Multi-Agent Safe Policy Learning for Power Management of Networked Microgrids

Qianzhi Zhang, *Student Member, IEEE,* Kaveh Dehghanpour, *Member, IEEE,* Zhaoyu Wang, *Member, IEEE,* Feng Qiu, *Senior Member, IEEE,* and Dongbo Zhao, *Senior Member, IEEE*

***Abstract*—This paper presents a supervised multi-agent safe policy learning (SMAS-PL) method for optimal power management of networked microgrids (MGs) in distribution systems. While conventional reinforcement learning (RL) algorithms are *black-box* decision models that could fail to satisfy grid operational constraints, our proposed method is constrained by AC power flow equations and other operational limits. Accordingly, the training process employs the gradient information of operational constraints to ensure that the optimal control policy functions generate safe and feasible decisions. Furthermore, we have developed a distributed consensus-based optimization approach to train the agents' policy functions while maintaining MGs' privacy and data ownership boundaries. After training, the learned optimal policy functions can be safely used by the MGs to dispatch their local resources, without the need to solve a complex optimization problem from scratch. Numerical experiments have been devised to verify the performance of the proposed method.**

***Index Terms*—Safe policy learning, multi-agent framework, networked microgrids, power management, policy gradient.**

NOMENCLATURE

Indices

i, j	Indices of buses, $\forall i, j \in \Omega_I$.
ij	Index of branch between bus i and bus j, $\forall ij \in \Omega_{Br}$.
k	Iteration index in distributed optimization, $k \in \{1, ..., k^{max}\}$.
m	Constraint index, $m \in \{1, ..., M_c\}$.
n	Agent index, $n \in \{1, ..., N\}$.
t'	Episode index in training process, $t' \in [t, t+T]$.

Parameters

a_n^f, b_n^f, c_n^f	Coefficients of the DG quadratic cost function for agent n.
$\boldsymbol{b_{m,\mu_n}}$	Gradient vector of the constraint return function m w.r.t. the parameters μ_n.
$\boldsymbol{b_{m,\Sigma_n}}$	Gradient vectors of the constraint return function m w.r.t. the parameters Σ_n.
D_n	Dimension of multivariate Gaussian distribution function for agent n.
d_m	Upper limit for constraint m.
E^{Cap}	Max. capacity of ESS unit.
H_n	Fisher information matrix of agent n.
$\boldsymbol{g_{\mu_n}}$	Gradient vector of the reward functions w.r.t. the parameters μ_n.
$\boldsymbol{g_{\Sigma_n}}$	Gradient vector of the reward functions w.r.t. the parameters Σ_n.
I_{ij}^M	Max. current limit on branch ij.
M_c	Number of constraints.
M_c^G	Number of global constraints.
M_c^L	Number of local constraints.
N	Number of MGs.
N_n	Number of neighboring MGs for agent n.
$P^{Ch,M}$	Max. ESS charging limits.
$P^{Dis,M}$	Max. ESS discharging limits.
P^D, Q^D	Active and reactive load power.
$P^{DG,M}$	Max. DG active power capacity.
$Q^{DG,M}$	Max. DG reactive power capacity.
$P^{DG,R}$	Max. DG ramp limit.
P^{PV}	PV active power output.
$P^{PCC,M}$	Max. active power flow at the PCCs.
$Q^{PCC,M}$	Max. reactive power flow at the PCCs.
$Q^{PV,M}$	Max. PV reactive power output limit.
SOC^M	Max. SOC limits.
SOC^m	Min. SOC limits.
T	Length of the moving decision window.
V_i^M, V_i^m	Max. and min. voltage limit on bus i.
$w_n(n')$	Weight parameters assigned of agent n to neighboring agent n'.
Y^{Re}, Y^{Im}	Real and imaginary parts of the nodal admittance matrix Y.
η_{Ch}, η_{Dis}	Charging and discharging efficiency of ESS.
λ^F	Diesel generator fuel price.
λ^R	Retail price signals at the PCCs.
$\boldsymbol{\theta_{\mu_n}}, \boldsymbol{\theta_{\Sigma_n}}$	Vector of DNN weights and bias of agent n.
$\boldsymbol{\mu_n}, \Sigma_n$	Mean vector and covariance matrices for control action of agent n.
δ, ρ_1	Step sizes for updating θ and λ.
ρ_2	Penalty factor for constraints violation.
Δt	Time step.
$\Delta\theta_n$	Threshold for parameter updating.
γ	Discount factor.
τ	Tightening multiplier.

Variables

$\boldsymbol{a_n}$	Vector of control actions of agent n.
$C_m(\pi)$	Return value of constraint m based on the control policy π.
$F_{i,n}$	Fuel consumption of DG at bus i of agent n.
I_i^{Re}, I_i^{Im}	Real and imaginary parts of the injected current at bus i.

Q. Zhang, K. Dehghanpour, and Z. Wang are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: wzy@iastate.edu).

F. Qiu and D. Zhao are with Energy Systems Division, Argonne National Laboratory, Lemont, IL 60439 USA (e-mail: fqiu@anl.gov, dongbo.zhao@anl.gov).

I_{ij}^{Re}, I_{ij}^{Im}	Real and imaginary parts of the branch current at branch ij.
$\boldsymbol{O_t}$	Vectors of observation variable.
P^{Ch}, P^{Dis}	Charging and discharging power of ESS unit.
P^{DG}, Q^{DG}	DG active and reactive power outputs
P^{PCC}	Active power flow at the PCC.
Q^{PCC}	Reactive power flow at the PCC.
Q^{ESS}	Reactive power outputs of ESS unit.
Q^{PV}	PV inverter reactive power output.
SOC	SOC of the battery system.
$\boldsymbol{S_n}$	Vectors of system state of agent n.
V_i^{Re}, V_i^{Im}	Real and imaginary parts of the bus voltage magnitude at bus i.
$\boldsymbol{\lambda_n}$	Vector of Lagrangian multipliers.
Functions	
J_{R_n}	Expected reward function of agent n.
J_{C_m}	Expected return function of constraint m.
π_n	Multivariate distribution function over control actions of agent n.
Δ	Kullback Leibler (KL)-divergence function.

I. Introduction

NETWORKED microgrids (MGs) can offer various benefits, including higher perpetration of local distributed energy resources (DERs), improved controllability, and enhancement of power system resilience and reliability [1], [2]. Solving the power management problem of networked MGs is a complex task. While previous works in this area have provided valuable insight, we have identified two shortcomings in the literature:

(1) *Limitations of model-based optimization methods:* In the existing literature, there are quite a few model-based methods for solving the optimal power management problem of networked MGs, such as centralized decision models [3]–[5] and distributed control frameworks [6]–[8]. However, with increasing number of MGs in distribution networks, these methods have to solve large-scale optimization problems with numerous nonlinear constraints that incur high computational costs and hinder real-time decision making. Furthermore, model-based methods are unable to adapt to the continuously evolving system conditions, as they need to re-solve the problem at each time step.

(2) *Potential infeasibility of model-free machine learning methods:* To address the limitations of model-based methods, model-free reinforcement learning (RL) techniques have been used to solve the optimal power management problem through repeated interactions between a control agent and its environment. This approach eliminates the need to solve a large-scale optimization problem at each time point and enables the control agent to provide adaptive response to time-varying system states. Existing examples of RL application in power systems include economic dispatch and energy consumption scheduling of individual MGs [9]–[11] and multi-area smart control of generation in interconnected power grids [12], [13]. Further, in our previous paper [14], we proposed a bi-level power management method for networked MGs, where a centralized RL agent determines retail prices in a cooperative business model for each MG under the incomplete information of physical model. Current RL-based solutions employ control agents to train *black-box* functions to approximate the optimal actions through trial and error. However, the trained black-box functions can fail to satisfy critical operational constraints, such as network nodal voltage and capacity limits, since these constraints have not been encoded in the training process. This can lead to unsafe operational states and control action infeasibility.

However, incorporating constraints into the training process of conventional black-box methods is challenging since these methods have generally relied on adding penalty terms to training objective functions for enforcing constraints, which cannot guarantee the safety of control policies as the number of constraints grows. Inspired by recent advances in constrained policy learning (PL) [15]–[17] and to address the shortcomings in the existing literature, we have cast the power management of networked MGs as a *supervised multi-agent safe PL problem* (SMAS-PL). Moreover, we have proposed a multi-agent policy gradient solution strategy to learn optimal control policies for the networked MGs in a distributed way. The proposed method introduces a trade-off between model-free and model-based methods and combines the benefits offered by both sides. The purpose is to leverage the advantages of both model-free and model-based methods, for scalable real-time decision making while also maintaining some level of safety by considering constraints in the training process. Hence, on one hand, MGs' power management policy functions are modeled using black-box deep neural networks (DNNs); while on the other hand, to ensure decision feasibility, a constrained gradient-based training method is proposed that exploits the derivatives of the constraints and objective functions of the power management problem w.r.t. control actions and learning parameters. The training process employs these gradient factors to provide a convex quadratically constrained linear program (QCLP) approximation to the power management problem at each episode. This enables the proposed method to be both adaptable to changes in the inputs of the black-box components, and feasible with respect to operational constraints, including AC power flow. Finally, a distributed consensus-based primal-dual optimization method [18] is adopted to decompose the training task among MG agents. In summary, compared to existing decision making solutions, the main advantages of this paper are as follows:

- Compared to the black-box learning-based methods, the proposed SMAS-PL leverages the gradient information of all the operational constraints to devise a tractable QCLP-based training process to promote the safety and feasibility of control policies. A backtracking mechanism is added into the PL framework to perform a final verification of feasibility before issuing control commands to the assets.
- Compared to conventional centralized training methods, the distributed training process in the SMAS-PL offers two advantages: it preserves the privacy of MG agents, including their control policies parameters and structures, operation cost functions, and local asset constraints; it

also enhances computational efficiency and maintains scalability as the number of learning parameters grows into a humongous size.

- The proposed SMAS-PL method does not need to solve a complex optimization problem in real-time. The agents' policy functions, that are trained offline, can be leveraged online to select optimal control actions in response to latest system state data.

The reminder of the paper is organized as follows: Section II presents the overall framework of the proposed solution. Section III introduces the SMAS-PL problem and integrates problem gradients into the solver. Section IV describes the multi-agent consensus-based training algorithm for SMAS-PL. Simulation results and conclusions are given in Section V and Section VI, respectively.

II. Overview of the Proposed Framework

The general framework of the proposed SMAS-PL method is shown in Fig. 1. Note that vectors are denoted in bold letters throughout the paper. The micro-sources within each MG are controlled by an agent that adopts a private control policy. Here, the *control policy* for the n'th agent, π_n, is a parametric probability distribution function, with parameters $\boldsymbol{\theta_n}$, over the agent's control actions ($\boldsymbol{a_{n,t}}$), including active/reactive power dispatching signals for local diesel generators (DGs), energy storage system (ESS) and solar photo-voltaic (PV) panels. Note that the control policy π_n is a function of the MG's state variables ($\boldsymbol{S_{n,t}}$), defined by the aggregate MG load and solar irradiance. To ensure the safety of the control policies, MG agents receive the observed variables from the grid, including network nodal voltages $\boldsymbol{V_t}$ and injection currents $\boldsymbol{I_t}$, to determine gradient factors of the problem constraints and objectives w.r.t. to learning parameters, $\nabla_{\boldsymbol{\theta}}\boldsymbol{J}$. These gradient factors are then integrated into a multi-agent constrained training algorithm, which employs local inter-MG communication to satisfy all global and local operational constraints through exchanging and processing dual Lagrangian variables, λ(t). The Lagrangian multipliers embody the interactions among the MGs and capture the impacts of MGs' decisions on each other. Theoretical analysis and numerical simulations are conducted to show that the proposed SMAS-PL method can minimize the MG agents' operational cost and satisfy operational constraints. Note that the proposed SMAS-PL is not a purely model-free approach, since the AC power flow equations are used to calculate gradient factors and ensure the decision feasibility when training the DNNs.

III. Safe Policy Learning for Power Management of Networked MGs

To facilitate the discussion, Section III-A introduces a general power management formulation that is commonly used in literature [4], [6], [14]. Sections III-B defines each component of the proposed SMAS-PL. In Sections III-C and III-D, we propose a tractable SMAS-PL method, employing the gradient factors of reward function and constraint return functions w.r.t. actions and learning parameters, to solve the power management of networked MGs.

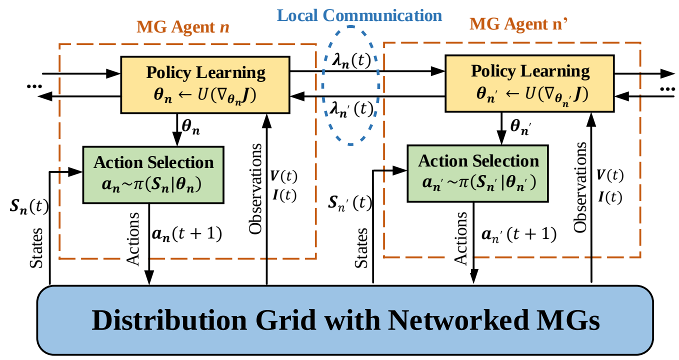


Fig. 1. Structure of the proposed SMAS-PL method for power management of networked MGs

A. Power Management Problem Statement

Each MG is assumed to have local DGs, ESS, solar PV panels and a number of loads. This optimization problem is solved over a moving look-ahead decision window $t' \in [t, t+T]$, using the latest estimations of solar and load power at different instants. Here, n is the MG index ($n \in \{1, ..., N\}$), i and j define the node numbers ($\forall i, j \in \Omega_i$), ij defines the branch numbers ($\forall ij \in \Omega_{Br}$).

*1) **Problem objective***: The objective function (1), with control action vector $[P^{DG}, P^{Ch}, P^{Dis}, Q^{DG}, Q^{PV}, Q^{ESS}] \in (\boldsymbol{x_p}, \boldsymbol{x_q})$, minimizes MGs' total cost of operation, which is composed of the income/cost from power transfer with the grid and cost of running local DG. Here, λ_n^F is the DG fuel price in $\$/L$, λ_n^R is the electricity price in $\$/kWh$, and $P_{n,t'}^{PCC}$ is active power transfer between grid and the n'th MG at the point of common coupling (PCC). The fuel consumption of DG, $F_{i,n,t'}$, can be expressed as a quadratic polynomial function of its power, $P_{i,n,t'}^{DG}$, with parameters $a_n^f = 0.0001773\ L/kW^2$, $b_n^f = 0.1709\ L/kW$, and $c_n^f = 14.67L$ adopted from [19], as shown in (2).

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

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

*2) **Global constraints***: These constraints are defined over variables that are impacted by control actions of all the MGs, including the voltage amplitude limits for the entire nodes, $[V_i^m, V_i^M]$, and the maximum permissible branch current flow magnitudes I_{ij}^M throughout the distribution grid and the MGs:

$$V_i^m \leq V_{i,t'} \leq V_i^M \tag{3}$$

$$-I_{ij}^M \leq I_{ij,t'} \leq I_{ij}^M \tag{4}$$

Note that the global constraints are implicitly determined by the AC power flow equations, which will be used to calculate the gradient factors of objective (1) and constraints (3)-(16) w.r.t. learning parameters as elaborated in Section III-D.

*3) **Local constraints***: These constraints are defined over the local control actions of each MG. Constraints (5)-(6) ensure that the DG active/reactive power outputs, $P_{i,n}^{DG}/Q_{i,n}^{DG}$, are within the DG power capacity $P_{i,n}^{DG,M}/Q_{i,n}^{DG,M}$, and (7) enforces the maximum DG ramp limit, $P_{i,n}^{DG,R}$. PV reactive power output, $Q_{i,n}^{PV}$, is constrained by its maximum limit $Q_{i,n}^{PV,M}$ per (8). The active power transfer $P_{n,t'}^{PCC}$ and the reactive power transfer $Q_{n,t'}^{PCC}$ at the PCCs are bounded with the constraints (9) and (10), respectively.

$$0 \leq P_{i,n,t'}^{DG} \leq P_{i,n}^{DG,M} \tag{5}$$

$$0 \leq Q_{i,n,t'}^{DG} \leq Q_{i,n}^{DG,M} \tag{6}$$

$$|P_{i,n,t'}^{DG} - P_{i,n,t'-1}^{DG}| \leq P_{i,n}^{DG,R} \tag{7}$$

$$|Q_{i,n,t'}^{PV}| \leq Q_{i,n}^{PV,M} \tag{8}$$

$$|P_{n,t'}^{PCC}| \leq P_{n}^{PCC,M} \tag{9}$$

$$|Q_{n,t'}^{PCC}| \leq Q_{n}^{PCC,M} \tag{10}$$

The operational ESS constraints are described by (11)-(16), where (11) determines the state of charge (SOC) of ESSs, $SOC_{i,n}$. $E_{i,n}^{Cap}$ denotes the maximum capacity of ESSs. To ensure safe ESS operation, the SOC and charging/discharging power of ESS, $P_{i,n}^{Ch}$, $P_{i,n}^{Dis}$, are constrained as shown in (12)-(16). Here, $[SOC_{i,n}^{m}, SOC_{i,n}^{M}]$, $P_{i,n}^{Ch,M}$ and $P_{i,n}^{Dis,M}$ define the permissible range of SOC, and maximum charging and discharging power, respectively. Constraint (15) indicates that ESSs cannot charge and discharge at the same time instant. And η_{Ch}/η_{Dis} represents the charging/discharging efficiency. The reactive power of ESS, $Q_{i,n}^{ESS}$, is kept within maximum limit, $Q_{i,n}^{ESS,M}$, through constraint (16).

$$SOC_{i,n,t'} = SOC_{i,n,t'-1} + \Delta t \frac{(P_{i,n,t'}^{Ch}\eta_{Ch} - P_{i,n,t'}^{Dis}/\eta_{Dis})}{E_{i,n}^{Cap}} \tag{11}$$

$$SOC_{i,n}^{m} \leq SOC_{i,n,t'} \leq SOC_{i,n}^{M} \tag{12}$$

$$0 \leq P_{i,n,t'}^{Ch} \leq P_{i,n}^{Ch,M} \tag{13}$$

$$0 \leq P_{i,n,t'}^{Dis} \leq P_{i,n}^{Dis,M} \tag{14}$$

$$P_{i,n,t'}^{Ch} P_{i,n,t'}^{Dis} = 0 \tag{15}$$

$$|Q_{i,n,t'}^{ESS}| \leq Q_{i,n}^{ESS,M} \tag{16}$$

B. Safe Policy Learning Setup

In this section, the optimal power management of networked MGs is transformed into a SMAS-PL problem. The purpose of the SMAS-PL is to provide a framework for control agents to collaboratively find control policies to maximize their total accumulated reward while satisfying all problem constraints. To do this, we have provided formulations to ensure that the outcome of the SMAS-PL also corresponds to the solution of optimal power management of networked MGs (1)-(16). To show this, first we provide a description of the components of the SMAS-PL method:

1) Control agents: The problem consists of N autonomous control agents, where each agent is in charge of dispatching the resources within an individual MG. The MGs are *collaborative*, in the sense that they depend on local communication with each other to optimize their behaviors.

2) State set: The state vector for the n'th MG agent at time t is defined as $\boldsymbol{S_{n,t}}$ over the time window $[t, t+T]$, as $\boldsymbol{S_{n,t}} = [\boldsymbol{\hat{I}_{n,t'}^{PV}}, \boldsymbol{\hat{P}_{n,t'}^{D}}]_{t'=t}^{t+T}$, where $\boldsymbol{\hat{I}_{n,t'}^{PV}}$ and $\boldsymbol{\hat{P}_{n,t'}^{D}}$ are the vectors of predicted aggregate internal load power and solar irradiance of the 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 adopted from [20]–[22].

3) Action Set: The control action vector for the n'th agent at time t is denoted as $\boldsymbol{a_{n,t}} \in \mathbb{R}^{D_n}$ and consists of the dispatching decision variables for the n'th MG over the time window $[t, t+T]$, as $\boldsymbol{a_{n,t}} = [P_{n,t'}^{DG}, P_{n,t'}^{Ch}, P_{n,t'}^{Dis}, Q_{n,t'}^{DG}, Q_{n,t'}^{PV}, Q_{n,t'}^{ESS}]_{t'=t}^{t+T}$.

4) Observation Set: The observation variable vector for the agents at time t is denoted as $\boldsymbol{O_t}$, and includes grid's nodal voltages $\boldsymbol{V_t}$ and current injections $\boldsymbol{I_t}$ at that time, $\boldsymbol{O_t} = [\boldsymbol{V_t}, \boldsymbol{I_t}]$. Note that the observations are implicitly determined by the agents' control actions, and thus, cannot be predicted independently of the agents' policies. However, unlike the observation variables, the state variables are independent of the agents' control actions and can be predicted for the whole decision window without the need to consider agents' policies. In the power management problem, nodal sensors or distribution grid's state estimation module will provide the latest values of observations.

5) Control policy: Following the suggestions from [23] and [24], the control policy for the n'th agent, denoted as π_n, is defined as a D_n-dimensional multivariate Gaussian distribution over control actions $\boldsymbol{a_n}$. The policy function determines the probability of the agent's optimal control action after training, as follows:

$$\boldsymbol{a_n} \sim \pi_n(\boldsymbol{a_n}|\boldsymbol{\theta_n}) = \frac{1}{\sqrt{|\Sigma_n|(2\pi)^{D_n}}} e^{-\frac{1}{2}(\boldsymbol{a_n}-\boldsymbol{\mu_n})^\top \Sigma_n^{-1}(\boldsymbol{a_n}-\boldsymbol{\mu_n})} \tag{17}$$

where $\boldsymbol{\mu_n} \in \mathbb{R}^{D_n \times 1}$ is the mean vector and $\Sigma_n \in \mathbb{R}^{D_n \times D_n}$ is the covariance matrix of of multivariate Gaussian distribution for the n'th agent. The Gaussian policy function explicitly determines the expected value and uncertainties of optimal control actions for each agent. Each agent's learning parameter vector, $\boldsymbol{\theta_n}$, consists of two parametric subsets $\boldsymbol{\theta_{\mu_n}}$ and $\boldsymbol{\theta_{\Sigma_n}}$, corresponding to the mean vector and the covariance matrix of the agent's policy function. To do this, two DNNs are used for each MG agent as parametric learning functions to represent control policy components. These DNNs receive the agent's states, $\boldsymbol{S_n}$, as input to fully quantify the sufficient statistics of optimal control policies of MGs, i.e., the mean vector and the covariance matrix of the agent's actions, as follows:

$$\boldsymbol{\mu}_n = DNN(\boldsymbol{S_n}|\boldsymbol{\theta_{\mu_n}}) \tag{18}$$

$$\Sigma_n = DNN(\boldsymbol{S_n}|\boldsymbol{\theta_{\Sigma_n}}) \tag{19}$$

The DNNs are maintained, continuously updated, and deployed in real-time by local control agents of each MG.

6) Reward function: The reward function for the n'th MG is defined as the discounted negative accumulated operational cost of individual MG over the decision window $[t, t+T]$, $R_{n,t'} = -[\sum_{t'=t}^{t+T}(-\lambda_n^R P_{n,t'}^{PCC} + \lambda_{i,n}^F F_{i,n,t'})]$, obtained from the objective functions of the networked MGs power management problem, (1), as follows:

$$J_{R_n}(\pi_n) = E_{\pi_n}[\sum_{t'=t}^{t+T} \gamma^{t'} R_{n,t'}], \forall n \in \{1, ..., N\} \tag{20}$$

where, $\gamma \in [0, 1)$ is a discount factor and $E_{\pi_n}\{\}$ is the expectation operation over the control policy of the agent, to take into account the inherent uncertainty of the states and observation variables.

7) Constraint return: The SMAS-PL consists of a total of M constraints, including M_c^L local and M_c^G global constraints, defined by (3)-(4) and (5)-(16), respectively, and denoted as $C_m(\pi) \leq d_m, m \in \{1, ..., M_c\}$, where $C_m(\pi)$ represents the return value of m'th constraint under the control policy π and d_m is the upper-boundary of the m'th constraint. Note that all constraints in the power management problem have been transformed into this format (equality constraint (15) can be transformed into two inequality constraints.) Constraint satisfaction is encoded into the SMAS-PL using the discounted constraint return values of agents' policies π as:

$$J_{C_m}(\pi) = E_{\pi}[\sum_{t'=t}^{t+T} \gamma^{t'} C_{m,t'}] \leq d_m, \forall m \in \{1, ..., M_c\} \tag{21}$$

where, expectation operation has been leveraged in (21) to handle the state and observation uncertainties.

C. Safe Policy Learning Formulation

Given the definitions of the components of the SMAS-PL (Section III-B), the power management problem of the networked MGs (1)-(16) is transformed into an iterative SMAS-PL problem, where the control policies of the agents are updated at time t, around their latest values, by maximizing a reward function (22), while satisfying constraint return criteria:

$$\boldsymbol{\pi}^{t+1} = \underset{\pi_1,...,\pi_N}{\arg\max} \sum_{n=1}^{N} J_{R_n}(\pi_n) \tag{22}$$

$$s.t. \;\; \boldsymbol{a_n} \sim \pi_n(\boldsymbol{S_n}) \tag{23}$$

$$J_{C_m}(\boldsymbol{\pi}) \leq d_m, \;\forall m \tag{24}$$

$$\Delta(\pi_n, \pi_n^t) \leq \delta, \;\forall n \tag{25}$$

where, $\boldsymbol{\pi} = \{\pi_1, ..., \pi_n\}$ denotes the set of control policies of all agents. In (23), the agent's policy is a function of the state vector, $\boldsymbol{S_n}$. In (24), the expected constraint return value are used to ensure the satisfaction of m'th constraint based on control policies. In (25), $\Delta(\cdot, \cdot)$ is the Kullback Leibler (KL)-divergence function [15] that serves as a distance measure between the previous policy, π_n^t, and the updated policy, π_n^{t+1}, and is constrained by a step size, δ. Note that (25) ensures that consecutive policies are within close distance from each other.

The intractable non-convex PL formulation, (22)-(25), can be solved in principle using a trust region policy optimization (TRPO) method [15]; however, in this paper we apply a further approximation to TRPO to transform the problem into a tractable convex iterative QCLP, which enables learning the PL parameters, $\boldsymbol{\theta} = \{\boldsymbol{\theta_1}, ..., \boldsymbol{\theta_N}\}$, in a more scalable and efficient manner. Our solution leverages the linear approximations of the objective and constraint returns around the latest parameter values $\boldsymbol{\theta^t}$:

$$\boldsymbol{\theta^{t+1}} = \underset{\boldsymbol{\theta_1},...,\boldsymbol{\theta_N}}{\arg\max} \sum_{n=1}^{N} \boldsymbol{g_n}^T(\boldsymbol{\theta_n} - \boldsymbol{\theta_n^t}) \tag{26}$$

$$s.t. \;\; J_{c_m}(\boldsymbol{\theta^t}) + \boldsymbol{b_m}^T(\boldsymbol{\theta} - \boldsymbol{\theta^t}) \leq d_m, \;\forall m \tag{27}$$

$$\frac{1}{2}(\boldsymbol{\theta_n} - \boldsymbol{\theta_n^t})^T H_n(\boldsymbol{\theta_n} - \boldsymbol{\theta_n^t}) \leq \delta, \;\forall n \tag{28}$$

where, $\boldsymbol{g_n} = \nabla_\theta J_R$ and $\boldsymbol{b_m} = \nabla_\theta J_{C_m}$ are the *gradient factors* of the reward and constraint return functions w.r.t. the learning parameters. Constraint (25) is transformed into (28) using the Fisher information matrix (FIM) of the policy functions, π_n, denoted by H_n. The FIM is a positive semi-definite matrix, whose (c, d)'th entry for policy functions with a Gaussian structure is determined as follows [25]:

$$H_n(c,d) = E[\frac{\partial \log \pi_n(\boldsymbol{a_n}|\boldsymbol{\theta_n})}{\partial \boldsymbol{\theta_n}(c)} \frac{\partial \log \pi_n(\boldsymbol{a_n}|\boldsymbol{\theta_n})}{\partial \boldsymbol{\theta_n}(d)}]$$
$$= 2(\frac{\partial \mu_n^H}{\partial \boldsymbol{\theta_n}(c)} \Sigma_n^{-1} \frac{\partial \boldsymbol{\mu_n}}{\partial \boldsymbol{\theta_n}(d)}) + \mathrm{Tr}\left\{\Sigma_n^{-1} \frac{\partial \Sigma_n}{\partial \boldsymbol{\theta_n}(c)} \Sigma_n^{-1} \frac{\partial \Sigma_n}{\partial \boldsymbol{\theta_n}(d)}\right\} \tag{29}$$

Note that (26)-(28) provides a convexified constrained gradient-based method for training the policy functions' parameters of the MG agents; using this QCLP-based strategy the agents do not need to learn an action-value function explicitly. Instead, the power-flow-based gradient factors, $\boldsymbol{g_n}$ and $\boldsymbol{b_m}$, have to be determined for the two sets of learning parameters, $[\boldsymbol{\theta_{\mu_n}}, \boldsymbol{\theta_{\Sigma_n}}]$. This process is outlined in Section III-D.

D. Gradient Factor Determination

To determine gradient factors, the following information are used: (i) the observation variables, $\boldsymbol{O_t}$, including nodal voltage $\boldsymbol{V}$ and current injections $\boldsymbol{I}$; (ii) the latest system states $\boldsymbol{S_{n,t}}$ for each MG agent; (iii) the latest control actions $\boldsymbol{a_n}$ of each MG agent; (iv) the latest learning parameters $\boldsymbol{\theta_n} = [\boldsymbol{\theta_{\mu_n}}, \boldsymbol{\theta_{\Sigma_n}}]$; (v) network parameters, including the nodal admittance matrix, Y. Using information (i)-(v) and chain rule, $\boldsymbol{g_n} = [\boldsymbol{g_{\mu_n}}, \boldsymbol{g_{\Sigma_n}}]$ and $\boldsymbol{b_m} = [\boldsymbol{b_{m,\mu_n}}, \boldsymbol{b_{m,\Sigma_n}}]$ in (26) and (27) can be written as:

$$\boldsymbol{g_{\mu_n}} = \frac{\partial J_{R_n}}{\partial \boldsymbol{a_n}} \frac{\partial \boldsymbol{a_n}}{\partial \pi_n} \frac{\partial \pi_n}{\partial \boldsymbol{\mu_n}} \frac{\partial \boldsymbol{\mu_n}}{\partial \boldsymbol{\theta_{\mu_n}}} \tag{30a}$$

$$\boldsymbol{b_{m,\mu_n}} = \frac{\partial J_{C_m}}{\partial \boldsymbol{a_n}} \frac{\partial \boldsymbol{a_n}}{\partial \pi_n} \frac{\partial \pi_n}{\partial \boldsymbol{\mu_n}} \frac{\partial \boldsymbol{\mu_n}}{\partial \boldsymbol{\theta_{\mu_n}}} \tag{30b}$$

$$\boldsymbol{g_{\Sigma_n}} = \frac{\partial J_{R_n}}{\partial \boldsymbol{a_n}} \frac{\partial \boldsymbol{a_n}}{\partial \pi_n} \frac{\partial \pi_n}{\partial \Sigma_n} \frac{\partial \Sigma_n}{\partial \boldsymbol{\theta_{\Sigma_n}}} \tag{31a}$$

$$\boldsymbol{b_{m,\Sigma_n}} = \frac{\partial J_{C_m}}{\partial \boldsymbol{a_n}} \frac{\partial \boldsymbol{a_n}}{\partial \pi_n} \frac{\partial \pi_n}{\partial \Sigma_n} \frac{\partial \Sigma_n}{\partial \boldsymbol{\theta_{\Sigma_n}}} \tag{31b}$$

where, each gradient factor, $\boldsymbol{g_{\mu_n}}$, $\boldsymbol{b_{m,\mu_n}}$, $\boldsymbol{g_{\Sigma_n}}$, and $\boldsymbol{b_{m,\Sigma_n}}$, consists of four elements. All the elements in (30) and (31) can be obtained as follows:

1) $\partial J_{R_n}/\partial \boldsymbol{a_n}$ and $\partial J_{C_m}/\partial \boldsymbol{a_n}$: The gradients of the expected reward J_{R_n} and the expected constraint return J_{C_m} w.r.t. control actions $\boldsymbol{a_n}$ can be obtained using a proposed four-step process, that leverages the current injection-based AC power flow equations. The details of this process are shown in Appendix A.

2) $\partial \boldsymbol{a_n}/\partial \pi_n$: Using the latest values for parameters $\boldsymbol{\mu_n}$, Σ_n, and actions $\boldsymbol{a_n}$, the gradient of control actions w.r.t. π_n is obtained from (17), as shown in (32):

$$\frac{\partial \boldsymbol{a_n}}{\partial \pi_n} = -\left(\frac{\Sigma_n^{-1}(\boldsymbol{a_n} - \boldsymbol{\mu_n})}{\sqrt{|\Sigma_n|(2\pi)^{D_n}}} e^{-\frac{1}{2}A}\right)^{-1} \tag{32}$$

where, $A = (\boldsymbol{a_n} - \boldsymbol{\mu_n})^\top \Sigma_n^{-1}(\boldsymbol{a_n} - \boldsymbol{\mu_n})$. The detailed derivation of (32) can be found in Appendix B.

3) $\partial \pi_n/\partial \boldsymbol{\mu_n}$ and $\partial \pi_n/\partial \Sigma_n$: using the latest values for parameters $\boldsymbol{\mu_n}$, Σ_n and actions $\boldsymbol{a_n}$, the gradients of control policies, w.r.t. $\boldsymbol{\mu_n}$ and Σ_n are determined using (17), as shown in (33) and (34):

$$\frac{\partial \pi_n}{\partial \boldsymbol{\mu_n}} = \frac{\Sigma_n^{-1}(\boldsymbol{a_n} - \boldsymbol{\mu_n})}{\sqrt{|\Sigma_n|(2\pi)^{D_n}}} e^{-\frac{1}{2}A} \tag{33}$$

$$\frac{\partial \pi_n}{\partial \Sigma_n} = -\frac{1}{2}\frac{(\Sigma_n^{-1} - \Sigma_n^{-1}(\boldsymbol{a_n} - \boldsymbol{\mu_n})(\boldsymbol{a_n} - \boldsymbol{\mu_n})^\top \Sigma_n^{-1})}{\sqrt{|\Sigma_n|(2\pi)^{D_n}}} e^{-\frac{1}{2}A} \tag{34}$$

where, the detailed derivations of (33) and (34) are shown in Appendix B.

4) $\partial \boldsymbol{\mu_n}/\partial \boldsymbol{\theta_{\mu_n}}$ and $\partial \Sigma_n/\partial \boldsymbol{\theta_{\Sigma_n}}$: A *back-propagation process* [26] is performed on the two DNNs within each MG agent's control policy function, (18) and (19), to determine the gradients of DNNs' outputs w.r.t. their parameters. 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 [26]. To enhance the stability of the back-propagation process, a sample batch approach is adopted, where the gradients obtained from several sampled actions are averaged to ensure robustness against outliers.

IV. Multi-Agent Consensus-based Safe Policy Learning

A. *Offline Policy Training*

Using the gradient factors (30) and (31), the QCLP, (26)-(28), is fully specified and can be solved at each policy update iteration for training the agents' PL frameworks. However, we have identified two challenges in this problem: (i) the size of the DNN parameters $\boldsymbol{\theta}$ can be extremely large, which results in high computational costs during training; (ii) the control policy privacy of the MG agents needs to be preserved during training, which implies that the agents might not have access to each other's control policies, cost functions, and local constraints on assets. Centralized solvers can be both time-consuming and lack guarantees for maintaining data ownership boundaries.

In order to address these two challenges, we have developed a *multi-agent consensus-based constrained training algorithm* [18]. Due to its distributed nature this method is both scalable and does not require sharing control policy parameters among agents. Thus, the proposed algorithm is able to efficiently solve the QCLP (26)-(28), while relying only on local inter-MG communication. The purpose of inter-MG interactions is to satisfy global constraints, (3)-(4). To do this, the agents repeatedly estimate and communicate dual variable $\boldsymbol{\lambda_n}$, corresponding to the Lagrangian multiplier of global constraints. Furthermore, a local primal-dual gradient step is included in the algorithm to move the primal and dual parameters towards their global optimum. The proposed distributed algorithm consists of four stages that are performed iteratively, as follows:

Stage I. *Initialize ($k \leftarrow 1$):* Gradient factors $\boldsymbol{g_n}$ and $\boldsymbol{b_m}$ are obtained from Section II-D. The previous values of learning parameters are input to the QCLP, $\boldsymbol{\theta_n^t}(0) \leftarrow \boldsymbol{\theta_n^{t-1}}$. Lagrangian multipliers are initialized as zero for each MG agent.

Stage II. *Weighted averaging operation:* MG agent n receives the Lagrangian multiplier $\boldsymbol{\lambda_{n'}}$, for global constraints (3)-(4), from its neighbouring MG agents $n' \in \{1, ..., N_n\}$ and combines the received estimates using weighted averaging:

$$\bar{\boldsymbol{\lambda}}_{\boldsymbol{n}}(k) = \sum_{n'=1}^{N_n} w_n(n')\boldsymbol{\lambda_{n'}}(k) \tag{35}$$

where, $w_n(n')$ is the weight that MG agent n assigns to the incoming message of the neighbouring MG agent n'. To guarantee convergence to consensus, the weight matrix, composed of the agents' weight parameters is selected as a doubly stochastic matrix [18], i.e., $w_n(n') = \frac{1}{N_n}$. This weight selection strategy implies that the MG agents assign equal importance to the information received from their neighboring agents.

Stage III. *Primal gradient update:* The n'th MG agent updates its primal parameters $\boldsymbol{\theta_n^t}$ employing a gradient descent operation, using the gradients of the agent's reward and the global constraint returns, $m' \in M_c^G$, and step size ρ_1:

$$\bar{\boldsymbol{\theta}}_{\boldsymbol{n}}(k) = \boldsymbol{\theta_n^t}(k) - \rho_1(\boldsymbol{g_n}(\boldsymbol{\theta_n^t}(k)) + \boldsymbol{b_{m'}}(\boldsymbol{\theta_n^t}(k))\bar{\boldsymbol{\lambda}}_{\boldsymbol{n}}(k)) \tag{36}$$

Stage IV. *Projection on local constraints:* The agent projects the local learning parameters to the feasible region defined by the gradients of the local constraints (5)-(16):

$$\boldsymbol{\theta_n^t}(k+1) = \arg\min_{\boldsymbol{\theta}} ||\bar{\boldsymbol{\theta}}_{\boldsymbol{n}}(k) - \boldsymbol{\theta}|| \tag{37}$$

$$s.t. \quad J_{c_m}(\boldsymbol{\theta_n^t}(0)) + \boldsymbol{b_m}^T(\boldsymbol{\theta_n^t}(0) - \boldsymbol{\theta}) \leq d_m, \ \forall m \in M_c^L \tag{38}$$

$$\frac{1}{2}(\boldsymbol{\theta_n^t}(0) - \boldsymbol{\theta})^T H_n(\boldsymbol{\theta_n^t}(0) - \boldsymbol{\theta}) \leq \delta, \ \forall n \tag{39}$$

Stage V. *Dual gradient update:* Each agent's estimations of dual variables $\boldsymbol{\lambda_n}$ for the global constraints, (3) and (4), will be updated using a gradient ascent process over $\bar{\boldsymbol{\lambda}}_{\boldsymbol{n}}$:

$$\boldsymbol{\lambda_n}(k+1) = [(\bar{\boldsymbol{\lambda}}_{\boldsymbol{n}}(k) + \rho_2(\boldsymbol{b_{m'}}\boldsymbol{\theta_n^t}(k+1) - d_{m'})]^+, \forall m' \in M_c^G \tag{40}$$

where, ρ_2 is a penalty factor for global constraints violation, and the operator $[\cdot]^+$ returns the non-negative part of its input.

Stage VI. *Stopping criteria:* Check algorithm convergence using the changes of $\boldsymbol{\theta_n^t}(k)$; stop when the changes in parameters falls below the threshold value $\Delta\theta_n$; otherwise, go back to Stage II.

The overall flowchart of the SMAS-PL training process using the proposed distributed training technique is shown in Algorithm 1. The calculations of Steps 8 and 9 can be found in Appendix A.

Algorithm 1 SMAS-PL Training

Select $t^{max}, T, \delta, k^{max}, w_n(n'), \rho_1, \rho_2, \Delta\theta_n$
Initialize $\boldsymbol{\theta_n^{to}}$
for $t \leftarrow 1$ to t^{max} **do**
  $\boldsymbol{S_n} \leftarrow [\boldsymbol{S_n}(t), ..., \boldsymbol{S_n}(t+T)]$
  $\boldsymbol{\mu_n} \leftarrow$ (18) [Parameter insertion]
  $\boldsymbol{\Sigma_n} \leftarrow$ (19) [Parameter insertion]
  $\boldsymbol{a_n} \sim \pi_n(\boldsymbol{S_n}|\boldsymbol{\theta_n}) \leftarrow$ (17) [Action selection]
  $\partial J_{R_n}/\partial \boldsymbol{a_n} \leftarrow$ (55)-(56)
  $\partial J_{C_m}/\partial \boldsymbol{a_n} \leftarrow$ (59), (57)-(58)
  $\partial \boldsymbol{a_n}/\partial \pi_n \leftarrow$ (32)
  $\partial \pi_n/\partial \boldsymbol{\mu_n} \leftarrow$ (33)
  $\partial \pi_n/\partial \Sigma_n \leftarrow$ (34)
  $\partial \boldsymbol{\mu_n}/\partial \boldsymbol{\theta_{\mu_n}} \leftarrow DNN_{\mu_n}$ [Back-propagation]
  $\partial \Sigma_n/\partial \boldsymbol{\theta_{\Sigma_n}} \leftarrow DNN_{\Sigma_n}$ [Back-propagation]
  $\boldsymbol{g_{\mu_n}}, \boldsymbol{b_{m,\mu_n}} \leftarrow$ (30) [Chain rule]
  $\boldsymbol{g_{\Sigma_n}}, \boldsymbol{b_{m,\Sigma_n}} \leftarrow$ (31) [Chain rule]
  $H_n \leftarrow$ (29) [FIM Construction]
  Initialize $\boldsymbol{\lambda_n}(k_0)$
  for $k \leftarrow 1$ to k^{max} **do**
    $\bar{\boldsymbol{\lambda}}_{\boldsymbol{n}}(k) \leftarrow$ (35) [Averaging operation]
    $\bar{\boldsymbol{\theta}}_{\boldsymbol{n}}(k) \leftarrow$ (36) [Primal gradient update]
    $\boldsymbol{\theta_n^t}(k+1) \leftarrow$ (37)-(39) [Projection on M^L]
    $\boldsymbol{\lambda_n}(k+1) \leftarrow$ (40) [Dual gradient update]
    if $||\boldsymbol{\theta_n^t}(k+1) - \boldsymbol{\theta_n^t}(k)|| \leq \Delta\theta_n$ **then**
      $\boldsymbol{\theta_n^{t+1}} \leftarrow \boldsymbol{\theta_n^t}(k+1)$; Break;
    end if
  end for
  if $||\boldsymbol{\theta_n^{t+1}} - \boldsymbol{\theta_n^t}|| \leq \Delta\theta_n$ **then**
    Output $\boldsymbol{\theta_n^*} \leftarrow \boldsymbol{\theta_n^{t+1}}$; Break;
  end if
end for
Output well-trained parameterized policy $\pi_n(\boldsymbol{\theta_n^*})$

B. Online Action Selection

The trained policy functions are used by the MG agents for online action selection. This process can be simply represented as sampling from the learned Gaussian policy functions (17). First, the agents receive the latest values of the states, including the predicted solar irradiance and aggregate internal load power of MGs. These values are inserted into the trained DNNs (18) and (19) to obtain the mean and covariance matrices of the policy functions. Finally, 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.

C. Backtracking Strategy

Due to convex approximations in the formulations (26)-(28), it is possible for few global constraints to be marginally violated in practice. To ensure feasibility, we can add a backtracking strategy into the proposed solution. This closed-loop backtracking strategy consists of two components, as shown in Fig. 2:

Component 1. *Power flow engine (PFE):* The PFE receives the control actions from MG agents and runs a simple power

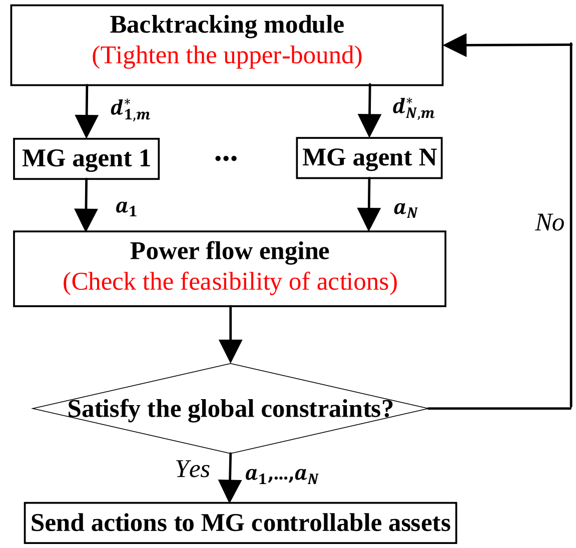


Fig. 2. Flowchart of the backtracking strategy

flow program to obtain the status of all constraints. If no global constraint is violated, the control signals are passed to controllable assets. If some constraints are violated, then the PFE will engage the backtracking process.

Component 2. *Backtracking module:* The backtracking module tightens the upper-bound limit (d_m) (only) for the constraints that have been violated. The parameters of the trained DNNs will be re-updated according to update rules (35)-(40) 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:

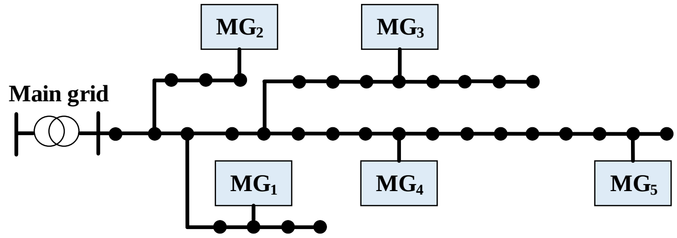
$$d^*_m = \tau d_m \tag{41}$$

Using simulations, we have observed that $\tau = 0.9$ can ensure feasibility for global constraints that have been marginally violated after one-to-two rounds of backtracking in almost all operation scenarios.

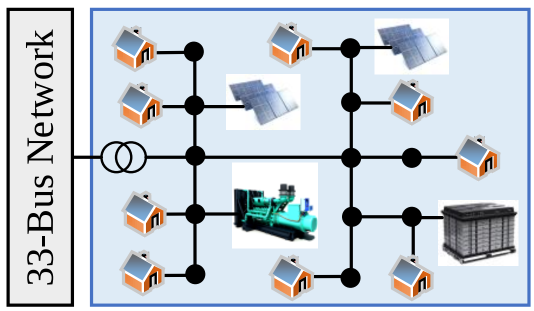
V. Simulation Results

The proposed method is tested on a modified 33-bus distribution network [27], which consists of five MGs as shown in Fig. 3a. Each MG is modeled as a modified IEEE 13-bus network [27] at a low voltage level as shown in Fig. 3b. When calculating the gradient factors, a single-phase AC power flow model is used for the sake of brevity. Note that the proposed SMAS-PL technique can be extended to unbalanced three-phase systems as well. To do this, each agent needs to learn three separate policy functions, each corresponding to controllable assets attached to a phase. The training process for each policy function remains the same.

In this case study, the energy price for the transferred power at the MG PCCs and the fuel price for the local DGs are adopted from [28] and [29], respectively. The input data load demands and PV generations data with 15 minutes time resolution are both obtained from smart meters to provide realistic numerical experiments [30]. All the case studies are simulated using a PC with Intel Core i7-4790 3.6 GHz CPU



(a) 33-bus system for distribution network



(b) 13-bus system for MGs

Fig. 3. Test system under study.

TABLE I
SELECTED DNN HYPERPARAMETERS AND OTHER USER-DEFINED COEFFICIENTS

Description	Notion	Value
Maximum eposide	t^{max}	1000
Length of the decision window	T	4
Discount factor	γ	0.99
Step size for updating θ	δ	1×10^{-3}
Maximum iteration	k^{max}	200
Weight assigned to received information	w_n	0.2
Step size for updating λ	ρ_1	0.01
Penalty factor for constraints violation	ρ_2	0.01
Threshold for parameter updating	$\Delta\theta$	1×10^{-4}
Tightening multiplier	τ	0.9
Number of hidden layer	-	3
Number of neurons per hidden layer	-	10
Size of minibatches	-	128
Activation function of DNNs for μ_n	-	*tansig*
Activation function of DNNs for Σ_n	-	*logsig*

and 16 GB RAM hardware. The simulations are performed in MATLAB [31], OpenDSS [32], and GAMS [33] to obtain the gradient factors, update the learning parameters, solve the distributed training problem, and validate the results. All hyperparameters have been selected through cross-validations, including repeated try-outs and Bayesian optimization in MATLAB environment. The activation functions for the DNNs corresponding to the mean vector and the covariance matrices are hyperbolic tangent-sigmoid (tansig) transfer function and log-sigmoid (logsig) transfer function, respectively. Table I summarizes selected DNN hyperparameters and other user-defined coefficients in simulations.

A. System Operation Outcomes

We have performed out of sample testing in 500 scenarios. The average outcomes are shown in Fig. 4, Fig. 5 and Table II. The aggregate MG demand, aggregate MG generation, and aggregate power transfer through PCCs of MGs over a day are shown in Fig. 4. It can be seen that the main MG demands are supplied by the local generation within MGs due to low DG fuel prices and renewable outputs. While most MGs are exporting power to the upstream distribution grid, MG_4 is importing power to satisfy the heavy local load that cannot be fully supplied internally. In all cases the power balance is maintained within the MGs. The ESS SOCs for each MG are shown in Fig. 5, where can be seen that ESSs charge during off-peak period and discharge during peak time to provide optimal power balancing support for MGs.

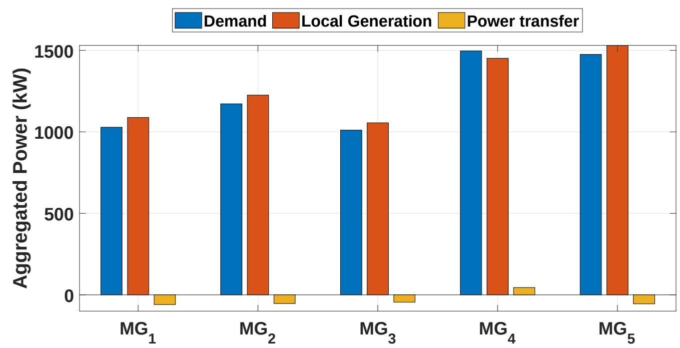


Fig. 4. Aggregated power of local demand, local generation and power transfer for MG_1-MG_5.

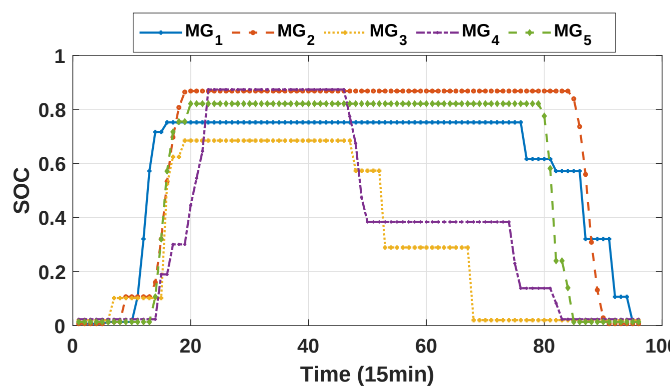


Fig. 5. ESS dispatching results for MG_1-MG_5.

Table II presents a comparison between a conventional commercial centralized optimization solver [33] and the proposed SMAS-PL, including the overall cost of operation over numerous scenarios, average online decision time, and MG privacy maintenance. It can be seen that The SMAS-RL is able to accurately track the underlying optimal solution of the power management problem (with average error of 1.14% over 500 test scenarios). The centralized solver has access to the full systemic information, and thus, has slightly better performance in terms of solution optimality.

In general, the SMAS-PL method has three fundamental advantages over centralized optimization method: 1) Even

TABLE II
COMPARISON BETWEEN CENTRALIZED SOLVER AND SMAS-PL METHOD

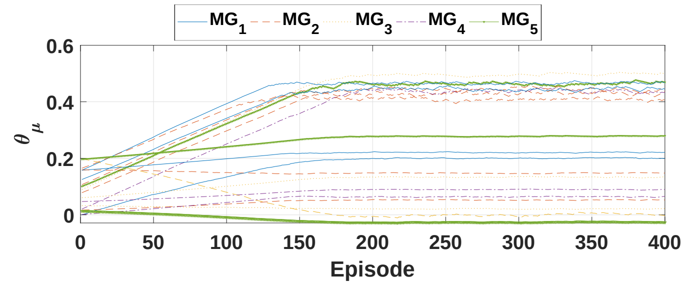
	Cen. solver.	SMAS-PL	Diff.
Total cost ($)	1356.60	1372.11	-1.14%
Average time (second)	145.50	1.40 (per agent)	99.04%
MG privacy maintenance	No	Yes	-

though the offline training process in our method takes a long time (around 35 minutes per agent), the average online decision time for the proposed SMAS-PL is about only 1.4 seconds per agent, which is much shorter than the average time 145.5 seconds for the centralized optimization solver. Thus, the real-time response of the trained policy function is almost 100 times faster than that of the OPF solver. The reason for this is that the OPF solver needs to find the optimal solution of a complex optimization problem in real-time, while our approach simply samples from multivariate Gaussian distributions that embody optimal control policies. Furthermore, we have observed that the computational cost of the centralized OPF solver rises almost quadratically with the size of the system; beyond a certain point the commercial solver is not able to provide solutions in a reasonable time. On the other hand, our SMAS-PL retains an almost constant online decision time, while the cost of offline training increases almost linearly. 2) The proposed PL method takes advantage of a multi-agent (distributed) framework to train the policy function of each MG agent; in practice, this distributed framework can be implemented using parallel computation techniques, which also enhances the scalability of the proposed SMAS-PL method compared to centralized solvers. 3) Due to its distributed nature, the proposed SMAS-PL method maintains the privacy and data ownership boundaries of individual MGs. During the training process, the MG agents do not need to share control policy parameters, policy functions, cost functions, and local asset constraints with each other. The only variables that are shared among MG agents are the Lagrangian multipliers corresponding to global network constraints. These multipliers do not have a physical meaning and thus, do not contain sensitive information.

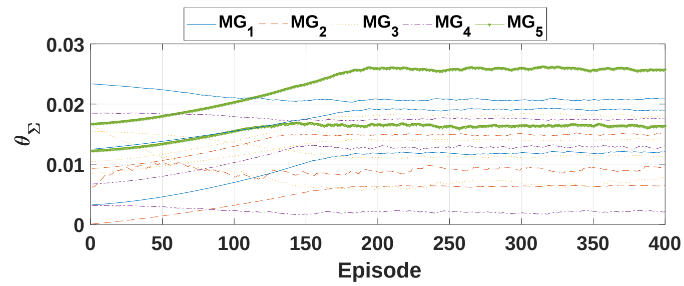
B. Algorithm Performance

Fig. 6a and Fig. 6b show the convergence of a selected group of learning parameters, $\boldsymbol{\theta_\mu}$ and $\boldsymbol{\theta_\Sigma}$ during the training process, for each MG agent. As can be seen, the changes in $\boldsymbol{\theta_\mu}$ are relatively larger than that of $\boldsymbol{\theta_\Sigma}$. This is due to the higher levels of sensitivity of MG agents' objective functions to the mean values of the control actions compared with their variance levels.

To better show the performance of the proposed SMAS-PL method and the importance of considering constraints during the training process, we have compared three cases: (i) DG capacity constraints are in place for all MGs; (ii) no DG capacity constraints in MG_1 and MG_2; (iii) no DG capacity constraints in MG_1-MG_5. As can be seen in Fig. 7, in cases (ii) and (iii), the agents obtain a higher reward compared to case (i) due to the constraint omission; however, this comes



(a) Selected θ_μ's during training process



(b) Selected θ_Σ's during training process

Fig. 6. Convergence of learning parameters θ_μ and θ_Σ for MG_1-MG_5 .

at the expense of decision infeasibility. In case (i), these operational constraints are satisfied, which also leads to a drop in total reward, as expected. This shows that our proposed constrained PL decision model can ensure the feasibility of the control actions w.r.t. the constraints of the power management problem. Note that the proposed SMAS-PL framework is an approximation to the TRPO [15]. While the TRPO has theoretical guarantees for monotonic increase in return, such guarantees do not exist for the approximate formulation QCLP in the proposed SMAS-PL. However, compared to TRPO our solution offers a simpler, more efficient, and tractable alternative, with fewer learning parameters.

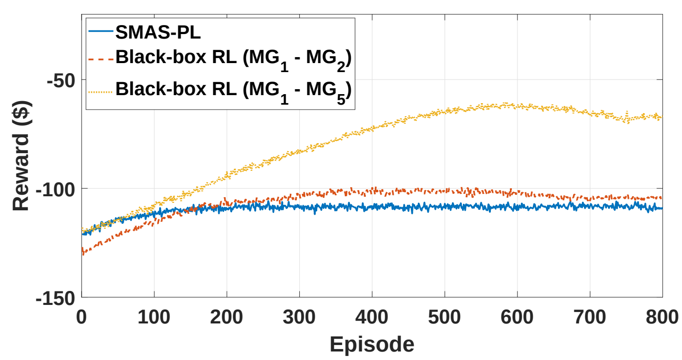


Fig. 7. Comparison of reward w/ and w/o DG capacity constraints for different MGs.

Furthermore, Fig. 8 shows the constraint return values during the training iterations for a 1-hour time window, for the two cases with and without DG capacity constraints in MG_1, where the dark blue and red curves represent averaged constraint returns, and the light blue and red areas represent the variations around the average curves for the SMAS-PL and black-box RL, respectively. During the training process, the black-box RL violates the upper boundary for DG generation limit (i.e., local constraint case study); on the other hand, the SMAS-PL solver satisfies the DG generation capacity constraints, which implies that the local constraints can be safely maintained. Therefore, compared to black-box RL, the proposed SMAS-PL has shown to be able to generate control actions that not only improve the reward function but also satisfy the constraints.

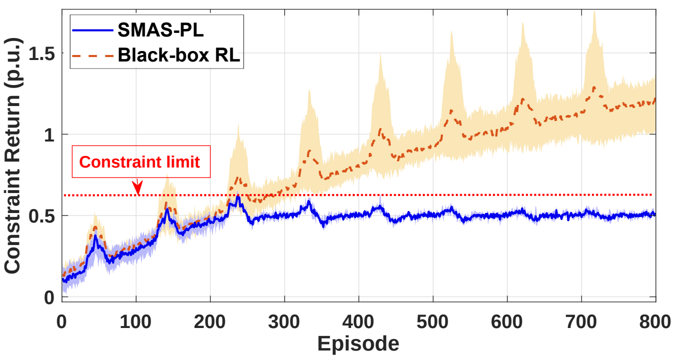


Fig. 8. Comparison of constraint return values w/ and w/o DG capacity constraints in MG_1.

One example of the distributed training convergence process is shown in Fig. 9 for a policy gradient update step. As can be seen, the Lagrangian multipliers λ_n reach zero over iterations of the proposed multi-agent algorithm, which indicates that all the global constraints, including nodal voltage and branch current limits, are satisfied and feasible solutions are obtained. This also means that the bus voltage and line current constraints are not binding for this case.

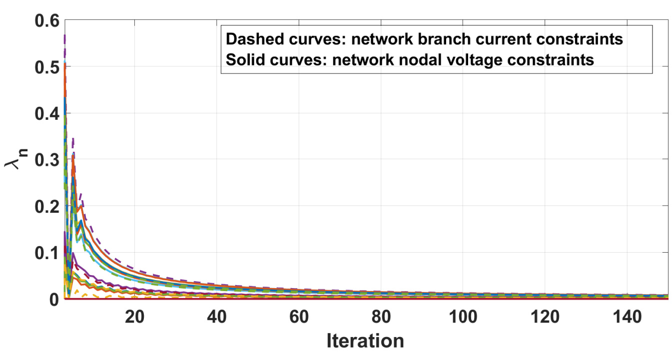


Fig. 9. The performance of the iterative distributed training method in one episode (no binding global constraints).

Another example is given to demonstrate the effectiveness of the SMAS-PL in handling binding global constraints. This case shows a line flow constraint in the grid under the proposed SMAS-PL and a black-box RL baseline; as is observed in Fig. 10, the black-box RL has generated infeasible decisions that violate the constraint, while our approach has prevented the flow to go above its upper bound. Further, as can be seen in Fig. 11, 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; as can be seen, using the proposed SMAS-PL the agents are capable of reaching consensus on the value of the multiplier without having any access to each other's policy functions, which corroborates the performance of our proposed method under incomplete information.

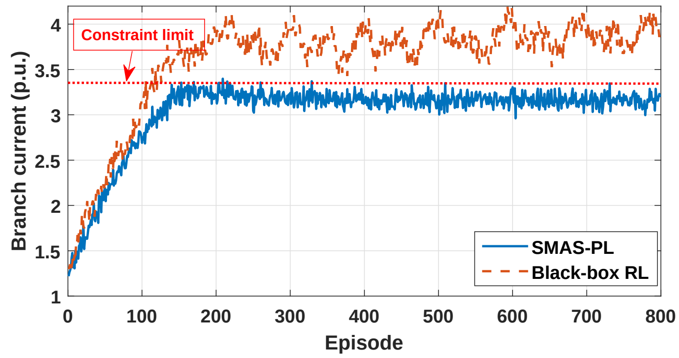


Fig. 10. Selected global branch current constraint return values for MG agents.

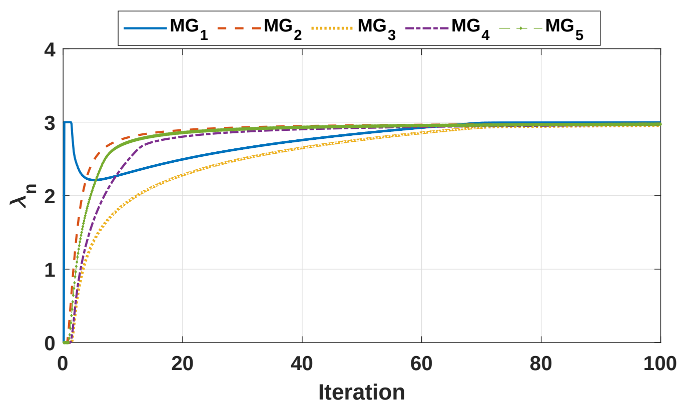


Fig. 11. MG agents' consensus on λ_n for the selected global constraint.

Overall, the current SMAS-PL still has certain drawbacks as follows: (i) To obtain a consensus-based solution, the SMAS-PL needs inter-agent communication infrastructure. (ii) In order to guarantee the global optimal solution, the SMAS-PL needs to first approximate the original problem with a convex surrogate, which despite enhancing the problem tractability, comes at the expense of loss of accuracy and a reduction in performance. (iii) In case of changes in system structure, the SMAS-PL will need an offline re-training phase to adapt to new system conditions. This could take some time, during which the agents will experience a temporary decline in their payoffs. In the future, we will expand the SMAS-PL solution to address these drawbacks.

VI. Conclusion

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. Motivated by these challenges, in this paper, a SMAS-PL method has been proposed for power management of networked MGs. Our proposed method exploits the gradients of the decision problem to learn control policies that achieve both optimality and

TABLE III
PARTIAL DERIVATIONS OF $\boldsymbol{I^{Re}}$ AND $\boldsymbol{I^{Im}}$ W.R.T. $\boldsymbol{a_n} = [P_n^{DG}, P_n^{Ch}, P_n^{Dis}, Q_n^{DG}, Q_n^{PV}, Q_n^{ESS}]$

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

feasibility. Furthermore, 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.

APPENDIX A
CALCULATION OF $\partial J_{R_n}/\partial \boldsymbol{a_n}$ AND $\partial J_{C_m}/\partial \boldsymbol{a_n}$

The major difficulty in determining $\partial J_{R_n}/\partial \boldsymbol{a_n}$ and $\partial J_{C_m}/\partial \boldsymbol{a_n}$ pertains to the agents' reward functions and global constraint returns, (1)-(4), which are only implicitly related to the control actions. Since the reward and all the global constraint returns are functions of the observation variables, $\boldsymbol{V}$ and $\boldsymbol{I}$, the gradients of these variables w.r.t. control actions are obtained and used to quantify $\partial J_{R_n}/\partial \boldsymbol{a_n}$ and $\partial J_{C_m}/\partial \boldsymbol{a_n}$. To do this, a four-step process is proposed that leverages the current injection-based AC power flow equations:

Step 1 - First, the gradients of real and imaginary parts of nodal current injection w.r.t. control actions are derived (denoted as $\partial \boldsymbol{I^{Re}}/\partial \boldsymbol{a_n}$ and $\partial \boldsymbol{I^{Im}}/\partial \boldsymbol{a_n}$, respectively.) To achieve this, the nodal power balance and nodal current injection relationships in the network are employed [34]:

$$I_{i,t'}^{Re} = \frac{p_{i,n,t'} V_{i,t'}^{Re} + q_{i,n,t'} V_{i,t'}^{Im}}{V_{i,t'}^2} \tag{42}$$

$$I_{i,t'}^{Im} = \frac{p_{i,n,t'} V_{i,t'}^{Im} - q_{i,n,t'} V_{i,t'}^{Re}}{V_{i,t'}^2} \tag{43}$$

$$p_{i,n,t'} = P_{i,n,t'}^{D} - P_{i,n,t'}^{DG} - P_{i,n,t'}^{PV} + P_{i,n,t'}^{Ch} - P_{i,n,t'}^{Dis} \tag{44}$$

$$q_{i,n,t'} = Q_{i,n,t'}^{D} - Q_{i,n,t'}^{DG} - Q_{i,n,t'}^{PV} + Q_{i,n,t'}^{ESS} \tag{45}$$

where, I_i^{Re}, I_i^{Im} and V_i^{Re}, V_i^{Im} denote the real and imaginary parts of nodal voltage and current injection at node i. Using these equations, $\partial \boldsymbol{I^{Re}}/\partial \boldsymbol{a_n}$ and $\partial \boldsymbol{I^{Im}}/\partial \boldsymbol{a_n}$ are derived and shown in Table III. Note that the entries of this table can be calculated using the real and imaginary parts of nodal voltages, which in practice are either measured or estimated [34].

Step 2 - Using $\partial \boldsymbol{I^{Re}}/\partial \boldsymbol{a_n}$ and $\partial \boldsymbol{I^{Im}}/\partial \boldsymbol{a_n}$ from Step 1 (Table III), $\partial \boldsymbol{V^{Re}}/\partial \boldsymbol{a}$ and $\partial \boldsymbol{V^{Im}}/\partial \boldsymbol{a}$ are obtained employing the network-wide relationship between nodal voltages and current injections:

$$\begin{bmatrix} \frac{\partial \boldsymbol{V^{Re}}}{\partial \boldsymbol{a_n}} \\ \frac{\partial \boldsymbol{V^{Im}}}{\partial \boldsymbol{a_n}} \end{bmatrix} = \begin{bmatrix} Y^{11} & Y^{12} \\ Y^{21} & Y^{22} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial \boldsymbol{I^{Re}}}{\partial \boldsymbol{a_n}} \\ \frac{\partial \boldsymbol{I^{Im}}}{\partial \boldsymbol{a_n}} \end{bmatrix} \tag{46}$$

where, the modified network bus admittance sub-matrices are determined as follows:

$$Y^{11} = Y^{Re} - Y_D^{(Re,Re)}, Y^{12} = -Y^{Im} - Y_D^{(Re,Im)} \tag{47}$$

$$Y^{21} = Y^{Im} - Y_D^{(Im,Re)}, Y^{22} = Y^{Re} - Y_D^{(Im,Im)} \tag{48}$$

here, Y^{Re} and Y^{Im} are the real and imaginary parts of the original bus admittance matrix. The elements in diagonal matrices $Y_D^{(Re,Re)}$, $Y_D^{(Re,Im)}$, $Y_D^{(Im,Re)}$ and $Y_D^{(Im,Im)}$ are calculated using the following equations [34]:

$$Y_D^{(Re,Re)}(i,i) = \frac{p_{i,n,t'}}{V_{i,t'}^2} - \frac{2V_{i,t'}^{Re}(p_{i,n,t'} V_{i,t'}^{Re} + q_{i,n,t'} V_{i,t'}^{Im})}{V_{i,t'}^4} \tag{49}$$

$$Y_D^{(Re,Im)}(i,i) = \frac{q_{i,n,t'}}{V_{i,t'}^2} - \frac{2V_{i,t'}^{Im}(p_{i,n,t'} V_{i,t'}^{Re} + q_{i,n,t'} V_{i,t'}^{Im})}{V_{i,t'}^4} \tag{50}$$

$$Y_D^{(Im,Re)}(i,i) = -\frac{q_{i,n,t'}}{V_{i,t'}^2} - \frac{2V_{i,t'}^{Re}(p_{i,n,t'} V_{i,t'}^{Im} - q_{i,n,t'} V_{i,t'}^{Re})}{V_{i,t'}^4} \tag{51}$$

$$Y_D^{(Im,Im)}(i,i) = \frac{p_{i,n,t'}}{V_{i,t'}^2} - \frac{2V_{i,t'}^{Im}(p_{i,n,t'} V_{i,t'}^{Im} - q_{i,n,t'} V_{i,t'}^{Re})}{V_{i,t'}^4} \tag{52}$$

Step 3 - Noting that the current flow constraint returns and the rewards are also functions of branch current flows, the gradients of branch current flows are required to obtain $\partial J_{R_n}/\partial \boldsymbol{a_n}$ and $\partial J_{C_m}/\partial \boldsymbol{a_n}$. Using the branch current flow equations, these gradients are determined as a function of the derivatives of nodal voltages and current injections, as follows:

$$\frac{\partial I_{ij,t'}^{Re}}{\partial \boldsymbol{a_{n,t'}}} = y_{ij}^{Im}\left(\frac{\partial V_{i,t'}^{Im}}{\partial \boldsymbol{a_{n,t'}}} - \frac{\partial V_{j,t'}^{Im}}{\partial \boldsymbol{a_{n,t'}}}\right) - y_{ij}^{Re}\left(\frac{\partial V_{i,t'}^{Re}}{\partial \boldsymbol{a_{n,t'}}} - \frac{\partial V_{j,t'}^{Re}}{\partial \boldsymbol{a_{n,t'}}}\right) \tag{53}$$

$$\frac{\partial I_{ij,t'}^{Im}}{\partial \boldsymbol{a_{n,t'}}} = y_{ij}^{Im}\left(\frac{\partial V_{i,t'}^{Re}}{\partial \boldsymbol{a_{n,t'}}} - \frac{\partial V_{j,t'}^{Re}}{\partial \boldsymbol{a_{n,t'}}}\right) + y_{ij}^{Re}\left(\frac{\partial V_{i,t'}^{Im}}{\partial \boldsymbol{a_{n,t'}}} - \frac{\partial V_{j,t'}^{Im}}{\partial \boldsymbol{a_{n,t'}}}\right) \tag{54}$$

where, I_{ij}^{Re} and I_{ij}^{Im} are the real and imaginary parts of branch currents, y_{ij}^{Re} and y_{ij}^{Im} are the real and imaginary parts of branch admittance.

Step 4 - Finally, using the derivatives obtained from Steps 1, 2, and 3, $\partial J_{R_n}/\partial \boldsymbol{a_n}$ and $\partial J_{C_m}/\partial \boldsymbol{a_n}$ are determined through straightforward algebraic manipulations. As an example, the gradient of reward function w.r.t. $P_{n,t'}^{DG}$ is calculated as:

$$\frac{\partial J_{R_n}}{\partial P_{n,t'}^{DG}} = \sum_{t'=t}^{t+T} \left(\lambda_{i,n}^F (2a_f + b_f) - \lambda_n^R \frac{\partial P_{n,t'}^{PCC}}{\partial P_{n,t'}^{DG}}\right) \tag{55}$$

where, $\partial P_{n,t'}^{PCC}/\partial P_{n,t'}^{DG}$ is obtained using the outcomes of Steps 2 and 3, as follows:

$$\frac{\partial P_{n,t'}^{PCC}}{\partial P_{n,t'}^{DG}} = \frac{\partial V_{i,t'}^{Re}}{\partial P_{n,t'}^{DG}} I_{ij,t'}^{Re} + V_{i,t'}^{Re} \frac{\partial I_{ij,t'}^{Re}}{\partial P_{n,t'}^{DG}} + \frac{\partial V_{i,t'}^{Im}}{\partial P_{n,t'}^{DG}} I_{ij,t'}^{Im} + V_{i,t'}^{Im} \frac{\partial I_{ij,t'}^{Im}}{\partial P_{n,t'}^{DG}} \tag{56}$$

Furthermore, $\partial J_{C_m}/\partial \boldsymbol{a_n}$ for the global constraints (3) and (4) can be calculated using the outcomes of Steps 2 and 3:

$$\frac{\partial V_{i,t'}}{\partial \boldsymbol{a_{n,t'}}} = \frac{V_{i,t'}^{Re}}{V_{i,t'}} \frac{\partial V_{i,t'}^{Re}}{\partial \boldsymbol{a_{n,t'}}} + \frac{V_{i,t'}^{Im}}{V_{i,t'}} \frac{\partial V_{i,t'}^{Im}}{\partial \boldsymbol{a_{n,t'}}} \tag{57}$$

$$\frac{\partial I_{ij,t'}}{\partial \boldsymbol{a_{n,t'}}} = \frac{I^{Re}_{ij,t'}}{I_{ij,t'}} \frac{\partial I^{Re}_{ij,t'}}{\partial \boldsymbol{a_{n,t'}}} + \frac{I^{Im}_{ij,t'}}{I_{ij,t'}} \frac{\partial I^{Im}_{ij,t'}}{\partial \boldsymbol{a_{n,t'}}} \tag{58}$$

As can be seen in (5)-(16), the local constraint returns are trivial functions of the control actions. For example, the constraint return value for (5) is $J_{C_{5,t'}} = P^{DG}_{n,t'}$ which induces a simple gradient element w.r.t. control action $P^{DG}_{n,t'}$:

$$\frac{\partial J_{C_{5,t'}}}{\partial P^{DG}_{n,t'}} = 1 \tag{59}$$

The gradients of constraint returns w.r.t. control actions for the remaining local constraints, (6)-(16), can be obtained in a similar way.

Appendix B
Derivation of $\partial \boldsymbol{a_n}/\partial \pi_n$, $\partial \pi_n/\partial \boldsymbol{\mu_n}$ and $\partial \pi_n/\partial \Sigma_n$

$\partial \boldsymbol{a_n}/\partial \pi_n$, $\partial \pi_n/\partial \boldsymbol{\mu_n}$ and $\partial \pi_n/\partial \Sigma_n$ are obtained using the probability density function of (D-dimensional) multivariate Gaussian distribution [35], which has the following general formulation:

$$f(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^D}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{60}$$

where $\boldsymbol{x}$ is a random vector. To derive the gradients, first, the log-likelihood function of this multivariate Gaussian distribution (60) is obtained as follows:

$$L = \ln(f) = \ln \frac{1}{\sqrt{|\Sigma|(2\pi)^D}} - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \tag{61}$$

The derivative of L w.r.t. mean vector $\boldsymbol{\mu}$ and covariance matrix Σ can be written as follows:

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{\mu}} &= -\frac{1}{2} \frac{\partial (\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} \\ &= -\frac{1}{2}(-2\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})) = \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\end{aligned} \tag{62}$$

$$\begin{aligned}\frac{\partial L}{\partial \Sigma} &= -\frac{1}{2}\Big(\frac{\partial \ln(|\Sigma|)}{\partial \Sigma} + \frac{\partial (\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{\partial \Sigma}\Big) \\ &= -\frac{1}{2}(\Sigma^{-1} - \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1})\end{aligned} \tag{63}$$

Thus, using (62) and (63), the derivatives of the function f w.r.t. $\boldsymbol{\mu}$ and Σ can be shown in (64) and (65), respectively:

$$\frac{\partial f}{\partial \boldsymbol{\mu}} = \frac{\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{\sqrt{|\Sigma|(2\pi)^D}} e^{-\frac{1}{2}A} \tag{64}$$

$$\frac{\partial f}{\partial \Sigma} = -\frac{1}{2} \frac{(\Sigma^{-1} - \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1})}{\sqrt{|\Sigma|(2\pi)^D}} e^{-\frac{1}{2}A} \tag{65}$$

where $A = (\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$. Similarly, the derivative of the function f w.r.t. $\boldsymbol{x}$ is shown as follows:

$$\frac{\partial f}{\partial \boldsymbol{x}} = -\frac{\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{\sqrt{|\Sigma|(2\pi)^D}} e^{-\frac{1}{2}A} \tag{66}$$

References

[1] Z. Li, M. Shahidehpour, F. Aminifar, A. Alabdulwahab, and Y. Al-Turki, "Networked microgrids for enhancing the power system resilience," *Proc. of the IEEE*, vol. 105, no. 7, pp. 1289–1310, July 2017.

[2] M. N. Alam, S. Chakrabarti, and A. Ghosh, "Networked microgrids: state-of-the-art and future prospectives," *IEEE Trans. Ind. Informat.*, vol. 15, pp. 1238–1250, Mar. 2019.

[3] M. Ban, M. Shahidehpour, J. Yu, and Z. Li, "A cyber-physical energy management system for optimal sizing and operation of networked nanogrids with battery swapping stations," *IEEE Trans. Sustain. Energy*, vol. 10, no. 1, pp. 491–502, Jan. 2019.

[4] 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.

[5] L. Xiao, N. B. Mandayam, and H. V. Poor, "Prospect theoretic analysis of energy exchange among microgrids," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 63–72, Jan. 2015.

[6] 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, Mar. 2016.

[7] 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.

[8] B. Zhao, X. Wang, D. Lin, M. M. Calvin, J. C. Morgan, R. Qin, and C. Wang, "Energy management of multiple microgrids based on a system of systems architecture," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 6410–6421, Nov. 2018.

[9] J. Duan, Z. Yi, D. Shi, C. Lin, X. Lu, and Z. Wang, "Reinforcement-learning-based optimal control for hybrid energy storage systems in hybrid ac/dc microgrids," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 5355–5364, Sept. 2019.

[10] P. Zeng, H. Li, H. He, and S. Li, "Dynamic energy management of a microgrid using approximate dynamic programming and deep recurrent neural network learning," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4435–4445, 2019.

[11] B. Kim, Y. Zhang, M. van der Schaar, and J. Lee, "Dynamic pricing and energy consumption scheduling with reinforcement learning," *IEEE Trans. Smart Grid*, vol. 7, no. 5, pp. 2187–2198, Sept. 2016.

[12] T. Yu, H. Z. Wang, B. Zhou, K. W. Chan, and J. Tang, "Multi-agent correlated equilibrium Q learning for coordinated smart generation control of interconnected power grids," *IEEE Trans. Power Syst.*, vol. 30, no. 4, pp. 1669–1679, July 2015.

[13] X. Lu, X. Xiao, L. Xiao, C. Dai, M. Peng, and H. V. Poor, "Reinforcement learning-based microgrid energy trading with a reduced power plant schedule," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10 728–10 737, Dec. 2019.

[14] 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*, Early access.

[15] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," *in Proc. of 34th International Conference on Machine Learning*, 2017.

[16] M. Wen and T. Ufuk, "Constrained cross-entropy method for safe reinforcement learning," *in Proc. of 32nd Conference on NeurIPS*, 2018.

[17] Y. Chow, O. Nachum, M. Ghavamzadeh, and E.Duen, "A lyapunov-based approach to safe reinforcement learning," *in Proc. of 32nd Conference on NeurIPS*, 2018.

[18] T. H. Chang, A. Nedic, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Trans. Automatic Control*, vol. 59, no. 6, pp. 1524–1538, June 2014.

[19] A. Pourmousavi, H. Nehrir, and R. K. Sharma, "Multi-timescale power management for islanded microgrids including storage and demand response," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1185–1195, May 2015.

[20] G. Mokryani, "Active distribution networks planning with integration of demand response," *Solar Energy*, vol. 122, pp. 1362–1370, Dec. 2015.

[21] M. Wahbah, T. H. M. El-Fouly, B. Zahawi, and S. Feng, "Hybrid Beta-KDE model for solar irradiance probability density estimation," *IEEE Trans. Sustain. Energy*, Early Access.

[22] H. Wu, M. Shahidehpour, Z. Li, and W. Tian, "Chance-constrained day-ahead scheduling in stochastic power system operation," *IEEE Trans. Power Syst.*, vol. 29, no. 4, pp. 1583–1591, July 2014.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. London, England: The MIT Press, 2017.

[24] H. Li, Z. Wan, and H. He, “Constrained EV charging scheduling based on safe deep reinforcement learning,” *IEEE Trans. Smart Grid*, Early Access.

[25] O. Besson and Y. I. Abramovich, “On the Fisher information matrix for multivariate elliptically contoured distributions,” *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1130–1133, Nov. 2013.

[26] J. Li, J. Cheng, J. Shi, and F. Huang, “Brief introduction of back propagation neural network algorithm and its improvement,” *Advances in computer science and information engineering*, pp. 553–558, 2012.

[27] W. Kersting, “Radial distribution test feeder,” *in Proc. of 2001 IEEE PES Winter Meeting*, vol. 2, pp. 908–912, 2001.

[28] Wholesale electricity and natural gas market data. [Online]. Available: https://www.eia.gov/electricity/wholesale/#history

[29] Gasoline and diesel fuel update. [Online]. Available: https://www.eia.gov/petroleum/gasdiesel/

[30] Iowa distribution test systems. [Online]. Available: http://wzy.ece.iastate.edu/Testsystem.html

[31] Matlab. [Online]. Available: https://www.mathworks.com/help/matlab/index.html

[32] Opendss basic tutorial. [Online]. Available: https://sourceforge.net/p/electricdss/discussion/beginners.html

[33] An introduction to GAMS. [Online]. Available: https://www.gams.com/products/introduction.html

[34] R. A. Jabr and I. Dafi, “Sensitivity-based discrete coordinate-descent for volt/var control in distribution networks,” *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4670–4678, Nov. 2016.

[35] P. K. Brandt and M. S. Pedersen, *The Matrix Cookbook*. Technical University of Denmark, 2008.