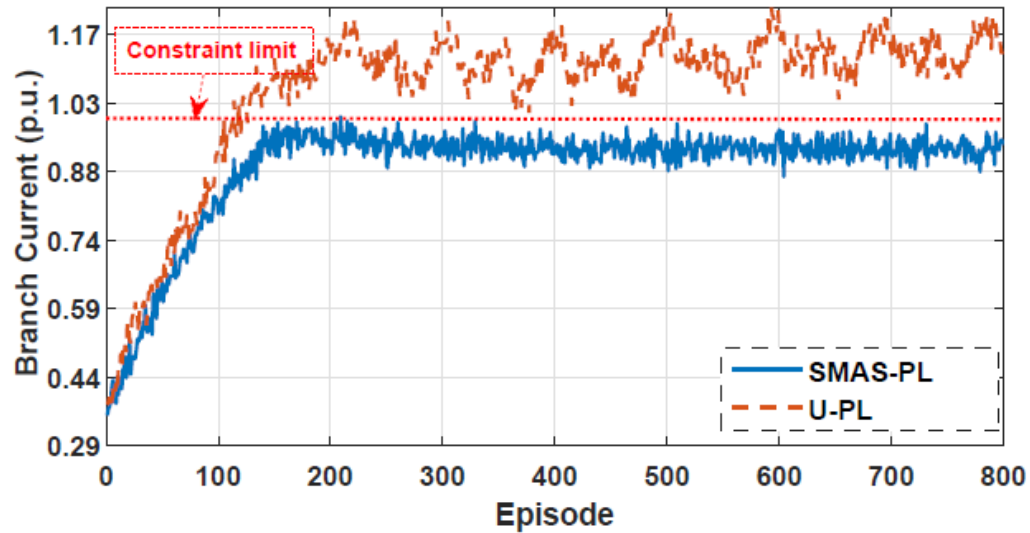
This figure shows one example of the iterative distributed training convergence process for a policy gradient update step.

- Dashed curves represent the Lagrangian multipliers for network branch current constraints
- Solid curves represent the Lagrangian multipliers for network nodal voltage constraints



- As can be seen, the Lagrangian multipliers reach zero over iterations of the proposed multi-agent algorithm,
- It indicates that all the global constraints, including nodal voltage and branch current limits, are satisfied and feasible solutions are obtained.
- It also means that the bus voltage and line current constraints are not binding for this case.

# Simulation: Check Global Constraints



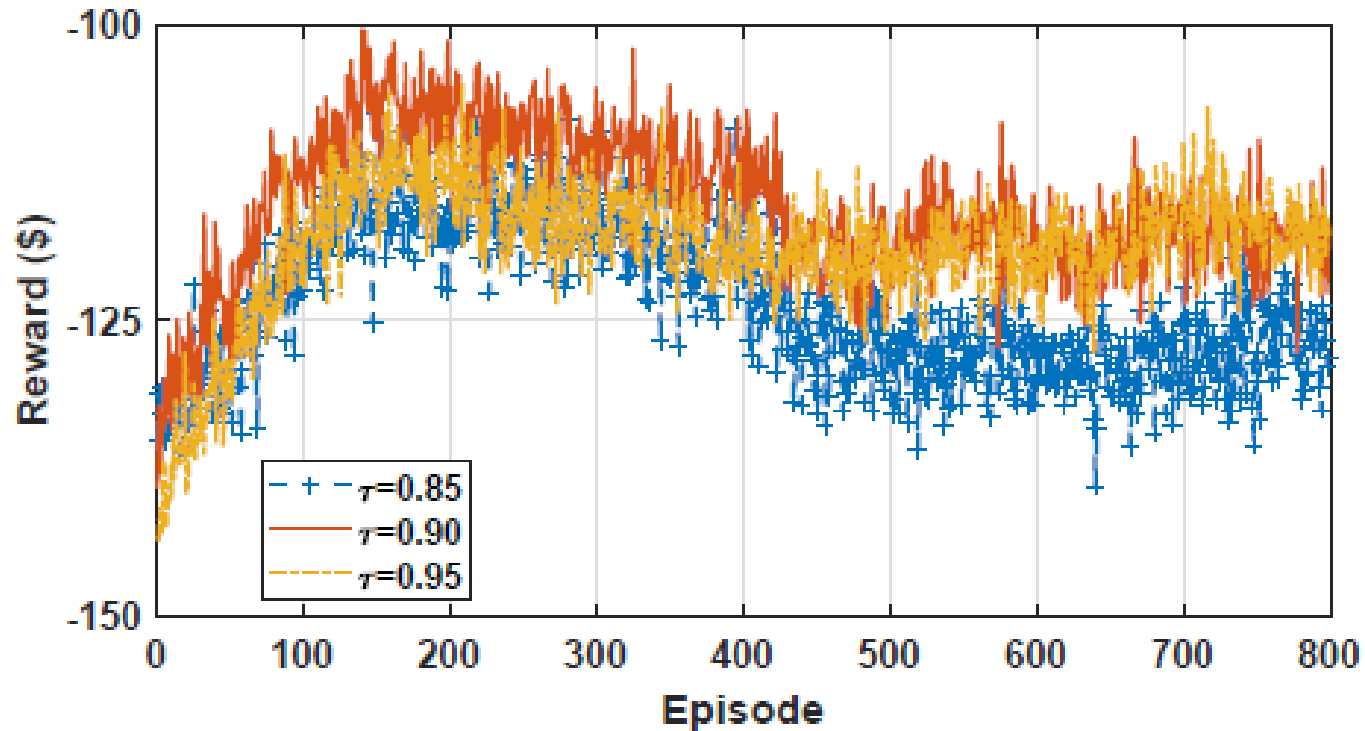
When handling binding global constraints.

- The first figure shows a line flow constraint in the grid under the proposed SMAS-PL and a U-PL baseline.
- The U-PL has generated infeasible decisions that violate the constraint, while SMAS-PL has prevented the flow to go above its upper bound.
- In second figure, the Lagrangian multipliers for this binding constraint reach a non-zero constant number over iterations.
- This also shows the agents' estimations of Lagrange multipliers for a global line flow constraint.
- The proposed SMAS-PL can reach consensus between agents on the value of the multiplier without having any access to each other's policy functions,

# Simulation: Impact of Backtracking

To validate the tightening parameters level ( $d_m^* = \tau \times d_m$ ) in backtracking strategy, we have studied the impact of different  $\tau$  values on the reward.

- At episode 400, the value of  $\tau$  is decreased from 1 to 0.95, 0.9 or 0.85.

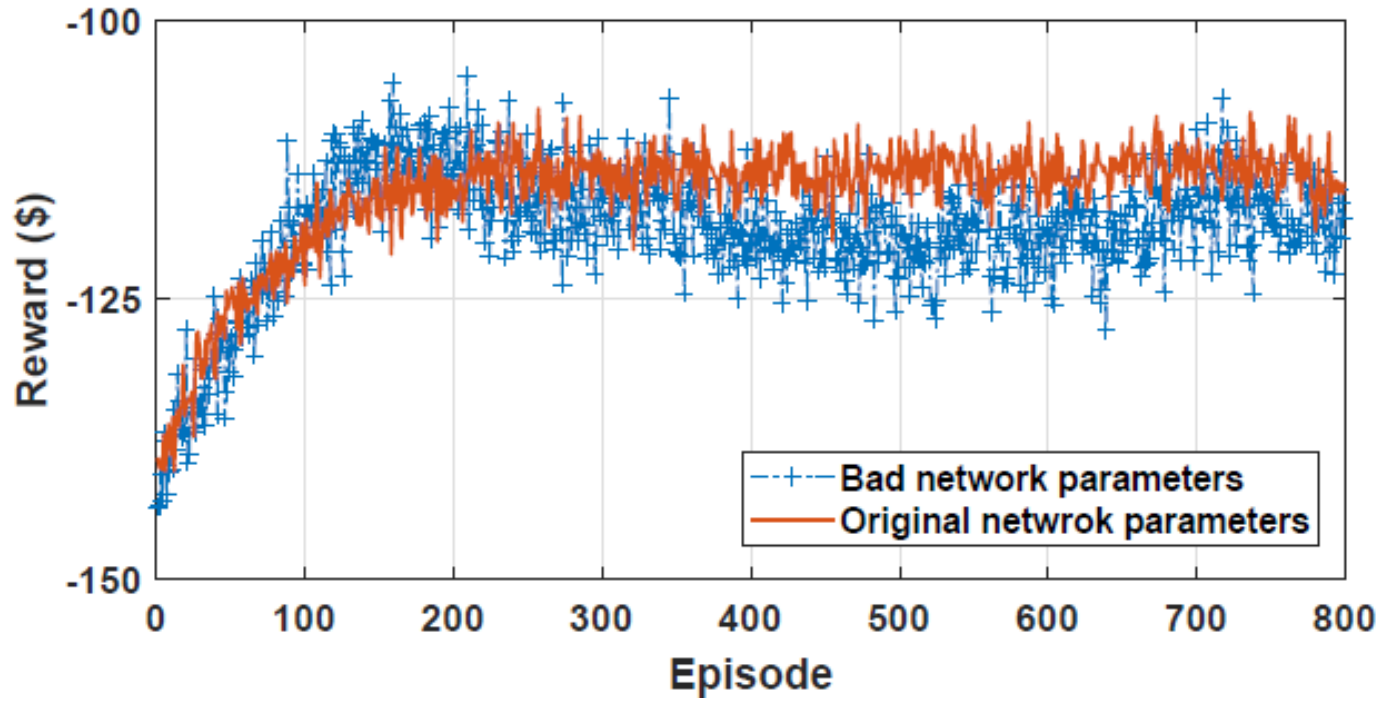


- When  $\tau$  is close to 1 (i.e., 0.95 or 0.9), the reward values are very close to each other. When  $\tau=0.85$ , the reward drops significantly.
- In our simulation, we have observed that  $\tau=0.9$  is sufficient for ensuring feasibility for those few constraints that have been marginally violated in certain operation scenarios after one to two rounds of backtracking.
- Note that this threshold needs to be fine-tuned for specific grids.

# Simulation: Impact of Bad Network Data

To validate the SMAS-PL under network data imperfection, we have compared the average reward obtained with perfect knowledge of network parameters and under bad network parameter information.

- To simulate the impact of bad network parameter data on model performance, we have added random errors (with a 10% variance) to the network resistance (R) and reactance (X) parameters during the training process.



- The bad network data will lead to errors in gradient factors.
- Even though the learning process with bad network data shows more volatility and needs more time to reach convergence, the model still reaches reward values close to the ideal case.
- However, due to the imperfect information, a loss of reward is inevitable.
- We put topology change in the future work.

# Conclusions

- Conventional model-based optimization methods suffer from high computational costs when solving large-scale multi-MG power management problems. On the other hand, the conventional model-free methods are black-box tools, which may fail to satisfy the operational constraints.
- Our proposed SMAS-PL method exploits the gradients of the decision problem to learn control policies that achieve both optimality and feasibility.
- To enhance computational efficiency and maintain the policy privacy of the control agents, a distributed consensus-based training process is implemented to update the agents' policy functions over time using local communication.



# Future Work

1. During the service restoration process for low-inertia inverter-dominated MGs, then the frequency response due to large load pick up shall be considered.
  - We are developing a frequency dynamics constrained sequential service restoration method.
2. If the topology changes, the policy function for each MG agent needs to be re-trained to guarantee the satisfaction of constraints.
  - Meta learning + RL: Meta learning is “learn to learning”. The meta-parameters such as learning rate, exploration rate and discount factor, can be pre-trained.
  - Topology embedded graph convolutional network (GCN) + RL: GCN develops an explicit way of integrating topological structures into the convolution algorithm. The basic idea behind GCN is to distill the high-dimensional information about a node’s graph neighborhood into a vector representation with dimension reduction.

# Publications

## Published

- Q. Zhang, K. Dehghanpour, Z. Wang, F. Qiu and D. Zhao, "Multi-agent safe policy learning for power management of networked microgrids," in IEEE Transactions on Smart Grid, vol. 12, no. 2, pp. 1048-1062, March 2021.
- Q. Zhang, K. Dehghanpour, Z. Wang and Q. Huang, "A learning-based power management method for networked microgrids under incomplete information," in IEEE Transactions on Smart Grid, vol. 11, no. 2, pp. 1193-1204, March 2020.
- Q. Zhang, K. Dehghanpour and Z. Wang, "Distributed CVR in unbalanced distribution systems with PV penetration," in IEEE Transactions on Smart Grid, vol. 10, no. 5, pp. 5308-5319, Sept. 2019.

## Under review

- Q. Zhang, Y. Guo, Z. Wang and F. Bu, "Distributed optimal conservation voltage reduction in integrated primary-secondary distribution systems," in IEEE Transactions on Smart Grid, under review.
- Q. Zhang, Z. Ma, Y. Zhu, and Z. Wang, "Frequency dynamics constrained sequential load restoration in inverter-based microgrids," in IEEE Transactions on Smart Grid, under review.
- Q. Zhang, Z. Wang, S. Ma and A. Arif, "Stochastic pre-event preparation for enhancing resilience of distribution systems with high DER penetration," Renewable and Sustainable Energy Reviews, under review.

# References

- D. E. Olivares et al., "Trends in Microgrid Control," IEEE Trans. Smart Grid, vol. 5, no. 4, pp. 1905-1919, July 2014.
- Z. Wang, B. Chen, J. Wang, M. Begovic, and C. Chen, "Coordinated energy management of networked microgrids in distribution systems," IEEE Trans. Smart Grid, vol. 6, no. 1, pp. 45–53, Jan. 2015.
- Tianguang Lu, Zhaoyu Wang, Qian Ai, Wei-Jen Lee, "Interactive Model for Energy Management of Clustered Microgrids", IEEE Trans. Ind. Appl., vol. 53, no. 3, pp. 1739-1750, 2017.
- H. Farzin, M. Fotuhi-Firuzabad and M. Moeini-Aghtaie, "Role of Outage Management Strategy in Reliability Performance of Multi-Microgrid Distribution Systems," IEEE Trans. Power Syst., vol. 33, no. 3, pp. 2359-2369, May 2018.
- Z. Wang, B. Chen, J. Wang, and J. Kim, "Decentralized Energy Management System for Networked Microgrids in Grid-connected and Islanded Modes," IEEE Trans. Smart Grid, vol. 7, no. 2, pp. 1097-1105, March 2016.
- Mojtaba Khederzadeh, "Multi-agent system design for automation of a cluster of microgrids", *CIREC - Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 1304-1307, 2017.
- Akhtar Hussain, Van-Hai Bui, Hak-Man Kim, "A Resilient and Privacy-Preserving Energy Management Strategy for Networked Microgrids", IEEE Trans. Smart Grid, vol. 9, no. 3, pp. 2127-2139, 2018.
- R. S. Sutton and A. G. Barto, Reinforcement learning: an introduction. London, England: The MIT Press, 2017
- D. Shi, X. Chen, Z. Wang, X. Zhang, Z. Yu, X. Wang, and D. Bian, "A distributed cooperative control framework for synchronized reconnection of a multi-bus microgrid," IEEE Trans. Smart Grid, vol. 9, no. 6, pp. 6646–6655, Nov. 2018.
- Q. Zhang, K. Dehghanpour, Z. Wang, and Q. Huang, "A learning-based power management method for networked microgrids under incomplete information," IEEE Trans. Smart Grid, vol. 11, no. 2, pp. 1193–1204, March 2020.
- Y. Ye, D. Qiu, X. Wu, G. Strbac and J. Ward, "Model-free real-time autonomous control for a residential multi-energy system using deep reinforcement learning," IEEE Trans. Smart Grid, vol. 11, no. 4, pp. 3068-3082, July 2020.
- X. Sun and J. Qiu, "Two-Stage Volt/Var Control in Active Distribution Networks with Multi-Agent Deep Reinforcement Learning Method," IEEE Trans. Smart Grid, Early Access.
- Y. Gao, W. Wang and N. Yu, "Consensus Multi-Agent Reinforcement Learning for Volt-VAR Control in Power Distribution Networks," IEEE Trans. Smart Grid, Early Access.
- H. Liu and W. Wu, "Online Multi-agent Reinforcement Learning for Decentralized Inverter-based Volt-VAR Control," IEEE Trans. Smart Grid, Early Access.
- Z. Yan and Y. Xu, "A Multi-Agent Deep Reinforcement Learning Method for Cooperative Load Frequency Control of a Multi-Area Power System," IEEE Trans. Power System, vol. 35, no. 6, pp. 4599-4608, Nov. 2020.
- J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in Proc. of 34th International Conference on Machine Learning, 2017.

Thank You!  
Q & A

# Backup Slides

# Gradient Factor Determination

The gradients  $\frac{\partial a_n}{\partial \pi_n}$ ,  $\frac{\partial \pi_n}{\partial \mu_n}$  and  $\frac{\partial \pi_n}{\partial \Sigma_n}$  are obtained using the probability density function of multivariate Gaussian distribution. The multivariate Gaussian distribution is:

$$f(x; \mu, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^D}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

The derivative of distribution function  $f$  w.r.t. mean vector  $\mu$  and covariance matrix  $\Sigma$  can be written as follows:

$$\frac{\partial f}{\partial \mu} = \frac{\Sigma^{-1}(x - \mu)}{\sqrt{|\Sigma|(2\pi)^D}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

$$\frac{\partial f}{\partial \Sigma} = -\frac{1}{2} \frac{(\Sigma^{-1} - \Sigma^{-1}(x - \mu)(x - \mu)^T \Sigma^{-1})}{\sqrt{|\Sigma|(2\pi)^D}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

The derivative of distribution function  $f$  w.r.t. variable  $x$  is shown as follows:

$$\frac{\partial f}{\partial x} = \frac{\Sigma^{-1}(x - \mu)}{\sqrt{|\Sigma|(2\pi)^D}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

# Gradient Factor Determination

The gradients of DNNs' outputs w.r.t. DNN parameters  $\frac{\partial \mu_n}{\partial \theta_{\mu_n}}$  and  $\frac{\partial \Sigma_n}{\partial \theta_{\Sigma_n}}$  are obtained by using a back-propagation method:

- In each iteration, the latest values of state variables are employed as inputs of the DNNs.
- The back-propagation process exploits chain rule for stage-by-stage spreading of gradient information through layers of the DNNs, starting from the output layer and moving towards the input layer.
- To enhance the stability of the back-propagation process, a simple batch approach is adopted, where the gradients obtained from several sampled actions are averaged to ensure robustness against outliers.