TRUST-REGION BASED POLICY OPTIMIZATION FOR EFFICIENT REINFORCEMENT LEARNING

Hepfeng Li
University of Rhode Island, hepengli@uri.edu

Follow this and additional works at: https://digitalcommons.uri.edu/oa_diss

Recommended Citation
https://digitalcommons.uri.edu/oa_diss/1559

This Dissertation is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Dissertations by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons-group@uri.edu.
TRUST-REGION BASED POLICY OPTIMIZATION FOR EFFICIENT
REINFORCEMENT LEARNING

BY

HEPENG LI

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND
2023
DOCTOR OF PHILOSOPHY DISSERTATION
OF
HEPENG LI

APPROVED:

Dissertation Committee:

Major Professor   Haibo He
Yan (Lindsay) Sun
Lisa DiPippo
Peter Swaszek
Brenton DeBoef
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2023
ABSTRACT

Reinforcement Learning (RL) has achieved remarkable success in solving complicated control and sequential decision-making tasks in recent years, such as board games, video games, robot control, dynamic energy management, and autonomous driving. In these tasks, RL has demonstrated unprecedented adaptability and self-learning capability in unknown and stochastic environments. However, existing RL algorithms still have significant disadvantages and shortcomings compared to human intelligence, such as learning stability, sample efficiency, safety, scalability, and collaboration with other agents. These drawbacks impede the widespread application of RL in real-life domains.

To address these challenges, this dissertation aims to propose novel theories and algorithms for more effective and efficient RL and to develop their applications in real-world domains. This work consists of three major parts. First, the dissertation establishes a novel theoretical result for trust-region-based RL methods by analyzing the lower bound of policy performance. A closed-form policy update rule is then derived based on the theoretical result, providing a monotonic improvement guarantee. Second, the dissertation develops an off-policy RL algorithm for continuous control problems, inspired by the closed-form update rule. This algorithm enables sample-efficient learning of deep neural network policies. Additionally, trust region policy optimization is extended to cooperative multi-agent systems through consensus optimization, resulting in a distributed MARL algorithm. Third, the dissertation develops the real-world applications of a trust-region-based safe RL method and the MARL method in smart grids. Numerous experiments are conducted to verify the effectiveness of these methods using various benchmark environments, including robotics, strategic games, and power systems.
Throughout my study at the University of Rhode Island (URI), numerous exceptional individuals have provided me with support and inspiration. I am profoundly grateful to all of them for contributing to this journey and facilitating the completion of this dissertation.

First and foremost, I would like to express my gratitude to my advisor, Dr. Haibo He. He has persistently guided and encouraged me to maintain professionalism and persevere through difficult times. Without his unwavering assistance, this work would not have been accomplished.

I would also like to extend my sincere appreciation to Dr. Yan (Lindsay) Sun, Dr. Lisa DiPippo, Dr. Peter Swaszek, and Dr. Marco Alvarez for their invaluable time and suggestions during my research. Their expertise and experience have greatly benefited me.

Additionally, I am grateful to the exceptional colleagues in the Computational Intelligence and Self-Adaptive System (CISA) Lab, including Zhen, Xiangnan, Jun, Bo, Yufei, Zhiqiang, Lusi, He, Zhenhua, William, Nick, and Alexis, for their support throughout various aspects of this work.

I would like to express my deep and sincere appreciation to my parents and sister for their unwavering support. Lastly, I am forever thankful to my wife, Ying, and my son, Shaoyu, for their love during my pursuit of a doctoral degree.

During my Ph.D. study, my research have been partially supported by the National Science Foundation (NSF) under ECCS 1917275 and Office of Naval Research (ONR) under N00014-18-1-2396. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the funding agencies. I gratefully acknowledge the support from NSF and ONR for my research.
# TABLE OF CONTENTS

ABSTRACT ......................................................... ii

ACKNOWLEDGMENTS ................................. iii

TABLE OF CONTENTS ............................... iv

LIST OF FIGURES .................................. viii

LIST OF TABLES ........................................ x

CHAPTER

1 Introduction ................................. 1

1.1 Research Motivations and Challenges ............. 1

1.2 The RL Problem and Algorithms ............... 3

1.2.1 Markov Decision Process .................. 4

1.2.2 RL Algorithms ......................... 5

1.2.3 Dissertation Outline .................. 11

2 Monotonic Improvement Guarantee for Trust-Region Policy Optimization .......... 14

2.1 Overview ........................................ 14

2.2 Related Work .................................. 16

2.3 Problem Formulation ..................... 18

2.3.1 Markov Decision Process .............. 18

2.3.2 Partially Observable Markov Game ......... 19

2.3.3 Trust Region Methods .................. 20

2.4 Theoretical Results ...................... 21
2.4.1 Monotonic Improvement Guarantee for Single-Agent RL 21
2.4.2 Monotonic Improvement Guarantee for Multi-Agent RL . 41
2.5 Connections with Prior Work . . . . . . . . . . . . . . . . . . 42
2.6 Limitations and Discussions . . . . . . . . . . . . . . . . . . . 47
2.7 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

3 Off-Policy Trust Region Policy Optimization Algorithm . . 49
  3.1 Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
  3.2 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
  3.3 An Off-Policy Trust-Region RL Algorithm . . . . . . . . . . 52
    3.3.1 Closed-Form Solution to Trust-Region Policy Optimization 52
  3.4 Case Studies . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
    3.4.1 MuJoCo Environment for Robot Control . . . . . . . . 56
    3.4.2 Comparison With Prior Trust-Region methods . . . . . 57
    3.4.3 Comparison With Other Off-Policy methods . . . . . . 59
  3.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60

4 Multi-Agent Trust Region Policy Optimization Algorithm . . 62
  4.1 Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
  4.2 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
  4.3 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
    4.3.1 Partially Observable Markov Game . . . . . . . . . . . . . 67
    4.3.2 The TRPO Algorithm . . . . . . . . . . . . . . . . . . . . . 68
    4.3.3 Alternating Direction Method of Multipliers . . . . . . . 69
  4.4 Multi-Agent Trust Region Policy Optimization . . . . . . . 70
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.1</td>
<td>Trust Region Optimization for Multiple Agents</td>
<td>70</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Distributed Consensus with Asynchronous ADMM</td>
<td>76</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Sequential Convexification for Convergence</td>
<td>79</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Information Required by Agents</td>
<td>80</td>
</tr>
<tr>
<td>4.5</td>
<td>Practical Implementation</td>
<td>81</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Sample-Based Approximation of the Consensus Constraint</td>
<td>81</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Agreement on Logarithmic Ratio</td>
<td>82</td>
</tr>
<tr>
<td>4.6</td>
<td>Empirical Study</td>
<td>88</td>
</tr>
<tr>
<td>4.6.1</td>
<td>The Baseline Methods</td>
<td>89</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Multi-Agent Particle Environment</td>
<td>90</td>
</tr>
<tr>
<td>4.6.3</td>
<td>StarCraft Multi-Agent Challenge</td>
<td>94</td>
</tr>
<tr>
<td>4.7</td>
<td>Summary</td>
<td>99</td>
</tr>
<tr>
<td>5</td>
<td>Applications of Trust-Region RL Algorithms in Smart Grid</td>
<td>100</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview</td>
<td>100</td>
</tr>
<tr>
<td>5.2</td>
<td>Optimal Operation of Distribution Grid with Safe RL</td>
<td>101</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Introduction</td>
<td>101</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Problem Formulation</td>
<td>104</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Safe Deep Reinforcement Learning Solution</td>
<td>109</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Case studies</td>
<td>115</td>
</tr>
<tr>
<td>5.3</td>
<td>Distributed Volt-Var Control with MATRPO</td>
<td>124</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Introduction</td>
<td>124</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Problem Formulation</td>
<td>127</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Multi-Agent Reinforcement Learning Solution</td>
<td>129</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>58</td>
</tr>
<tr>
<td>5</td>
<td>59</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
</tr>
<tr>
<td>7</td>
<td>90</td>
</tr>
<tr>
<td>8</td>
<td>93</td>
</tr>
<tr>
<td>9</td>
<td>93</td>
</tr>
<tr>
<td>10</td>
<td>95</td>
</tr>
<tr>
<td>11</td>
<td>96</td>
</tr>
<tr>
<td>12</td>
<td>97</td>
</tr>
<tr>
<td>13</td>
<td>97</td>
</tr>
<tr>
<td>14</td>
<td>117</td>
</tr>
<tr>
<td>15</td>
<td>118</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>16</td>
<td>Test performance of different algorithms on the modified IEEE 34 system: a) distribution of constraint violation, and b) cumulative cost.</td>
</tr>
<tr>
<td>17</td>
<td>A modified IEEE-123 node test feeder system.</td>
</tr>
<tr>
<td>18</td>
<td>Learning curves of different algorithms on the modified IEEE-123 node system: a) constraint value, and b) return.</td>
</tr>
<tr>
<td>19</td>
<td>Test performance of different algorithms on the modified IEEE 123 system: a) distribution of constraint violation, and b) cumulative cost.</td>
</tr>
<tr>
<td>20</td>
<td>A modified IEEE 34 Bus Test Feeder.</td>
</tr>
<tr>
<td>21</td>
<td>Comparison of episode rewards using different methods during the training process.</td>
</tr>
<tr>
<td>22</td>
<td>Distributions of average bus voltage violation on 366 test days using different methods.</td>
</tr>
<tr>
<td>23</td>
<td>Cumulative system line losses on 366 test days using different methods.</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>Parameters of the proposed off-policy trust-region deep reinforcement learning algorithm.</td>
</tr>
<tr>
<td>2</td>
<td>Parameters of MATRPO for the MPE tasks.</td>
</tr>
<tr>
<td>3</td>
<td>Parameter settings of the proposed method.</td>
</tr>
<tr>
<td>4</td>
<td>Hyper-parameters of the MATRPO algorithm for the Distributed VVC Problem.</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

1.1 Research Motivations and Challenges

Reinforcement learning (RL) is a crucial learning paradigm present in many intelligent creatures, including humans. During infancy, we interacted with the world around us through sensorimotor connections. We played with toys, deliberately dropped them to observe the outcome, cried to our parents to get what we wanted, and broke things to satisfy our curiosity. Through trial and error, we learned from the information perceived from the environment, predicted the consequences of our actions, and reinforced the actions that helped us achieve our goals. The algorithms inspired by this concept of goal-directed learning from interactions are referred to as reinforcement learning, or RL [1].

RL focuses on how an intelligent agent should take actions in an unfamiliar and unpredictable environment to optimize a sequence of rewards given by the environment. The agent explores the environment through trial and error, and learns from the feedback received in the form of rewards. Due to its capacity for self-learning and adaptability, RL is arguably regarded as one of the most promising approaches to general-purpose artificial intelligence (AI), as evidenced by the recent breakthrough of ChatGPT, which uses RL to learn from human feedback [2]. However, conventional RL algorithms face challenges in complex real-world environments due to the curse of dimensionality. As the number of possible states and actions increases exponentially with the number of dimensions, the environment becomes intractable for an RL agent to explore and learn effectively.

To overcome the challenges posed by high-dimensional state and action spaces, deep reinforcement learning (DRL) has emerged in recent years as a powerful solution that combines RL with deep learning techniques. By leveraging the represen-
tation capabilities of deep neural networks, DRL algorithms can learn high-quality and sophisticated behaviors in an end-to-end manner, which has been successful in solving various complex decision-making applications, such as games [3, 4] and robot control [5]. DRL has also achieved remarkable success in other domains, such as improving the efficiency of power systems, optimizing the allocation of resources in data centers [6], predicting protein structures in their three-dimensional shape [7], optimizing large language models for online chat conversation [2].

However, the widespread adoption of RL in real-life domains still faces fundamental challenges:

(1) Instability. The instability arises due to the interaction between the reinforcement learning algorithm and the deep neural network. One common issue is the tendency of DRL algorithms to suffer from the problem of divergence, where the gradients become unstable and lead to the policy or value function oscillating or diverging. This can result in poor performance or even complete failure of the learning process.

(2) Sample inefficiency. DRL algorithms typically require a large number of interactions with the environment to learn an optimal policy or value function, especially when using high-capacity function approximators such as deep neural networks. This can be a problem when applying RL to real-world domains where data collection is expensive and time-consuming.

(3) Multi-agent RL. In real-world applications, it is common to have multiple agents that may have different behaviors, observations, and interests. The non-stationarity issue in multi-agent environments arises because the behavior of each agent affects the environment, which in turn affects the behavior of the other agents. As a result, the agents may have difficulty in reaching convergence because they are constantly adjusting their policies to adapt to the changing behavior of
other agents.

(4) Safety. Safety is another important challenge in DRL, particularly in real-world applications such as autonomous driving, robotics, and power systems. To learn a good policy, DRL algorithms typically allow agents to explore the environment and learn a good policy by freely selecting any behavior that leads to performance improvement. However, this freedom can potentially lead to unsafe actions that could put humans or other agents at risk.

1.2 The RL Problem and Algorithms

RL stands apart from both supervised and unsupervised learning approaches. RL deals with the problem of learning how to make decisions in an uncertain environment. The goal is to design an agent that can interact with an environment, learn from the feedback it receives, and make optimal decisions to achieve a specific objective over time. Figure 1 shows the general architecture of RL. At each time step, the agent takes an action based on its current state and receives a reward from the environment. The reward reflects how well the agent is doing in achieving its objective. The agent’s goal is to learn a policy, which is a mapping from states to actions, that maximizes the expected cumulative reward over time.

![Figure 1. Architecture of RL [1].](attachment:figure1.png)
1.2.1 Markov Decision Process

The RL problem can be formalized as a Markov decision process (MDP) [8], which consists of a set of states, actions, rewards, and transition probabilities. At each time step, the agent observes the current state of the environment and selects an action based on its policy. The environment then transitions to a new state and provides the agent with a reward signal that reflects the quality of the action taken. The objective of RL is to learn a policy that maximizes the cumulative reward over a sequence of actions. RL algorithms typically use trial-and-error to learn a good policy, by iteratively updating the policy based on the observed rewards and the states visited.

An MDP is defined by a tuple \((S, A, p, r, \rho_0, \gamma)\), where \(S\) is the state space, \(A\) is the action space, \(P : S \times A \times S \to \mathbb{R}_{\geq 0} \) (\(\to [0, 1]\) for discrete states) is the transition probability density (mass) function, \(r : S \times A \to [r_{\text{min}}, r_{\text{max}}]\) is the reward function, \(\rho_0 : S \to \mathbb{R}_{\geq 0} \) (\(\to [0, 1]\) for discrete states) is the probability density (mass) function of the initial state \(s_0\), \(\gamma \in [0, 1)\) is the discount factor.

Denote a stochastic policy \(\pi : S \times A \to \mathbb{R}_{\geq 0} \) (\(\to [0, 1]\) for discrete actions) by \(\pi(a|s)\), which represents the probability density (mass) function of the action \(a\) given the state \(s\). The goal is to find an optimal policy that maximizes the expected cumulative discounted rewards:

\[
J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \tag{1}
\]

where \(\tau\) denotes the trajectory \(\tau := (s_0, a_0, s_1, \ldots)\), and \(\tau \sim \pi\) indicates that the distribution over the trajectory depends on \(\pi : s_0 \sim \rho_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)\).

For a trajectory \(\tau_t := (s_t, a_t, s_{t+1}, \ldots)\) starting from any time step \(t\), define its
return as the discounted sum of the rewards

\[ R(\tau_t) = \sum_{l=t}^{\infty} \gamma^{l-t} r(s_l, a_l). \] (2)

The state-value function or value function of a given state \( s \) is defined by

\[ V_\pi(s) = \mathbb{E}_{\tau_t \sim \pi}[R(\tau_t) | s_t = s], \] (3)

the action-value function or Q-function of a given state-action pair \((s, a)\) is defined by

\[ Q_\pi(s, a) = \mathbb{E}_{\tau_t \sim \pi}[R(\tau_t) | s_t = s, a_t = a], \] (4)

and the advantage function is defined by

\[ A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s). \] (5)

Note that \( V_\pi(s) \), \( Q_\pi(s, a) \), and \( A_\pi(s, a) \) are defined with respect to the policy \( \pi \).

1.2.2 RL Algorithms

RL algorithms can be broadly categorized into three categories: value-based methods, policy-based methods, and actor-critic methods. Value-based methods learn an estimate of the optimal value function and use this to determine the best action to take in each state. Policy-based methods learn a direct mapping from states to actions, without explicitly computing the value function. Actor-critic methods combine value-based and policy-based approaches, by learning both a value function and a policy and using the former to guide the updates of the latter. In the following sections, more detailed explanations of each category and some representative algorithms within each will be provided.

**Value-Based Methods**

Value-based methods \([8, 9, 10]\) learn an estimate of the optimal value function, which assigns a value to each state or state-action pair indicating how desirable it
is to be in that state or take that action. The optimal value function is defined as:

$$V^*(s) = \max_\pi \mathbb{E}_{\tau_t \sim \pi} [R(\tau_t) | s_t = s],$$  \hspace{1cm} (6)

$$Q^*(s, a) = \max_\pi \mathbb{E}_{\tau_t \sim \pi} [R(\tau_t) | s_t = s, a_t = a],$$  \hspace{1cm} (7)

which are the value function or Q-function with respect to the optimal policy $\pi^*$, i.e. $V^*(s) = \max_\pi V_\pi(s) = V_{\pi^*}(s)$, $Q^*(s, a) = \max_\pi Q_\pi(s, a) = Q_{\pi^*}(s, a)$.

Based on Bellman Optimality Equation [9], the optimal value function can be decomposed as

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P(\cdot|s, a)} [r(s, a) + \gamma V^*(s')],$$  \hspace{1cm} (8)

$$Q^*(s, a) = \mathbb{E}_{s' \sim P(\cdot|s, a)} [r(s, a) + \gamma \max_{a'} Q^*(s', a')]$$  \hspace{1cm} (9)

Value-based methods learn the optimal value function by iteratively updating an estimate to balance the Bellman Optimality Equation based on the observed rewards and transitions, using techniques such as dynamic programming [9] or temporal difference (TD) learning [10]. They then use the estimated value function to determine the best action to take in each state, typically by selecting the action with the highest estimated value.

Typical examples of value-based methods include Q-learning [11] and SARSA [12]. Q-learning and SARSA use $\epsilon$-greedy policies to overcome the exploration-exploitation dilemma. Since the value function estimation is inherently uncertain, especially in the early stages of learning, if the agent chooses to always exploit the action with the highest estimated value, it may miss out on other potentially better actions that it has not yet explored. In other words, greedy policies can get stuck in local optima and fail to explore the state space sufficiently.

The $\epsilon$-greedy policy (10) selects actions in two ways: with probability $(1 - \epsilon)$, it selects the action with the highest estimated value based on the current value function (i.e., it exploits its current knowledge). With probability $\epsilon$, it selects a
random action uniformly from the action space (i.e., it explores).

\[
a = \begin{cases} 
    \max_a Q(s, a), & 1 - \epsilon, \\
    \text{a random action}, & \epsilon.
\end{cases}
\]  

The advantage of using an \( \epsilon \)-greedy policy is that it allows the agent to explore unexplored actions with a small probability, which can lead to better long-term outcomes.

Additionally, both Q-learning and SARSA adopt one-step TD error to update an estimated optimal Q-value function [11, 12]. However, they differ in the way they update the Q-function.

**SARSA:**

\[
Q(s, a) \leftarrow Q(s, a) + \alpha Q[r + \gamma Q(s', a') - Q(s, a)], a' \sim \epsilon\text{-greedy}
\]  

\[ (11) \]

**Q-Learning:**

\[
Q(s, a) \leftarrow Q(s, a) + \alpha Q[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]
\]  

\[ (12) \]

Eqs. (11) and (12) present respectively the TD updates of the two algorithms for the Q-function, where \( \alpha \) is a hyper-parameter for stepsize. The main difference is in how they handle the exploration-exploitation tradeoff. SARSA uses an on-policy approach, meaning that it learns the value function and policy while following the same policy (i.e. the \( \epsilon \)-greedy) that it is currently using to select actions. On the other hand, Q-Learning is an off-policy approach, meaning that it learns the value function and policy while following a different policy that maximizes the expected return, which is a greedy policy.

While SARSA and Q-Learning both guarantee convergence to an optimal policy for a lookup table, they may suffer from the "curse of dimensionality" when the state space is very large or continuous. To address the curse of dimensionality, DRL algorithms, such as the Deep Q-Network (DQN) [13], have been developed. DQNs combine Q-learning with deep neural networks to approximate the value function for high-dimensional or continuous state spaces. Instead of storing the
value of each state-action pair in a lookup table, DQNs use a deep neural network to approximate the value function, which takes the state as input and outputs the value of each action. The neural network is trained using a variant of Q-learning, where the target values are computed using a separate ”target network” that is periodically updated to stabilize the learning process. However, training DQNs can be challenging due to issues such as non-stationarity and credit assignment. Therefore, improvements like Double DQN [14], Dueling DQN [15], and prioritized experience replay [16] have been proposed to address these issues.

Policy-Based Methods

Policy-based methods [17] learn directly a policy, which is a mapping from states to actions. Instead of estimating the optimal value function and using it to select the best action, policy-based methods learn a parameterized policy, e.g. \( \pi_\theta \), that outputs a probability distribution over the possible actions given the current state. Policy-based methods update the parameters \( \theta \) of the policy iteratively based on the observed rewards and transitions, using techniques such as gradient-free or policy gradient.

Gradient-free methods, such as random search [18, 19] and genetic algorithms [20, 21], do not rely on the gradient of the expected reward with respect to the policy parameters. Instead, they use some form of heuristic or search algorithm to explore the space of possible policy parameters and find the one that maximizes the expected reward. Gradient-free methods can be useful in settings where the policy space is highly non-linear or non-differentiable, but they can be less efficient than gradient-based methods for high-dimensional problems.

Policy gradient methods [17, 19], on the other hand, use gradient-based optimization to update the policy parameters in the direction that maximizes the expected cumulative reward. Policy gradient methods calculate the derivative of
the expected return with respect to the policy parameters, i.e. $\nabla J_\theta(\theta_k)$, and use this gradient to update the parameters in each iteration,

$$\theta_{k+1} = \theta_k + \alpha_\pi \nabla J_\theta(\theta_k), \quad (13)$$

where $\alpha_\pi$ is a hyperparameter for step size. Based on the Policy Gradient Theorem [19], the derivative of the expected return has a closed form:

$$\nabla J_\theta(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \right]. \quad (14)$$

REINFORCE [17] is a typical policy gradient RL algorithm, which estimate the above gradient by Monte Carlo sampling. While policy gradient methods can be effective in many situations, especially for high-dimensional parameter spaces, they can also suffer from high variance in the gradient estimates. The high variance in policy gradient estimates can occur due to the stochasticity of the rewards and the policies, as well as the non-stationarity of the policy distribution during training. This high variance can lead to slow convergence, instability, and poor performance.

**Actor-Critic Methods**

Actor-critic methods [22] use a value function to estimate the expected return and reduce the variance of the gradient estimates of policy gradient methods. This can help to reduce the variance of the policy gradient and improve the stability and convergence of the algorithm. The actor component of the algorithm updates the policy based on the policy gradient, while the critic component updates the value function based on temporal difference learning or another value-based method. By combining value-based and policy-based approaches, actor-critic methods can leverage the strengths of both methods to achieve faster and more stable learning, while also providing a flexible framework for incorporating additional features such as eligibility traces, off-policy learning, and function approximation.
Advantage Actor Critic (A2C) and Asynchronous Advantage Actor Critic (A3C) [23] are standard actor-critic methods. One key innovation in A2C and A3C is the use of an advantage function to reduce the high variance of the policy gradient estimates.

\[
\nabla J_\theta(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) A_{\pi_\theta}(s_t, a_t) \right].
\]

Intuitively, the advantage function captures the “advantage” of taking a particular action in a particular state, and it provides a way to reduce the variance of the policy gradient estimates by subtracting the baseline value from the actual return. In A2C and A3C, the advantage function is usually estimated using a temporal difference (TD) error, which it provides a signal for the actor to update the policy in the direction of higher expected returns.

Deep Deterministic Policy Gradients (DDPG) [24] is a specific type of actor-critic algorithm that builds upon the standard actor-critic architecture by using deep neural networks to represent the policy and the value function. In DDPG, the actor network outputs the action directly, instead of a probability distribution over the actions, making it suitable for continuous action spaces. The critic network estimates the expected reward given the current state and the action taken by the actor. DDPG uses experience replay and target networks to improve the stability of the algorithm and mitigate the issue of overestimation of the value function.

Trust-region method is another type of actor-critic algorithm that aims to restrict the size of the policy updates based on a trust region constraint, which is defined in terms of a maximum step size or a maximum change in the policy parameters. One popular trust region method is Trust Region Policy Optimization (TRPO) [25], which uses the Kullback-Leibler (KL) divergence between the new policy and the old policy as the trust region constraint. TRPO has several advantages over other actor-critic algorithms, including improved sample efficiency,
faster convergence, and more stable updates. However, TRPO can be computationally expensive because it requires second-order gradient information to estimate the KL-divergence constraint. To address the limitation, a more scalable and easier-to-use variant of TRPO, called Proximal Policy Optimization (PPO), was proposed in [26]. PPO uses a clipped objective function to constrain the policy updates, instead of computing the KL-divergence. The clipped objective function limits the change in the probability ratio between the new and old policies to a pre-specified range. PPO has been shown to achieve state-of-the-art performance in many benchmark RL tasks, and it has become a popular algorithm for researchers and practitioners.

1.2.3 Dissertation Outline

This dissertation focuses on the use of trust-region methods in model-free reinforcement learning (RL) and aims to develop fundamental theories, principled algorithms, and practical applications to overcome the challenges associated with RL. Through this work, the goal is to advance the field of RL and make it more applicable to real-world problems by providing effective and efficient solutions.

The outline of this dissertation is illustrated in Fig. 2. Chapter 2 of this dissertation presents a novel theoretical result of trust-region-based RL on a closed-form policy update rule with monotonic improve guarantee. Building upon this theoretical result, Chapter 3 introduces an off-policy RL algorithm that enables sample-efficient learning. To extend RL to multi-agent systems, Chapter 4 proposes a distributed multi-agent RL (MARL) algorithm for multi-agent collaboration systems based on trust-region methods and distributed consensus optimization. In addition, Chapter 5 develops real-world applications of safe RL (SRL) and MARL algorithms in the smart grid domain.

Chapter 2 of this dissertation presents a novel theoretical result for trust-
region-based RL that bridges the gap between theory and practice by proposing a lower bound on policy performance. The lower bound tightens previous results of trust-region methods by connecting itself to an expected KL-Divergence. Furthermore, this chapter derives a closed-form policy update rule with monotonic improvement guarantee based on the theoretical result.

Chapter 3 builds upon this result by proposing an off-policy RL algorithm that enables sample-efficient learning of deep neural network policies while retaining the stability of trust-region methods. The proposed algorithm demonstrates superior
performance on benchmark continuous control tasks over prior trust-region-based DRL algorithms in terms of both the final return and sample efficiency.

In Chapter 4, the dissertation proposes a distributed MARL algorithm that extends trust-region policy optimization from single-agent domains to cooperative multi-agent systems. This algorithm enables heterogeneous autonomous agents to collaborate and achieve a common objective through consensus over a peer-to-peer communication network. The algorithm is fully decentralized and does not require agents to share private information.

Chapter 5 focuses on the real-world applications of safe RL and MARL algorithms in the smart grid domain. The trust-region-based SRL algorithm is applied to solve the optimal operation problem of distribution grids considering uncertain renewable energy sources, addressing the safety issue of the power system environment. The MARL algorithm proposed in Chapter 4 is applied to solve the distributed Volt-VAR optimization (VVO) problems in distribution grids, considering the increasing installation of distributed energy sources. The effectiveness of these methods is justified through real-world datasets and simulation studies on IEEE benchmark systems.
CHAPTER 2

Monotonic Improvement Guarantee for Trust-Region Policy Optimization

2.1 Overview

Policy search methods have gained great popularity in reinforcement learning for the last decade. As opposed to value function methods, in which the policy is represented implicitly by a greedy action-selection strategy with respect to an estimated value function, policy search methods search directly in the space of policy representations for a good policy. The advantages of policy search methods include being able to learn stochastic policies [27], better convergence, and effectiveness in high-dimensional or continuous action spaces. Generally, policy search approaches use function approximators, such as neural networks, to construct a parametric policy. The parametric policy is then optimized using policy gradient [17, 19] or derivative-free algorithms [28] by searching in the parameter space.

However, policy gradient based approaches have two fundamental limitations. First, it lacks a universal criteria for determining the step-size to guarantee improvement in the direction of gradient ascent. Second, it is sample inefficient because the calculation of policy gradients requires integration over the states and actions from the current policy. This chapter presents a closed-form policy update rule that is independent of parametric function approximators. The update rule has a monotonic improvement guarantee and is suitable for optimizing general stochastic policies with continuous or discrete actions. The update rule provides a new theoretical foundation for policy-based RL, which traditionally restricts the policy search to a family of parametric functions, such as policy gradient [19], DDPG [24, 29], actor critic [22, 30], soft actor-critic (SAC) [31, 32], and so on. The update rule is derived from a closed-form solution to a trust-region method.
using calculus of variation.

The trust-region method is a vital tool in the field of RL. The primary objective of this method is to improve the existing policy by exploring the local area around it, where the objective function is well-approximated by a manageable surrogate model. One of the well-known trust-region methods is TRPO [25], which offers a simple and effective surrogate model that can be evaluated using the current best policy. Additionally, TRPO provides an upper bound of the approximation error of the surrogate model, which is crucial because subtracting this bound from the surrogate model yields the worst-case performance degradation, or a lower bound, of the true objective. Therefore, maximizing the lower bound results in an enhanced policy with performance that does not decrease [25].

The TRPO theory holds significant importance for policy-based RL as it provides a guaranteed approach for policy improvement. However, the derived bound in TRPO is dependent on the maximum KL-Divergence of the proposed policy $\pi'$ and the current policy $\pi$ on the entire state space, which can be extremely large or even infinite, even if $\pi'$ and $\pi$ are relatively close in most states. TRPO addresses this issue by heuristically imposing a strict constraint to bound the KL-Divergence at every state. However, implementing this constraint becomes intractable when dealing with a large or continuous state space. To overcome this challenge, an empirical approximation using the expected KL-Divergence, such as $\mathbb{E}_{s \sim d^\pi} [D_{KL}[\pi'\|\pi](s)]$, is usually adopted [25, 33]. Despite this, the monotonic improvement property is no longer guaranteed.

This dissertation establishes a new theoretical result on the bound of the surrogate approximation error by connecting it to the expected KL-Divergence. This result leads to a more practical lower bound of the objective, which improves previous analysis on this topic in terms of KL-Divergence, such as [25, 33, 34]. It also
closes the gap between theory and practice in TRPO and the related approaches. Furthermore, this result enables us to derive a closed-form solution for policy optimization. The closed-form solution introduces a very simple policy update rule that guarantees to produce monotonically improving policies. Furthermore, the update rule extends immediately to partially observable Markov games with cooperative agents and the monotonic improvement guarantee still holds when updates are performed by one agent at a time.

2.2 Related Work

The idea of restricting policy search to a local area of the current policy is common in model-free RL. For instance, instead of imposing a hard boundary on the searching area, Kakade & Langford [35] proposed a conservative update scheme mixing the current policy and a greedy update via a weighted sum. A lower bound on the performance improvement as a function of the weighting coefficient was proven. Following this line of work, Pirotta et al. [36] proposed two more general lower bounds connecting to the difference between two policies. Then, two conservative update algorithms were developed by maximizing the proposed bounds, respectively. Zhu & Matsubara [37] proposed a similar bound and a practical algorithm for entropy-regularized RL. While monotonic improvement guarantee is derived in the previous studies, the update scheme cannot apply to non-mixture policies. Schulman et al. [25] extended this line of work to general stochastic policies and proposed a new bound that connected it to the maximum KL-Divergence between two successive policies on the state space. However, this bound is intractable when the state space is large. Although a tighter bound relating it to an average total variation distance is proposed by Achiam et al. in [33], deriving a closed-form policy update rule from the lower bound is still challenging.

In practice, many approaches use a hard constraint to bound the searching
area but they generally lose the monotonic improvement guarantee. Peters et al. [38] proposed relative entropy policy search (REPS) to restrict the relative entropy between observed data distribution of the state-action pairs and the distribution generated by the new policy. A closed-form update rule in a softmax form was derived using the method of Lagrange multipliers. However, this approach is not straightforwardly extendable to general non-linear policies. To apply nonlinear policies, TRPO [25] and constrained policy optimization [33] approximately constrained the on-policy expected KL-Divergence by using second-order Taylor expansion, which was closely related to natural policy gradient [39]. Extending the work in TRPO, Akrour et al. [34] provided a monotonic improvement guarantee for bounding the expected KL-divergence, but the result only held for linear-Gaussian policies. Nachum et al. [40, 41] presented multi-step softmax consistencies under entropy regularization and adopted a discounted relative entropy trust-region constraint to improve exploration and stability. By relating policy search to probabilistic inference, Levine [42] and Abdolmaleki et al.[43] proposed the maximum a posterior policy optimization (MPO) algorithm based on Expectation-Maximization, where the policy update was decomposed into E-step and M-step. A closed-form E-step combined with a maximum-a-posteriori-estimation M-step for Gaussian policies was provided. Although a monotonic improvement guarantee is claimed, the guarantee is for the KL-Divergence regularized objective rather than the true expected return. Besides, suffering from the same issue as in [38], the policy update rule needs to determine the optimal Lagrangian multipliers of the dual problem, which requires a costly nonlinear optimization in the inner loop. Different from previous works, Otto et al. [44] proposed projection-based solutions to impose trust-region constraints on the individual state, which enabled exact guarantees of monotonic improvement. Three closed-form projection
layers based on Wasserstein L2 distance, Frobenius norm, and KL-Divergence were proposed to project the updated policy onto trust regions. However, the proposed approach only applies to Gaussian policies.

2.3 Problem Formulation

In this section, the problem formulations for single-agent RL and multi-agent RL settings are presented. Then, the mathematical model of the trust-region method for RL policy optimization is introduced.

2.3.1 Markov Decision Process

For single-agent RL, the problem can be formulated as an MDP, which is presented in Section 1.2.1 of Chapter 1. For the MDP \((\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)\) and the policy \(\pi\), the objective is to maximize the expected return

\[
J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right].
\] (16)

Let \(\rho_t^\pi(s)\) denote the probability density (mass) function of the state at step \(t\) given the policy \(\pi\), where \(\rho_0^\pi(s) = \rho_0(s)\) is the probability density (mass) function of the initial state \(s_0\). Given that, the discounted state visitation probability density (mass) function is denoted by

\[
d^\pi(s) = (1 - \gamma) \left[ \rho_0^\pi(s) + \gamma \rho_1^\pi(s) + \gamma^2 \rho_2^\pi(s) + \cdots \right]
= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \rho_t^\pi(s).
\] (17)

Also, let \(\nu_\pi(s'|s)\) denote the one-step state transition density given the policy \(\pi\):

\[
\nu_\pi(s'|s) = \int_{\mathcal{A}} p(s'|s, a)\pi(a|s)da,
\] (18)

and \(\nu_t^\pi(s'|s)\) be the \(t\)-step state transition density given the policy \(\pi\) (the Chapman Kolmogorov equation):

\[
\nu_t^\pi(s'|s) = \int_{\tilde{s}} \nu_\pi^m(s'|\tilde{s})\nu_{\pi}^{t-m}(\tilde{s}|s)d\tilde{s},
\] (19)
where $0 \leq m \leq t$, and $\nu^0\pi(s'|s)$ is a Dirac delta distribution:

$$\nu^0\pi(s'|s) = \begin{cases} 
\infty, & \text{if } s' = s, \\
0, & \text{otherwise.} 
\end{cases}$$  (20)

Given these definitions, the discounted state transition PDF under the policy $\pi$ is

$$\mu\pi(s'|s) = (1 - \gamma) \left[ \nu^0\pi(s'|s) + \gamma \nu^1\pi(s'|s) + \gamma^2 \nu^2\pi(s'|s) + \cdots \right]$$

$$= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \nu^t\pi(s'|s).$$  (21)

Then, the discounted visitation PDF can be written as

$$d\pi(s') = \int_s r_0(s) \mu\pi(s'|s) ds.$$  (22)

And the objective can be compactly written as

$$J(\pi) = \mathbb{E}_{s \sim d^\pi, a \sim \pi(\cdot|s)}[r(s, a)].$$  (23)

2.3.2 Partially Observable Markov Game

For multi-agent RL, the problem can be formulated as a partially observable Markov game (POMG) [45], which is defined on a state space, $\mathcal{S}$, and a collection of action spaces, $\mathcal{A}^1, \ldots, \mathcal{A}^N$, one for each agent in the environment. The state transition $s \mapsto s'(s, s' \in \mathcal{S})$ happens following the probability density $P : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ when the actions $a = [a^1, \ldots, a^N]$, $a^i \in \mathcal{A}^i$, $i \in \{1, \ldots, N\}$, are exerted on the environment at state $s$. Each agent is rewarded based on a local reward function $r^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \mapsto [r^i_{\min}, r^i_{\max}]$, which depends on the current state $s$ and the joint action $a$.

In a POMG, each agent has a local observation of the environment, $o^i$, which contains incomplete information of the state $s$. At state $s$, $o^i$ is observed with a likelihood, $P^o_\pi : \mathcal{S} \times \mathcal{O}^i \mapsto \mathbb{R}_{\geq 0}$, where $\mathcal{O}^i$ is the observation space of the agent.
Each agent acts according to a policy $\pi^i : \mathcal{O}^i \times \mathcal{A}^i \mapsto \mathbb{R}_{\geq 0}$ (or $\mapsto [0, 1]$), which is a probability distribution (or a probability mass function) over the action space $\mathcal{A}^i$ given the observation $o^i$. The joint policy $\pi(a|s)$ and the joint policy $\pi^{-i}(a^{-i}|s)$ except $i$ are defined by:

$$\pi(a|s) = \prod_{i \in \mathcal{N}} \int_{o^i} \pi^i(a^i|o^i)P_o(o^i|s)do^i,$$  \hspace{1cm} (24)

$$\pi^{-i}(a^{-i}|s) = \prod_{j \in \mathcal{N}\backslash\{i\}} \int_{o^j} \pi^j(a^j|o^j)P_o(o^j|s)do^j. \hspace{1cm} (25)$$

where $\mathcal{N} = \{1, 2, \ldots, N\}$ is a set of agent’s IDs. The goal of the agents is to learn a set of distributed policies $\{\pi^i(a^i|o^i)|i \in \mathcal{N}\}$ to maximize the expected return

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t (r_t^1 + \cdots + r_t^N) \right]$$ \hspace{1cm} (26)

where $\tau \sim \pi$ indicates that $s_0 \sim \rho_0, o_t^i \sim P_o(\cdot|s_t), a_t^i \sim \pi^i(\cdot|o_t^i), s_{t+1} \sim P(\cdot|s_t, a_t^1, ..., a_t^N)$.

### 2.3.3 Trust Region Methods

Trust region methods restrict policy search to a local region around the current best solution, where the objective function is well-approximated by a surrogate model. Trust Region Methods solve the following optimization:

$$\pi_{k+1} = \arg \max_{\pi' \in \Pi} J(\pi'), \text{ s.t. } D(\pi', \pi_k) \leq \delta \hspace{1cm} (27)$$

where $J(\pi')$ is some surrogate model, $D$ is a distance measure, and $\delta > 0$ is the size of the trust region, in which we search for an improved policy. A simple and effective choice for the surrogate model is

$$L_{\pi_k}(\pi') = J(\pi_k) + \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^k, a \sim \pi'} [A_{\pi_k}(s, a)]. \hspace{1cm} (28)$$
Schulman, et al. [25] prove that the difference between the surrogate model and the true objective is bounded by:

$$|J(\pi') - L_{\pi_k}(\pi')| \leq C \max_s D_{\text{KL}}[\pi' \parallel \pi_k](s),$$

where $C = \frac{4\gamma\epsilon}{(1 - \gamma)^2}$, $\epsilon = \max_{s,a} |A_{\pi_k}(s, a)|$,

which connects it to the maximum KL-Divergence over the state space, $\max_s D_{\text{KL}}[\pi' \parallel \pi_k](s)$. By using this bound, the worst-case performance degradation of the true objective is derived:

$$J(\pi') \geq L_{\pi_k}(\pi') - C \max_s D_{\text{KL}}[\pi' \parallel \pi_k](s),$$

It follows that maximizing the right-hand side of the inequality, which is a lower bound of the true objective function, can lead to guaranteed improvement in the performance. This result has fostered a branch of practical trust-region algorithms, i.e. [25, 26, 33, 41, 46], which approximately optimize the lower bound to improve policies.

### 2.4 Theoretical Results

The principle theoretical result is a closed-form solution to policy optimization based on trust-region methods, following a new bound on the difference between the surrogate model and the true objective, which is proved by Li et al. [47]. The closed-form solution leads to a policy update rule that guarantees monotonic policy improvement. The policy update rule is suitable for optimizing general stochastic policies with discrete or continuous actions. Moreover, the update rule extends immediately to cooperative multi-agent systems when updates are performed by one agent at a time.

#### 2.4.1 Monotonic Improvement Guarantee for Single-Agent RL

The first theoretical result is a new bound on the difference between the surrogate model and the true objective, which improves existing bounds by relating
it to an expected KL Divergence:

\[ |J(\pi') - L_\pi(\pi')| \leq \frac{1}{1 - \gamma} C_\pi \mathbb{E}_{s \sim d^\pi} [D_{KL}[\pi' \parallel \pi](s)], \]

where \( C_\pi = \frac{\gamma^2 \epsilon}{(1 - \gamma)^3}, \epsilon = \max_{s,a} |A_\pi(s,a)|. \) (31)

This proof uses techniques from the proof of Lemma 3 in [25], exploiting them to derive a new bound that relates to an average divergence between policies, \( \pi', \pi. \) An informal overview is as follows. First, using Lemma 1 in [25], the gap between the surrogate and the objective is decomposed into the difference of two expected advantages over the policies \( \pi', \pi. \) Then, the coupling technique is used to measure the coincidence of two trajectories resulted from \( \pi', \pi \) before an arbitrary timestep \( t. \) Finally, the gap is constrained to an average KL-Divergence using Pinsker’s inequality.

To present the theoretical result and its proof in a clear way, the definition of \( \alpha \)-coupled policies from the Definition 1 in [25] is introduced first.

**Definition 2.4.1 (\( \alpha \)-coupled policies).** A coupling of two probability distributions \( \mu \) and \( \nu \) is a pair of random variables \((X, Y)\) defined on a single probability space such that the marginal distribution of \( X \) is \( \mu \) and the marginal distribution of \( Y \) is \( \nu \) [48].

The policies \( \pi'(a'|s) \) and \( \pi(a|s) \) are called \( \alpha \)-coupled if they define a coupling of \((\pi', \pi)\) such that

\[ P(a' \neq a|s) \leq \alpha(s). \] (32)

Numerically, \( \alpha \)-coupling means that the actions \( a' \) and \( a \) given state \( s \) match with probability of at least \( 1 - \alpha(s) \) when their samples are drawn using the same seed.

The technique of coupling is useful because it relates two policies to their total variation distance. According to the lemma 4.7 in [48], for policies \( \pi' \) and \( \pi, \) there
exists a coupling that satisfies

\[ D_{TV}[\pi'|\pi](s) = \inf \{P(a' \neq a|s), a' \text{ and } a \text{ is a coupling of } \pi' \text{ and } \pi\}. \quad (33) \]

where \( D_{TV}[\pi'|\pi](s) \) represents the total variation distance between policies \( \pi' \) and \( \pi \) given the state \( s \). This means that \( D_{TV}[\pi'|\pi](s) \) is the infimum of the probability \( P(a' \neq a|s) \), and therefore \( \alpha(s) \) can be selected to be \( D_{TV}[\pi'|\pi](s) \).

Note that the definition of \( \alpha(s) \) here, depending on the state \( s \), is different from the definition in [25], which is the maximum over the state space, i.e. \( \max_{s \in S} \alpha(s) \).

Next, a lemma from [35] and [25] is presented, which shows that the performance difference between two arbitrary policies can be expressed as an expected advantage of one policy over a trajectory resulted from the other. Following that, another two important lemmas will be introduced.

**Lemma 2.4.2.** Given two policies \( \pi', \pi \), we have

\[ J(\pi') = J(\pi) + \frac{1}{1-\gamma}E_{s \sim d^{\pi'}, a \sim \pi'} [A_{\pi}(s, a)]. \quad (34) \]

**Proof.** Note that \( A_{\pi}(s, a) = E_{s' \sim P(\cdot|s, a)} [r(s, a) + \gamma V_{\pi}(s') - V_{\pi}(s)] \). Therefore,

\[
E_{s \sim d^{\pi'}, a \sim \pi'} [A_{\pi}(s, a)] \\
= E_{s \sim d^{\pi'}, a \sim \pi'} [r(s, a) + \gamma V_{\pi}(s') - V_{\pi}(s)] \\
= (1-\gamma)E_{s_t \sim d_0^{\pi'}, a_t \sim \pi', s_{t+1} \sim P} \left[ \sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \gamma V_{\pi}(s_{t+1}) - V_{\pi}(s_t)) \right] \\
= (1-\gamma)E_{s_t \sim d_0^{\pi'}, a_t \sim \pi'} \left[ -V_{\pi}(s_0) + \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \\
= (1-\gamma) \left[ -E_{s_0 \sim d_0} [V_{\pi}(s_0)] + E_{s_t \sim d_0^{\pi'}, a_t \sim \pi'} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \right] \\
= (1-\gamma) [-J(\pi) + J(\pi')] \]

Rearranging it, the result follows. \( \square \)
Lemma 2.4.3. Given two stochastic policies \( \pi', \pi \) and their discounted state transition PDFs, \( \mu_{\pi'}(s'|s), \mu_{\pi}(s'|s) \), the following inequality holds:

\[
\int_S |\mu_{\pi'}(s'|s) - \mu_{\pi}(s'|s)| ds' \leq \frac{2\gamma^2}{1 - \gamma} \int_S \mu_{\pi}(s'|s) D_{TV}[\pi' || \pi](s') ds'.
\] (36)

**Proof.** First note that

\[
\gamma \int_S \nu_{\pi}(s'|\bar{s}) \mu_{\pi}(\bar{s}|s) d\bar{s} = \gamma \int_S \nu_{\pi}(s'|\bar{s})(1 - \gamma) [\nu_{\pi}^0(\bar{s}|s) + \gamma \nu_{\pi}^1(\bar{s}|s) + \gamma^2 \nu_{\pi}^2(\bar{s}|s) + \cdots] d\bar{s}
\]

\[
= (1 - \gamma) \left[ \gamma \nu_{\pi}^1(\bar{s}|s) + \gamma^2 \nu_{\pi}^2(\bar{s}|s) + \gamma^3 \nu_{\pi}^3(\bar{s}|s) + \cdots \right]
\]

\[
= \mu_{\pi}(s'|s) - (1 - \gamma) \nu_{\pi}^0(\bar{s}|s).
\] (37)

Then, we have

\[
\gamma \int_{S \times S} \mu_{\pi'}(s'|\bar{s}) [\nu_{\pi'}(\bar{s}|\bar{s}) - \nu_{\pi}(\bar{s}|\bar{s})] \mu_{\pi}(\bar{s}|s) d\bar{s} d\bar{s} = \int_S \left( \gamma \int_S \mu_{\pi'}(s'|\bar{s}) \nu_{\pi'}(\bar{s}|s) d\bar{s} \right) \mu_{\pi}(\bar{s}|s) d\bar{s} - \int_S \mu_{\pi'}(s'|\bar{s}) \left( \gamma \int_S \nu_{\pi}(\bar{s}|\bar{s}) \mu_{\pi}(\bar{s}|s) d\bar{s} \right) d\bar{s}
\]

\[
= \int_S \left[ \nu_{\pi'}(s'|\bar{s}) - (1 - \gamma) \nu_{\pi'}^0(\bar{s}'|s) \right] \mu_{\pi}(\bar{s}|s) d\bar{s} - \int_S \mu_{\pi'}(s'|\bar{s}) [\mu_{\pi}(\bar{s}|s) - (1 - \gamma) \nu_{\pi'}^0(\bar{s}|s)] d\bar{s}
\]

\[
= (1 - \gamma) \int_S \nu_{\pi'}^0(\bar{s}'|s) \mu_{\pi}(\bar{s}|s) d\bar{s} + (1 - \gamma) \int_S \nu_{\pi'}^0(\bar{s}'|s) \mu_{\pi}(\bar{s}|s) d\bar{s} - \int_S \nu_{\pi'}^0(\bar{s}'|s) \mu_{\pi}(\bar{s}|s) d\bar{s}
\]

\[
= \frac{1 - \gamma}{\gamma} \left[ \left( \mu_{\pi'}(s'|s) - (1 - \gamma) \nu_{\pi'}^0(\bar{s}'|s) \right) - \left( \mu_{\pi}(s'|s) - (1 - \gamma) \nu_{\pi'}^0(\bar{s}'|s) \right) \right]
\]

\[
= \frac{1 - \gamma}{\gamma} \left[ \mu_{\pi'}(s'|s) - \mu_{\pi}(s'|s) \right].
\] (38)

Rearranging the equation, we have

\[
\mu_{\pi'}(s'|s) - \mu_{\pi}(s'|s) = \frac{\gamma^2}{1 - \gamma} \int_{S \times S} \mu_{\pi'}(s'|\bar{s}) [\nu_{\pi'}(\bar{s}|\bar{s}) - \nu_{\pi}(\bar{s}|\bar{s})] \mu_{\pi}(\bar{s}|s) d\bar{s} d\bar{s}.
\] (39)
Recalling the definition of one-step state transition density in Equation (18), we have

\[
\nu_{\pi'}(\tilde{s}|\bar{s}) - \nu_{\pi}(\tilde{s}|\bar{s}) = \int_{\mathcal{A}} p(\tilde{s}|\bar{s}, \bar{a}) \left[ \pi'(\bar{a}|\tilde{s}) - \pi(\bar{a}|\tilde{s}) \right] d\bar{a}.
\] (40)

Then, we have

\[
\gamma^2 \int_{\mathcal{S}} |\mu_{\pi'}(s'|s) - \mu_{\pi}(s'|s)| ds' \\
\leq \gamma^2 \int_{\mathcal{S} \times \mathcal{S} \times \mathcal{S} \times \mathcal{A}} \mu_{\pi'}(s'|\tilde{s}) p(\tilde{s}|s, \bar{a}) |\pi'(\bar{a}|\tilde{s}) - \pi(\bar{a}|\tilde{s})| \mu_{\pi}(s|s) d\tilde{s} d\bar{a} d\bar{a} d\bar{s} \\
= \gamma^2 \int_{\mathcal{S}} \mu_{\pi'}(s'|\tilde{s}) ds' \int_{\mathcal{S} \times \mathcal{S} \times \mathcal{A}} p(\tilde{s}|s, \bar{a}) |\pi'(\bar{a}|\tilde{s}) - \pi(\bar{a}|\tilde{s})| \mu_{\pi}(s|s) d\tilde{s} d\bar{a} d\bar{s} \\
= \gamma^2 \int_{\mathcal{S}} \mu_{\pi}(\tilde{s}|s) \int_{\mathcal{A}} |\pi'(\bar{a}|\tilde{s}) - \pi(\bar{a}|\tilde{s})| d\bar{a} d\bar{s} \\
= \gamma^2 \int_{\mathcal{S}} \mu_{\pi}(\tilde{s}|s) D_{TV} [\pi'||\pi](\tilde{s}) d\tilde{s}.
\] (41)

Replacing all \( \tilde{s} \) with \( s' \), the result follows.

Lemma 2.4.4. Let \( \bar{\alpha}_t \) and \( \gamma \) be any real numbers within \( \bar{\alpha}_t \in [0, 1], \forall t \in \mathbb{N} \) and \( \gamma \in [0.5, 1) \), the following inequality holds:

\[
(1 - \gamma)^2 \sum_{t=0}^{\infty} \gamma^t \bar{\alpha}_t \bar{\alpha}_{0t} \leq \sum_{t=0}^{\infty} \gamma^t \bar{\alpha}_{t}^2.
\] (42)

where \( \bar{\alpha}_{0t} = 1 - \prod_{i=0}^{t}(1 - \bar{\alpha}_i) \).

Proof. First note that \( \bar{\alpha}_{0t} \) can be expressed as

\[
\bar{\alpha}_{0t} = \bar{\alpha}_t + (1 - \bar{\alpha}_t) \bar{\alpha}_{t-1} + (1 - \bar{\alpha}_t)(1 - \bar{\alpha}_{t-1}) \bar{\alpha}_{t-2} + \cdots + \prod_{i=0}^{t}(1 - \bar{\alpha}_i) \bar{\alpha}_0,
\] (43)

or in a recursive form:

\[
\bar{\alpha}_{0t} = \bar{\alpha}_t + (1 - \bar{\alpha}_t) \bar{\alpha}_{0t-1},
\] (44)
where $\sigma_{00} = \sigma_0$. Then, we have

\[
(1 - \gamma)^2 \sum_{t=0}^{\infty} \gamma^t \alpha_t \alpha_{0t} \\
= (1 - \gamma)^2 \sum_{t=0}^{\infty} \gamma^t \alpha_t^2 + (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \alpha_t (1 - \alpha_t) \alpha_{0t-1} \\
= \sum_{t=0}^{\infty} \gamma^t \alpha_t^2 - (2\gamma - \gamma^2) \sum_{t=0}^{\infty} \gamma^t \alpha_t^2 + (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \alpha_t (1 - \alpha_t) \alpha_{0t-1} \\
= \sum_{t=0}^{\infty} \gamma^t \alpha_t^2 - \left[ (2\gamma - \gamma^2) \sum_{t=0}^{\infty} \gamma^t \alpha_t^2 - (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \alpha_t (1 - \alpha_t) \alpha_{0t-1} \right]
\]

(45)

For the inequality (42) to hold, all that is needed is to prove that the subtrahend on the rightmost-hand side of (45) is greater than 0. Note that

\[
(2\gamma - \gamma^2) \sum_{t=0}^{\infty} \gamma^t \alpha_t^2 - (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \alpha_t (1 - \alpha_t) \alpha_{0t-1} \\
= \sum_{t=0}^{\infty} \gamma^t \left[ \gamma + \gamma(1 - \gamma) \right] \alpha_t^2 - (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \alpha_t (1 - \alpha_t) \alpha_{0t-1} \\
= \sum_{t=0}^{\infty} \gamma^t \left[ (1 - \gamma) \sum_{i=1}^{\infty} \gamma^i + (1 - \gamma)^2 \sum_{i=1}^{\infty} \gamma^i \right] \alpha_t^2 - (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \alpha_t (1 - \alpha_t) \alpha_{0t-1} \\
= (1 - \gamma)^2 \sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{\infty} \gamma^i \left[ \frac{1}{1 - \gamma} + 1 \right] \alpha_i^2 - (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \alpha_t (1 - \alpha_t) \alpha_{0t-1} \\
= \gamma (1 - \gamma) \alpha_0^2 + (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \left[ \sum_{n=0}^{t-1} (t - n) \alpha_n^2 + \frac{\gamma}{1 - \gamma} \alpha_t^2 - \alpha_t (1 - \alpha_t) \alpha_{0t-1} \right] \\
\geq \gamma (1 - \gamma)^2 \alpha_0^2 + (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \left[ \sum_{n=0}^{t-1} (t - n) \alpha_n^2 + \alpha_t^2 - \alpha_t \alpha_{0t-1} \right].
\]

(46)
In the expanded form, the rightmost-hand side of (46) can be expressed as

\[
\gamma(1 - \gamma)^2 \bar{\alpha}_0^2 + (1 - \gamma)^2 \sum_{t=1}^{\infty} \gamma^t \left[ \sum_{n=0}^{t-1} (t-n)\bar{\alpha}_n^2 + \bar{\alpha}_t^2 - \bar{\alpha}_t\bar{\alpha}_{0t-1} \right]
\]

\[
= \gamma(1 - \gamma)^2 \left[ \bar{\alpha}_0^2 + \bar{\alpha}_0^2 + \bar{\alpha}_1^2 - \bar{\alpha}_1\bar{\alpha}_0 \right] +
\]

\[
\gamma^2(1 - \gamma)^2 \left[ 2\bar{\alpha}_0^2 + \bar{\alpha}_1^2 + \bar{\alpha}_2^2 - \bar{\alpha}_2\bar{\alpha}_01 \right] +
\]

\[
\gamma^3(1 - \gamma)^2 \left[ 3\bar{\alpha}_0^2 + 2\bar{\alpha}_1^2 + \bar{\alpha}_2^2 + \bar{\alpha}_3^2 - \bar{\alpha}_3\bar{\alpha}_02 \right] +
\]

\[
\gamma^4(1 - \gamma)^2 \left[ 4\bar{\alpha}_0^2 + 3\bar{\alpha}_1^2 + 2\bar{\alpha}_2^2 + \bar{\alpha}_3^2 + \bar{\alpha}_4^2 - \bar{\alpha}_4\bar{\alpha}_03 \right] +
\]

\[
\ldots
\]

Since

\[
\gamma^t(1 - \gamma)^2 = \gamma^t(1 - \gamma)^2(1 - \gamma + \gamma) = \gamma^t(1 - \gamma)^3 + \gamma^{t+1}(1 - \gamma)^2,
\]

Equation (47) can be rewritten as

\[
\gamma(1 - \gamma)^3 \left[ 2\bar{\alpha}_0^2 + \bar{\alpha}_1^2 - \bar{\alpha}_1\bar{\alpha}_0 \right] +
\]

\[
\gamma^2(1 - \gamma)^3 \left[ 4\bar{\alpha}_0^2 + 2\bar{\alpha}_1^2 + \bar{\alpha}_2^2 - \bar{\alpha}_2\bar{\alpha}_01 \right] +
\]

\[
\gamma^3(1 - \gamma)^3 \left[ 7\bar{\alpha}_0^2 + 4\bar{\alpha}_1^2 + 2\bar{\alpha}_2^2 + \bar{\alpha}_3^2 - \bar{\alpha}_3\bar{\alpha}_01 - \bar{\alpha}_3\bar{\alpha}_01 \right] +
\]

\[
\gamma^4(1 - \gamma)^3 \left[ 11\bar{\alpha}_0^2 + 7\bar{\alpha}_1^2 + 4\bar{\alpha}_2^2 + 2\bar{\alpha}_3^2 + \bar{\alpha}_4^2 - \bar{\alpha}_4\bar{\alpha}_01 - \bar{\alpha}_4\bar{\alpha}_01 - \bar{\alpha}_4\bar{\alpha}_02 - \bar{\alpha}_4\bar{\alpha}_03 \right] +
\]

\[
\ldots
\]

\[
= (1 - \gamma)^3 \sum_{t=1}^{\infty} \gamma^t \left[ a_t\bar{\alpha}_0^2 + \sum_{i=1}^{t} \left( a_{t-i}\bar{\alpha}_i^2 - \bar{\alpha}_i\bar{\alpha}_{0i-1} \right) \right]
\]

\[
= (1 - \gamma)^3 \sum_{t=1}^{\infty} \gamma^t H_t
\]

where

\[
a_t = 1 + \sum_{j=0}^{t} j.
\]

and

\[
H_t = a_t\bar{\alpha}_0^2 + \sum_{i=1}^{t} \left( a_{t-i}\bar{\alpha}_i^2 - \bar{\alpha}_i\bar{\alpha}_{0i-1} \right).
\]
Next, we prove $H_t \geq 0$ for all $t \in \mathbb{N}^+$ by using convex optimization. Decompose $H_t$ into:

$$H_t = a_t \alpha_0^2 + \sum_{i=1}^{t-1} \left(a_{t-i} \alpha_i^2 - \alpha_i \alpha_{0t-1} \right) + a_0 \alpha_t^2 - \alpha_t \alpha_{0t-1}$$  \hspace{1cm} (52)

$$= h_{t-1} + a_0 \alpha_t^2 - \alpha_t \alpha_{0t-1}$$

Taking the partial derivative of $H_t$ with respect to $\alpha_t$ and setting it to be zero, $H_t$ attains its minimum value, i.e.,

$$H_t \geq h_{t-1} - \frac{1}{4a_0} \alpha_{0t-1}^2.$$  \hspace{1cm} (53)

Denoting $b_1 = 1/4a_0$ and decomposing $h_{t-1}$, we get

$$H_t \geq h_{t-2} + a_t \alpha_{t-1}^2 - \alpha_{t-1} \alpha_{0t-2} - b_t \alpha_{0t-1}^2$$

$$\geq h_{t-2} + a_t \alpha_{t-1}^2 - \alpha_{t-1} \alpha_{0t-2} - b_t \left[\alpha_{t-1} + \alpha_{0t-2}\right]^2$$  \hspace{1cm} (54)

$$= h_{t-2} + (a_t - b_t) \alpha_{t-1}^2 - (2b_t + 1) \alpha_{t-1} \alpha_{0t-2} - b_t \alpha_{0t-2}^2$$

Again, taking the partial derivative with respect to $\alpha_{t-1}$ and setting it to be zero, we get

$$H_t \geq h_{t-2} - \frac{b_t + a_t b_t + 1/4}{a_t - b_t} \alpha_{0t-2}^2.$$  \hspace{1cm} (55)

Recursively, as long as $b_i \geq 0$ and $a_i - b_i \geq 0$ hold for all $i \leq t \in \mathbb{N}^+$, we can repeatedly apply the previous procedure and get

$$H_t \geq h_{t-1} - b_1 \alpha_{0t-1}^2 \geq h_{t-2} - b_2 \alpha_{0t-2}^2 \geq \cdots \geq a_t \alpha_0^2 - b_t \alpha_0^2 \geq 0,$$  \hspace{1cm} (56)

where

$$b_{i+1} = \frac{b_i + a_i b_i + 1/4}{a_i - b_i}, \quad i = 0, \ldots, t-1,$$  \hspace{1cm} (57)

$b_0 = 0$ and $a_i$ is defined in Equation (50). Next, let us prove $b_i \geq 0$ and $a_i - b_i \geq 0$ for all $i \leq t \in \mathbb{N}^+$. 

28
First, it is easy to manually verify that \( b_i \geq 0 \) and \( a_i - b_i \geq 0 \) when \( i < 15 \). In addition, for \( i = 15 \), we can verify that the following inequalities holds

\[
a_i - b_i \geq 4(b_i + \frac{1}{2})^2, \quad b_i \leq \frac{1}{4}i,
\]

since \( a_{15} = 121 \) and \( b_{15} \approx 3.6945 \).

Next, I prove the inequalities in (58) holds for \( i > 15 \). Note that

\[
b_{i+1} - b_i = \frac{b_i + a_i b_i + 1/4}{a_i - b_i} - b_i = \frac{(b_i + \frac{1}{2})^2}{a_i - b_i},
\]

\[
a_{i+1} - a_i = i + 1.
\]

Therefore, we have

\[
b_{i+1} = b_i + \frac{(b_i + \frac{1}{2})^2}{a_i - b_i} \leq b_i + \frac{1}{4} \leq \frac{1}{4}(i + 1),
\]

and

\[
a_{i+1} - b_{i+1} = i + 1 + a_i - b_i - \frac{(b_i + \frac{1}{2})^2}{a_i - b_i}
\]

\[
\geq i + 1 + 4(b_i + \frac{1}{2})^2 - \frac{1}{4}
\]

\[
\geq i + 1 + 4\left(b_{i+1} + \frac{1}{2} - \frac{1}{4}\right)^2 - \frac{1}{4}
\]

\[
= 4(b_{i+1} + \frac{1}{2})^2 - 2b_{i+1} + i
\]

\[
\geq 4(b_{i+1} + \frac{1}{2})^2 - \frac{1}{2}(i + 1) + i \quad (i > 15)
\]

\[
\geq 4(b_{i+1} + \frac{1}{2})^2.
\]

Based on Equations (58), (60) and (61), it can be proven that \( b_i \geq 0 \) and \( a_i - b_i \geq 0 \) hold for \( i \geq 15 \) using mathematical induction. Combining the fact that they also hold for \( i < 15 \), we have \( b_i \geq 0 \) and \( a_i - b_i \geq 0 \) for all \( i \leq t \in \mathbb{N}^+ \). As a result, the inequality (56) holds, i.e., \( H_t \geq 0 \), which concludes the proof.

Now, the new bound is formally given as the following theorem [47].
Theorem 2.4.5. For any stochastic policies $\pi', \pi$ and discount factor $\gamma \in [0.5, 1)$, the following bound holds:

$$|J(\pi') - L_\pi(\pi')| \leq \frac{1}{1 - \gamma} C_\pi \mathbb{E}_{s \sim d^\pi} [D_{KL}[\pi'||\pi](s)],$$

where $C_\pi = \frac{\gamma^2 \epsilon}{(1 - \gamma)^3}$, $\epsilon = \max_{s,a} |A_\pi(s, a)|$.

Proof. Define $A(s)$ to be the expected advantage of $\pi'$ over $\pi$ at state $s$:

$$A(s) = \mathbb{E}_{a \sim \pi'(s)} [A_\pi(s, a)]$$

Then, Lemma 2.4.2 can be rewritten as follows:

$$J(\pi') = J(\pi) + \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^\pi'} [A(s)] = J(\pi) + \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim p_{\pi'}^t} [A(s_t)].$$

Note that the surrogate model can be written as

$$L_\pi(\pi') = J(\pi) + \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^\pi} [A(s)] = J(\pi) + \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim p_{\pi}^t} [A(s_t)].$$

Then, the difference between the surrogate $L_\pi(\pi')$ and the true objective $J(\pi')$ can be written as

$$J(\pi') - L_\pi(\pi') = \sum_{t=0}^{\infty} \gamma^t \left[ \mathbb{E}_{s_t \sim p_{\pi'}^t} [A(s_t)] - \mathbb{E}_{s_t \sim p_{\pi}^t} [A(s_t)] \right]$$

$$= \sum_{t=1}^{\infty} \gamma^t \left[ \mathbb{E}_{s_t \sim p_{\pi'}^t} [A(s_t)] - \mathbb{E}_{s_t \sim p_{\pi}^t} [A(s_t)] \right].$$

(since $p_0^{\pi'} = p_0^\pi = \rho_0$)

Next, the proof is split into three parts. (1) By using the coupling technique, the difference terms in (66), i.e. $\mathbb{E}_{s_t \sim p_{\pi'}^t} [A(s_t)] - \mathbb{E}_{s_t \sim p_{\pi}^t} [A(s_t)]$, is decomposed to derive an equivalent expression. (2) Based on the result from the first part, an upper bound of $|J(\pi') - L_\pi(\pi')|$ that depends on a bunch of state-dependent total variation distances of $\pi', \pi$ is derived using Lemma 2.4.3. (3) the bound derived from the second part is related to the expected KL-Divergence between $\pi', \pi$.  

30
i) The first part of the proof is given as follows.

The techniques from the proof of Lemma 3 in [25] will be used to measure the coincidence of two trajectories resulted from $\pi', \pi$ before an arbitrary timestep $t$. Let $n_t$ denote the number of times that $a'_i \neq a_i | s_i$ at state $s_i$ for $i < t$. For instance, $n_t = 0$ means the trajectories $\tau, \tau'$ completely match before timestep $t$, i.e., $a'_i = a_i | s_i$ for all $i < t$.

The expected advantage at state $s_t$ on the trajectory $\tau' \sim \pi'$ decomposes as follows:

$$E_{s_t \sim \rho_t}^{\pi'} [\overline{A}(s_t)] = P(n_t = 0)E_{s_t \sim \rho_t^t | n_t=0} \overline{A}(s_t) + P(n_t > 0)E_{s_t \sim \rho_t^t | n_t>0} \overline{A}(s_t).$$  \hspace{1cm} (67)$$

The expected advantage on the trajectory $\tau \sim \pi$ decomposes similarly:

$$E_{s_t \sim \rho_t}^{\pi} [\overline{A}(s_t)] = P(n_t = 0)E_{s_t \sim \rho_t^t | n_t=0} \overline{A}(s_t) + P(n_t > 0)E_{s_t \sim \rho_t^t | n_t>0} \overline{A}(s_t).$$  \hspace{1cm} (68)$$

Subtracting Equation (68) from (67), we get

$$E_{s_t \sim \rho_t}^{\pi'} [\overline{A}(s_t)] - E_{s_t \sim \rho_t}^{\pi} [\overline{A}(s_t)] = P(n_t > 0) \left( E_{s_t \sim \rho_t^t | n_t>0} \overline{A}(s_t) - E_{s_t \sim \rho_t^t | n_t=0} \overline{A}(s_t) \right),$$  \hspace{1cm} (69)$$

because $E_{s_t \sim \rho_t^t | n_t=0} \overline{A}(s_t) = E_{s_t \sim \rho_t^t | n_t>0} \overline{A}(s_t)$ when $n_t = 0$.

Note that

$$n_t > 0 \Rightarrow \begin{cases} n_{t-1} = 0 \text{ and } a'_{t-1} \neq a_{t-1} | s_{t-1} \text{ for every } s_{t-1}, \text{ or} \\ n_{t-1} > 0, \end{cases}$$  \hspace{1cm} (70)$$

so we have

$$P(n_t > 0) = P(n_{t-1} = 0) \cdot E_{s_{t-1} \sim \rho_{t-1}^t} \left[ P(a'_{t-1} \neq a_{t-1} | s_{t-1}) \right] + P(n_{t-1} > 0).$$  \hspace{1cm} (71)$$

In a recursive form, it can be expressed as:

$$P(n_t > 0) = \sum_{i=0}^{t-1} P(n_i = 0)E_{s_i \sim \rho_i^t} \left[ P(a'_i \neq a_i | s_i) \right]$$  \hspace{1cm} (72)$$
Then, Equation (75) can be expressed as
\[
E_{s_t \sim \rho_t^n} \left[ \mathcal{A}(s_t) \right] - E_{s_t \sim \rho_t^n} \left[ \overline{\mathcal{A}}(s_t) \right]
= \sum_{i=0}^{t-1} P(n_i = 0)E_{s_t \sim \rho_t^n} \left[ P(a_i' \neq a_i | s_i) \right] \left( E_{s_t \sim \rho_t^n | n_i=0} \left[ \mathcal{A}(s_t) \right] - E_{s_t \sim \rho_t^n | n_i=0} \left[ \overline{\mathcal{A}}(s_t) \right] \right).
\]

(73)

Note that
\[
E_{s_t \sim \rho_t^n | n_i=0} \left[ \mathcal{A}(s_t) \right] - E_{s_t \sim \rho_t^n | n_i=0} \left[ \overline{\mathcal{A}}(s_t) \right]
= \int_S \left[ \rho_t^n(s_t) - \rho_t^n(s_t) \right]_{n_i=0} \mathcal{A}(s_t) ds_t
= \int_S \left( \int_S [\rho_t^n(s_i) \nu_{t-i}^n(s_t | s_i) - \rho_t^n(s_i) \nu_{t-i}^n(s_t | s_i)]_{n_i=0} ds_i \right) \mathcal{A}(s_t) ds_t
= \int_S \rho_t^n(s_i) \left[ \nu_{t-i}^n(s_t | s_i) - \nu_{t-i}^n(s_t | s_i) \right] \mathcal{A}(s_t) ds_i ds_t. \quad \text{(since } \rho_t^n = \rho_t^n \text{ when } n_i = 0)\]
\[
= \int_S \rho_t^n(s_i) \delta^{t-i}(s_t | s_i) \mathcal{A}(s_t) ds_i ds_t \quad \text{(let } \delta^{t-i}(s_t | s_i) = \nu_{t-i}^n(s_t | s_i) - \nu_{t-i}^n(s_t | s_i) )\]
\[
= \int_S \rho_t^n(s) \delta^{t-i}(s' | s) \overline{\mathcal{A}}(s') ds ds'.
\]

(74)

Substituting (74) into (73), we get
\[
E_{s_t \sim \rho_t^n} \left[ \mathcal{A}(s_t) \right] - E_{s_t \sim \rho_t^n} \left[ \overline{\mathcal{A}}(s_t) \right]
= \sum_{i=0}^{t-1} P(n_i = 0)E_{s_t \sim \rho_t^n} \left[ P(a_i' \neq a_i | s_i) \right] \int_S \rho_t^n(s) \delta^{t-i}(s' | s) \overline{\mathcal{A}}(s') ds ds'.
\]

(75)

For notational simplicity, denote
\[
P_{n_i=0 \bar{P}_{a_i' \neq a_i}} := P(n_i = 0)E_{s_t \sim \rho_t^n} \left[ P(a_i' \neq a_i | s_i) \right].
\]

(76)

Then, Equation (75) can be expressed as
\[
E_{s_t \sim \rho_t^n} \left[ \mathcal{A}(s_t) \right] - E_{s_t \sim \rho_t^n} \left[ \overline{\mathcal{A}}(s_t) \right] = \sum_{i=0}^{t-1} P_{n_i=0 \bar{P}_{a_i' \neq a_i}} \int_S \rho_t^n(s) \delta^{t-i}(s' | s) \overline{\mathcal{A}}(s') ds ds'.
\]

(77)

ii) The second part of the proof is given as follows.
Substituting (77) into (66), we get

\[
J(\pi') - L_\pi(\pi')
\]

\[
= \sum_{t=1}^\infty \sum_{i=0}^{t-1} P_{n_t=0} P_{a_i' \neq a_i} \int_S \int_S J_1(0) \delta'(s') \delta(s) d\pi(s) d\pi(s')
\]

\[
= \left( P_{n_0=0} P_{a_0' \neq a_0} \int_S \int_S \gamma \rho_0^\pi(s) \delta'(s') \delta(s) d\pi(s) d\pi(s') \right) + \left( P_{n_0=0} P_{a_0' \neq a_0} \int_S \int_S \gamma^2 \rho_0^\pi(s) \delta'(s') \delta(s) d\pi(s) d\pi(s') \right) + \cdots
\]

\[
= \frac{1}{1 - \gamma} \sum_{t=0}^\infty P_{n_t=0} P_{a_t' \neq a_t} \int_S \int_S \gamma^t \rho_0^\pi(s) [\mu_\pi(s') - \mu_\pi(s)] d\pi(s) d\pi(s').
\]

Taking absolute values on both sides and applying Hölder’s inequality, we get

\[
|J(\pi') - L_\pi(\pi')|
\]

\[
\leq \frac{1}{1 - \gamma} \sum_{t=0}^\infty P_{n_t=0} P_{a_t' \neq a_t} \int_S \int_S \gamma^t \rho_0^\pi(s) [\mu_\pi(s') - \mu_\pi(s)] d\pi(s) d\pi(s')
\]

\[
\leq \frac{1}{1 - \gamma} \sum_{t=0}^\infty P_{n_t=0} P_{a_t' \neq a_t} \int_S \gamma^t \rho_0^\pi(s) \int_S |\mu_\pi(s') - \mu_\pi(s)| d\pi(s') d\pi(s) \max_{s', a'} |A_\pi(s', a')|
\]

Applying Lemma 2.4.3, we have

\[
|J(\pi') - L_\pi(\pi')|
\]

\[
\leq \frac{2\gamma^2}{(1 - \gamma)^2} \sum_{t=0}^\infty P_{n_t=0} P_{a_t' \neq a_t} \int_S \int_S \gamma^t \rho_0^\pi(s) \mu_\pi(s') d\pi(s') \mu_\pi(s) d\pi(s) D_{TV}[\pi || \pi'] d\pi(s') d\pi(s).
\]
Note that the integral part in the above inequality can be expressed as

$$
\int_{S \times S} \gamma \rho_t^\pi(s) \mu_\pi(s'|s) D_{TV}[\pi'||\pi](s') ds ds'
= \int_S \left( \int_S \gamma \rho_t^\pi(s) \mu_\pi(s'|s) ds \right) D_{TV}[\pi'||\pi](s') ds' \quad \text{(See the definition of } \mu_\pi \text{ in (21).)}
= \int_S \left( d_\pi(s') - (1 - \gamma) \sum_{i=0}^{t-1} \gamma^i \rho_t^\pi(s') \right) D_{TV}[\pi'||\pi](s') ds' \quad \text{(for all } t > 0)$$

$$= \mathbb{E}_{s' \sim d_\pi} [D_{TV}[\pi'||\pi](s')] - (1 - \gamma) \sum_{i=0}^{t-1} \gamma^i \mathbb{E}_{s' \sim \rho_t^\pi} [D_{TV}[\pi'||\pi](s')] \quad \text{(for all } t > 0)$$

$$= \mathbb{E}_{s \sim d_\pi} [D_{TV}[\pi'||\pi](s)] - (1 - \gamma) \sum_{i=0}^{t-1} \gamma^i \mathbb{E}_{s \sim \rho_t^\pi} [D_{TV}[\pi'||\pi](s)] \quad \text{(for all } t > 0)$$

(81)

In the following, all total variations $D_{TV}[\pi'||\pi](s)$ will be replaced with $\alpha(s)$ (see Definition 2.4.1) and use the following notations for simplicity:

$$\overline{\alpha} := \mathbb{E}_{s \sim d_\pi}[\alpha(s)], \quad \overline{\alpha}_i := \mathbb{E}_{s \sim \rho_t^\pi}[\alpha(s)].$$

(82)

Plugging (81) into (80), we have

$$|J(\pi') - L_\pi(\pi')|$$

$$\leq \frac{2\gamma^2 \epsilon}{(1 - \gamma)^2} \left[ \sum_{t=0}^{\infty} P_{n_t=0} P_{a_i^t \neq a_i} \cdot \overline{\alpha} - (1 - \gamma) \sum_{t=1}^{\infty} \sum_{i=0}^{t-1} P_{n_t=0} P_{a_i^t \neq a_i} \cdot \gamma^i \overline{\alpha}_i \right]$$

(83)

Using Equations (72) and (76), we have

$$\sum_{t=1}^{k-1} P_{n_t=0} P_{a_i^t \neq a_i} = P[n_k > 0] \to 1 \quad \text{when } k \to \infty.$$  

(84)

Therefore, the first term in the parentheses on the rightest-hand side of (83) is just

$$\sum_{t=1}^{\infty} P_{n_t=0} P_{a_i^t \neq a_i} \cdot \overline{\alpha} = \overline{\alpha}.$$  

(85)

The second term in the parentheses on the rightest-hand side of (83) can be ex-
pressed as

\[(1 - \gamma) \sum_{t=1}^{\infty} \sum_{i=0}^{t-1} P_{nt=0} P_{a_i' \neq a_t} \cdot \gamma^i \bar{\alpha}_i \]

\[= (1 - \gamma) \left( P_{n_1=0} P_{a_1' \neq a_1} \cdot \bar{\alpha}_0 + P_{n_2=0} P_{a_2' \neq a_2} \cdot \gamma \bar{\alpha}_1 + P_{n_3=0} P_{a_3' \neq a_3} \cdot \gamma^2 \bar{\alpha}_2 + \ldots \right) \]

\[= (1 - \gamma) \left( \alpha_0 \sum_{t=1}^{\infty} P_{nt=0} P_{a_i' \neq a_t} + \alpha_1 \sum_{t=2}^{\infty} P_{nt=0} P_{a_i' \neq a_t} + \alpha_2 \sum_{t=3}^{\infty} P_{nt=0} P_{a_i' \neq a_t} + \ldots \right) \]

\[= (1 - \gamma) \left( \alpha_0 [1 - P(n_1 > 0)] + \gamma \bar{\alpha}_1 [1 - P(n_2 > 0)] + \gamma^2 \bar{\alpha}_2 [1 - P(n_3 > 0)] + \ldots \right) \]

\[= (1 - \gamma) \left( \sum_{t=0}^{\infty} \gamma^t \bar{\alpha}_t - \sum_{t=0}^{\infty} \gamma^t \bar{\alpha}_t P(n_{t+1} > 0) \right) \]

\[= \bar{\alpha} - (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \bar{\alpha}_t P(n_{t+1} > 0) \] \hspace{1cm} (86)

Substituting (85) and (86) into (83), we get

\[|J(\pi') - L_\pi(\pi')| \leq \frac{2\gamma^2 \epsilon}{1 - \gamma} \sum_{t=0}^{\infty} \gamma^t \bar{\alpha}_t P(n_{t+1} > 0). \] \hspace{1cm} (87)

iii) The third part of the proof is given as follows.

Recall that \(n_i\) denote the number of times that \(a_i' \neq a_i | s_i\) at state \(s_i\) for \(i < t\), and \(n_t = 0\) means that \(a_i' = a_i | s_i\) for all \(i < t\). Based on Definition 2.4.1 (\(\alpha\)-coupled policy), we have \(P(a_i' = a_i | s_i) \geq 1 - \alpha(s_i)\) for every \(s_i\). Thus,

\[P(n_{t+1} > 0) = 1 - P(n_{t+1} = 0) \]

\[= 1 - \prod_{i=0}^{t} \mathbb{E}_{s_i \sim \rho_i} [P(a_i' = a_i | s_i)] \]

\[\leq 1 - \prod_{i=0}^{t} (1 - \bar{\alpha}_i) \] \hspace{1cm} (88)
Substituting (88) and (82) into (87), we have

$$|J(\pi') - L_\pi(\pi')| \leq \frac{2 \gamma^2 \epsilon}{1 - \gamma} \sum_{t=0}^\infty \gamma^t \overline{\alpha}_t \left(1 - \prod_{i=0}^t (1 - \overline{\alpha}_i)\right). \quad (89)$$

Using Lemma 2.4.4, the inequality (89) can be further simplified as

$$|J(\pi') - L_\pi(\pi')| \leq \frac{2 \gamma^2 \epsilon}{(1 - \gamma)^3} \sum_{t=0}^\infty \gamma^t \overline{\alpha}_t^2. \quad (90)$$

Replacing $\overline{\alpha}_t$ with $\mathbb{E}_{s \sim \rho_t}[D_{TV}[\pi'|\pi](s)]$ and applying $\mathbb{E}[X^2] \leq \mathbb{E}[X^2]$, we get

$$|J(\pi') - L_\pi(\pi')| \leq \frac{2 \gamma^2 \epsilon}{(1 - \gamma)^3} \sum_{t=0}^\infty \gamma^t \mathbb{E}_{s \sim \rho_t}[D_{TV}[\pi'|\pi](s)]. \quad (91)$$

Last, applying Pinsker’s inequality, $2D^2_{TV}[\pi' || \pi](s) \leq D_{KL}[\pi' || \pi](s)$, the result follows.

The new bound is tighter in terms of KL-Divergence compared with (29) derived from [25]. While the improvement in tightness is at a cost of $\gamma/(4(1 - \gamma)^2)$, this result directly relates the bound to the expected KL-Divergence $\mathbb{E}_{s \sim \pi}[D_{KL}[\pi' || \pi](s)]$, which closes the gap between theory and practice in TRPO and related algorithms. In addition, the new bound improves prior analysis in the literature, such as [33, 34], in terms of either KL-Divergence or total variation distance.

Furthermore, using this result, a new lower bound of the true objective can be derived:

$$J(\pi') \geq L_{\pi_k}(\pi') - \frac{1}{1 - \gamma} C_\pi \mathbb{E}_{s \sim \rho}[D_{KL}[\pi' || \pi](s)]. \quad (92)$$

Maximizing the lower bound results in a guaranteed improvement of the policy.

More interestingly, an analytical solution to the maximization of the lower bound can be obtained. This leads to a closed-form policy update rule with monotonic improvement guarantee.
Theorem 2.4.6. For any stochastic policies \( \pi_{\text{new}}, \pi_{\text{old}} \) that are continuously differentiable on the state space \( S \), the inequality, \( J(\pi_{\text{new}}) \geq J(\pi_{\text{old}}) \), holds when [47]

\[
\pi_{\text{new}} = \pi_{\text{old}} \cdot \frac{e^{\alpha_{\pi_{\text{old}}}}}{\mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\pi_{\text{old}}}}]}
\] (93)

where \( \alpha_{\pi_{\text{old}}} = A_{\pi_{\text{old}}}/C_{\pi_{\text{old}}} \).

Proof. With Theorem 2.4.5, we can get a lower bound of the objective function \( J(\pi') \) when approximating around \( \pi_{\text{old}} \):

\[
J(\pi') \geq L_{\pi_{\text{old}}} (\pi') - \frac{1}{1 - \gamma} C_{\pi_{\text{old}}} \mathbb{E}_{s \sim d_{\pi_{\text{old}}}} [D_{\text{KL}}[\pi' \parallel \pi_{\text{old}}](s)]
\] (94)

It follows that maximizing the lower bound will give us a new policy that is not worse than \( \pi_{\text{old}} \). To see this, let \( I(\pi') \) denote the lower bound and \( \pi_{\text{new}} \) denote its maximum solution:

\[
I(\pi') = L_{\pi_{\text{old}}} (\pi') - \frac{1}{1 - \gamma} C_{\pi_{\text{old}}} \mathbb{E}_{s \sim d_{\pi_{\text{old}}}} [D_{\text{KL}}[\pi' \parallel \pi_{\text{old}}](s)]
\] (95)

\[
\pi_{\text{new}} = \arg \max_{\pi'} I(\pi')
\] (96)

where \( L_{\pi_{\text{old}}} (\pi') \) is the surrogate model. Then, we have

\[
J(\pi_{\text{new}}) \geq I(\pi_{\text{new}}) \geq I(\pi_{\text{old}}) = J(\pi_{\text{old}}).
\]

Next, the proof shows that the expression of \( \pi_{\text{new}} \) in (133) is a necessary and sufficient condition for the optimal solution of the problem in (96).

Continuous action space

Calculus of variation will be used to derive the analytical expression for \( \pi_{\text{new}} \). Let \( \pi' \in C^1(U) \) be functions defined on \( U = S \times A \). Note that the lower bound
\( I(\pi') \) can be rewritten as follows:

\[
I(\pi') = J(\pi_{\text{old}}) + \frac{1}{1 - \gamma} \int_{S \times A} d^{\pi_{\text{old}}}(s) \left[ \pi'(a|s) A_{\pi_{\text{old}}}(s, a) - C_{\pi_{\text{old}}} \pi'(a|s) \log \frac{\pi'(a|s)}{\pi_{\text{old}}(a|s)} \right] ds da.
\]

(97)

Note that the policy \( \pi' \) should be a probability distribution, which means that it integrates to 1. To ensure that, the following constraint is added:

\[
H(\pi') = \frac{1}{1 - \gamma} \int_S d^{\pi_{\text{old}}}(s) \left[ \int_A \pi'(a|s) da - 1 \right] ds = 0.
\]

(98)

Now, consider all functions in Equations (97) and (98) as variables in function spaces, and define

\[
F(s, a, \pi') = d^{\pi_{\text{old}}} \left( \pi' A_{\pi_{\text{old}}} - C_{\pi_{\text{old}}} \pi' \log \pi' + C_{\pi_{\text{old}}} \pi' \log \pi_{\text{old}} - \lambda \right) = 0,
\]

(99)

\[
G(s, a, \pi') = d^{\pi_{\text{old}}} \pi' - d^{\pi_{\text{old}}} \pi_{\text{old}}.
\]

(100)

Based on Euler-Lagrange equation [49], there must exist a real number \( \lambda \) such that the optimal policy \( \pi^* \) satisfies

\[
\nabla_{\pi'} F(s, a, \pi') - \lambda \nabla_{\pi'} G(s, a, \pi') = 0,
\]

(101)

where \( \lambda \) is the Lagrange multiplier. Solving Equation (101), we have

\[
d^{\pi_{\text{old}}} \left( A_{\pi_{\text{old}}} \log \pi^* - C_{\pi_{\text{old}}} \log \pi_{\text{old}} - \lambda \right) = 0,
\]

(102)

and

\[
\pi_{\text{new}} = \pi_{\text{old}} \cdot \exp \left\{ \frac{A_{\pi_{\text{old}}}}{C_{\pi_{\text{old}}}} - 1 - \frac{\lambda}{C_{\pi_{\text{old}}}} \right\}.
\]

(103)

Since \( \pi' \) integrates to 1, we have

\[
\int_A \pi_{\text{new}}(a|s) da = \int_A \pi_{\text{old}}(a|s) \exp \left\{ \frac{A_{\pi_{\text{old}}}(s, a)}{C_{\pi_{\text{old}}}} - 1 - \frac{\lambda}{C_{\pi_{\text{old}}}} \right\} da
\]

\[
= e^{-1-\lambda/C_{\pi_{\text{old}}}} \int_A \pi_{\text{old}}(a|s) \exp \left\{ \frac{A_{\pi_{\text{old}}}(s, a)}{C_{\pi_{\text{old}}}} \right\} da
\]

(104)

\[
= 1
\]
Rearranging it, we get

$$\int_A \pi_{\text{old}}(a|s) \exp \left\{ \frac{A_{\pi_{\text{old}}}(s,a)}{C_{\pi_{\text{old}}}} \right\} da = e^{1+\lambda/C_{\pi_{\text{old}}}}. \quad (105)$$

Taking logarithm on both sides and rearranging it, we get

$$\lambda = C_{\pi_{\text{old}}} \log \int_A \pi_{\text{old}}(a|s) \exp \left\{ \frac{A_{\pi_{\text{old}}}(s,a)}{C_{\pi_{\text{old}}}} \right\} da - C_{\pi_{\text{old}}}. \quad (106)$$

Substituting (106) into (103), we get

$$\pi_{\text{new}}(a|s) = \pi_{\text{old}} \cdot \exp \left\{ \frac{A_{\pi_{\text{old}}}(s,a)}{C_{\pi_{\text{old}}}} - \log \int_A \pi_{\text{old}}(a|s) \exp \left\{ \frac{A_{\pi_{\text{old}}}(s,a)}{C_{\pi_{\text{old}}}} \right\} da \right\}. \quad (107)$$

Denote $\alpha_{\pi_{\text{old}}} = \frac{A_{\pi_{\text{old}}}(s,a)}{C_{\pi_{\text{old}}}}$. Then, the optimal policy can be simplified as

$$\pi_{\text{new}} = \pi_{\text{old}} \cdot e^{\alpha_{\pi_{\text{old}}}} \mathbb{E}_{a \sim \pi_{\text{old}}} \left[ e^{\alpha_{\pi_{\text{old}}}} \right]. \quad (108)$$

Until now, the sufficient condition has been proved. Next, I prove that the policy $\pi_{\text{new}}$ in Eq. (108) is also the necessary condition for the optimal solution to the maximization of $I(\pi')$.

Consider weak variations $\epsilon \eta$ such that $\pi' = \pi_{\text{new}} + \epsilon \eta$, where $\eta \in C^1(U)$ and $\epsilon$ is a real number. The second variation can be expressed as,

$$\delta^2 I = \frac{\epsilon^2}{1-\gamma} \int_{S \times A} \nabla^2_{\pi' \pi'} F(s,a,\pi') \eta^2 dsda$$

$$= \frac{\epsilon^2}{1-\gamma} \int_{S \times A} \frac{d_{\pi_{\text{old}}} C_{\pi_{\text{old}}}}{\pi'} \eta^2 dsda \quad (109)$$

$$\leq 0 \quad \text{(for all weak variations } \eta \text{)}$$

because $C_{\pi_{\text{old}}} \geq 0$, and $d_{\pi_{\text{old}}}, \pi'$ are probability distributions and thus always greater or equal to 0. Based on second-variation condition [50], the functional $I(\pi')$ reaches a maximum at $\pi_{\text{new}}$. 

39
Discrete action space

For discrete actions, the functionals \( I(\pi') \) and \( H(\pi') \) can be rewritten as follows:

\[
I(\pi') = J(\pi_{\text{old}}) + \frac{1}{1 - \gamma} \int_S d\pi_{\text{old}}(s) \sum_{i=1}^{k} \left[ \pi'(a_i|s) A_{\pi_{\text{old}}}(s, a_i) - C_{\pi_{\text{old}}} \pi'(a_i|s) \log \frac{\pi'(a_i|s)}{\pi_{\text{old}}(a_i|s)} \right] ds,
\]

\[ (110) \]

\[
H(\pi') = \frac{1}{1 - \gamma} \int_S d\pi_{\text{old}}(s) \left[ \sum_{i=1}^{k} \pi'(a_i|s) - 1 \right] ds.
\]

\[ (111) \]

Now, consider the policy as a vector of functions, \( \pi' = [\pi'_1, \pi'_2, \ldots, \pi'_k] \), where \( \pi'_i = \pi'(a_i|s) \in C^1(S) \) is a function defined on \( S \) given the action \( a_i \). Then, define the Lagrange functions by

\[
F(s, a, \pi') = d\pi_{\text{old}} \sum_{i=1}^{k} \left( \pi'_i A_{\pi_{\text{old}}} - C_{\pi_{\text{old}}} \pi'_i \log \pi'_i + C_{\pi_{\text{old}}} \pi'_i \log \pi_{\text{old}} \right).
\]

\[ (112) \]

\[
G(s, a, \pi') = \sum_{i=1}^{k} \pi'_i - 1.
\]

\[ (113) \]

The Euler-Lagrange Equation (101) becomes

\[
\nabla_{\pi'_i} F(s, a, \pi') - \lambda \nabla_{\pi'_i} G(s, a, \pi') = 0, \quad \forall i \in \{1, \ldots, k\}.
\]

\[ (114) \]

Solving the Euler-Lagrange Equations (114), we get

\[
\pi'^*_i = \pi_{i,\text{old}} \cdot \exp \left\{ \frac{A_{\pi_{\text{old}}}}{C_{\pi_{\text{old}}}} - 1 - \frac{\lambda}{C_{\pi_{\text{old}}}} \right\}, \quad \forall i \in \{1, \ldots, k\}.
\]

\[ (115) \]

Note that \( \pi'^*_i \) should satisfy

\[
\sum_{i=1}^{k} \pi'^*_i = 1.
\]

\[ (116) \]

Then, the Lagrange multiplier \( \lambda \) can be calculated by:

\[
\lambda = C_{\pi_{\text{old}}} \log \sum_{i=1}^{k} \pi_{i,\text{old}}(a_i|s) \exp \left\{ \frac{A_{\pi_{\text{old}}}(s, a_i)}{C_{\pi_{\text{old}}}} \right\} - C_{\pi_{\text{old}}}.
\]

\[ (117) \]
Substituting (117) into (115) and use the vector form, we get

\[
\pi^* = \pi_{\text{new}} = \pi_{\text{old}} \cdot \frac{e^{\alpha_{\text{old}}}}{\mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\text{old}}}]}.
\] (118)

Use the same method as in Equation (109), we can prove that the second-variation condition is satisfied.

2.4.2 Monotonic Improvement Guarantee for Multi-Agent RL

Another interesting result of Theorem 2.4.6 is that the update rule immediately extends to cooperative multi-agent RL problems while the monotonic improvement guarantee still holds if the agents perform local policy updates in turn. This result is presented in the following corollary.

**Corollary 2.4.7.** For any stochastic policies \( \pi_{\text{new}}^i, \pi_{\text{old}}^i \) of agent \( i \) that are continuously differentiable on the local observation space \( \mathcal{O}^i \), and the corresponding joint policies \( \pi_{\text{new}}, \pi_{\text{old}} \), the inequality, \( J(\pi_{\text{new}}) \geq J(\pi_{\text{old}}) \), holds when \[47\]

\[
\pi_{\text{new}}^i = \pi_{\text{old}}^i \cdot \frac{e^{\alpha_{\text{old}}}}{\mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\text{old}}}]},
\]

\[
\pi_{\text{new}}^{-i} = \pi_{\text{old}}^{-i},
\]

where \( \pi_{\text{new}}^{-i}, \pi_{\text{old}}^{-i} \) are joint policies of all agents except \( i \).

**Proof.** Based on Theorem 2.4.6, we have

\[
\pi_{\text{new}}(a|s) = \pi_{\text{old}}(a|s) \cdot \frac{e^{\alpha_{\text{old}}}}{\mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\text{old}}}]}.
\] (120)

Note that the joint policy can be decomposed as follows:

\[
\pi(a|s) = \pi^i(a^i|s)\pi^{-i}(a^{-i}|s),
\] (121)

where \( \pi^i(a^i|s) = \int_{\mathcal{O}^i} \pi^i(a^i|o^i)P_o^i(o^i|s)do^i. \) Thus, Eq. (120) can be rewritten as follows:

\[
\pi_{\text{new}}^{-i} \int_{\mathcal{O}^i} \pi_{\text{new}}^i(a^i|o^i)P_o^i(o^i|s)do^i
\]

\[
= \pi_{\text{old}}^{-i} \int_{\mathcal{O}^i} \pi_{\text{old}}^i(a^i|o^i)P_o^i(o^i|s)do^i \cdot \frac{e^{\alpha_{\text{old}}}}{\mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\text{old}}}]};
\] (122)
when \( \pi_{-i}^{\text{new}} = \pi_{-i}^{\text{old}} \), they cancel each other on both sides. Then, simplifying the above equation, the result follows.

\[ \pi_{-i}^{\text{new}} = \pi_{-i}^{\text{old}} \]

### 2.5 Connections with Prior Work

This section connects the closed-form policy update rule with some state-of-the-art algorithms and discuss how the update rule can help explain these algorithms from a different perspective.

**TRPO and PPO**

Note that the exponential factor in (133) can be written as

\[
\alpha_{\pi_{\text{old}}} = \frac{A_{\pi_{\text{old}}}(s, a)}{\max_{s, a} |A_{\pi_{\text{old}}}(s, a)|} \cdot \frac{(1 - \gamma)^3}{\gamma^2},
\]

where \( \gamma \in [0.5, 1) \). The first term on the right-hand side is a normalized advantage and the second term is a positive constant smaller than 1. Letting \([\alpha_{\min}, \alpha_{\max}]\) denote the range of \( \alpha_{\pi_{\text{old}}} \), then we have \( \alpha_{\min} \leq 0 \leq \alpha_{\max} \), as shown in Figure 3. In addition, since \( \alpha_{\pi_{\text{old}}} \) is a random variable given \( s \), we have \( e^{\alpha_{\min}} \leq \mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\pi_{\text{old}}}}] \leq e^{\alpha_{\max}} \). Then, based on the update rule (133), the ratio of the new policy to the old policy is bounded by

\[
\frac{\pi_{\text{new}}}{\pi_{\text{old}}} \in \left[ \frac{e^{\alpha_{\min}}}{Z}, \frac{e^{\alpha_{\max}}}{Z} \right] = [1 - \epsilon_1, 1 + \epsilon_2],
\]

where \( Z = \mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\pi_{\text{old}}}}] \) and \( \epsilon_1, \epsilon_2 \) are positive numbers (\( \epsilon_1 < 1 \)). Equation (124) indicates that bounding the policy ratio is an effective way to confine the searching area. This helps explain the success of the PPO algorithm [26], which clips the policy ratio by \([1 - \epsilon, 1 + \epsilon], 0 < \epsilon < 1 \).

It is also noted that the policy ratio \( \pi_{\text{new}}/\pi_{\text{old}} \) will be greater than 1 if \( e^{\alpha_{\pi_{\text{old}}}} > \mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\pi_{\text{old}}}}] \), and vice versa (shown in Figure 3). Note that the exponential term \( e^{\alpha_{\pi_{\text{old}}}} \) is monotonically increasing with respect to \( A_{\pi_{\text{old}}}(s, a) \), and so is the policy ratio \( \pi_{\text{new}}/\pi_{\text{old}} \). Less rigorously, consider the term \( \mathbb{E}_{a \sim \pi_{\text{old}}} [e^{\alpha_{\pi_{\text{old}}}}] \) as an “average”
advantage of the policy $\pi_{\text{old}}$. Then, selecting the action $a$ at state $s$ is encouraged, i.e. $\pi_{\text{new}}(a|s) > \pi_{\text{old}}(a|s)$, if it leads to an advantage that is above average. On the contrary, selecting the action $a$ at state $s$ is discouraged, i.e. $\pi_{\text{new}}(a|s) < \pi_{\text{old}}(a|s)$, if it leads to an advantage that is below average. To what extent the action $a$ is encouraged or discouraged is determined by the value of $A_{\pi_{\text{old}}}(s, a)$. This result
matches the TRPO algorithm [25], which maximizes
\[
\max _{\pi} \mathbb{E}_{s \sim d^\text{old}, a \sim \pi^\text{old}} \left[ \frac{\pi(a|s)}{\pi^\text{old}(a|s)} A^\text{old}(s, a) \right],
\] (125)
where \(\pi(a|s)\) is increased to gain weights for large advantages and decreased to lose weights for small advantages. Although the proposed update rule suggests that the policy ratio is proportional to an exponential advantage, rather than a linear advantage as suggested in TRPO and PPO, it is easy to verify that \(e^{A^\text{old}/C^\text{old}} \approx A^\text{old}/C^\text{old} + 1\) when the policy ratio is bounded around 1.

**Value-Based Methods and Dynamic Programming**

The closed-form policy update rule can also connect to value function methods. By multiplying the numerator and denominator both by \(e^{V^\pi^\text{old}(s)/C^\pi^\text{old}}\), the update rule can be rewritten as
\[
\pi^\text{new}(a|s) = \pi^\text{old}(a|s)\frac{\omega^i_j}{\sum_j \pi^\text{old}(a^j|s)\omega^j_j},
\] (126)
\[
\omega^i_j = \exp\{Q^\pi^\text{old}(s, a^i)/C^\pi^\text{old}\}
\]
As shown in (126), the new policy is a weighted probability mass function of the old policy in a softmax form. The weights are the exponential terms, \(\exp\{Q^\pi^\text{old}(s, a^i)/C^\pi^\text{old}\}\). That indicates actions with larger Q values will get better chance to be selected in the future. In fact, the policy update rule can be deemed as a stochastic analogy of the \(\epsilon\)-greedy policy used in value function methods, such as SARSA [12].

In addition, the monotonic improvement guarantee of the policy update rule can be verified through dynamic programming. To see this, note that
\[
V^\pi^\text{old}(s) = \sum_i \pi^\text{old}(a^i|s)Q^\pi^\text{old}(s, a^i)
\leq \sum_i \pi^\text{old}(a^i|s)\frac{\omega^i_j}{\sum_j \pi^\text{old}(a^j|s)\omega^j_j}Q^\pi^\text{old}(s, a^i)
= \sum_i \pi^\text{new}(a^i|s)Q^\pi^\text{old}(s, a^i).
\]
For brevity, let $P_{sa}' := P(s'|s, a)$; then,

$$V_{\pi_{old}}(s) \leq \mathbb{E}_{\pi_{new}} [Q_{\pi_{old}}(s, a)]$$

$$= \mathbb{E}_{\pi_{new}} \left[ r(s, a) + \gamma \mathbb{E}_{P_{sa}'} [V_{\pi_{old}}(s')] \right]$$

$$\leq \mathbb{E}_{\pi_{new}} \left[ r(s, a) + \gamma \mathbb{E}_{P_{sa}'} \left[ \mathbb{E}_{\pi_{new}} [Q_{\pi_{old}}(s', a')] \right] \right]$$

$$\vdots$$

$$\leq \mathbb{E}_{\pi_{new}} \left[ r(s, a) + \gamma \mathbb{E}_{P_{sa}'} [r(s', a')] + \cdots \right]$$

$$= V_{\pi_{new}}(s).$$

Therefore, by applying the update rule (126), a sequence of monotonically improving policies and value functions can be obtained:

$$\pi_0 \rightarrow V_{\pi_0} \rightarrow \pi_1 \rightarrow V_{\pi_1} \rightarrow \cdots \rightarrow \pi_\star \rightarrow V_{\pi_\star},$$

where $V_{\pi_0}(s) \leq V_{\pi_1}(s) \leq \cdots \leq V_{\pi_\star}(s)$ for all $s \in \mathcal{S}$.

**Relative Entropy Policy Search and Maximum a Posterior Policy Optimization**

The REPS (Relative Entropy Policy Search) algorithm [38] can be obtained as a special case of the update rule by replacing $\pi_{old}$ with the observed data distribution and the coefficient $C_{\pi_{old}}$ with the Lagrange multiplier $\eta$. However, the REPS algorithm is based on finite MDPs with discrete actions and not extendable to general continuous policies. A similar closed-form update rule has also been derived in the MPO (Maximum a posterior Policy Optimization) algorithm [43] in its E-step for evaluating a variational policy, which is then used to optimize policy parameters.

The policy update rule is different from the previous work because it directly expresses the new policy as a closed-form function of the current policy. That means the policy update can be accurately calculated using the current policy without involving policy gradient or policy optimization. Especially, the proposed
update rule provides an explicit formula for determining the coefficient $C_{\pi_{\text{old}}}$ and guarantees monotonic improvement on performance. However, the update rules in [38, 43] need to numerically determine the optimal Lagrangian multiplier $\eta$, which requires a costly nonlinear optimization in the inner loop and no monotonic improvement is guaranteed.

**Soft Actor-Critic**

The SAC (Soft Actor-Critic) algorithm [31, 32] can also be derived as a special case of the policy update rule. Note that the update rule (133) can be expressed as a Gibbs measure (Boltzmann distribution in case of discrete actions):

$$
\pi_{\text{new}}(a|s) = \pi_{\text{old}}(a|s) \frac{e^{A\pi_{\text{old}}(s,a)/C_{\pi_{\text{old}}}}}{\mathbb{E}_{a \sim \pi_{\text{old}}} [e^{A\pi_{\text{old}}(s,a)/C_{\pi_{\text{old}}}}]}
$$

$$
= \pi_{\text{old}}(a|s) \frac{e^{Q\pi_{\text{old}}(s,a)/C_{\pi_{\text{old}}}}}{\mathbb{E}_{a \sim \pi_{\text{old}}} [e^{Q\pi_{\text{old}}(s,a)/C_{\pi_{\text{old}}}}]}
$$

$$
= \frac{1}{Z} \exp \left\{ \frac{Q\pi_{\text{old}}(s,a)}{C_{\pi_{\text{old}}}} + \log \pi_{\text{old}}(a|s) \right\},
$$

where $Z = \mathbb{E}_{a \sim \pi_{\text{old}}} [e^{Q\pi_{\text{old}}(s,a)/C_{\pi_{\text{old}}}}]$ is the partition function.

To optimize a policy $\pi$, the KL-Divergence between $\pi$ and $\pi_{\text{new}}$ can be minimized:

$$
\min_{\pi} D_{\text{KL}} \left( \pi(\cdot|s) \left\| \text{exp} \left( \frac{1}{C_{\pi_{\text{old}}}} \widetilde{Q}_{\pi_{\text{old}}}(s,\cdot) \right) \right\| \right),
$$

where $\widetilde{Q}_{\pi_{\text{old}}}$ is the soft Q-function:

$$
\widetilde{Q}_{\pi_{\text{old}}}(s, a) = Q_{\pi_{\text{old}}}(s, a) + C_{\pi_{\text{old}}} \log \pi_{\text{old}}(a|s).
$$

Replacing $C_{\pi_{\text{old}}}$ with a temperature parameter $\alpha$, we immediately get the SAC algorithm.

A slight difference of the algorithm (130) than SAC is that it minimizes the policy entropy instead of maximizing it. Note that the soft state value function derived from the proposed update rule is given by

$$
\tilde{V}_{\pi}(s) = \mathbb{E}_{a \sim \pi} [\widetilde{Q}_{\pi}(s,a)] = V_{\pi}(s) - C_{\pi} H(\pi(\cdot|s)),
$$
where \( H(\pi(s)) \) is the policy entropy. Since \( C_\pi \) is always positive, the policy entropy is penalized in the soft state value function. Thus, applying (130) will minimize the policy entropy. This is reasonable because the policy distribution should be concentrating more and more on the optimal action as the policy improves monotonically.

2.6 Limitations and Discussions

The theoretical results have the following limitations:

- **Tightness of the Bound in Terms of \( \gamma \):** The bound in Theorem 2.4.5 improves prior analysis in terms of KL-Divergence, but not in terms of \( \gamma \), which could be a limitation of the policy update rule. Compared to the bound in TRPO, the improvement is at a cost of \( \gamma/(4(1-\gamma)^2) \). When \( \gamma \) is close to 1, the penalty coefficient \( C_{\pi_{\text{old}}} \) for the KL-Divergence can be large, resulting in small step sizes for policy updates. While \( C_{\pi_{\text{old}}} \) can be tuned to allow larger step-sizes in practice, a proven bound that is tighter in terms of \( \gamma \) will be an interesting direction for future work.

- **Monotonic Guarantee and Function Approximation:** The policy update rule is derived as a closed-form solution, which assumes that both an exact advantage function and an exact maximum of its absolute value are available. However, in large Markov Decision Processes (MDPs), these quantities typically need to be estimated using function approximators, which can introduce errors and undermine the monotonic improvement guarantee. While it is important to acknowledge this limitation, the focus should remain on developing efficient algorithms that leverage function approximation. Despite its assumptions, the policy update rule provides a simple and widely connected framework for RL, and it is expected that new RL theories will
continue to build upon it in the future.

- **Simultaneous Update for Multi-Agent RL** The extension of the update rule to multi-agent RL requires agents to take turns updating their policies. Thus, if there are many agents, the learning process could become slow. From Equation (122), it is clear that this requirement arises because we need to ensure that $\pi_{\text{new}}^{-i}$ is equal to $\pi_{\text{old}}^{-i}$. It would be worthwhile to investigate the possibility of relaxing this requirement so that agents can update their policies simultaneously without compromising the monotonic improvement guarantee in the future.

2.7 Summary

This chapter has presented a closed-form update rule for general stochastic policy optimization with monotonic improvement guarantee. A new theoretical result has been provided by relating the lower bound of the performance to an expected KL-Divergence, which closes the gap between theory and practice in the literature. Based on the theoretical result, calculus of variation has been introduced to derive the policy update rule. Furthermore, it is proven that the policy update rule can extend to cooperative multi-agent RL when agents take turns performing policy updates.
CHAPTER 3

Off-Policy Trust Region Policy Optimization Algorithm

3.1 Overview

Trust-region-based RL algorithms have demonstrated remarkable success in challenging domains, including playing video games [46], controlling robots [25], and training large language models [2], through their combination with high-capacity nonlinear function approximators, such as deep neural networks (DNN). However, the broad adoption of these approaches in real-life applications faces two major obstacles. Firstly, most trust-region based RL methods require a large number of samples, making them inefficient. Even relatively simple tasks can necessitate millions of steps of data collection, and complex environments with high-dimensional observations may require significantly more. Secondly, the performance of these methods is heavily influenced by their hyper-parameters. To obtain good results, a variety of hyper-parameters, including learning rates, trust-region size, and policy ratio boundaries, must be meticulously adjusted for different tasks. Both of these challenges severely restrict the usefulness of these methods in real-world scenarios.

Trust-region based methods suffer from poor sample efficiency, in part, because of on-policy learning. This technique necessitates collecting fresh samples according to the most recent policy for every gradient step, which rapidly becomes prohibitively costly. As task complexity increases, the number of per-update samples required to learn an effective policy also grows. Off-policy algorithms do not suffer from this issue because they can use previous experience to compute gradient steps. While this is relatively simple for value-based approaches such as Q-learning [11], it is typically impractical with traditional policy-based formulations, especially with trust-region based methods.
To overcome this challenge, researchers have attempted to merge off-policy learning with policy-based approaches. For instance, a widely used off-policy RL algorithm, known as DDPG [29], was developed as a deep version of the deterministic policy gradient (DPG) algorithm [24]. DDPG employs an actor-critic design, learning a deterministic actor to maximize an off-policy trained Q-function estimator. However, the interaction between the deterministic actor and the Q-function makes convergence difficult and sensitive to various hyper-parameter settings. In contrast, SAC [31] uses a stochastic actor in the off-policy actor-critic framework, resulting in a significantly more robust and scalable algorithm in practice. A key reason for this improvement is that SAC combines off-policy actor-critic with an entropy maximization [51, 52, 53, 54] objective regarding the stochastic policy that maximizes the actor’s entropy.

This chapter aims to enhance the performance and sample efficiency of trust-region RL methods for continuous control tasks. The objective is to create a stable and sample-efficient trust-region policy optimization algorithm for model-free RL. The key contributions are two-fold. First, building on the closed-form policy update rule, a practical deep RL algorithm is developed, enabling the use of off-policy data for policy optimization with trust-region methods. Second, a detailed implementation of the algorithm is provided for parameterized policies, such as deep neural networks, based on first-order gradients, making it practical for real-world issues. The algorithm’s efficiency is evaluated on a variety of Multi-Joint dynamics with Contact (MuJoCo) robot control tasks. Results indicate that the proposed algorithm effectively solves high-dimensional continuous control tasks and outperforms prior trust-region methods in terms of final return and sample efficiency.
3.2 Related Work

Trust-region RL algorithms have been developed based on various trust region distance measurements. Natural policy gradient [39] uses a quadratic metric over policy parameters to measure the trust region distance, which is calculated using Fisher information matrix. Trust region policy optimization [25] employs KL-Divergence measurement and provides a guarantee of monotonic improvement in policy performance. However, it requires second-order gradients, making it computationally inefficient. Proximal policy optimization [26] approximates the KL distance by clipping the policy ratio in the objective, allowing the use of first-order optimization methods. Differentiable trust-region layers for Gaussian policies are proposed in [44] based on the Wasserstein L2 distance, Frobenius norm, and KL-Divergence, with guarantees of monotonic improvement. However, these methods are typically sample inefficient, requiring a large number of on-policy interactions with the environment, limiting their practical application in real-world scenarios.

To improve the sample efficiency of trust-region methods, off-policy DRL algorithms have been proposed lately. In [41], a trust region path consistency learning algorithm is proposed to exploit off-policy data for policy learning and value function estimation. Considering RL as a variational inference paradigm [42], an off-policy trust-region RL algorithm is proposed in [43] via a maximum-a-posteriori estimation for Gaussian policies using expectation-maximization (EM). In [55], an off-policy trust-region policy optimization algorithm is proposed based on a new surrogate function. In [56], an off-policy DRL algorithm is proposed using mirror descent policy optimization, which projects gradients in a dual space of the policy onto a trust region under a Bregman divergence. In [57], off-policy variants of trust-region policy optimization, proximal policy optimization, and mirror descent policy optimization are developed by constructing a general class of sur-
rogate functions based on functional mirror ascent. Despite the progress made by these results, their ability to compete against other kinds of policy-based DRL algorithms, such as DDPG and SAC, on high-dimensional continuous control tasks is still unsatisfying.

3.3 An Off-Policy Trust-Region RL Algorithm

This section presents a practical off-policy deep reinforcement learning (DRL) algorithm based on trust-region policy optimization methods. The theoretical result from Chapter 2 is reinstated, and a new algorithm based on the theoretical result is designed using parameterized function approximators. A detailed implementation of the algorithm for deep neural network-based policies is also provided.

3.3.1 Closed-Form Solution to Trust-Region Policy Optimization

Recall the closed-form policy update rule (133) in chapter 2, which states that for any continuously differentiable policies $\pi_{k+1}, \pi_k$, we have $J(\pi_{k+1}) \geq J(\pi_k)$ when

$$
\pi_{k+1}(a|s) = \pi_k(a|s) \cdot \frac{e^{\alpha_{\pi_k}(s,a)}}{\mathbb{E}_{a \sim \pi_k} [e^{\alpha_{\pi_k}(s,a)}]}, \forall s, a
$$

(133)

where $\alpha_{\pi_k}(s,a) = A_{\pi_k}(s,a)/C_{\pi_k}$. The policy update rule mentioned earlier guarantees monotonic performance improvement, but it requires the accurate calculation of both the policy $\pi_{k+1}(a|s)$ and the advantage function $A_{\pi_k}(s,a)$ for all $(s,a)$ pairs. In continuous control tasks, function approximators are necessary to represent the policy and estimate the advantage values. Therefore, a practical algorithm based on this policy update rule is proposed.

Differentiable function approximators will be used to represent the policy and estimate advantage values. A parameterized policy $\pi_{\theta}(a|s)$ and a parameterized Q-function $Q_{\phi}(s,a)$ will be utilized for this purpose, with $\theta$ and $\phi$ being the parameters of these differentiable approximators, respectively. As neural networks are effective and commonly used differentiable function approximators, the policy
network and Q-network will be referred to as the parameterized policy and Q-function, respectively. The networks will be updated alternatively using mini-batch stochastic gradient descent (SGD). The Q-network will be employed to estimate the advantage value using the definition provided in Equation (5):

\[ A_{\pi_{\theta}}(s, a) = Q_{\phi}(s, a) - \mathbb{E}_{a \sim \pi_{\theta}}[Q_{\phi}(s, a)]. \]  

The Q-network can be trained to minimize the mean square Bellman error as follows:

\[
J_Q(\phi) = \frac{1}{2} \mathbb{E}_{(s_t, a_t) \sim D} \left[ \left( Q_{\phi}(s_t, a_t) - \hat{Q}(s_t, a_t) \right)^2 \right]
\]

where the target Q-value is

\[
\hat{Q}(s_t, a_t) = r_t + \gamma \mathbb{E} [Q_{\phi}(s_{t+1}, a_{t+1})], a_{t+1} \sim \pi_{\theta}.
\]

In the proposed algorithm, the Q-network is optimized with the mini-batch SGD

\[
\hat{\nabla}_{\phi} J_Q(\phi) = \frac{1}{B} \sum_{b=1}^{B} \nabla_{\phi} Q_{\phi}(s_t^b, a_t^b) \cdot \left( Q_{\phi}(s_t^b, a_t^b) - r_t^b - \gamma Q_{\bar{\phi}}(s_{t+1}^b, a_{t+1}^b) \right),
\]

where \( B \) is the batch size, \( s_t^b, a_t^b, r_t^b \), and \( s_{t+1}^b \) are sampled from a reply buffer \( D \), and \( a_{t+1}^b \) is sampled from \( \pi_{\theta} \). The update makes use of a target Q-function to compute the Bellman error, and the parameters \( \bar{\phi} \) of the target Q-function are obtained as a moving average of the SGD iterates of the Q-function parameters, which has been shown to help stabilize Q-function learning [58]. The update also makes use of two Q-functions to mitigate overestimated value estimates, as proposed in [14]. Specifically, we construct two Q-networks with parameters \( \phi_i, i = 1, 2 \), which are trained independently using the mini-batch SGD in Eq. (137). We also obtain two target Q-functions with \( \bar{\phi}_i, i = 1, 2 \) using Polyak moving averages of the SGD updates of \( \phi_i, i = 1, 2 \). Then, we use the minimum of the target Q-functions as the target Q-value in Eq. (137).
The policy parameters $\theta$ can be learned by minimizing:

$$J_\pi(\theta) = \frac{1}{2} \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_{\theta_k}} [(r_\theta(s_t, a_t) - \hat{r})^2]$$

(138)

where

$$r_\theta(s, a) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, \quad \hat{r} = \frac{e^{\alpha_{\theta_k}(s_t, a_t)}}{\mathbb{E}_{a \sim \pi_{\theta_k}}[e^{\alpha_{\theta_k}(s_t, a_t)}]}.$$ 

The objective (138) aims to balance the closed-form update rule (133) with a mean square error (MSE) loss. To this end, the proposed algorithm makes use of a target policy with parameters $\bar{\theta}$ as the most recent policy $\pi_k$. The policy network is then optimized with the mini-batch SGD [59]:

$$\hat{\nabla}_\theta J_\pi(\theta)_{|\theta = \bar{\theta}} = \frac{1}{B} \sum_{b=1}^{B} \sum_{n=1}^{N} (1 - \hat{r}^b_n) \nabla \log \pi_\theta(a_t^n|s^b_t)$$

(139)

where the target policy ratio $\hat{r}^b_n$ is calculated by

$$\hat{r}^b_n = \frac{\exp\{A_\pi_\theta(s^b_t, a^n_t)/C_{\pi}\}}{\frac{1}{N} \sum_{n=1}^{N} \exp\{A_\pi_\theta(s^b_t, a^n_t)/C_{\pi}\}}.$$ 

(140)

The states $s^b_t$ are sampled from the replay buffer $\mathcal{D}$, and the actions $a^n_t$ are sampled from the most recent policy $\pi_{\bar{\theta}}$ given $s^b_t$. Note that obtaining samples from $\pi_{\bar{\theta}}$ does not require additional interactions with the environment, and therefore we can update the policy in an off-policy manner using (139).

To compute the target policy ratio $\hat{r}^b_n$, we also need the value of $C_{\pi_{\bar{\theta}}}$, which requires the exact value of $\max_{s,a} |A_{\pi_{\bar{\theta}}}(s, a)|$. Computing the exact value of $\max_{s,a} |A_{\pi_{\bar{\theta}}}(s, a)|$ can be challenging for high-dimensional, continuous problems. Besides, if the coefficient $C_{\pi_{\bar{\theta}}}$ recommended by the theory in Theorem 2.4.5 is used, the step sizes would be very small. In the proposed algorithm, the coefficient $C_{\pi_{\bar{\theta}}}$ is approximated by using a sliding window maximum of sampled estimates with a window size $w$:

$$C_{\pi_{\bar{\theta}}} = \max\{C_{k-w}, \ldots, C_{k-1}, C_k\}$$

(141)
where
\[ C_k = \frac{\gamma^2 \epsilon}{(1 - \gamma)^2}, \quad \epsilon = \max\{|A_{\pi^k}(s^b_t, a^n_t)|, \forall b, n\}. \tag{142} \]
where \( \bar{\theta}^k \) is the parameters of the target policy network at the iteration \( k \).

The advantage function is computed by
\[
A_{\pi^k}(s_t, a_t) = Q_\phi(s_t, a_t) - \frac{1}{N} \sum_{n=1}^{N} [Q_\phi(s_t, a^n_t)], a^n_t \sim \pi_{\bar{\theta}}. \tag{143}
\]
The policy parameters are updated by applying \( \theta := \theta - \lambda \pi \nabla_\theta J_\pi(\theta) \), and the target policy parameters are then completely replaced with the most recent iterate of the policy parameters, i.e. \( \bar{\theta} := \theta \).

To ensure bounded actions, a squashed Gaussian is used as the policy distribution, as proposed in [31]. In particular, let \( u \) be a multivariate random variable following the diagonal gaussian distribution \( \mu(u|s) \). Then, the action \( a \) is a function of \( u \):
\[
a = l + \frac{h - l}{2} (\tanh(u) + 1) \tag{144}
\]
where \( \tanh \) is applied elementwise; \( l \) and \( h \) are the lower and upper bound of the action \( a \). Based on distribution function technique, the action distribution can be calculated by \( \pi(a|s) = \mu(a|s) \left| \det \left( \frac{da}{du} \right) \right|^{-1} \). Then, we can calculate the log-likelihood of action \( a \) by
\[
\log \pi(a|s) = \log \mu(a|s) - \sum_{i=1}^{|A|} \log \left( 1 - \tanh^2(u_i) \right) - \sum_{i=1}^{|A|} \log \left( \frac{h_i - l_i}{2} \right) \tag{145}
\]
where \( h_i, l_i, u_i \) are the \( i \)th elements of \( h, l, u \), respectively. The algorithm is summarized in 1.

### 3.4 Case Studies

To verify the effectiveness of the proposed algorithm, experiments are conducted on various robot control tasks using the MuJoCo simulator [60] and interfaced through the OpenAI Gym environment [61]. The objective is to assess
Algorithm 1: An Off-policy Trust-Region DRL Algorithm

Input: Initial parameters $\theta$, $\bar{\theta}$, $\phi_1$, $\phi_2$, $\bar{\phi}_1$, $\bar{\phi}_2$;

for $k = 0, 1, 2, \ldots$ do
  for each environment step do
    $a_t \sim \pi_\theta(\cdot|s_t)$;
    $s_{t+1} \sim P(\cdot|s_t, a_t)$;
    $D \leftarrow D \cup \{s_t, a_t, r_t, s_{t+1}\}$;
  end for
  for each gradient step do
    $\phi_i \leftarrow \phi_i - \lambda Q \hat{\nabla}_\phi J_Q(\phi_i), i \in \{1, 2\}$;
    $\bar{\phi}_i \leftarrow \beta \bar{\phi}_i + (1 - \beta)\phi_i, i \in \{1, 2\}$;
    $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta J_\pi(\theta)$;
    $\bar{\theta} \leftarrow \bar{\theta}$;
  end for
end for

the proposed method’s effectiveness in solving high-dimensional, continuous control
tasks, and to compare its performance and sample efficiency with prior trust-region
DRL methods and state-of-the-art off-policy algorithms.

3.4.1 MuJoCo Environment for Robot Control

In the experiments, six MuJoCo continuous control tasks on robotic loco-
motion are used as benchmarks, including (a) Swimmer-v3, (b) Hopper-v3, (c)
Walker2D-v3, (d) HalfCheetah-v3, (e) Ant-v3, and (f) Humanoid-v3, as shown in
Fig. 4. The state and action spaces of these tasks are continuous, and have varying
dimensions, ranging from 8 states and 2 actions (Swimmer-v3) to 376 states and 17
actions (Humanoid-v3). Some tasks with low dimensions of state and action spaces
are easier to solve, but the high-dimensional benchmarks, such as Humanoid-v3,
are extremely difficult with prior trust-region DRL algorithms.

To demonstrate the superiority of the proposed algorithm, a comparison is
made with three trust-region DRL methods, namely, (1) TRPO, which is a stable
on-policy algorithm based on second-order gradients and a line search procedure for
policy parameters updating; (2) PPO, a variant of TRPO that utilizes first-order
gradients and policy ratio clipping; and (3) Trust-PCL, an off-policy trust-region algorithm that has been shown to perform better and more efficiently than TRPO. Additionally, two different kinds of off-policy DRL algorithms are compared, including (1) DDPG, which is an efficient algorithm for learning deterministic policies in continuous control tasks, and (2) SAC, a state-of-the-art DRL algorithm based on maximum entropy policy optimization.

Both the proposed algorithm and the baselines use a feedforward neural network with two hidden layers of 256 ReLU units to parameterize the policy. Although the value network (for TRPO, PPO, Trust-PCL) and Q-network (for DDPG, SAC, and the proposed method) have the same hidden layer architecture, they differ in their input and output dimensions. The hyperparameters of the proposed method can be found in Table I. TensorFlow [62] is used to conduct the experiments. During training, the policy is trained for 3 million environment steps and evaluated for 10 episodes after every 2000 environment steps. For the best performance, the action of the agent at each step is selected as the mean of the policy distribution during evaluation.

### 3.4.2 Comparison With Prior Trust-Region methods

Figure 5 displays a comparison of the average episode return from 10 evaluation rollouts during training for TRPO, PPO, Trust-PCL, and the proposed method. The solid curves indicate the mean results of each algorithm, while the shaded areas correspond to 95% confidence intervals. To ensure a fair comparison, each algorithm is executed for five times with different random seeds.

Based on the comparison, it is apparent that the proposed algorithm significantly outperforms the baseline trust-region methods in terms of both sample efficiency and final average return. Although TRPO performs relatively well on easier tasks, such as Swimmer-v3 and Hopper-v3, it fails to learn a suitable policy
Figure 4. Six representative MuJoCo robot control tasks [60] used in the evaluation.

Table 1. Parameters of the proposed off-policy trust-region deep reinforcement learning algorithm.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimizer</td>
<td>Adam [63]</td>
</tr>
<tr>
<td>learning rate $\lambda_\pi$</td>
<td>0.001</td>
</tr>
<tr>
<td>learning rate $\lambda_Q$</td>
<td>0.001</td>
</tr>
<tr>
<td>discount factor $\gamma$</td>
<td>0.99</td>
</tr>
<tr>
<td>replay buffer size $</td>
<td>D</td>
</tr>
<tr>
<td>number of hidden layers (all networks)</td>
<td>2</td>
</tr>
<tr>
<td>number of units per hidden layer</td>
<td>256</td>
</tr>
<tr>
<td>activation function</td>
<td>ReLU</td>
</tr>
<tr>
<td>batch size $B$</td>
<td>256</td>
</tr>
<tr>
<td>number of action samples per state $N$</td>
<td>50</td>
</tr>
<tr>
<td>Polyak coefficient $\beta$</td>
<td>0.995</td>
</tr>
<tr>
<td>Sliding window size for estimating $C_{\pi_\theta}$</td>
<td>200</td>
</tr>
</tbody>
</table>
Figure 5. Learning curves of the proposed algorithm in contrast to prior trust-region DRL methods on six representative MuJoCo continuous control tasks. The solid curves represent the means of different algorithms over five random seeds and the shaded areas correspond to 95% confidence intervals.

for harder tasks like HalfCheetah, Ant-v3, and Humanoid-v3. Although Trust-PCL slightly outperforms TRPO and PPO on more high-dimensional tasks like HalfCheetah and Ant-v3, it can only learn sub-optimal policies and fails to solve these tasks. None of the benchmark trust-region methods make any progress on the most complex task, Humanoid-v3.

3.4.3 Comparison With Other Off-Policy methods

Figure 6 depicts the learning curves of the proposed algorithm, DDPG, and SAC, which are two different kinds of off-policy DRL algorithms. Based on the comparison, it is evident that the proposed method performs similarly to the state-of-the-art SAC algorithm on almost all tasks, except for Swimmer-v3, where SAC fails to learn a satisfactory control policy. Moreover, the proposed method outperforms DDPG by a considerable margin on Hopper-v3, Walker2D-v3, and Humanoid-v3, and achieves comparable performance on the remaining
Figure 6. Learning curves of the proposed algorithm in contrast to advanced off-policy DRL methods on six representative MuJoCo continuous control tasks. The solid curves represent the means of different algorithms over five random seeds and the shaded areas correspond to 95% confidence intervals.

3.5 Summary

In this chapter, an off-policy trust-region DRL algorithm is introduced, which enables efficient learning of deep neural network policies while maintaining the stability of trust-region methods. The empirical results on a set of MuJoCo continuous control tasks demonstrate that the proposed approach outperforms prior trust-region deep RL algorithms, including TRPO, PPO, and Trust-PCL, in terms of both final return and sample efficiency. Interestingly, the proposed approach achieves comparable performance to SAC, a state-of-the-art off-policy algorithm, and even outperforms DDPG on most tasks. This highlights the potential of trust-region methods not only to stabilize policy optimization but also to facilitate
sample-efficient learning of high-quality policies for complex continuous control
tasks. These findings suggest that the off-policy trust-region RL algorithm can be
a valuable addition to the existing arsenal of reinforcement learning algorithms,
particularly in the context of continuous control tasks.
CHAPTER 4

Multi-Agent Trust Region Policy Optimization Algorithm

4.1 Overview

Multi-agent reinforcement learning (MARL) is a rapidly growing field that focuses on developing intelligent agents that can interact and cooperate with each other in complex environments. The goal of MARL is to enable agents to learn from their experiences and adapt to their environment to achieve better outcomes. In recent years, MARL has gained significant attention due to its potential applications in various domains, including multiplayer games [64], coordinating self-driving vehicles [65], multi-robot systems [66], traffic signal control [67], and smart grids [68].

Most existing MARL algorithms can be classified into three categories: (1) fully centralized, such as joint action learning [69], which learns a centralized policy by reducing the problem to a single-agent RL over the global observation and action space, (2) centralized training decentralized execution, such as [70], which learns local policies (that select actions based on local observations) in a centralized manner by using global information, such as aggregated observations, joint rewards, policy parameters or gradients of other agents, etc., and (3) fully decentralized, such as independent learning [71], which learns local policies in a decentralized manner, and does not require global information in the training and execution stages.

Centralized training decentralized execution is preferred in many studies since the learned policies can be executed in a distributed way while the nonstationarity issue [72] can be solved using centralized training. Nonstationarity issue arises when the agents learn policies independently. If one agent changes its policy, the state transition and reward perceived by others will change since the agent is part of the environment from the perspective of others. If all agents change policy in-
dependently, the environment perceived by every agent would be non-stationary, which makes it difficult to converge for independent learning. Centralized training solves this issue by sharing local observations, rewards, and policies among the agents. Many centralized training decentralized execution algorithms are developed based on this idea, such as multi-agent deep deterministic policy gradient (MADDPG) [70], value decomposition network (VDN) [73], monotonic value function factorization (QMIX) [74], counterfactual multi agent policy gradient (COMA) [75], multi-agent actor-attention-critic (MAAC) [76], multiagent reinforcement learning for unshaped cooperative scenarios (UNMAS) [77], and so on.

Nevertheless, the centralized training decentralized execution framework exhibits many limitations. First, the existence of an unavoidable central controller makes the framework vulnerable to single-point failure and cyber-attack. Second, training distributed policies in one single controller may cost massive computation and communication resources, raising the issue of scalability and flexibility. Third, collecting agents’ local information may raise concerns about personal information leakage, making it difficult to deploy in real-world applications.

The limitations motivate the design of fully decentralized or distributed policy optimization. However, distributed policy optimization faces the challenge of partial observability: agents may possess incomplete information about the environment state. To improve observability, many approaches adopt different information structures, such as partially nested [78], periodic sharing [79], delayed sharing [80], event trigger [81], etc. In distributed stochastic control, the common information approach is often employed through partial history sharing [82]. An information state is usually used to represent the agent’s posterior belief of the environment state. The conditions under which an information state are sufficient is studied in [83]. In [84], information state embedding is designed by compressing agents’
local histories. However, constructive algorithms are not offered in these studies. In MARL, learnable communication protocols [85] are preferred, which map local information to actionable messages such that agents can improve observability by incorporating neighbors’ messages to augment observations. Various aggregation methods, such as averaging [85], encoding [86], and attention mechanism [87], are studied to incorporate neighbors’ messages efficiently.

This chapter proposes a fully decentralized algorithm for partially observable MARL based on trust-region methods. The proposed algorithm extends TRPO to cooperative MARL problems where agents have partial observations and private rewards, yielding a fully decentralized algorithm, which is named by multi-agent TRPO (MATRPO). In MATRPO, distributed policies are optimized using a distributed asynchronous alternating direction method of multipliers (ADMM) algorithm, which only requires local communication between neighboring agents. During training, each agent shares the ratio of the local policy to its most recent iterate with neighbors for consensus. Agents do not need to transmit their private information on observations, rewards, or policy and value network parameters.

4.2 Related Work

There exists a series of insightful works in the literature that address the decentralized MARL problem, and a comprehensive overview of decentralized MARL methods can be found in [88, 89]. Most decentralized MARL methods fall into two categories: (1) value-base methods [90], which alternate between policy evaluation and policy improvement; (2) policy gradient methods [19], which directly optimizes the policy according to an estimated gradient of the expected return.

For value-based methods in decentralized MARL, the key is to evaluate the global value function in a decentralized manner. A distributed policy evaluation algorithm is proposed in [91] by extending the classic single-agent gradient
temporal-difference (GTD) algorithm to multi-agent systems by using diffusion strategies [92]. A convergence analysis with linear function approximation under sufficiently small step-size updates was established. Following this line of study, in [93] a consensus-based distributed GTD algorithm is studied with a weak convergence analysis under time-varying network topologies using ordinary differential equation [94]. A similar approach formulating the consensus optimization as a primal-dual problem is developed in [95]. In [96], an asymptotic convergence of the distributed TD(0) algorithm for both discounted and average reward MARL problems is established considering gossiping agents [97]. While the convergence or asymptotic convergence is established in these works, analysis about the rate of convergence or finite-time convergence is missing. In [98], a double averaging scheme combining the dynamic consensus and stochastic average gradient algorithm is proposed with a convergence guarantee at a global geometric rate. In [99], the finite-time convergence of a consensus-based distributed TD(0) algorithm under constant and time-varying step-sizes was investigated. Different from the above methods attempting to estimate the value function, the DQ-learning algorithm in [100] is to learn the action-value function by distributed Q-factors, which were updated by using a consensus + innovation algorithm. However, the aforementioned studies are based on linear function approximation, which limits their application in many complicated MARL problems where general non-linear function approximators, such as neural networks, are required. To address this problem, [101] presents a neural network based multi-agent deep Q-network (DQN) algorithm, where distributed optimal action-value functions are learned by aggregating neighboring feature sets for state representation via an attentive relational encoder.

Policy gradient methods enable agents to learn stochastic policies, which are
essential to partially observable environments [27]. Furthermore, policy gradient methods are more stable in learning complicated nonlinear policies since it can directly search in the space of policy parameters. Most policy gradient methods adopt the actor-critic architecture [1]. In [102], two decentralized actor-critic algorithms are designed by using a consensus action-value function approximator and the policy gradient theorem for decentralized MARL is established. While the proposed method guarantees convergence for linear approximate functions, it does not consider the partial observability issue. In [103], an off-policy actor-critic algorithm is studied by learning consensus policies. By the decomposition of the global value function and action-value function, each agent can estimate the gradient of the consensus-based local policy based on a local critic. The convergence guarantee for linear approximate functions was also established for fully observable environments. However, the proposed method is limited to homogeneous policies since all policies must have the same architecture for consensus. In [104], a semi-centralized MARL algorithm is proposed for cooperative multi-agent StarCraft games. A hierarchical structure with two levels of actor-critic is designed where a global actor-critic learns to provide some global signals based on limited information shared by agents and each agent learns to play using local actor-critics. While this method can reduce the communication burden compared with centralized training methods, it still requires a coordinator simultaneously interacting with all agents during training.

Similar to policy gradient methods, the proposed algorithm can train stochastic policies parameterized by large nonlinear function approximators using local policy search with trust region constraints. Compared to the policy gradient approaches mentioned above, the proposed method considers partial observable environments and is suitable for heterogeneous policies. Specifically, in the proposed
algorithm the agents’ policies are improved by solving a consensus optimization in a decentralized way using an asynchronous ADMM. Since the consensus is imposed on the policy ratio rather than policy parameters, the policies can have different architectures and the agents only need to share the policy ratio values with neighbors via a peer-to-peer network. Therefore, the proposed algorithm does not require a central controller or a coordinator to collect and distribute every agent’s private information on reward, observations, or policy and value function parameters.

4.3 Preliminaries

In this section, some important preliminaries that serve as building blocks for understanding cooperative MARL are introduced. Firstly, a model for cooperative MARL is introduced, which involves multiple agents working together to achieve a common goal. Secondly, the TRPO algorithm is discussed. Lastly, the standard ADMM algorithm is introduced, which is a distributed optimization algorithm that can be used in MARL to improve communication efficiency between agents. These preliminaries are essential for gaining a comprehensive understanding of the cooperative MARL framework and its underlying algorithms.

4.3.1 Partially Observable Markov Game

For MARL, the problem can be formulated as a POMG, which is presented in Section 2.3.2 of Chapter 2. In the distributed MARL setting, it is assumed that $N$ agents communicate with immediate neighbors through a peer-to-peer network, which is represented by an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, \ldots, N\}$ is the set of $N$ agents and $\mathcal{E} = \{1, \ldots, L\}$ is the set of $L$ communication links.

Letting $R^i(\tau) = \sum_{t=0}^{\infty} \gamma^t r^i_t$ denote the discounted return of a trajectory perceived by agent $i$, define the local value function as $V^i_\pi(s) = \mathbb{E}_{\tau \sim \pi} [R^i(\tau) \mid s_0 = s]$, local action-value function as $Q^i_\pi(s, a) = \mathbb{E}_{\tau \sim \pi} [R^i(\tau) \mid s_0 = s, a_0 = a]$, and local
advantage function as $A^i_\pi(s, a) = Q^i_\pi(s, a) - V^i_\pi(s)$, respectively. With these definitions, the global value function can be expressed by $V^\pi(s) = \sum_{i=1}^{N} V^i_\pi(s)$, the global action-value function can be expressed by $Q^\pi(s, a) = \sum_{i=1}^{N} Q^i_\pi(s, a)$, and the global advantage function can be expressed by $A^\pi(s, a) = \sum_{i=1}^{N} A^i_\pi(s, a)$.

Also, denote $\rho_\pi$ as the discounted state visitation frequencies according to the joint policy $\pi$:

$$\rho_\pi(s) = P(s_0 = s | \pi) + \gamma P(s_1 = s | \pi) + \gamma^2 P(s_2 = s | \pi) + \cdots .$$

Likewise, denote $\rho^i_\pi$ as the discounted observation visitation frequencies of agent $i$ according to $\pi$:

$$\rho^i_\pi(o) = P(o^i_0 = o | \pi) + \gamma P(o^i_1 = o | \pi) + \gamma^2 P(o^i_2 = o | \pi) + \cdots .$$

### 4.3.2 The TRPO Algorithm

TRPO is a practical trust-region-based algorithm, which tries to optimize a parameterized policy by maximizing the lower bound of the true objective function in Equation (29). However, optimizing the lower bound directly is impractical for high-dimensional or continuous state space since it requires an accurate estimate of the expected value over $a$ $\pi$ in the objective and the maximal KL over the state space in the constraint. In practice, TRPO approximates the optimization by using the following importance sampling objective with a constraint on the expected KL:

$$\max_{\pi} \mathbb{E}_{s \sim \rho^\pi_{old}} \left[ \frac{\pi(a|s)}{\pi^\text{old}(a|s)} A^\text{old}(s, a) \right]$$

$$\text{s.t. } \mathbb{E}_{s \sim \rho^\pi_{old}} [D_{KL}([\pi^\text{old}]|\pi](s)] < \delta,$$

where $D_{KL}^\rho_{\pi^\text{old}}(\pi^\text{old}, \pi) = \mathbb{E}_{s \sim \rho^\pi_{old}} [D_{KL}(\pi^\text{old}(|s|) \| \pi(|s|))]$ is the average KL-Divergence. In the optimization (146), the objective is an importance sampling [105] estimate of the advantage function $A^\text{old}_\pi(s, a)$. The KL constraint restricts the searching area in the neighborhood of $\pi^\text{old}$ via the trust-region parameter $\delta$. 

68
4.3.3 Alternating Direction Method of Multipliers

In this study, the TRPO algorithm is extended to partially observable MARL by transforming the optimization in (146) into a consensus optimization problem, which has the following canonical form:

$$\min_{x,v} f(x) = \sum_{i=1}^{N} f_i(x_i)$$

s.t. $x_i = z, \forall i \in \mathcal{N}$

(147)

where the objective function $f$ is separable with respect to the decision variables $x = (x_1, \ldots, x_N)$, i.e., $f(x) = \sum_{i=1}^{N} f_i(x_i)$; $f_i$ is the objective function with respect to the local variables $x_i \in \mathbb{R}^{m \times 1}$; the constraints $x_i = z$ enforce consensus for all $x_i$ on $z \in \mathbb{R}^{m \times 1}$.

Alternating Direction Method of Multipliers (ADMM) is an effective algorithm to solve problems of such kind with a more general formulation [106]:

$$\min_{x,z} f(x) + g(z)$$

s.t. $Ax + Bz = C$;

(148)

where the matrix $A$ is partitioned to $A = [A_1, \ldots, A_N]$ so that $Ax = \sum_{i=1}^{N} A_i x_i$, $A_i \in \mathbb{R}^{p \times m}$, $B \in \mathbb{R}^{p \times m}$, and $C \in \mathbb{R}^{p \times 1}$; the objective is a sum of two convex functions, $f(x)$ and $g(z)$, with respect to the decision variables $x$ and $z$, respectively.

The ADMM algorithm solves the problem (148) by optimizing the following augmented Lagrangian,

$$\mathcal{L}_\beta(x, z, y) = f(x) + g(z) + y^T (Ax + Bz - C)$$

$$+ \frac{\beta}{2} \| Ax + Bz - C \|_2^2, \beta > 0,$$

(149)

where $y \in \mathbb{R}^{m \times 1}$ is the dual variable or Lagrange multiplier. The optimization is performed through sequentially updating the primal variables $x$ and $z$, and the
dual variable $y$:

\begin{align}
  x_i^{(k+1)} &:= \arg \min_{x_i} \mathcal{L}_\rho(x_i, z^{(k)}, y^{(k)}), \quad \forall i \\
  z^{(k+1)} &:= \arg \min_z \mathcal{L}_\rho(x^{(k+1)}, z, y^{(k)}), \\
  y^{(k+1)} &:= y^{(k)} + \beta(Ax^{(k+1)} + Bz^{(k+1)} - C). 
\end{align}

(150a, 150b, 150c)

### 4.4 Multi-Agent Trust Region Policy Optimization

Consider the TRPO policy optimization (146) for the multi-agent case. By decomposing the advantage function $A_{\pi_{old}}(s, a)$, the objective can be rewritten as follows:

\[
\max_\pi \mathbb{E}_{s \sim \rho_{\pi_{old}}, a \sim \pi_{old}} \left[ \frac{\pi(a|s)}{\pi_{old}(a|s)} \left( A^1_{\pi_{old}}(s, a) + \cdots + A^N_{\pi_{old}}(s, a) \right) \right]
\]

(151)

s.t. $\mathcal{D}_{KL}^{\rho_{\pi_{old}}}(\pi_{old}, \pi) \leq \delta$.

The purpose is to split the objective into $N$ independent sub-objectives so that the optimization problem can be solved in a distributed way by $N$ agents. However, the sub-objectives are coupled through the joint policy ratio $\pi(a|s)/\pi_{old}(a|s)$, where $\pi(a|s) = \prod_{i=1}^N \pi^i(a^i|s)$.

#### 4.4.1 Trust Region Optimization for Multiple Agents

To solve this problem, each agent is trained to learn a local model of the joint policy, $\pi^i(a|s), i \in \mathcal{N}$, and to reach an agreement on the policy ratio $\pi^i(a|s)/\pi_{old}^i(a|s)$. Then, the policy optimization (151) can be equivalently transformed into the the consensus optimization problem, as given in the following lemma [107].

**Lemma 4.4.1.** Assume the local policy has the form $\pi^i(a|s) = \pi^i(a^1|s)\pi^i(a^2|s)\cdots\pi^i(a^N|s)$. Then, the TRPO policy update (151) can be
equivalently transformed into:

\[
\max_{\{\pi^i\}_{i \in \mathcal{N}}} \sum_{i=1}^{N} \mathbb{E}_{s \sim \rho_{\pi_{old}}} \left[ \frac{\pi^i(s) A^i_{\pi_{old}}(s, a)}{\pi^i_{old}(s, a)} \right] \\
\text{s.t.} \quad \frac{\pi^1(s)}{\pi^1_{old}(s)} = \cdots = \frac{\pi^N(s)}{\pi^N_{old}(s)}, \forall i \in \mathcal{N} \quad (152)
\]

\[
D_{KL}^{\rho_{\pi_{old}}} (\pi_{old}, \pi) \leq \delta.
\]

**Proof.** We expand the likelihood ratio \(\frac{\pi^i(a|s)}{\pi^i_{old}(a|s)}\) of agent \(i\) into

\[
\frac{\pi^i(a|s)}{\pi^i_{old}(a|s)} = \frac{\pi^i(a_1|s)}{\pi^i_{old}(a_1|s)} \frac{\pi^i(a_2|s)}{\pi^i_{old}(a_2|s)} \cdots \frac{\pi^i(a_N|s)}{\pi^i_{old}(a_N|s)}. \tag{153}
\]

When the consensus constraints in (152) are satisfied, we can rewrite the expansion (153) by replacing \(\frac{\pi^i(a_j|s)}{\pi^i_{old}(a_j|s)}\) for all \(i \neq j\) with \(\pi^j(a_j|s)\) as follows:

\[
\frac{\pi^i(a|s)}{\pi^i_{old}(a|s)} = \frac{\pi^i(a_1|s)}{\pi^i_{old}(a_1|s)} \cdots \frac{\pi^i(a_N|s)}{\pi^i_{old}(a_N|s)} = \frac{\pi(a|s)}{\pi_{old}(a|s)} \tag{154}
\]

Substituting (154) into the objective of (152), we have

\[
\sum_{i=1}^{N} \mathbb{E}_{s \sim \rho_{\pi_{old}}} \left[ \frac{\pi^i(s) A^i_{\pi_{old}}(s, a)}{\pi^i_{old}(s, a)} \right] \\
= \mathbb{E}_{s \sim \rho_{\pi_{old}}} \left[ \frac{\pi(s) A_{\pi_{old}}(s, a)}{\pi_{old}(s, a)} \right] \sum_{i=1}^{N} \pi^i_{old}(s, a) \tag{155}
\]

Now, the objective is separable with respect to \(\{\pi^i(a|s)\}_{i \in \mathcal{N}}\). However, the local model \(\pi^i(a|s)\) is conditioned on the global \(s\), which is unavailable in a POMG. Moreover, the KL constraint is imposed on the joint policy \(\pi_{old}\), which is difficult to satisfy in a distributed way. To address this problem, the following definition, lemma, and theorem are introduced from [107].
Definition 4.4.2 (Sufficient Observation Information). The local observation $o^i$ of agent $i$ is said to be sufficient if it satisfies:

1. $E_{o^i \sim P^i_o(s', a \sim \pi)}[r^i(s, a)] = E_{s \sim P_s(s', a \sim \pi)}[r^i(s, a)]$,

2. $P^i_\pi(s', o^i | s) = P^i_\pi(s', o^i | o^i)$,

where $P^i_\pi(s', o^i | s)$ and $P^i_\pi(s', o^i | o^i)$ are the transition probabilities from $s$ and $o^i$ to $(s', o^i)$ under $\pi$, respectively.

Lemma 4.4.3. If the observation $o^i, \forall i \in N$ is sufficient (Definition 1), $E_{s \sim \rho^i, a \sim \tilde{\pi}}[A^i_s(s, a)] = E_{o^i \sim \rho^i, a \sim \tilde{\pi}}[A^i_\pi(o^i, a)]$ holds.

Proof. First note that the discounted state visitation frequencies $\rho^i$ and the discounted local observation visitation frequencies (defined in Section 4.3-A) have the following relationship:

$$\sum_s \rho^i(s) P^i_o(o^i | s) = P(o^i = o^i | \pi) + \gamma P(o^i = o^i | \pi) + \cdots = \rho^i(o^i).$$

Define $V^i_\pi(s, o^i) = E_{\tau \sim \pi}[R^i(\tau) | s_0 = s, o^i_0 = o^i]$ as the local value function of agent $i$ with respect to $(s, o^i)$. The Bellman equation for $V^i_\pi(s, o^i)$ can be expressed by

$$V^i_\pi(s, o^i) = \sum_a \pi(a | o^i) r^i(s, a) + \gamma \sum_{s', o^i'} P^i_\pi(s', o^i' | s, o^i) V^i_\pi(s', o^i'),$$

where $P^i_\pi(s', o^i | s, o^i) = \sum_a \pi^i(a | o^i) P(s' | s, a) P^i_o(o^i | s')$ is the transition probability.

Then, the local value functions $V^i_\pi(s)$ and $V^i_\pi(o^i)$ can be expressed respectively as:

$$V^i_\pi(s) = \sum_{o^i} P^i_o(o^i | s) V^i_\pi(s, o^i) = \sum_{o^i} P^i_o(o^i | s) \sum_a \pi(a | o^i) r^i(s, a) + \gamma \sum_{s', o^i'} P^i_\pi(s', o^i' | s) V^i_\pi(s', o^i'), \quad (156)$$

$$V^i_\pi(o^i) = \sum_s P^i_\pi(s | o^i) V^i_\pi(s, o^i) = \sum_s P^i_\pi(s | o^i) \sum_a \pi(a | o^i) r^i(s, a) + \gamma \sum_{s', o^i'} P^i_\pi(s', o^i' | o^i) V^i_\pi(s', o^i'). \quad (157)$$
Based on the sufficient information assumptions on local observations: (1) 
\[ \mathbb{E}_{o^i \sim \mathcal{O}^i, a \sim \mathcal{A}^i} [r^i(s, a)|s] = \mathbb{E}_{s \sim \mathcal{S}^i, a \sim \mathcal{A}^i} [r^i(s, a)|o^i] \]; (2) \[ \mathcal{P}_\pi(s', o^i|s) = \mathcal{P}_\pi(s', o^i'|o^i) \], we can easily prove that

\[ V^i_\pi(s) = V^i_\pi(o^i) \].  

(158)

Letting \( \mathcal{P}^i : \mathcal{O}^i \times \mathcal{A}^i \times \cdots \times \mathcal{A}^\mathcal{N} \times \mathcal{O}^i \mapsto [0, 1] \) denote the transition distribution of the observations of agent \( i \), then the local advantage function can be rewritten by

\[
A^i_\pi(s, a) = r^i(s, a) - V(s) + \gamma \mathbb{E}_{s' \sim \mathcal{P}^i(s, a)} [V(s')] \\
= r^i(s, a) - V(o^i) + \gamma \mathbb{E}_{s' \sim \mathcal{P}^i(s, a)} [\mathbb{E}_{o^i' \sim \mathcal{P}^i(o^i, a)} [V(s')]] \\
= r^i(s, a) - V(o^i) + \gamma \mathbb{E}_{s' \sim \mathcal{P}^i(s, a)} [\mathbb{E}_{o^i' \sim \mathcal{P}^i(o^i, a)} [V(o^i)]] \\
= r^i(s, a) - V(o^i) + \gamma \mathbb{E}_{o^i' \sim \mathcal{P}^i(o^i, a)} [V(o^i)] \\
\]

(159)

Then, based on Eqs. (156) and (159), we have

\[
\mathbb{E}_{s \sim \mathcal{P}_\pi, a \sim \mathcal{A}} [A^i_\pi(s, a)] = \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A^i_\pi(s, a) \\
= \sum_s \rho_\pi(s) \sum_a \left( \sum_{o^i} \tilde{\pi}(a|o^i) \mathcal{P}^i(o^i|s) \right) A^i_\pi(s, a) \\
= \sum_{o^i} \sum_a \tilde{\pi}(a|o^i) \sum_s \rho_\pi(s) \mathcal{P}^i(o^i|s) r^i(s, a) \quad \text{(use (156), (159))} \\
- \sum_{o^i} \sum_a \tilde{\pi}(a|o^i) r^i(o^i) \left[ V(o^i) - \gamma \mathbb{E}_{o^i' \sim \mathcal{P}^i(o^i, a)} [V(o^i)] \right] \\
= \sum_s \rho_\pi(s) \sum_{o^i} \mathcal{P}^i(o^i|s) \sum_a \tilde{\pi}(a|o^i) r^i(s, a) \\
- \sum_{o^i} \rho_\pi(o^i) \sum_a \tilde{\pi}(a|o^i) \left[ V(o^i) - \gamma \mathbb{E}_{o^i' \sim \mathcal{P}^i(o^i, a)} [V(o^i)] \right] \\
= \sum_{o^i} \rho_\pi(o^i) \sum_{s} \mathcal{P}_s(s|o^i) \sum_a \tilde{\pi}(a|o^i) r^i(s, a) \\
- \sum_{o^i} \rho_\pi(o^i) \sum_a \tilde{\pi}(a|o^i) \left[ V(o^i) - \gamma \mathbb{E}_{o^i' \sim \mathcal{P}^i(o^i, a)} [V(o^i)] \right] \\
= \mathbb{E}_{o^i \sim \mathcal{P}^i, a \sim \mathcal{A}} [A^i_\pi(o^i, a)].
\]

(160)
where $J^i(\pi)$ is the expected discounted rewards of agent $i$. \hfill \Box

**Theorem 4.4.4.** If the local observations $o^i, \forall i \in \mathcal{N}$ is sufficient, then the problem (255) is equivalent to the problem (151) with a stricter trust region, i.e.,

$$D_{KL}^{\rho_{\pi, old}}(\pi_{old}, \pi) \leq \Delta, \text{ where } \Delta \leq \delta.$$ 

$$\begin{align*}
\max_{\{\pi_i\}} & \sum_{i=1}^{N} \sum_{o^i \sim \rho^i_{\pi_{old}}} \mathbb{E}_{a \sim \pi^i_{old}} \left[ \frac{\pi^i(a|o^i)}{\pi^i_{old}(a|o^i)} A^i_{\pi_{old}}(o^i, a) \right] \\
\text{s.t. } & \frac{\pi^1(a^1|o^1)}{\pi^1_{old}(a^1|o^1)} = \cdots = \frac{\pi^N(a^N|o^N)}{\pi^N_{old}(a^N|o^N)}, \forall i \in \mathcal{N} \\
& D_{KL}^{\rho^i_{\pi_{old}}}(\pi^i_{old}, \pi^i) \leq \frac{\delta}{N}, \forall i \in \mathcal{N},
\end{align*}$$

(161)

where $\pi^i(a|o^i) = \prod_{n=1}^{N} \pi^i(a^n|o^n)$, and $a \sim \pi_{old}$ means that $a$ is selected according to the joint policy $\pi_{old}$; the KL Divergence is $D_{KL}^{\rho^i_{\pi_{old}}}(\pi^i_{old}, \pi^i) = \mathbb{E}_{o^i \sim \rho^i_{\pi_{old}}} [D_{KL}(\pi^i_{old} \parallel \pi^i)]$.

**Proof.** First, we show that the objective of problem (255) is equivalent to that of (151) if the local observation $o^i$ is sufficient.

Note that the objective of (151) can be rewritten as

$$\begin{align*}
\mathbb{E}_{s \sim \rho^i_{\pi_{old}}, a \sim \pi^i_{old}} \left[ \frac{\pi(a|s)}{\pi_{old}(a|s)} \left( A^i_{\pi_{old}}(s, a) + \cdots + A^N_{\pi_{old}}(s, a) \right) \right] \\
= \sum_{i=1}^{N} \mathbb{E}_{o^i \sim \rho^i_{\pi_{old}}, a \sim \pi^i_{old}} \left[ A^i_{\pi_{old}}(o^i, a) \right] \quad \text{(use Lemma 4.4.3)} \\
= \sum_{i=1}^{N} \mathbb{E}_{o^i \sim \rho^i_{\pi_{old}}, a \sim \pi^i_{old}} \left[ \frac{\pi(a|o^i)}{\pi_{old}(a|o^i)} A^i_{\pi_{old}}(o^i, a) \right].
\end{align*}$$

(162)

Using Lemma 4.4.1, Eq. (162) can be equivalently transformed into the following objective with consensus constraints:

$$\begin{align*}
\text{maximize } & \sum_{i=1}^{N} \sum_{o^i \sim \rho^i_{\pi_{old}}} \mathbb{E}_{a \sim \pi^i_{old}} \left[ \frac{\pi^i(a|o^i)}{\pi^i_{old}(a|o^i)} A^i_{\pi_{old}}(o^i, a) \right] \\
\text{subject to } & \frac{\pi^1(a^1|o^1)}{\pi^1_{old}(a^1|o^1)} = \cdots = \frac{\pi^N(a^N|o^N)}{\pi^N_{old}(a^N|o^N)}, \forall i \in \mathcal{N}
\end{align*}$$

(163)
Next, we show that the KL constraint over the joint policy in problem (151) will be enforced within a stricter trust region when the individual KL constraints over the local policy are satisfied in the distributed optimization problem (255).

Note that

\[ D_{KL}(\pi_{old}^i(\cdot|s) \parallel \pi^i(\cdot|s)) = \sum_{a^i} \pi_{old}^i(a^i|s) \log \frac{\pi_{old}^i(a^i|s)}{\pi^i(a^i|s)} \]

\[ = \sum_{a^i} \sum_{o^i} P_o^i(o^i|s) \pi_{old}^i(a^i|o^i) \log \frac{\sum_{o^i} P_o^i(o^i|s) \pi_{old}^i(a^i|o^i)}{\sum_{o^i} P_o^i(o^i|s) \pi^i(a^i|o^i)} \]

\[ \leq \sum_{o^i} P_o^i(o^i|s) \sum_{a^i} \pi_{old}^i(a^i|o^i) \log \frac{\pi_{old}^i(a^i|o^i)}{\pi^i(a^i|o^i)} \]

\[ = \sum_{o^i} P_o^i(o^i|s) D_{KL}(\pi_{old}^i(\cdot|o^i) \parallel \pi^i(\cdot|o^i)), \]

where the inequality is derived from the logarithmic property

\[ b \log \frac{b}{c} \leq \sum_{i=1}^{n} b_i \log \frac{b_i}{c_i}, \]

and \( b_i, c_i \) are nonnegative numbers, \( b \) is the sum of \( b_i \)s and \( c \) is the sum of \( c_i \)s.

Based on the additive and non-negative properties of KL-Divergence, we have

\[ \overline{D}_{KL}^{\rho_{old}}(\pi_{old}, \pi) = \mathbb{E}_{s \sim \rho_{old}}[D_{KL}(\pi_{old}(\cdot|s) \parallel \pi(\cdot|s)))] \]

\[ = \sum_{i=1}^{N} \mathbb{E}_{s \sim \rho_{old}}[D_{KL}(\pi_{old}^i(a^i|s) \parallel \pi^i(a^i|s))] \]

\[ \leq \sum_{i=1}^{N} \sum_{s} \rho_{old}(s) \left[ \sum_{o^i} P_o^i(o^i|s) D_{KL}(\pi_{old}^i(\cdot|o^i) \parallel \pi^i(\cdot|o^i)) \right] \]

\[ = \sum_{i=1}^{N} \mathbb{E}_{o^i \sim \rho_{old}^i}[D_{KL}(\pi_{old}^i(\cdot|o^i) \parallel \pi^i(\cdot|o^i))] \]

\[ = \sum_{i=1}^{N} \overline{D}_{KL}^{\rho_{old}^i}(\pi_{old}^i, \pi^i). \]

Using the KL constraints on the local policy in (255), we have

\[ \overline{D}_{KL}^{\rho_{old}}(\pi_{old}, \pi) \leq \sum_{i=1}^{N} \overline{D}_{KL}^{\rho_{old}^i}(\pi_{old}^i, \pi^i) \leq \sum_{i=1}^{N} \frac{\delta}{N} = \delta. \]
Denoting $\Delta = \sum_{i=1}^{N} D_{KL}^{d_{dd}}(\pi_{i,old}, \pi_{i})$, the result follows.

Theorem 4.4.4 provides a theoretical foundation for extending TRPO to POMG problems. It means that the policy optimization in TRPO can be transformed equivalently into a consensus optimization if the agents have sufficient observation. Various approaches can be used to improve observability, such as extracting information from histories. This is beyond the scope of this dissertation and interested readers are referred to [82, 108].

### 4.4.2 Distributed Consensus with Asynchronous ADMM

There are many techniques that can solve distributed consensus problems, such as gossip algorithms [97], distributed gradient methods [109], and ADMM [106]. Gossip algorithms generally focus on the consensus averaging problem for a network of computational nodes. Distributed gradient methods combine consensus averaging with local gradient descent steps and usually apply to unconstrained problems. ADMM adopts primal-dual formulation for constrained distributed optimization problems, and it is suitable for the considered problem.

Traditional distributed ADMM algorithms generally require a synchronous process at each iteration. As shown in Eq. (150), to optimize the primal variables $x_i$, the values of $z^{(k)}$ and $y^{(k)}$ at the most recent update need to be broadcasted. Then, the new primal variables $x^{(k+1)} = (x_1^{(k+1)}, \ldots, x_N^{(k+1)})$ must be synchronized to update the variables $z$ and $y$. This can increase the communication burden in a large-scale network and slow down the pace of the optimization.

In this study, the asynchronous distributed ADMM algorithm [110] is employed. The algorithm allows asynchronous updates by exchanging information between randomly activated neighboring agents. Moreover, the algorithm is robust to random failure of the communications. These advantages make it suitable for the proposed decentralized setting.
Let $\mathcal{N}(e) = \{i, j\}$ denote the agents at the endpoints of the communication link $e \in \mathcal{E}$. For each agent $q \in \{i, j\}$, the variable $z_q^e(o^q, a^n), \forall n \in \mathcal{N}$ is introduced to estimate the likelihood ratio of its neighbor $\mathcal{N}(e) \setminus \{q\}$. The consensus constraint for the agent $i$ and $j$ can then be replaced by:

\[
\frac{\pi^i(a^n|o^i)}{\pi_{old}^i(a^n|o^i)} = z_i^e(o^i, a^n), \quad -\frac{\pi^j(a^n|o^j)}{\pi_{old}^j(a^n|o^j)} = z_j^e(o^j, a^n),
\]

(168a)

\[
z_i^e(o^i, a^n) + z_j^e(o^j, a^n) = 0.
\]

(168b)

Let the constant $C_q^e$ be the weight on the information transmitted between agents $q \in \mathcal{N}(e)$. $C_q^e$ takes the value of 1 or $-1$ such that $C_i^e + C_j^e = 0$. Combining Eqs. (168), the problem (255) can be transformed into

\[
\min_{\pi^i, z^e} - \sum_{i=1}^{N} \mathbb{E}_{o^i \sim \rho_{old}^i} \left[ \frac{\pi^i(a|o^i)}{\pi_{old}^i(a|o^i)} A_{\pi_{old}^i}^i(o^i, a) \right]
\]

s.t. $C_q^e \frac{\pi^i(a^n|o^i)}{\pi_{old}^i(a^n|o^i)} = z_q^e(o^q, a^n), \ q \in \mathcal{N}(e),
\]

(169)

\[
z_i^e(o^i, a^n) + z_j^e(o^j, a^n) = 0, \ (i, j) \in \mathcal{N}(e),
\]

\[
D_{KL}(\pi_{old}^i, \pi^i) < \delta/N, \ \forall i \in \mathcal{N}.
\]

Note that $\pi^i(a^n|o^i)$ and $z_q^e(o^q, a^n)$ are dependent on $o^i$ and $a^n$. For notational simplicity, denote $\pi^i$ as a vector of the decision variables $\pi^i(a|o^i)$ for all realizations of $o^i$ and $a^n$. Similarly, denote $z_q^e$ as a vector of the decision variables $z_q^e(o^q, a^n)$ for all realizations of $o^i$ and $a^n$. The notations $\Upsilon_n(\pi^i) := \pi^i(a^n|o^i)/\pi_{old}^i(a^n|o^i)$ and $\Upsilon(\pi^i) := \pi^i(a|o^i)/\pi_{old}^i(a|o^i)$ are used to represent the policy ratios for all realizations of $o^i$ and $a^n$. Then, the augmented Lagrangian of (169) can be expressed
by:

\[ \mathcal{L}_\beta(\pi, z, y) = -\sum_{i=1}^{N} \mathbb{E}_{o^i \sim \rho_{o^{i\text{old}}}^{i\text{old}}, a \sim \pi_{o^{i\text{old}}}} [\mathcal{Y}^i A^i_{\pi_{o^{i\text{old}}}}(o^i, a)] \]

\[ + \sum_{e \in \mathcal{E}} \sum_{q \in \mathcal{N}(e)} \sum_{n \in \mathcal{N}} y^n_e (C^n_{\pi}(\pi^q) - z^n_q) \]

\[ + \sum_{e \in \mathcal{E}} \sum_{q \in \mathcal{N}(e)} \sum_{n \in \mathcal{N}} \frac{\beta}{2} \| C^n_{\pi}(\pi^q) - z^n_q \|^2, \]

where \( \pi = \{\pi^i\}_{i \in \mathcal{N}} \) and \( z = \{z^n_q\}_{e \in \mathcal{E}, q \in \mathcal{N}(e), n \in \mathcal{N}} \) are primal variables and \( y = \{y^n_e\}_{e \in \mathcal{E}, q \in \mathcal{N}(e), n \in \mathcal{N}} \) are dual variables; \( \beta > 0 \) is a penalty parameter. The primal variables \( \pi^i \) and \( z^n_q \) satisfy the constraints: \( \Pi^i = \{\pi | D_{KL}^{r_{o^{i\text{old}}}^{i\text{old}}} (\pi^i, \pi) < \delta/N\} \) and \( Z^n_e = \{z^n_e, z^n_j | z^n_e + z^n_j = 0, i, j \in \mathcal{N}(e)\} \).

To solve problem (169), the asynchronous ADMM minimizes the augmented Lagrangian by sequentially updating the local policies \( \pi \), the estimators \( z \), and the dual variables \( y \). Specifically, let \( \pi^{(k)}, z^{(k)}, y^{(k)} \) be the values at iteration \( k \). At iteration \( k + 1 \), a communication link \( e = (i, j) \) is selected and the two agents \( q \in \mathcal{N}(e) \) at the endpoints are activated to update \( \pi^q, z^n_q, \) and \( y^n_e \) according to:

\[ \pi^{q(k+1)} := \arg \min_{\pi^q \in \Pi^q} \mathcal{L}_\beta(\pi, z^{(k)}, y^{(k)}), \] (171a)

\[ z^{qn(k+1)} := \arg \min_{z^n_q \in Z^n_e} \mathcal{L}_\beta(\pi^{(k+1)}, z, y^{(k)}), \] (171b)

\[ y^{qn(k+1)} := y^{qn(k)} + \beta (C^n_{\pi}(\pi^q(k+1)) - z^{qn(k+1)}). \] (171c)

The algorithm (171) converges almost surely to the optimal solution at the rate of \( \mathcal{O}(1/k) \) as \( k \to \infty \) if the objectives and the sets \( \Pi^i \) and \( Z^n_e \) are convex [110]. However, the convex assumption does not hold for \( \Pi^i \) due to the KL constraints. Besides, parameterized policies \( \pi^i(a|o^i; \theta^i) \) are usually required when the observation and action space is huge. For general nonlinear policies, like neural networks, the convex assumption does not hold, and the convergence of the asynchronous ADMM algorithm in (171) is not guaranteed [106, 110].
4.4.3 Sequential Convexification for Convergence

To address the convergence issue, the idea of sequential convex programming [111] is used to approximate the problem (169) with a convex model. Since a parameterized policy $\pi^i(a|o^i; \theta^i)$ is considered, the parameter vector $\theta^i$ is used to represent the policy $\pi^i$, and overload previous notations, e.g. $\Upsilon^n(\pi^i) := \Upsilon^n(\pi^i)$, $f^i(\pi^i) := f^i(\theta^i)$. Note that, for a small trust region $\delta$, the likelihood ratio $\Upsilon^n(\theta^i)$ and $\Upsilon(\theta^i)$ can be well approximated by first order Taylor expansion round $\theta^i_{old}$:

$$\Upsilon^n(\theta^i) \approx 1 + \nabla^T \theta^i \Upsilon^n(\theta^i)(\theta^i - \theta^i_{old}),$$

$$\Upsilon(\theta^i) \approx 1 + \sum_{n=1}^{N} \nabla^T \theta^i \Upsilon^n(\theta^i)(\theta^i - \theta^i_{old})$$

and the KL-divergence can be well approximate by the second-order Taylor expansion round $\theta^i_{old}$:

$$\bar{D}^{\theta^i_{old}}_{KL}(\theta^i; \theta^i_{old}) \approx \frac{1}{2} (\theta^i - \theta^i_{old})^T H_i (\theta^i - \theta^i_{old}),$$

where $H_i = \nabla^2 \Upsilon^n(\theta^i_{old}, \theta^i)$. Substituting (172) - (173) into (169) and (170), we can derive a convex approximation of the primal problem:

$$\min_{\theta^i, z^e_q} - \sum_{i=1}^{N} \sum_{n=1}^{N} A^i_n(\theta^i_{old})(\theta^i - \theta^i_{old})$$

s.t. $C^q_e + C^q_e \nabla^T \Upsilon^n(\theta^i)(\theta^i - \theta^i_{old}) = z^q_e, \ q \in \mathcal{N}(e),$

$$z^i_{en} + z^j_{en} = 0, \ (i, j) \in \mathcal{N}(e),$$

$$\frac{1}{2} (\theta^i - \theta^i_{old})^T H_i (\theta^i - \theta^i_{old}) < \delta/N, \ \forall i \in \mathcal{N},$$

$$e \in \mathcal{E}, n \in \mathcal{N}.$$
where $\overline{A}_n^i(\theta_{old}^i) = \mathbb{E}_{o^i \sim \rho_{old}^i, a \sim \theta_{old}} [A_{\theta_{old}}^i(o^i, a) \nabla_{\theta_{old}} \Upsilon_n(\theta^i)]$. The corresponding augmented Lagrangian is

$$L_\beta(\theta, z, y) = -\sum_{i=1}^N \sum_{n=1}^N \overline{A}_n^i(\theta_{old}^i)(\theta^i - \theta_{old}^i) + \sum_{e \in E} \sum_{q \in N(e)} \sum_{n \in N} y_{en}^{qn^2} (C_e^q + C_e^q \nabla_{\theta_e} \Upsilon_n(\theta^i)(\theta^i - \theta_{old}^i) - z_{en}^{qn^2}) + \sum_{e \in E} \sum_{q \in N(e)} \sum_{n \in N} \frac{\beta}{2} \|C_e^q + C_e^q \nabla_{\theta_e} \Upsilon_n(\theta^i)(\theta^i - \theta_{old}^i) - z_{en}^{qn^2} \|^2_2, \quad (175)$$

where the decision variables $\theta^i$ are restricted to: $\Theta^i = \{ \frac{1}{2} (\theta^i - \theta_{old}^i)^T H_i (\theta^i - \theta_{old}^i) < \delta/N \}, i \in \mathcal{N}$. Then, the asynchronous ADMM algorithm becomes

$$\theta_{\theta}^{q(k+1)} := \arg \min_{\theta^i \in \Theta^i} L_\beta(\theta^i, z^{q(k)}, y^{q(k)}), \quad (176a)$$
$$y_{en}^{qn(k+1)} := \arg \min_{z_{en}^{qn} \in Z_{en}^p} L_\beta(\theta^{q(k+1)}, z, y^{q(k)}), \quad (176b)$$
$$y_{en}^{qn(k+1)} := y_{en}^{qn(k)} + \beta [C_e^q + C_e^q \nabla_{\theta_e} \Upsilon_{n}(\theta^{i(k+1)}) (\theta^i(k+1) - \theta_{old}^{i(k+1)}) - z_{en}^{qn(k+1)}]. \quad (176c)$$

Due to the convex approximation, the algorithm may converge to stationarity instead of optimality. However, since deep neural networks are highly nonlinear and non-convex, it is difficult for the algorithm to surely converge to optimality. Despite this limitation, the proposed approach demonstrates good performance in practice as shown in the experiments.

4.4.4 Information Required by Agents

To optimize the local policy, each agent needs to compute the local advantage $A_{\theta_{old}}^i(o^i, a)$ and the policy ratio $\Upsilon_n(\theta^i), \forall n \in \mathcal{N}$. To that end, each agent will need the local observation $o^i$, the joint action $a = (a^1, \ldots, a^N)$, and the personal reward $r^i$. The joint action $a$ is only global information required by each agent. Being able to observe other agents’ actions is not a restrictive assumption in many cooperative
tasks [102, 103]. However, it is worth mentioning that knowing the actions of other agents does not mean knowing their policies because the local policy $\pi^i_{\text{old}}(a|o^i)$ is conditioned on local observation $o^i$, which is owned solely by agent $i$.

4.5 Practical Implementation

This section demonstrates the practical implementation of the MATRPO algorithms in scenarios where the observation space and action space are large. Initially, the expected advantage function and consensus constraint are estimated using samples. Subsequently, the policy ratio in the consensus constraint is substituted with the logarithmic policy ratio, facilitating the creation of a closed-form solution.

4.5.1 Sample-Based Approximation of the Consensus Constraint

To satisfy the consensus constraint, the agents need to reach a consensus on the policy ratio $\Upsilon^n(\theta^i) = \pi^i(a^n|o^i, \theta^i) / \pi^i(a^n|o^i, \theta^i_{\text{old}})$, $\forall n \in \mathcal{N}$ for every observation and action. This is challenging when the observation and action spaces are large. In practice, the consensus constraints are implemented based on samples. Specifically, the joint policy $\pi(a|o)$ is implemented for $T$ steps in an environment to generate a batch of trajectories of the states and actions, and each agent $i$ has a batch of $D$ locally observed trajectories, i.e. $\mathcal{D}^i = \{(o^i_{0,d}, a_{0,d}, o^i_{1,d}, \ldots, a_{T-1,d}, o^i_{T-1,d}), d=1,\ldots,D\}$. After that, the agents compute the policy ratios $\Upsilon^n(\theta^i)$ and communicate with their neighbors. Then, the agents are constrained to reach consensus on these sampled policy ratios.

For the calculation of local advantage function $A^{i}_{\pi_{\text{old}}}(o^i, a)$, the generalized advantage estimation (GAE) method [112] is used:

$$A^{i} = \epsilon^{i}_t + (\gamma \lambda) \epsilon^{i}_{t+1} + \cdots + (\gamma \lambda)^{T-t+1} \epsilon^{i}_{T-1} , \tag{177}$$

$$\epsilon^{i}_t = r^{i}_t + \gamma V^{i}_{\pi_{\text{old}}}(a^{i}_{t+1}) - V^{i}_{\pi_{\text{old}}}(a^{i}_t), \tag{178}$$
where $V_{\pi_{\text{old}}}^{i}(o^{i})$ is the local value function approximated by a neural network.

### 4.5.2 Agreement on Logarithmic Ratio

In practice, it is better to use the logarithmic likelihood ratio $\log(\Upsilon^{n}(\theta^{i}))$ for consensus since it leads to a closed-form solution [107] to the asynchronous ADMM updates (171). As the logarithmic function is monotonic, consensus on $\log(\Upsilon^{n}(\theta^{i}))$ is equivalent to consensus on $\Upsilon^{n}(\theta^{i})$. In addition, the first order Taylor expansion of the logarithmic policy ratio is the same as that of the policy ratio:

$$\nabla_{\theta^{i}}\log(\Upsilon^{n}(\theta^{i}))(\theta^{i} - \theta_{\text{old}}^{i}) = \nabla_{\theta^{i}}\Upsilon^{n}(\theta^{i})(\theta^{i} - \theta_{\text{old}}^{i}).$$

Then, Eq. (175) can be approximated by the following Lagrangian based on the Monte-Carlo sampling

$$L_{\beta}(\theta, z, y) = -\frac{1}{M} \sum_{i=1}^{N} \sum_{n=1}^{N} A_{i}^{T}J_{\text{in}}(\theta^{i} - \theta_{\text{old}}^{i})$$

$$+ \frac{1}{M} \sum_{e=1}^{L} \sum_{q \in N(e)} \sum_{n=1}^{N} g_{e}^{q, n}T \left[ C_{e}^{q}J_{q}^{m}(\theta^{q} - \theta_{\text{old}}^{q}) - z_{e}^{q, n} \right]$$

$$+ \frac{1}{M} \sum_{e=1}^{L} \sum_{q \in N(e)} \sum_{n=1}^{N} \frac{\beta}{2} \| C_{e}^{q}J_{q}^{m}(\theta^{q} - \theta_{\text{old}}^{q}) - z_{e}^{q, n} \|_{2}^{2},$$

where $A_{i}^{i} \in \mathbb{R}^{M \times 1}$ is a vector of the sampling values of the advantage function $A_{\pi_{\text{old}}}^{i}(o^{i}, a)$, $J_{\text{in}}^{n} \in \mathbb{R}^{M \times D}$ is the Jacobian of the sampled likelihood ratios $\Upsilon^{n}(\theta^{i})$ with respect to $\theta^{i} \in \mathbb{R}^{D \times 1}$. The policy parameters $\theta^{i}$ is defined in the feasible set $\theta^{i} \in \Theta^{i} = \{\theta | \frac{1}{2}(\theta - \theta_{\text{old}}^{i})^{T}\overline{H}^{i}(\theta - \theta_{\text{old}}^{i}) \leq \delta / N\}$, where $\overline{H}^{i} \in \mathbb{R}^{D \times D}$ is the Hessian of the sample average KL-Divergence $\frac{1}{M} \sum_{m=1}^{M} D_{KL}(\pi_{\text{old}}^{i}(\cdot | o^{i}; \theta_{\text{old}}^{i}) || \pi^{i}(\cdot | o^{i}; \theta^{i})).$

The asynchronous ADMM updates for the Lagrangian (180) have a closed-form solution [107], which is given in the following proposition.

**Proposition 4.5.1.** *If there exists at least one strictly feasible point in the feasible
sets $\Theta^i$ and $Z^n_e$, the optimal solution of (180) is:

\[
\theta^{q(k+1)} := \theta^{q}_{\text{old}} + \sqrt{\frac{2\delta/N}{V^T H^n q^{-1} V}} V, \\
z^{qn(k+1)}_e := \frac{1}{\beta}(y^{qn(k)}_e - \nu^n_e) + C^n_e J^n q (\theta^{q(k+1)} - \theta^{q}_{\text{old}}), \\
y^{qn(k+1)}_e := \nu^n_e
\]

when $M \to \infty$, where

\[
V = \frac{1}{M} \sum_{n=1}^N J^n q A^q - \sum_{e \in \mathcal{E}(q)} C^n_e y^{qn(k)}_e + \beta \sum_{e \in \mathcal{E}(q)} C^n_e z^{qn(k)}_e,
\]

\[
\nu^n_e = \frac{1}{2} \sum_{q \in \mathcal{N}(e)} \left[ y^{qn(k)}_e + \beta C^n_e J^n q (\theta^{q(k+1)} - \theta^{q}_{\text{old}}) \right].
\]

and $\mathcal{E}(q)$ is the set of the communication links connecting to agent $q$.

**Proof.** Define $x^q = \theta^q - \theta^q_{\text{old}}$ and $x^q \in X^q = \{x | \frac{1}{2} x^T H^q x \leq \delta/N\}$. The Lagrangian (180) can be simplified as

\[
\mathcal{L}_\beta(x, z, y) = -\frac{1}{M} \sum_{n=1}^N \sum_{i=1}^N A_i^T J^{min,i} x
\]

\[
+ \frac{1}{M} \sum_{e=1}^L \sum_{q \in \mathcal{N}(e)} \sum_{n=1}^N y^{qnT}_e (C^n_e J^n q x^q - z^{qn}_e)
\]

\[
+ \frac{1}{M} \sum_{e=1}^L \sum_{q \in \mathcal{N}(e)} \sum_{n=1}^N \frac{\beta}{2} \| C^n_e J^n q x^q - z^{qn}_e \|_2^2,
\]

(1) We first consider the update of $\theta^q$, which is transformed into the following problem

\[
\min_{x^q \in X^q} \mathcal{L}_\beta(x, z, y), \quad q \in \mathcal{N}(e).
\]

This is a convex optimization with quadratic constraint. According to strong duality theory, the optimal value $p^*$ satisfies

\[
p^* = \min_{x^q} \max_{\lambda^q > 0} \mathcal{L}_\beta(x, z, y) + \lambda^q \left( \frac{1}{2} x^T H^q x - \delta/N \right)
\]

\[
= \max_{\lambda^q > 0} \min_{x^q} \mathcal{L}_\beta(x, z, y) + \lambda^q \left( \frac{1}{2} x^T H^q x - \delta/N \right)
\]

\[
= \max_{\lambda^q > 0} \min_{x^q} f(x^q, \lambda^q)
\]

83
Based on the first order condition, the optimal solution $x^q_*$ should satisfy
\[
\frac{\partial f(x^q, \lambda^q)}{\partial x^q} = 0,
\]
i.e.
\[
\left( \lambda^q \mathcal{H}^q + \frac{|\mathcal{E}(q)|\beta}{M} \sum_{n=1}^N J_{qn}^T J_{qn} \right) x^q - \frac{1}{M} \sum_{n=1}^N J_{qn}^T \left( A^q - \sum_{e \in \mathcal{E}(q)} C_e^q y_{e}^{qn} + \beta \sum_{e \in \mathcal{E}(q)} C_e^q z_{e}^{qn(k)} \right) = 0
\]
(185)
where $|\mathcal{E}(q)|$ is the cardinality of $\mathcal{E}(q)$. Solving (185), we get
\[
x^q_* = \left( \lambda^q \mathcal{H}^q + \frac{|\mathcal{E}(q)|\beta}{M} \sum_{n=1}^N J_{qn}^T J_{qn} \right)^{-1} V
\]
(186)
where
\[
V = \frac{1}{M} \sum_{n=1}^N J_{qn}^T (A^q - \sum_{e \in \mathcal{E}(q)} C_e^q y_{e}^{qn} + \beta \sum_{e \in \mathcal{E}(q)} C_e^q z_{e}^{qn(k)}).
\]
When $M \to \infty$, we have $\mathcal{H}^q = \frac{1}{M} \sum_{n=1}^N J_{qn}^T J_{qn}$ because
\[
\nabla_{\theta_i}^2 D_{KL}(\pi_{\theta_{old}}^i(a^n|\theta^i) || \pi^i(a^n|\theta^i)) \big|_{\theta_{old}^i = \theta_{old}^i} = \sum_{n=1}^N \nabla_{\theta_i}^2 D_{KL}(\pi_{\theta_{old}}^i(a^n|\theta^i) || \pi^i(a^n|\theta^i)) \big|_{\theta_{old}^i = \theta_{old}^i}
\]
(187)
Therefore, the optimal solution $x^q_*$ can be simplified as
\[
x^q_* = \frac{1}{\lambda^q + |\mathcal{E}(q)|\beta} \mathcal{H}^{q-1} V
\]
(188)
Substituting (188) into (184), we derive

\[ p^* = \max_{\lambda^q > 0} f(x^q, \lambda^q) \]

\[ = \max_{\lambda^q > 0} - \frac{1}{M(\lambda^q + |E(q)|\beta)} \sum_{n=1}^{N} A^q n \cdot J_{qn} \cdot H^{-1}_q \cdot V \]

\[ + \frac{1}{M} \sum_{e \in E(q)} \sum_{n=1}^{N} y_{en}^{qnT} \left( \frac{1}{\lambda^q + |E(q)|\beta} C_{e}^{q} n \cdot J_{qn} \cdot H^{-1}_q \cdot V - z_{e}^{qn} \right) \]

\[ + \frac{1}{M} \sum_{e \in E(q)} \sum_{n=1}^{N} \beta \frac{1}{2} \left\| \frac{1}{\lambda^q + |E(q)|\beta} C_{e}^{q} n \cdot J_{qn} \cdot H^{-1}_q \cdot V - z_{e}^{qn} \right\|_2^2 \]

\[ + \lambda^q \left( \frac{1}{2(\lambda^q + |E(q)|\beta)^2} V^T H^{-1}_q V - \delta/N \right) \]

(189)

Based on the first order condition, the optimal \( \lambda^q \) should satisfy

\[ \frac{\partial f(x^q, \lambda^q)}{\partial \lambda^q} = \frac{1}{M(\lambda^q + |E(q)|\beta)^2} \sum_{n=1}^{N} A^q n \cdot J_{qn} \cdot H^{-1}_q \cdot V - \]

\[ \frac{1}{M(\lambda^q + |E(q)|\beta)^2} \sum_{e \in E(q)} \sum_{n=1}^{N} C_{e}^{q} n \cdot J_{qn} \cdot H^{-1}_q \cdot V - \]

\[ \frac{n(E(q))\beta}{(\lambda^q + |E(q)|\beta)^3} V^T H^{-1}_q V + (\text{Use } H^q = \frac{1}{M} \sum_{n=1}^{N} J_{qn} J_{qn}^T) \]

\[ - \frac{\beta}{M(\lambda^q + |E(q)|\beta)^2} \sum_{e \in E(q)} \sum_{n=1}^{N} C_{e}^{q} V^T H^{-1}_q J_{qn} z_{e}^{qn} + \]

\[ \frac{|E(q)|\beta - \lambda^q}{2(\lambda^q + |E(q)|\beta)^2} V^T H^{-1}_q V - \delta/N \]

(190)

\[ = \left( \frac{1}{(\lambda^q + |E(q)|\beta)^2} - \frac{1}{2(\lambda^q + |E(q)|\beta)^2} \right) V^T H^{-1}_q V - \delta/N \]

(Use \( a^T H^{-1}_q b = b^T H^{-1}_q a \) since \( H^{-1}_q \) is symmetric)

\[ = \frac{1}{2(\lambda^q + |E(q)|\beta)^2} V^T H^{-1}_q V - \delta/N \]

\[ = 0 \]

Solving Eq. (190), we get

\[ \lambda^{q*} = \sqrt{\frac{V^T H^{-1}_q V}{2\delta/N}} - |E(q)|\beta. \]

(191)
Substituting (191) into (188), the optimal solution $x^q*$ is

$$x^q* = \sqrt{\frac{2\delta/N}{V^T H_q^{-1} V}}.$$

(192)

Since $x^q = \theta^q - \theta^q_{old}$, the optimal update $\theta^q(k+1) := \arg\min_{\theta^q \in \Theta} L_\beta(\theta, z^{(k)}, y^{(k)})$ satisfies

$$\theta^q(k+1) = \theta^q_{old} + \sqrt{\frac{2\delta/N}{V^T H_q^{-1} V}}.$$

(193)

where

$$V = \frac{1}{M} \sum_{n=1}^N J_q^{nT}(A^q - \sum_{e \in \mathcal{E}(q)} C^q e y^{q(k)}_e + \beta \sum_{e \in \mathcal{E}(q)} C^q e z^{q(k)}_e).$$

(2) Next, we consider the update of $z^{qn}_e$:

$$\min_{z^{qn}_e \in Z_e} L_\beta(x, z, y), \quad q \in \mathcal{N}(e).$$

(194)

where $Z^q_e = \{z^{in}_e, z^{jn}_e | z^{in}_e + z^{jn}_e = 0, q = (i, j) \in \mathcal{N}(e)\}$. This is a convex optimization problem. The optimal value $p^*$ satisfies

$$p^* = \min_{z^{in}_e \in Z_e} \max_{\nu^q > 0} L_\beta(x, z, y) + \nu^q (z^{in}_e + z^{jn}_e).$$

(195)

Based on the first order optimality conditions, the optimal $z^{in}_e, z^{jn}_e$ should satisfy:

$$-\frac{1}{M} y^{in}_e - \frac{1}{M} \beta (C^i e J^{in} x^i - z^{in}_e) + \nu^n_e = 0$$

(196)

$$-\frac{1}{M} y^{jn}_e - \frac{1}{M} \beta (C^j e J^{jn} x^j - z^{jn}_e) + \nu^n_e = 0$$

(197)

Solving Eqs. (196) and (197), we get

$$z^{in*}_e = \frac{1}{\beta} (y^{in}_e - M \nu^n_e) + C^i e J^{in} x^i,$$

(198)

$$z^{jn*}_e = \frac{1}{\beta} (y^{jn}_e - M \nu^n_e) + C^j e J^{jn} x^j.$$

(199)

Since $z^{in*}_e, z^{jn*}_e$ should satisfy $z^{in*}_e + z^{jn*}_e = 0$, we have

$$\frac{1}{\beta} (y^{in}_e + y^{jn}_e) - \frac{2M}{\beta} \nu^n_e + (C^i e J^{in} x^i + C^j e J^{jn} x^j) = 0.$$

(200)
Therefore, the optimal dual variable $\nu^*_e$ is

$$\nu^*_e = \frac{1}{2M} \left( y_e^{(i)} + y_e^{(j)} \right) + \frac{\beta}{2M} \left( C_e^i J^m x^i + C_e^j J^m x^j \right)$$

$$= \frac{1}{2M} \sum_{q \in N(e)} \left[ y_q^{(e)} + \beta C_q^e J^m x^q \right].$$

(201)

Since $x^q = \theta^q - \theta^q_{old}$, the optimal update $z_q^{(e)(k+1)} := \arg \min_{z_q \in Z_q} L_{\beta}(\theta^{(k+1)}, z, y^{(k)})$ satisfies

$$z_q^{(e)(k+1)} := \frac{1}{\beta} \left( y_q^{(e)} - \nu_e \right) + C_q^e J^m \left( \theta^{(k+1)} - \theta^q_{old} \right),$$

where

$$\nu_e = \frac{1}{2} \sum_{q \in N(e)} \left[ y_q^{(e)} + \beta C_q^e J^m \left( \theta^{(k+1)} - \theta^q_{old} \right) \right].$$

(203)

(3) Substituting $z_q^{(e)(k+1)}$ into the update

$$y_q^{(e)(k+1)} := y_q^{(e)} + \beta \left( C_q^e J^m \left( \theta^{(k+1)} - \theta^q_{old} \right) - z_q^{(e)(k+1)} \right),$$

we get $y_q^{(e)(k+1)} = \nu_e$.

$$\theta^{(k+1)} := \theta^q_{old} + \sqrt{\frac{2\delta/N}{V^T H^{-1} V}} H^{-1} V,$$

$$z_q^{(e)(k+1)} := \frac{1}{\beta} \left( y_q^{(e)} - \nu_e \right) + C_q^e J^m \left( \theta^{(k+1)} - \theta^q_{old} \right),$$

$$y_q^{(e)(k+1)} := \nu_e,$$

where

$$V = \frac{1}{M} \sum_{n=1}^N J^{mT} (A^q - \sum_{e \in \mathcal{E}(q)} C_q^e y_q^{(e)} + \beta \sum_{e \in \mathcal{E}(q)} C_q^e z_q^{(e)}),$$

$$\nu_e = \frac{1}{2} \sum_{q \in N(e)} \left[ y_q^{(e)} + \beta C_q^e J^m \left( \theta^{(k+1)} - \theta^q_{old} \right) \right].$$

and $\mathcal{E}(q)$ is the set of all communication links that directly connect to agent $q$.

The pseudocode for the proposed algorithm is given as Algorithm 2.
### Algorithm 2 Multi-Agent Trust Region Policy Optimization

**Input:** Initial local policies $\pi^i(\theta_0^i), \forall i \in \mathcal{N}$; tolerance $\delta$

```plaintext
for $itr = 0, 1, 2, \ldots$ do
    Set the joint policy $\pi(\theta_{old}) = \prod_{i=1}^{N} \pi^i(a^i|o^i; \theta_{itr}^i)$
    for $i = 0, 1, \ldots, N$ do
        Sample a set of trajectories $D^i \sim \pi(\theta_{old})$ for agent $i$
        Compute $A^i, J^i, H^i, \forall i, n \in \mathcal{N}$ using samples
    end for
    Initial estimators $z^{(0)}$ and dual variables $y^{(0)}$
    for $k = 0, 1, 2, \ldots$ do
        Randomly select a communication link $e$
        For agents $q := (i, j) \in \mathcal{N}(e)$ at the endpoints of link $e$, update $\theta^q, z^m_e$ and $y^m_e$ according to (205)
        Update $\theta^{(k)}, z^{(k)}, y^{(k)} \rightarrow \theta^{(k+1)}, z^{(k+1)}, y^{(k+1)}$
    end for
end for
```

### 4.6 Empirical Study

This section presents the experiments designed to assess the effectiveness of the proposed MATRPO algorithm. The following questions are addressed through these experimental studies:

- Can MATRPO successfully learn cooperative policies in a fully decentralized manner, even when agents have distinct observation spaces and reward functions?

- How does MATRPO compare to other MARL methods, such as centralized training and decentralized execution algorithms? Does it perform better than independent learning?

- Is MATRPO capable of solving large-scale MARL problems? Does its performance deteriorate as the number of agents increases?

First, experiments are performed on the Multi-agent Particle Environment (MPE) [113], where two tasks are evaluated: 1) Cooperative Navigation and 2)
Cooperative Treasure Collection. These tasks demand multiple agents to cooperate with partial observations and individual rewards. Then, the proposed and baseline algorithms are assessed on the StarCraft Multi-Agent Challenge (SMAC) [114], which emphasizes decentralized micromanagement of a group of agents.

### 4.6.1 The Baseline Methods

**I-TRPO**: An independent learning algorithm where the agents learn individual policies based on TRPO in the single-agent settings. For each agent $i$, a policy $\pi^i(a_i|o^i)$ is learned by a neural network to maximize the agent’s expected return based on the local observation $o^i$. A value network is also trained for policy evaluation using $o^i$ and the private reward $r^i$.

**FDMARL**: A fully decentralized algorithm [102] based on the actor-critic design. In the algorithm, every agent learns a policy network and a consensus critic network. The policy network is trained by using typical policy gradient. The critic network is trained based on the action-value TD-learning followed by a diffusion update, where a linear combination of its neighbors’ parameter estimates are used for the agents to reach a consensus on the critic network parameters.

**MAPPO**: A centralized training and decentralized execution algorithm proposed in [115], which resembles the structure of PPO by learning a policy network and a value network for each agent. The value network is trained by using the global information on shared observations and shared rewards. The policy network is trained by using the agent’s local observations based on PPO.

For all experiments, the policy and value networks are feedforward neural networks with two hidden layers of 128 SeLU units [116]. The decentralized communication network has a ring topology. For the implementations of FDMARL and MAPPO, the recommended settings of the hyper-parameters are used. All experiments are run for 5 times with different random seeds. The simulations are
Figure 7. (a) Cooperative Navigation [113]. The agents must cover all the landmarks through coordinated movements. While all agents are punished for colliding with each other, only agent 1 is rewarded based on the proximity of any agent to each landmark. (b) Cooperative Treasure Collection [113]. The hunters need to collect treasures, which are represented by small colored circles, and deposit them into correctly colored banks. The banks are rewarded for the successful collection and depositing of treasure. The hunters do not receive any reward, but are penalized for colliding with each other.

carried out on a computer with an Intel Core i7-7700X Processor 3.60GHz and a 16 GB memory. The operating system is Ubuntu 20.04.3 LTS. The code is written in Python 3.7.6 using the deep learning toolbox TensorFlow 2.2.0 [62], and the RL toolboxes OpenAI Gym [61] and Baselines-tf2 [117].

4.6.2 Multi-Agent Particle Environment

The proposed algorithm is tested on two cooperative MPE tasks as shown in Fig. 7:

a) Cooperative Navigation: $N$ agents must reach a set of $N$ landmarks through coordinated movements. In the original version, agents are collectively rewarded based on the minimum agent distance to each landmark and individually punished when colliding with each other. To make it more challenging for fully decentralized
Table 2. Parameters of MATRPO for the MPE tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Cooper. Navigation</th>
<th>Cooper. Treas. Collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Agents</td>
<td>( N = 3 )</td>
<td>( N = 8 ) ( N = 12 )</td>
</tr>
<tr>
<td>Sim. steps per iter.</td>
<td>10k</td>
<td>10k</td>
</tr>
<tr>
<td>Stepsize ((\delta/N))</td>
<td>0.003</td>
<td>0.001</td>
</tr>
<tr>
<td>Discount ((\gamma))</td>
<td>0.995</td>
<td>0.995</td>
</tr>
<tr>
<td>GAE Para. ((\lambda))</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>ADMM iter.</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Penalty para. ((\beta))</td>
<td>1.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

MARL, the task is modified so that only agent 1 is rewarded, and the other agents do not receive any rewards while being penalized.

b) Cooperative Treasure Collection: \( N_h \) treasure hunters and \( N_b \) treasure banks search around the environment to gather treasures. The treasures are generated with different colors and re-spawn randomly upon being collected. The hunters are responsible for collecting treasures and depositing them into correctly colored banks. The hunters do not receive any reward from collecting treasures, but they are penalized for colliding with each other. The banks are rewarded based on the function:

\[
r_{\text{bank}} = 5 \left( \sum_{j \in H} \mathbb{1}_{\text{collect}}^{j} + \sum_{i \in N} \mathbb{1}_{\text{deposit}}^{i} \right) - 0.1 \min_{j,k} \{ d(h_j, t_k) | \text{iff hunter } j \text{ is empty} \}, \tag{206}
\]

where \( \mathbb{1}_{\text{collect}}^{j} \) is 1 if a treasure is collected by hunter \( j \) and 0 otherwise; \( \mathbb{1}_{\text{deposit}}^{i} \) is 1 if a treasure is deposited to bank \( i \) and 0 otherwise; \( d(h_j, t_k) \) is the distance between hunter \( j \) and treasure \( k \).

Training Performance

The proposed algorithm (MATRPO) and the baseline methods are trained for 5 million timesteps. The episode length is 100 timesteps. Other settings of the hyper-parameters are given in Table 2. For comparison, a fully centralized method (C-TRPO), which learns a global policy by reducing the problem to a single-agent
RL based on TRPO, is introduced as another baseline method. The learning curves of MATRPO and the baselines are plotted in Fig. 8. As shown in this figure, MATRPO can learn collaborative policies for the networked agents to increase returns. In addition, MATRPO achieves comparable performance with C-TRPO, which almost finds the global optimum. I-TRPO performs poorly, especially on the treasure collection task. Since the hunters are punished for colliding with each other, agents trained by I-TRPO may move far away from each other to avoid collision. As a result, some hunters may end up with being distant from the treasures.

Furthermore, MATRPO demonstrates superior performance to FDMARL and similar performance to MAPPO on these tasks. Compared to FDMARL, MATRPO and MAPPO have better learning stability since they both restrict policy updates to a trust region for each policy update. MAPPO achieves performance very close to MATRPO, but it requires a control center to coordinate the training. The proposed MATRPO algorithm is fully decentralized and less dependent on powerful computational resources in a control center, or a centralized communication network. Although FDMARL is also fully decentralized, its learning process is less stable and the learning speed is slower.

To test the scalability of MATRPO, additional experiments are conducted by increasing the number of agents to $N = 6$ and $N = 12$ for the two tasks, respectively. The learning curves are plotted in Fig. 9. It can be observed that for the tasks with a larger number of agents, MARTPO still achieves comparable or superior performance to the baselines. This result demonstrates the scalability of the proposed algorithm.

**Impact of Hyperparameters**

The influence of three key hyperparameters are analyzed:
Figure 8. Learning curves over 5 random seeds on the (a) Navigation ($N = 3$), and (b) Treasure Collection ($N = 8$) tasks.

Figure 9. Learning curves over 5 random seeds on the (a) Navigation ($N = 6$), and (b) Treasure Collection ($N = 12$) tasks.

a) KL stepsize ($\delta/N$): The impact of the KL stepsize on MATRPO’s final performance is demonstrated in Fig. 10(a). It is observed that for the Navigation task with $N = 3$ and the Treasure Collection task with $N = 8$, KL stepsize has minor effects on the performance. When $N$ increases to 6 and 12, detrimental effects can be observed for large KL stepsize, e.g. $\delta/N = 0.003$. Note that when $N$ is fixed, a large $\delta/N$ leads to a large $\delta$, which may result in an overly broad trust region for policy search, and thus possible bad updates. This result suggests that small KL stepsizes may be better in problems where there are a large number of
agents.

b) ADMM penalty parameter ($\beta$): The impact of the ADMM penalty parameter on MATRPO's final performance is demonstrated in Fig. 10(b). It is observed that the impact of the penalty parameter $\beta$ varies with the number of agents. For $N = 3$, smaller values of $\beta$ can benefit the performance. For larger $N = 6, 8, 12$, the performance may be hurt if the penalty parameter is too small ($\beta = 1$) or too large ($\beta = 9$). Since $\beta$ penalizes disagreement between agents, a small $\beta$ may produce inadequate penalty for a large group of agents to reach consensus within limited iterations; if $\beta$ is too large, the penalty may dominate the objective and have a negative impact on the policy optimization.

c) Number of ADMM iterations: The impact of the number of ADMM iterations on MATRPO's final performance is demonstrated in Fig. 10(c). It is observed that in nearly all cases, the final performance of MATRPO gets better as the number of ADMM iterations increases; when it reaches a certain point, continuing to increase the number of ADMM iterations does not result in performance improvement. The number of ADMM iterations is a key factor in the distributed ADMM algorithm, and a large value setting is usually preferred to guarantee consensus. However, once the algorithm converges, more ADMM iterations are needless and inefficient in terms of training time and computational resources.

4.6.3 StarCraft Multi-Agent Challenge

SMAC [114] is a complex environment for MARL research, focusing on micro-management in StarCraft II. The proposed MATRPO algorithm is evaluated on four SMAC scenarios (Fig. 11):

a) 2s3z: 2 Stal-units & 3 Zeal-units are controlled by a group of RL agents, competing against an opposing team of 2 Stal-units & 3 Zeal-units under the control of the built-in heuristic AI.
Figure 10. Final performance (returns of the last 100 episodes) of MATRPO using different hyper-parameter settings.
b) 3s5z: A larger version of 2s3z, where each group has 3 Stal-units & 5 Zeal-units. Note that both 2s3z and 3s5z scenarios have heterogeneous agents.

c) 10m_vs_11m: 10 Mari-units are trained to defeat 11 Mari-units that are controlled by the heuristic AI. Although the agents are homogeneous, the game is asymmetric and more challenging.

d) MMM2: 1 Medi-unit, 2 Mara-units & 7 Mari-units are trained to defeat a larger group of opponents with 1 Medi-unit, 2 Mara-units & 8 Mari-units. The Medi-units are responsible for healing its teammates and other units are responsible for attacking. This task is challenging even for human players since it requires
coordinate control of three different types of units. This scenario is both heterogeneous and asymmetric.

At each timestep, agents receive observation information within their sight range (a circular area), including opponent features, teammate features, agent movement features, and agent unit features. Attacker units are allowed to take the following actions: move[direction], attack[opponent_id], stop and no-op. As healer units, Medi-units take heal[agent_id] actions instead of attack[opponent_id]. The default setting for a shaped reward in the SMAC environment [114] is adopted, which is calculated from the damage dealt and received by agents, having opponents or teammates eliminated, and winning/losing the game. To make it challenging for decentralized training, the environment is modified such that only one agent receives the reward signal and the other agents receive 0 reward. This is different than the original setting in SMAC, where all agents receive a shared reward. Details on observation features, action space, and reward function are found in the
SMAC paper [114].

Training Performance

The proposed and baseline methods are trained for 3 million timesteps over 5 random seeds. For each random seed, we run 32 evaluation games after every 10,000 timesteps of training and take the evaluation winning rate as an index of the performance. For the hyperparameters of MATRPO, we set $\delta/N = 0.001$ and $\beta = 5$, and set the number of ADMM iterations to 500. Other hyper-parameter settings are the same as in the MPE environment.

The episode return and evaluation winning rate during the training process are plotted in Fig. 12 and Fig. 13, respectively. It can be observed that MATRPO achieves performance superior to the fully decentralized baselines, I-TRPO and FDMARL, on all scenarios in terms of both episode return and evaluation winning rate. In addition, MATRPO achieves performance comparable to the centralized training baseline method, MAPPO, on nearly all scenarios except MMM2. On the 2s3z and 3s5z scenarios, MATRPO and MAPPO completely solve the task and achieve over 95% winning rate against the built-in heuristic AI. On the 10m_vs_11m scenario, MATRPO achieves over 80% winning rate within 1.5 million training steps and performs slightly better than MAPPO. On the MMM2 scenario, MATRPO obtains close performance to MAPPO in terms of the final episode return and achieves around 60% winning rate. Although MAPPO achieves better performance on this scenario, it utilizes global observation and shared reward, and adopts parameter sharing for agents’ policy and value networks during training. However, MATRPO only uses agents’ local observation and reward signals, and neighboring communications. Compared with MAPPO, the proposed method facilitates a decentralized solution to cooperative MARL for networked agents.
4.7 Summary

This chapter has extended TRPO to partially observable multi-agent systems via proper decomposition and transformation. As a result, a fully decentralized MARL algorithm named MATRPO has been developed, which optimizes distributed policies for networked agents. Additionally, it has been proved that the distributed policy optimization in the proposed method is equivalent to single-agent TRPO when each agent’s local observation is adequate. Simulation studies on the MPE tasks and the SMAC scenarios have demonstrated that MATRPO effectively learns collaborative policies in a fully decentralized manner. Moreover, the proposed method’s effectiveness and superiority have been tested through a comparison with several baseline methods, including centralized training and decentralized execution, independent learning, and fully-decentralized training.
CHAPTER 5

Applications of Trust-Region RL Algorithms in Smart Grid

5.1 Overview

This chapter presents the development of trust-region-based RL and MARL algorithms for smart grid applications. RL and MARL algorithms have recently been applied to several power system control and management tasks in smart grid. These algorithms have shown promising results in improving the performance of the power system while maintaining its stability. One of the main advantages of RL in power system control and management is its ability to learn and adapt to changing conditions in the power system. RL algorithms can learn from historical data and adjust the control policies to optimize the system’s performance. This ability is particularly important in the smart grid, where the power system is dynamic and constantly changing.

However, RL also has some challenges in the smart grid domain. First, safe exploration is a critical challenge when applying RL to power system control and management. The conventional model-free RL framework typically permits the agent to choose any action to investigate the environment, as long as it results in an improvement in performance. However, it is unacceptable to do so in power systems since improper control behaviors could result in catastrophic consequences, such as collapse of system frequency, over/under-voltage, overflow of power lines, and even blackout. In real-life domains like this, safe exploration during the training process becomes extremely important.

Second, the increasing penetration of distributed energy sources (DES) poses a significant challenge for the efficient control and management of power systems. With a growing number of controllable devices, single-agent RL approaches may not be sufficient to manage and control the system efficiently. Therefore, multi-
agent RL approaches may be necessary to address the challenges posed by the increasing penetration of DES and to achieve efficient control and management of the power system. Multi-agent RL algorithms can enable the coordination and cooperation of multiple agents to manage and control the system more efficiently, leading to better performance and stability of the power system.

This chapter aims to develop practical applications of trust-region-based RL and MARL algorithms to enable efficient control and management of power systems. The focus of the chapter is on two specific problems: distribution system operation and distributed Volt-Var control (VVC). To address the challenge of safe exploration, the chapter will apply a trust-region-based safe RL (SRL) algorithm to solve the distribution system operation problem. This algorithm ensures safe exploration during the training process, preventing catastrophic consequences like system collapse or blackout. Furthermore, the chapter will use the MATRPO algorithm proposed in chapter 3 to solve the distributed VVC problem. This is because MATRPO enables effective coordination and cooperation of multiple agents in a decentralized manner, which makes it particularly suitable for addressing the increasing penetration of DES in distribution grids. By applying the MATRPO algorithm, the chapter aims to develop efficient and reliable VVC strategies that ensure optimal voltage and reactive power control, leading to improved system stability and reliability in the presence of DES.

5.2 Optimal Operation of Distribution Grid with Safe RL
5.2.1 Introduction

Distributed grids are traditionally operated to avoid loading and voltage limit violations for one-way power flow [118]. The increasing penetration of distributed resources violates this basic assumption and can disrupt the operation of distribution networks. For instance, intermittent RSs, such as photovoltaics and wind
turbines, may cause swing of voltages due to their rapid power variations [119]. Distributed generators (DGs) injecting real power back upstream into the distribution networks can cause voltage boosts [120] and interfere with conventional Volt/VAR control (VVC) devices. This problem may become worse if BSSs are installed and dispatched to charge and discharge intermittently. Besides, unregulated charging power of BSSs may increase the burden of distribution lines and reduce the loading margin of distribution systems.

To coordinate VVC devices and distributed resources, extensive model-based methods have been proposed, including mixed-integer non-linear programming [121], mixed-integer second-order cone programming (MISOCp) [122], two-stage stochastic programming (SP) [123], robust optimization (RO) [124, 125, 126], model predictive control [127], etc. Generally, model-based methods require an explicit physical model to formulate the distribution grid, an accurate statistical model to characterize the uncertainty, and an efficient solver to obtain the optimal solution in a limited time. Developing such a method relies on extensive domain knowledge and human-effort on model selection, parameter estimation, and algorithm design. Improper physical models or inaccurate parameters may result in performance deterioration or unrealistic solutions.

Recently, RL-based model-free methods have been used in the distribution grid operation problem because they do not require a physical model of the distribution grid. For example, in [128], a two-timescale voltage control scheme was proposed to maintain bus voltage in distribution networks, where the on-off commitment of capacitor units was optimized by using DQN. In [129], a safe DRL algorithm was developed to optimize the tap position of On-Load Tap Changer (OLTCs) and Voltage Regulator (VRs) and on/off switching of Switchable Capacitor Bank (SCBs) based on SAC. In [119], the multi-agent deep deterministic
policy optimization was adopted to solve the voltage regulation problem by coordinating the reactive power output of PV inverters. However, the VVC devices, such as SCBs, OLTCs, and VRs, have not been considered. In [130], a multi-agent DQN algorithm was developed to solve the VVC problem by controlling the VVC devices and PV inverters.

Nevertheless, the \textit{model-free} methods mentioned above did not consider the co-optimization of conventional VVC devices and the emerging DGs and Battery Storage System (BSS). As pointed out earlier, dispatchable DGs and BSSs can interfere with conventional VVC devices and undermine the operation of distribution networks. However, co-optimization of VVC devices, dispatchable DGs, and BSSs may pose several challenges to traditional RL based methods. First, there are many inequality constraints in the distribution grid operation problem, which can be tricky for \textit{reward-driven} RL methods. Second, there exist plenty of heterogeneous devices that are controlled via discrete or continuous actions. Third, distribution systems exhibit serious uncertainty and nonlinearity, which is a major challenge to the representation and learning ability of a completely \textit{model-free} algorithm.

These challenges have motivated the development of a learning-based solution to the optimal operation distribution grids using SDRL, which has been shown to be effective by Li et al. [131]. In this chapter, the optimal operation of distribution networks (OODN) is formulated in the framework of constrained MDP (CMDP). To effectively restrict the constraints and maximize the reward, the constrained policy optimization (CPO) algorithm, which is an effective safe RL algorithm [33] is employed. CPO can train complicated nonlinear policies for high-dimensional control problems with constraints on states and actions, and guarantee monotonic performance improvement and constraint satisfaction. These advantages make
CPO suitable for the distribution grid operation problem. Additionally, to deal with mixed discrete and continuous actions, a stochastic control policy defined by a joint probabilistic distribution of discrete and continuous random variables is developed. The policy can output discrete and continuous actions simultaneously by sampling from the joint distribution.

5.2.2 Problem Formulation

In the formulation, the operational horizon of a distribution network is divided into $T$ time slots. The formulation uses the subscripts $t \in \Omega_t$ to index time intervals, $i \in \Omega_n$ to index network nodes, and $ij \in \Omega_b$ to index branches, where $\Omega_t$, $\Omega_n$, and $\Omega_b$ represent the set of time intervals, the set of nodes, and the set of branches, respectively.

The next section presents the models of all controllable devices in the distribution network, and the operational limits and constraints of the distribution network. A CMDP model will be used to formulate the OODN problem, where the distribution network is considered as a black box. This means that the network topology, line parameters, and load fluctuation are unknown. Control policies and scheduling decisions have to be learned and made based on observations of the system state.

Switchable Capacitor Bank (SCB)

The control variable of an SCB is the number of units in operation. For the SCB at node $i$, the control variable is denoted by $n_{i,t}^{\text{scb}}$, which can take integer values in the range

$$0 \leq n_{i,t}^{\text{scb}} \leq \pi_i^{\text{scb}}, \quad i \in \Omega_n, \quad t \in \Omega_t,$$

(207)

where $\pi_i^{\text{scb}}$ represents the maximum number of units of the SCB. The total reactive power $Q_{i,t}^{\text{scb}}$ injected by the SCB is dependent on the susceptance of each unit, the
number of units connected at the node, and the nodal voltage. A model of \( Q_{i,t}^{scb} \) can be found in [130]. In this study, no explicit model of \( Q_{i,t}^{scb} \) is required.

**On-Load Tap Changer (OLTCs) and Voltage Regulator (VRs)**

The control variable of an OLTC/VR is the tap position. Commonly, an OLTC/VR can provide a voltage regulation from –10% to +10% with 5 or 33 steps [120]. For the OLTC/VR connected to branch \( ij \), the control variable is denoted by \( l_{vr}^{ij,t} \), which can take integer values in the range

\[
-l_{ij}^{vr} \leq l_{ij,t}^{vr} \leq l_{ij}^{vr}, \quad ij \in \Omega_b, \ t \in \Omega_t,
\]

where \( l_{ij}^{vr} \) is the maximum number of the regulation up/down steps of the OLTC/VR.

**Dispatchable DGs**

The control variables of a dispatchable DG are the active and reactive power outputs of the DG. For the DG at node \( i \), the active and reactive power outputs are denoted by \( P_{i,t}^{dg} \) and \( Q_{i,t}^{dg} \), respectively. It is assumed that dispatchable DGs operate with a restricted power factor [122]; thus \( P_{i,t}^{dg} \) and \( Q_{i,t}^{dg} \) are constrained by:

\[
0 \leq P_{i,t}^{dg} \leq p_{f}^{dg} \cdot \overline{S}_{i}^{dg}, \quad i \in \Omega_n, \ t \in \Omega_t,
\]

\[
p_{f}^{dg} \leq \cos\left(\tan^{-1}\left(\frac{Q_{i,t}^{dg}}{P_{i,t}^{dg}}\right)\right) \leq 1, \quad i \in \Omega_n, \ t \in \Omega_t,
\]

where \( \overline{S}_{i}^{dg} \) is the nominal capacity of the DG, and \( p_{f}^{dg} \) is the minimum power factor.

**Battery Storage System (BSS)**

The control variable of a BSS is the charging and discharging power. For the BSS at node \( i \), the control variable is denoted by \( P_{i,t}^{bss} \), and a positive value of \( P_{i,t}^{bss} \) represents the BSS is charging, and a negative value represents discharging. The
value of $P_{i,t}^{\text{bss}}$ is restricted in the following range:

$$-P_{i,dch}^{\text{bss}} \leq P_{i,t}^{\text{bss}} \leq P_{i,ch}^{\text{bss}}, \ i \in \Omega_n, t \in \Omega_t. \quad (211)$$

where $P_{i,ch}^{\text{bss}}$ and $P_{i,dch}^{\text{bss}}$ represent the maximum charging power and maximum discharging power, respectively. Due to the capacity limit of a energy storage, the energy $E_{i,t}^{\text{bss}}$ stored in the BSS at time interval $t$ is constrained by

$$E_i^{\text{bss}} \leq E_{i,t}^{\text{bss}} \leq E_i^{\text{bss}}, \ i \in \Omega_n, t \in \Omega_t, \quad (212)$$

where $E_i^{\text{bss}}$ is the energy capacity of the BSS; $E_{i}^{\text{bss}}$ denotes the allowable minimum energy stored in the BSS.

**Limits and Constraints of the Distribution Networks**

For the considered distribution network, the notation $V_{i,t}$ is used to represent the nodal voltage at node $i$, and $I_{i,t}$ to represent the current on branch $ij$. $P_t^s$ and $Q_t^s$ are used to denote the active and reactive power injected from the substation, respectively.

The distribution network operates with the following limits:

$$\left( P_t^s \right)^2 + \left( Q_t^s \right)^2 \leq \left( S_t \right)^2, \ t \in \Omega_t. \quad (213)$$

$$V_i \leq V_{i,t} \leq \overline{V}_i, \ i \in \Omega_n, t \in \Omega_t, \quad (214)$$

$$0 \leq I_{ij,t} \leq \overline{I}_{ij}, \ ij \in \Omega_b, t \in \Omega_t, \quad (215)$$

Equation (213) constrains the complex power exchanged at the substation between the distribution network and the upper level grid. Equation (249) restricts the nodal voltages to their upper and lower limits. Equation (250) defines the maximum branch currents.
CMDP Model

One major challenge in modeling the distribution grid operation problem is how to handle the constraints. In most model-free methods, constraints are modeled as a negative rewards in the framework of Markov decision process (MDP) by using penalty methods. However, as discussed in [132], it is difficult to determine a good penalty coefficient to balance the constraint violation and the reward. Besides, penalty methods usually cannot guarantee that constraints are strictly satisfied even if a very large penalty coefficient is used. To overcome this issue, a CMDP formulation for the distribution grid operation problem is proposed. In the following subsections, the basic elements of the proposed CMDP formulation are elaborated.

The system states at any time interval $t$ are defined as

\[ s_t = (P_{1,t-T}, \ldots, P_{1,t-1}, Q_{1,t-T}, \ldots, Q_{1,t-1}, E_{1,t}^{\text{bss}}, \ldots, P_{i,t-T}, \ldots, P_{i,t-1}, Q_{i,t-T}, \ldots, Q_{i,t-1}, E_{i,t}^{\text{bss}}, \ldots, R_{s,t-T}, \ldots, R_{s,t-1}), i \in \Omega_n, t \in \Omega_t, \]

(216)

where $P_{i,t-T}, \ldots, P_{i,t-1}$ denote the historical net active power demand at node $i$ over the past $T$ slots; $Q_{i,t-T}, \ldots, Q_{i,t-1}$ denote the historical net reactive power demand at node $i$ over the past $T$ slots; $R_{s,t-T}, \ldots, R_{s,t-1}$ denote the historical energy prices over the past $T$ slots. The net active and reactive power demand at node $i$ are calculated by

\[ P_{i,t} = P_{i,t}^d - P_{i,t}^{\text{rs}}, \quad Q_{i,t} = Q_{i,t}^d, \quad i \in \Omega_n, t \in \Omega_t, \]

(217)

where $P_{i,t}^{\text{rs}}$ represents the active power generated by the RS at node $i$, $P_{i,t}^d$ denotes the active power demand at node $t$, and $Q_{i,t}^d$ denotes the reactive power demand at node $i$. To sufficiently utilize RSs, it is assumed that RSs are non-dispatchable sources operating with unity power factor.
The actions include the number of in-operation units of the SCBS, the tap position of OLTCs/VRs, the active and reactive power of DGs, and the charging/discharging power of BSSs:

\[ a_t = (n_{scb1,t}, l_{vr1,t}, P_{dg1,t}, Q_{dg1,t}, P_{bss1,t}, \ldots, n_{scbi,t}, l_{vri,t}, P_{dgi,t}, Q_{dgi,t}, P_{bssi,t}, \ldots), i \in \Omega_n, t \in \Omega_t. \]  

(218)

The reward is the negative sum of the purchasing costs of energy at the substation and the fuel costs of DGs

\[ r_t = - \left( R_s P_s \Delta t + \sum_{i \in \Omega_n} [a^d_{dg_i}(P^d_{it})^2 + b^d_{dg_i}P^d_{it} + c^d_{dg_i}] \Delta t \right), \]  

(219)

where \( a^d_{dg_i}, b^d_{dg_i}, \) and \( c^d_{dg_i} \) are generation cost coefficients of the DG at node \( i. \)

The constraint reflects the degree of constraint violations, which is defined by

\[ c_t = C^s_t + \sum_{i \in \Omega_n} C^v_{i,t} + \sum_{ij \in \Omega_b} C^l_{ij,t} + \sum_{i \in \Omega_n} C^d_{i,t} + \sum_{i \in \Omega_n} C^b_{i,t}. \]  

(220)

The first term \( C^s_t \) measures the violation of the substation capacity constraint (213), which is calculated by:

\[ C^s_t = \max(0, (P^s_t)^2 + (Q^s_t)^2)/S^s - 1). \]  

(221)

The second term \( C^v_{i,t} \) reflects the degree of violation of the nodal voltage limits (249), which is calculated by:

\[ C^v_{i,t} = \max(0, V_{i,t} - V_i) + \max(0, V_i - V_{i,t}). \]  

(222)

The third term \( C^l_{ij,t} \) reflects the degree of violation of branch loading limits (250), which is calculated by:

\[ C^l_{ij,t} = \max(0, I_{ij,t}/T_{ij} - 1). \]  

(223)

The fourth term \( C^d_{i,t} \) assesses the violation of the power factor constraint (210), which is calculated by:

\[ C^d_{i,t} = \max(0, \left(\frac{P_{ft}^d_{i}}{Q_{ft}^d_{i}}\right) - \cos(\tan^{-1}(Q_{ft}^d_{i}/P_{ft}^d_{i}))). \]  

(224)

108
The fifth term $C_{i,t}^{\text{bss}}$ measures the violation of the BSS capacity constraint (212), which is calculated by:

$$C_{i,t}^{\text{bss}} = \left[ \max(0, E_{bss}^{i,t} - \bar{E}_{i,t}) + \max(0, \bar{E}_{bss}^{i,t} - E_{i,t}^{bss}) \right] / \bar{E}_i.$$  (225)

Define $J(\pi)$ as the expected discounted return from time step 0 to $T$, which is calculated by

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[ r_0 + \gamma r_1 + \cdots + \gamma^{T-1} r_T \right]$$

where $\gamma \in [0,1]$ is the discount factor, $\tau$ denotes a trajectory ($\tau = (s_0, a_0, a_1, \ldots, s_T)$), $\mathbb{E}_{\tau \sim \pi} [\cdot]$ is the expected value of the distribution over the trajectory $\tau$, and $\tau \sim \pi$ is short for $s_0 \sim \mu, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$. The trajectory $\tau$ is a random process, in which the initial state $s_0$ follows the distribution $\mu$, denoted by $s_0 \sim \mu$; the action $a_t$ follows the policy distribution $\pi(\cdot|s_t)$, denoted by $a_t \sim \pi(\cdot|s_t)$; the next state $s_{t+1}$ follows the state transition probability distribution $p(\cdot|s_t, a_t)$, denoted by $s_{t+1} \sim p(\cdot|s_t, a_t)$. Since the CMDP formulation is model-free, the initial state distribution $\mu$ and the state transition probability distribution $p(\cdot|s_t, a_t)$ are unknown.

In addition, define $J_C(\pi)$ as the expected discounted constraint violations (also denoted as C-return) from time step 0 to $T$, which is calculated by

$$J_C(\pi) = \mathbb{E}_{\tau \sim \pi} \left[ c_0 + \gamma c_1 + \cdots + \gamma^{T-1} c_T \right].$$

### 5.2.3 Safe Deep Reinforcement Learning Solution

This section introduces a trust-region-based SRL algorithm to solve the CMDP. To handle the mixed discrete and continuous action space, a stochastic policy based on a multivariate joint distribution is designed.

**The CPO Algorithm**

For MDP problems, local policy search are usually used to find an optimal policy. Local policy search algorithms optimize a policy by iteratively searching
for an improved one in a neighborhood of the most recent iterate $\pi_k$ to maximize $J(\pi)$:

$$
\pi_{k+1} = \arg \max_{\pi \in \Pi} J(\pi) \quad \text{s.t. } D(\pi, \pi_k) \leq \delta
$$

(226)

where $D$ is a distance measure and $\delta$ defines the size of the neighborhood. A typical local policy search algorithm is trust region policy optimization [25], which uses the average KL-Divergence $\bar{D}_{KL}(\pi||\pi_k) = \mathbb{E}_{s \sim \rho_k}[D_{KL}(\pi(\cdot|s)||\pi_k(\cdot|s))]$ to measure the searching area. Another one is the standard policy gradient, which uses the $l$-2 measure $D(\pi, \pi_k) = ||\theta - \theta_k||^2$ (policy $\pi$ parameterized by $\theta$) and maximizes a linearized objective $J(\pi_k) + \nabla_\theta J(\pi)(\theta - \theta_k)$ in the neighborhood of $\pi_k$.

For the CMDP problem, the searching area in each iteration is additionally confined by the constraint:

$$
J_C(\pi) \leq d.
$$

(227)

This makes local policy search algorithms difficult to implement because it requires evaluation of the constraint function $J_C(\pi)$ to determine whether a proposed policy is feasible.

To address this problem, CPO uses surrogate functions that are easy to evaluate from samples collected on $\pi_k$ to approximate the constraint and the objective [33]. The surrogate functions are expressed by

$$
\tilde{J}(\pi) = J(\pi_k) + \mathbb{E}_{s \sim \rho_k} [A^{\pi_k}(s, a)] - \alpha_k \sqrt{D_{KL}(\pi||\pi_k)}
$$

(228)

$$
\tilde{J}_C(\pi) = J_C(\pi_k) + \mathbb{E}_{s \sim \rho_k} [A^C(\pi_k)] + \beta_k \sqrt{D_{KL}(\pi||\pi_k)}
$$

(229)

where $\alpha_k = \max_s [\mathbb{E}_{a \sim \pi}[A^{\pi_k}(s, a)] \cdot \sqrt{2}\gamma/(1 - \gamma)]$, $\beta_k = \max_s [\mathbb{E}_{a \sim \pi}[A^C(\pi_k)] \cdot \sqrt{2}\gamma/(1 - \gamma)]$, $A^{\pi_k}(s, a) = Q^\pi(s, a) - V^\pi(s)$ is the advantage function, and
\( A_C^\pi(s, a) = Q_C^\pi(s, a) - V_C^\pi(s) \) is the advantage functions with respect to the constraint.

According to [33], the surrogate functions satisfy the following properties:

\[
\tilde{J}(\pi) \leq J(\pi), \quad \tilde{J}_C(\pi) \geq J_C(\pi).
\] (230)

This means that \( \tilde{J} \) is a lower bound of the objective and \( \tilde{J}_C(\pi) \) is an upper bound of the constraint. If the objective and the constraint with their surrogates and update the policy are replaced by

\[
\pi_{k+1} = \arg \max_{\pi \in \Pi} \tilde{J}(\pi) \quad \text{s.t.} \quad \tilde{J}_C(\pi) \leq d,
\] (231)

we can improve the worst-case performance and bound the worst-case constraint violation. This means that the policy update (231) can guarantee monotonic improvement in objective performance and constraint satisfaction.

One difficulty in applying the policy update (231) is the computation of the coefficients \( \alpha_k \) and \( \beta_k \) because it involves solving the optimization \( \max_s | \cdot | \). To solve this problem, CPO adopts a trust region constraint on the KL-Divergence instead of penalizing it by \( \alpha_k \) and \( \beta_k \). Consequently, the policy update (231) can be transformed into

\[
\pi_{k+1} = \arg \max_{\pi \in \Pi} \mathbb{E}_{s \sim \rho_k} [A_{C}^\pi(s, a)]
\]

\[
s.t. \quad J_C(\pi_k) + \mathbb{E}_{a \sim \pi_k} [A_C^\pi(s, a)] \leq d \quad \text{and} \quad \bar{D}_{KL}(\pi || \pi_k) \leq \delta.
\] (232)

Since the policy \( \pi(a|s) \) is a function of \( s \) and \( a \), it is difficult to directly optimize \( \pi(a|s) \) using (232). Next, a neural network is designed to approximate the policy and optimize the neural network’s weights to improve the policy.
Parameterized Policy for Discrete and Continuous Actions

In the distribution grid operation problem, SCBs, OLTCs, and VRs operate in discrete steps whereas dispatchable DGs and BSSs operate with continuous outputs. Therefore, the action space contains both discrete and continuous control variables:

\[ a_t = (n_{scb}^{i,t}, l_{vr}^{i,t}, P_{dg}^{i,t}, Q_{dg}^{i,t}, P_{bss}^{i,t}), \forall i \in \Omega_n. \]  \hspace{1cm} (233)

To deal with the mixed discrete and continuous action space, the policy is approximated by using a joint distribution:

\[ \pi(a_t|s_t) = \prod_{i \in \Omega_n} p(n_{scb}^{i,t}|s_t)p(l_{vr}^{i,t}|s_t)f(P_{dg}^{i,t}|s_t)f(Q_{dg}^{i,t}|s_t)f(P_{bss}^{i,t}|s_t), \]  \hspace{1cm} (234)

where \( p(\cdot|s_t) \) is the probability mass function (PMF) of a categorical distribution; \( f(\cdot|s_t) \) is the probability density function (PDF) of a normal distribution. Note that the actions \((n_{scb}^{i,t}, l_{vr}^{i,t}, P_{dg}^{i,t}, Q_{dg}^{i,t}, P_{bss}^{i,t})\) are considered as random variables and assumed to be independent from each other. In practice, this assumption generally holds because all actions are executed simultaneously at each time slot. The probability of each action being executed is dependent on only the system state \( s_t \) but not the observation of other actions.

For a discrete control variable \( x \) whose sample space has \( C \) individually identified items, the probability of \( x \) taking on the value \( c \) given \( s_t = s \) is

\[ p(x = c|s_t = s) = \prod_{c=1}^{C} p_c(s)^{[x=c]} \]  \hspace{1cm} (235)

where \( p_c(s) \) represents the probability of seeing element \( c \) given \( s \), and \([x = c]\) evaluates to 1 if \( x = c \), 0 otherwise.

For a continuous control variable \( y \), the conditional probability given \( s_t = s \) is

\[ f(y|s_t = s) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(y - \mu(s))^2}{2\sigma^2} \right) \]  \hspace{1cm} (236)
where $\mu(s)$ is the mean and $\sigma$ is the standard deviation. Combining (235) and (236), the probability of an action $a_t$ given $s_t = s$ can be calculated by (256). Also, we can generate discrete and continuous actions by sampling from the joint distribution (256).

The challenge now is to determine the optimal distribution parameters $p_c(s)$, $\mu(s)$, and $\sigma$ to solve the CMDP. Since the distribution parameters $p_c(s)$ and $\mu(s)$ depends on the state $s$, neural networks are used to learn these distribution parameters.

For discrete control variables, the probabilities $(p_1(s), p_2(s), \ldots, p_C(s))$ are calculated by a softmax function:

$$
p_c(s) = \frac{e^{z_c}}{e^{z_1} + e^{z_2} + \ldots + e^{z_C}}, \quad \forall c \in \{1, \ldots, C\},
$$

(237)

where $z = [z_1, \ldots, z_C]^T$ is computed by

$$
z = W_c \cdot logits(s) + b_c,
$$

(238)

where $W_c$ and $b_c$ are trainable parameters.

For continuous control variables, the mean $\mu(s)$ and the standard deviation $\sigma$ is calculated by:

$$
\mu(s) = W_n \cdot logits(s) + b_n,
$$

$$
\sigma = \exp(r_n),
$$

(239)

where $W_n$, $b_n$, and $r_n$ are trainable parameters. The notation $logits(s)$ represents the feature vector extracted from $s$ by the neural network.

**Practical Implementation**

The trainable parameters of the neural network, denoted by the vector $\theta$, are utilized to parameterize $\pi(a_t|s_t; \theta)$. Also, the previous notations depending on $\pi$ will be replaced with the function of $\theta$, e.g. $J(\theta) := J(\pi)$, $J_C(\theta) := J_C(\pi)$, $\rho_\pi := \rho_\theta$.
and $\bar{D}_{KL}(\pi||\pi_k) := \bar{D}_{KL}(\theta||\theta_k)$. The parameters $\theta$ are then updated by solving the optimization problem (232).

To efficiently solve (232) in practice, a convex approximation of (232) is used. Note that in a local neighborhood of $\theta_k$, the expected advantage functions can be well approximated by

$$E_{s \sim \rho_{\theta_k}} \left[ A^\theta_k(s, a) \right] \approx E_{s \sim \rho_{\theta_k}} \left[ A^{\theta_k}(s, a) \right] + g^T(\theta - \theta_k)$$  \hspace{1cm} (240)

$$E_{s \sim \rho_{\theta_k}} \left[ A^{\theta_k}_C(s, a) \right] \approx E_{s \sim \rho_{\theta_k}} \left[ A^{\theta_k}_C(s, a) \right] + b^T(\theta - \theta_k)$$  \hspace{1cm} (241)

where $g$ is the gradient $\nabla_\theta E_{s \sim \rho_{\theta_k}, a \sim \theta}[A^{\theta_k}(s, a)]|_{\theta = \theta_k}$, and $b$ is the gradient $\nabla_\theta E_{s \sim \rho_{\theta_k}, a \sim \theta}[A^{\theta_k}_C(s, a)]|_{\theta = \theta_k}$. Also, the policy divergence can be well approximated by

$$\bar{D}_{KL}(\theta||\theta_k) = \bar{D}_{KL}(\theta_k||\theta_k) + \nabla_\theta \bar{D}_{KL}(\theta||\theta_k)|_{\theta = \theta_k}(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k)$$  \hspace{1cm} (242)

where $H$ is the hessian $\nabla^2_{\theta \theta} \bar{D}_{KL}(\theta||\theta_k)|_{\theta = \theta_k}$.

Since $E_{s \sim \rho_{\theta_k}, a \sim \theta}[A^{\theta_k}(s, a)] = E_{s \sim \rho_{\theta_k}, a \sim \theta}[A^{\theta_k}_C(s, a)] = \bar{D}_{KL}(\theta_k||\theta_k) = \nabla_\theta \bar{D}_{KL}(\theta||\theta_k)|_{\theta = \theta_k} = 0$, the problem (232) is well approximated by

$$\theta_{k+1} = \arg\max_{\theta \in \Theta} g^T(\theta - \theta_k)$$  \hspace{1cm} (243)

s.t. $c + b^T(\theta - \theta_k) \leq 0$

$$\frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta.$$  

where $g$ is the gradient $\nabla_\theta E_{s \sim \rho_{\theta_k}, a \sim \theta}[A^{\theta_k}(s, a)]|_{\theta = \theta_k}$, $b$ is the gradient $\nabla_\theta E_{s \sim \rho_{\theta_k}, a \sim \theta}[A^{\theta_k}_C(s, a)]|_{\theta = \theta_k}$, and $c = J_C(\theta_k) - d$. The problem is convex and has a closed-form solution:

$$\theta_{k+1} = -\frac{1}{\lambda^*} H^{-1}(g + \nu^* b)$$  \hspace{1cm} (244)

where $\lambda^*$ and $\nu^*$ are the optimal dual solutions.
In practice, the values of $g$, $b$, $H$, and $c$ are estimated by using their sampling means. The generalized advantage estimation (GAE) [112] is used to estimate the advantage functions $A^\theta_k(s, a)$ and $A^\theta_C(s, a)$:

\[
\hat{A}^\theta_k(s_t = s, a_t = a) = \epsilon_t + (\gamma \lambda)\epsilon_{t+1} + \cdots + (\gamma \lambda)^{T-t+1}\epsilon_{T-1}
\]

where $\epsilon_t = r_t + \gamma V^\pi_k(s_{t+1}) - V^\pi_k(s_t)$,

\[
\hat{A}^\theta_C(s_t = s, a_t = a) = \epsilon_C^t + (\gamma \lambda)\epsilon_{C,t+1} + \cdots + (\gamma \lambda)^{T-t+1}\epsilon_{C,T-1}
\]

where $\epsilon_C^t = c_t + \gamma V^\pi_C(s_{t+1}) - V^\pi_C(s_t)$, where $\lambda$ is the GAE parameter. The value functions $V^\pi_k(s_t)$ and $V^\pi_C(s_t)$ are approximated by a neural network parameterized by $\phi$, which is trained by minimizing a square-error loss:

\[
L_\phi = \sum_t \left[ (V^\pi_k(s_t; \phi) - V^{t\text{arg}}_t)^2 + (V^\pi_C(s_t; \phi) - V^{t\text{arg}}_{C,t})^2 \right],
\]

where $V^{t\text{arg}}_t = \sum_{l=t}^{T-1} \gamma^{l-t}r_l$ and $V^{t\text{arg}}_{C,t} = \sum_{l=t}^{T-1} \gamma^{l-t}c_l$. The pseudocode of the CPO-based methods for the distribution grid operation problem is presented in Alg. 3.

### 5.2.4 Case studies

The performance of the proposed learning method is evaluated on a modified IEEE-34 node test feeder system [133]. Figure 14 shows that the system contains two VRs with 33 tap positions ($-16; -15; \ldots; 0; +1; +2; \ldots; +16$), a regulation range of $-10\%$ to $+10\%$ ($0.625\%$ per tap), two SCBs of $0.48$ MVAR with four units ($0.12$ MVAR/unit) at nodes 864 and 840, two dispatchable DGs with capacities of $0.825$ MVA and $0.625$ MVA, a minimum power factors of $0.8$ at node 848 and 890, one BSS at node 810 with a capacity of $2$ MWh, a maximum charging/discharging power of $0.5$ MW, three photovoltaic RSs with power peaks of $0.1$ MW at nodes 822, 856, and 838, and three wind RSs with power peaks of $0.1$ MW at nodes 822, 856, and 838, and three wind RSs with power peaks of $0.1$ MW at nodes 822, 856, and 838, and three wind RSs with power peaks of $0.1$ MW at nodes 822, 856, and 838.
Algorithm 3 CPO-based Learning Algorithm for distribution grid operation

1: Initialize network parameters $\theta_0$, $\phi_0$.
2: for $k = 1, 2, \ldots$ do
3: Initialize an empty set $D$
4: for $n = 1, 2, \ldots, N$ do in parallel
5: Initialize an empty set $D$
6: for $t = 0, 1, \ldots, T - 1$ do
7: Choose $a_t \sim \pi(\cdot|s_t; \theta_k)$ and do simulation
8: Observe $s_{t+1}$, $r_t$, and $c_t$
9: end for
10: Store the trajectory $\tau = (s_0, a_0, r_0, c_0, s_1, \ldots)$ in $D$
11: end for
12: Use the sampled trajectories $\tau$ in $D$ to calculate
13: $\{\hat{A}_{\theta}^k(s_0, a_0), \ldots, \hat{A}_{\theta}^k(s_T, a_T)\}$ according to (245)
14: Use the sampled trajectories $\tau$ in $D$ to calculate
15: $\{\hat{A}_{\phi}^k(s_0, a_0), \ldots, \hat{A}_{\phi}^k(s_T, a_T)\}$ according to (246)
16: Use the sampled rewards $(r_0, \ldots, r_T)$ in $D$ to calculate
17: $\{V_{0, \text{targ}}, \ldots, V_{T, \text{targ}}\}$ according to $V_{t, \text{targ}}^{\text{targ}} = \sum_{l=t}^{T-1} \gamma^{l-t} r_l$
18: Use the samples $(c_0, \ldots, c_T)$ in $D$ to calculate
19: $\{V_{C,0, \text{targ}}, \ldots, V_{C,T, \text{targ}}\}$ according to $V_{C,t, \text{targ}} = \sum_{l=t}^{T-1} \gamma^{l-t} c_l$
20: Estimate $g$, $b$, $H$ and $c$ using their sampling means
21: Update $\phi_{k+1} \leftarrow \phi_k - \alpha \nabla_{\phi} L_{\phi|\phi=\phi_k}$ using (247)
22: Update $\theta_{k+1}$ by (244)
23: end for

822, 826, and 838. The objective is to minimize the total cost of energy purchased from the substation and the dispatchable DGs. The capacity of the substation is 2.5 MVA. The nodal voltages are bounded within $0.95 \text{ p.u.} - 1.05 \text{ p.u.}$ It is also assumed that the BSS has an efficiency of $\eta_{i, \text{ch}}^{\text{bss}} = \eta_{i, \text{dch}}^{\text{bss}} = 0.98$, and an allowable minimum energy of 0.2 MWh.

In this study, we utilized a three-year time-series dataset from the California Independent System Operator (CAISO) containing information on electricity prices, load demand, and RS power generation. The training set consists of the first two years of data (2018-2019), while the remaining year (2020) is used as the test set. To ensure proper scaling, the load demands are first normalized to unity and then multiplied by a base power. This approach ensures that the load data is
Figure 14. A modified IEEE-34 node test feeder system [133].

Table 3. Parameter settings of the proposed method.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint tolerance ($d$)</td>
<td>1e-3</td>
</tr>
<tr>
<td>GAE parameter ($\lambda$)</td>
<td>0.95</td>
</tr>
<tr>
<td>Discount factor ($\gamma$)</td>
<td>0.995</td>
</tr>
<tr>
<td>KL-Divergence Limit ($\delta$)</td>
<td>0.02</td>
</tr>
<tr>
<td>Stepsize of value network update ($\alpha$)</td>
<td>3e-4</td>
</tr>
<tr>
<td>Number of steps in one episode ($T$)</td>
<td>24</td>
</tr>
<tr>
<td>Number of total episodes</td>
<td>1M</td>
</tr>
</tbody>
</table>

scaled appropriately for the considered system. To ensure appropriate scaling for each node, the base power used for scaling the load demands varies from node to node. In this study, we utilized the nodal load data of the standard IEEE 34-node system [133] as the base power for scaling. The data on the photovoltaic and wind RSs were also processed using the same scaling approach to ensure consistency in the data processing methodology. The cost coefficients of the dispatchable DGs are $a_{dg1}^{d} = 100\$/MWh, $b_{dg1}^{d} = 72.4\$/MWh, $c_{dg1}^{d} = 0.5\$/h for DG1; $a_{dg2}^{d} = 100\$/MWh, $b_{dg2}^{d} = 51.6\$/MWh, $c_{dg2}^{d} = 0.46\$/h for DG2. The scheduling horizon is $T = 24$h.
Figure 15. Learning curves over 5 runs with different random seeds on the modified IEEE-34 system using different algorithms: a) constraint value, and b) return.

The hyperparameters used in the proposed method are presented in Table 4. The policy network consists of three layers of 128 ReLU neurons, and the size of the logits is 64. The value network architecture is identical to that of the policy network. The network parameters \( \theta \) and \( \theta^v \) are orthogonally initialized. The simulations were conducted on a workstation equipped with an Intel Core i7-7800X Processor @ 3.50GHz and running Ubuntu 20.04.3 LTS. The code was implemented in Python 3.7.6, utilizing the deep learning package TensorFlow 2.2.0[62], and DRL packages OpenAI Gym [61] and Baselines-tf2 [117].

Training Performance

In this study, we compare the performance of DDPG, PPO, and SAC algorithms. As DDPG and SAC are only capable of handling continuous actions, the control decisions are rounded to the nearest integer. Meanwhile, PPO utilizes the mixed joint distribution policy in the proposed method to handle both discrete and continuous actions. To ensure constraints are met, a penalty term is added to the reward for all three algorithms,

\[
    r_t := r_t + \rho \cdot c_t \tag{248}
\]
where $\varrho$ is the penalty coefficient, which is set to 500. Each method is run for 5 times with different random seeds.

Figure 15 compares the training performance of the proposed CPO and the state-of-the-arts DRL methods over 5 independent runs with different random seeds. From Figure 15(a), it is observed that the constraint value of CPO (the purple line) decreases quickly to a level at $1e^{-3}$, which is an acceptable tolerance defined in the parameter $d$. After that, the constraint value remains at this level whereas the reward starts to increase steadily, as shown in Figure 15(b). This means that CPO learned a safe operation strategy for the distribution network in the first place, and then consistently improved it without undermining the safety of the operation. This makes the proposed method more practical than the other DRL based approaches. As observed from Figure 15(a), DDPG and PPO failed to learn a safe operation strategy because of the large constraint value. Besides, even though SAC outperforms DDPG and PPO by restricting the constraint to a very small level, mostly below $1e^{-2}$, the performance is not uniformly convergent on the entire training episodes. Also, as shown in Figure 15(b), the reward of SAC
does not improve much during the training process. This may result from the reason that the constraint is over-penalized.

Test Performance

After training, the well-trained model is tested on the test set. To verify the optimality of the proposed approach, it is compared with a model-based method, wherein the distribution grid operation problem is formulated as a MISOCP using DistFlow model [134]. In the MISOCP method, it is assumed that all uncertainties can be accurately predicted. To solve the MISOCP model, the optimization toolbox PySCIPOpt [135] is used.

Figure 16 presents a comparison of the testing results obtained by the proposed CPO method and the benchmark methods. In the comparison, the constraint value is calculated by $\sum_{t=0}^{T-1} c_t$ and the operational cost is calculated by $-\sum_{t=0}^{T-1} r_t$. As shown in the figure, the proposed CPO method outperforms the state-of-the-art DRL methods by achieving the lowest cost and the least constraint violation. Specifically, in Figure 16(a), it can be observed that for the CPO algorithm, there are only a few cases of constraint violation on the whole-year testing data. It is important to note that it is impossible to guarantee the safety of hourly-ahead operation in any situation due to the existence of uncertainty. However, CPO can learn to safely operate the distribution system in most situations and guarantee near-constraint satisfaction. Although PPO and SAC can also confine the constraint violations to some extent, their performance varies largely on different days. Figure 16(b) shows the cumulative operational cost on the 366 testing days. The total operating costs for DDPG, PPO, SAC, and CPO are $373.95K$, $292.01K$, $503.21K$, and $252.06K$, respectively. Compared to DDPG, PPO, and SAC, CPO reduces the cost by 32.5%, 13.6%, and 99.6%, respectively. It is notable that although DDPG obtains a lower cost than SAC, it causes serious violations.
of the operating constraints, which makes it impossible to implement in real distribution networks. The total operating cost of MISOCP is $224.30K. Compared to CPO, the MISOCP method only reduces the operating cost by 11.01% even though it uses perfect forecast information of the uncertainty. It is worth mentioning that the performance of MISOCP is ideal and cannot be achieved in practice. These results confirm the effectiveness of the proposed CPO-based method against uncertainty in the operation of distribution networks.

IEEE-123 Node System

To test the scalability of the proposed method, experiments on a modified IEEE-123 node system [136] are conducted. Figure 17 presents the system, which contains two OLTCs with 5 tap positions (−2; −1; 0; +1; +2) and a regulation range of −10% to +10% (2.5% per tap); two VRs with 33 tap positions (−16; −15; . . . ; 0; +1; +2; . . . ; +16) and a regulation range of −10% to +10% (0.625% per tap); two SCBs of 1.2 MVAR with four units (0.3 MVAR/unit) at
nodes 108 and 76; three dispatchable DGs with capacities of 0.825 MVA, 0.625 MVA, and 0.625 MVA and a minimum power factors of 0.8 at node 24, 94, and 114, respectively; two BSSs both with a capacity of 2 MWh and a maximum charging/discharging power of 0.5 MW at nodes 20 and 56, respectively; five photovoltaic RSs with power peaks of 0.1 MW at nodes 22, 250, 41, 450, and 39, respectively; five wind RSs with power peaks of 0.1 MW at nodes 4, 59, 46, 75, and 83, respectively. The capacity of the substation is 5 MVA. The nodal voltages are bounded within 0.95 p.u. - 1.05 p.u. It is also assumed that the BSS has an efficiency of $\eta_{\text{bss},i,ch} = \eta_{\text{bss},i,dch} = 0.98$, and an allowable minimum energy of 0.2 MWh. The cost coefficients of the dispatchable DGs are $a_{\text{dg}}_1 = 100\$/MWh^2$, $b_{\text{dg}}_1 = 72.4\$/MWh$, $c_{\text{dg}}_1 = 0.5\$/h$ for DG1; $a_{\text{dg}}_2 = a_{\text{dg}}_3 = 100\$/MWh^2$, $b_{\text{dg}}_2 = b_{\text{dg}}_3 = 51.6\$/MWh$, $c_{\text{dg}}_2 = c_{\text{dg}}_3 = 0.46\$/h$ for DG2 and DG3.

For the CPO algorithm, we use the same parameter settings as presented in Table 4. The training and test datasets are the same as those used in the IEEE-34 node system. The policy and value network have three layers of 256 ReLU neurons and the size of the logits is 128.

**Training Performance**

Figure 18 compares the training performance of CPO and other DRL methods over 5 independent runs with different random seeds. From Figure 18(a), it can be observed that CPO successfully learned a safe policy to restrict the constraint value to $1e-3$ in less than 20k episodes. However, the benchmark DRL methods, especially DDPG and PPO, fail to do so and lead to large constraint violations. In addition, from Figure 18 (b), it can be observed that DDPG and CPO obtain higher returns during the training than PPO and SAC do. Compared to DDPG, however, CPO learns to improve policy without violating operating constraints, which makes it more practical to be trained in real distribution networks.
Figure 18. Learning curves of different algorithms over 5 runs with different random seeds on the modified IEEE-123 node system: a) constraint value, and b) return.

Figure 19. Test performance of different algorithms on the modified IEEE 123 system: a) distribution of constraint violation, and b) cumulative cost.
Test Performance

Figure 19 compares the testing results of CPO and the benchmark methods on IEEE-123 node system. In the comparison, it is observed that CPO outperforms the other DRL methods in terms of operating cost reduction and constraint satisfaction. As shown in Figure 19 (a), CPO results in almost no constraint violation except a few testing days. DDPG and PPO, however, causes serious constraint violation on most of the testing days and is infeasible to implement in practice. Besides, although SAC shows comparable performance to CPO in handling constraints, it results in high operating costs, as shown in Figure 19 (b). The total operating cost for DDPG, PPO, SAC, and CPO are $520.09K, $773.85K, $930.59K, and $498.33K, respectively. Compared to DDPG, PPO, and SAC, CPO reduces the operational cost by 4.1%, 35.6%, and 46.4%, respectively. Besides, the total operating cost of MISOCP is $434.87K, which is only 12.7% lower than that of CPO.

5.3 Distributed Volt-Var Control with MATRPO

5.3.1 Introduction

VVC is a power system control technique used to regulate voltage and reactive power levels within acceptable limits. VVC involves adjusting the voltage magnitude and phase angles of power system components, such as OLTCs, VRs, and SCs [137], in response to changes in power demand and supply. The goal of VVC is to maintain power quality standards and improve system efficiency. VVC is particularly important in distribution systems with distributed energy resources (DERs) because these resources can cause voltage fluctuations and other power quality issues.

Traditional VVC methods generally adopt centralized control and uses model-based algorithms. For example, In [138], a MILP model for VVC of unbalanced
distribution networks is formulated, and the characteristics of different components, e.g., transformer winding connections, types of loads, embedded generators are considered. In [139], an OPF-based two-layer VVC method was proposed, where set points for VV control devices were calculated by solving MILP in the first layer and an optimal solution was obtained by tuning the coefficients of VV control devices and solving to approximate nonlinear OPF problem. In [140], a model predictive control-based VVC method was designed and the variabilities of distributed generation output and load-to-voltage relationship were considered. In [141], a centralized VVC method based on oriented discrete coordinate descent algorithm was proposed. The algorithm in [141] used fast power flow model and had good convergence speed for real-time application but only suboptimal solution could be obtained. In [142], the VVC problem in distribution system with uncertainty of wind turbines was solved using a fuzzy optimization method. In [143], the VVC problem was decomposed into two sub problems: one subproblem at substation level aiming to optimize the dispatch of OLTCs and SCs, the other subproblem at feeder level aiming to optimize SCs control along with feeders. However, centralized VVC algorithms can provide limited scalability because all computation tasks have to be carried out in a computing server. Moreover, centralized VVC algorithms highly rely on high-bandwidth and reliable communication links to estimate the system state and transmitting control signals, which can be challenging for practical implementation in real-world distribution systems.

To overcome these issues, distributed VVC algorithms have been investigated in recent years based on multi-agent system (MAS) and distributed optimization techniques. For example, In [144], a multi-agent-based distributed VVC algorithm was proposed, where voltage regulator agents and shunt capacitor agents were adjusted cooperatively to obtain the feasible and optimal solution to VVC problem.
In [145], a two-level distributed VVC control framework with high penetration of PV inverters was developed. In the real-time scale level, each PV inverter was aggregated using consensus and droop control method. In the 15-minute timescale level, the reactive power of PV inverters was dispatched for minimum power loss. In [146], by decomposing distribution system into a regional sub-system, a distributed algorithm to solve VVC problem in an unbalanced distribution system with DG was introduced. In [147], an ADMM-based distributed algorithm for VVC problem was investigated and taken into account both continuous control variables. Nevertheless, the methods mentioned above require a specific physical model to formulate the distribution system. The performance of these methods may deteriorate when implementing in practice due to model inaccuracy.

To address these limitations, learning-based methods that do not require any physical model have been proposed recently. These methods can directly learn a control policy by using DRL techniques, and have been successfully used in many power system applications, such as microgrid energy management [148], demand response [149], electric vehicle charging management [150, 132] etc. For distributed VVC, some multi-agent RL/DRL algorithms have been proposed in state-of-the-art methods. For example, in [151], a distributed Q-learning was proposed based on global information discovery to solve the optimal reactive power dispatch. In [130], a model-free VVC algorithm based on multi-agent DQN was developed for unbalanced distribution systems. However, these methods require to learn a centralized state-action value function and may suffer from the curse of dimensionality when the number of actions increases. In [119], a MADDPG-based approach for distribution system voltage regulation with high penetration of photovoltaics (PVs) was proposed. However, this method did not consider the coordination control of OLTCs, VRs, and SCs. Besides, all of these methods adopted the central-
ized training and decentralized execution scheme, which still requires a centralized computing server to perform complicated data processing and policy training. In [152], a decentralized multi-agent RL algorithm based on value function consensus and maximum entropy optimization was proposed to solve the VVC. However, this method did not consider the impact of inverter-based DGs, such as renewable energy resources, in distribution systems.

In this section, the VVC problem is formulated as a cooperative MARL problem with a group of networked agents. Each agent is responsible for monitoring and control of a sub-region of the distribution system and can only communicate with its neighbor over a decentralized communication network [153]. The objective is to find a joint policy to minimize the total system loss while maintaining the bus voltages in a normal range. To solve this problem, the proposed MATRPO algorithm in Chapter 3 is applied. Simulation studies using real-world data are carried out to verify the effectiveness of the proposed approach.

5.3.2 Problem Formulation

Consider a distribution grid with a set of OLTCs, VRs, SCs, inverter-based DGs, and loads. Let \( \Omega_b \) denote the set of the system buses and \( \Omega_l \) denote the set of distribution lines. Let also \( V_{k,t} \) denote the complex voltage of the bus \( k \) and \( I_{kj,t} \) the complex current on the branch between bus \( k \) and bus \( j \) in the time slot \( t \). To maintain the bus voltages within a normal range and avoid overloading of the distribution lines, the following constraints should be satisfied:

\[
\underline{V}_k \leq V_{k,t} \leq \overline{V}_k, \quad k \in \Omega_b,
\]

\[
0 \leq I_{kj,t} \leq \overline{I}_{kj}, \quad kj \in \Omega_l,
\]

where \( \underline{V}_k \) and \( \overline{V}_k \) are the lower and upper limits of the voltage at bus \( k \), and \( \overline{I}_{kj} \) is the maximum current on the line \( kl \).
In the considered distributed VVC framework, the distribution system is divided into $N$ regions. It is assumed that there are $N$ agents responsible for the monitoring and control of the $N$th regions. The agents are connected through a distributed communication network. They can exchange information through the communication network with their immediate neighbors.

Let $\Omega^i_b$ denote the set of buses and $\Omega^i_l$ the set of lines in the region $i$. The distribution lines between neighboring regions are put under the region that has a smaller index number.

The observation of the agent $i$ in the time slot $t$ is defined as

$$o^i_t = (P^i_{k,t}, Q^i_{k,t}), \quad \forall k \in \Omega^i_b,$$  \hspace{1cm} (251)

where $P^i_{k,t}$ and $Q^i_{k,t}$ are the active and reactive power injected into the bus $k$ in region $i$.

The action of the agent $i$ includes the number of in-operation units of SCBs, the tap position of OLTCs/VRs, and the reactive power of inverter-based DGs in the region $i$:

$$a^i_t = (n^{\text{scb}}_k, n^{\text{vr}}_{kj}, Q^{\text{dg}}_{k,t}) \quad \forall k \in \Omega^i_b, \forall kj \in \Omega^i_l.$$  \hspace{1cm} (252)

The reward of the agent $i$ is calculated based on the line losses and bus voltage constraints in the region $i$:

$$r^i_t = -\omega_1 \sum_{kj \in \Omega^i_l} P^{\text{loss}}_{kj,t} - \omega_2 \sum_{k \in \Omega^i_b} \left[ \max(0, Q^{\text{dg}}_{k,t} - Q^{\text{dg}}_{k,t}) + \max(0, Q^{\text{dg}}_{k,t} - Q^{\text{dg}}_{k,t}) \right]$$

$$\hspace{4cm} - \omega_3 \sum_{k \in \Omega^i_b} \left[ \max(0, V_{k,t} - V_k) + \max(0, V_k - V_{k,t}) \right]$$  \hspace{1cm} (253)

where the first term denotes the negative sum of the power losses on the distribution lines in the region $i$; the second term is the negative sum of the reactive power violation of the inverter-based DGs; the third term is the negative sum of the bus voltages deviations from the normal range; $\omega_1$, $\omega_2$, and $\omega_3$ are coefficients.
Let $\pi^i(a^i|o^i) : \mathcal{O}^i \times \mathcal{A}^i \mapsto [0, 1]$ denote a stochastic policy of agent $i$, and then the joint policy of all agents can be denoted by $\pi(a|o) = \prod_{i=1}^N \pi^i(a^i|o^i)$, where $o = (o^1, o^2, \ldots, o^N)$ denotes the aggregated observation of all agents and $a = (a^1, a^2, \ldots, a^N)$ represents the joint action. The objective of the agents is to collaborate to jointly maximize the expected sum of the total discounted rewards:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^T \sum_{i=1}^N \gamma^t r^i_t \right]$$

where $\tau$ denotes a trajectory $(o_0, a_0, o_1, \ldots)$ following the policy $\pi$.

### 5.3.3 Multi-Agent Reinforcement Learning Solution

To solve the distributed VVC problem, the MATRPO algorithm proposed in chapter 4 is applied. In MATRPO, each agent learns a local policy $\pi^i(a^i|o^i)$ to predict the joint action $a$ based on its local observation $o^i$. In order for the agents to jointly optimize the policy $\pi(a|o) = \prod_{i=1}^N \pi^i(a^i|o^i)$, the MATRPO algorithm iteratively solves the following distributed consensus optimization problem:

$$\max_{\{\pi^i\}_{i\in\mathcal{N}}} \sum_{i=1}^N \mathbb{E}_{o^i \sim \rho_{o,\text{old}}^i} \left[ \frac{\pi^i(a|o^i)}{\frac{\pi^i_{\text{old}}(a^i|o^i)}{A_{n,\text{old}}^i(a^i,o^i)}} \right]$$

$$\text{s.t. } \frac{\pi^1(a^1|o^1)}{\pi^1_{\text{old}}(a^1|o^1)} = \cdots = \frac{\pi^N(a^N|o^N)}{\pi^N_{\text{old}}(a^i|o^N)}, \forall i \in \mathcal{N}$$

To learn a set of optimal cooperative policies for the distributed VVC problem, feedforward neural networks are used. Since the actions $a^i_t = (n_{\text{sch}}^i, n_{\text{vr}}^i, Q_{d}^i) \forall k \in \Omega_k^i$ contain both integer and continuous variables, the local policy for each agent $i$ is formulated as a joint distribution

$$\pi(a_t|o_t^i) = P(n_{\text{sch}}^i)P(n_{\text{vr}}^i)f(Q_{d}^i),$$

where $P(\cdot)$ is the probability mass function (PMF) of a categorical distribution; $f(\cdot)$ is the probability density function (PDF) of a Gaussian distribution. The
PMF and PDF parameters are learnt by using a feedforward neural network parameterized by $\theta^i$. Then, the parameters are updated by the MATRPO algorithm in Equation (205) in chapter 4.

5.3.4 Case Studies

In this section, the proposed distributed VVC method is evaluated on a modified IEEE-34 bus system as shown in Fig. 20. The system is divided into 8 regions, and each region is governed by an agent. The agents communicate through a ring network to share information with their neighbors. The test system contains two VRs with a maximum regulation of $-10\%$ to $+10\%$ of the nominal voltage and 33 tap positions ($-16; -15; \ldots; 0; +1; +2; \ldots; +16$) each; two SCBs of 0.48 MVAR with four units (0.12 MVAR/unit) at nodes 856 and 838; 6 inverter-based DGs with capacities of 0.1 MVA at nodes 810, 814, 820, 864, 844, and 840, respectively. The nodal voltages are confined to $0.95 \text{ p.u.} - 1.05 \text{ p.u.}$ The coefficients $\omega_1$, $\omega_2$, and $\omega_3$ are set to 100, 1, and 1, respectively. These coefficients determines the weights of the power loss, reactive power violation, and bus voltage deviation in the reward. In our experiments, these values are determined based on a heuristic to have a good tradeoff among these quantities.

The hourly time-series data in CAISO [154] is used to simulate the system load demand and renewable energy sources generation. The data in 2019 are used to train the neural network models and the data in 2020 are used to do test. The time-series data on the load are normalized to unity and then scaled to a proper level by multiplying every value in the normalized time series data by a base demand. The load demand at each bus in the standard IEEE 34-bus system [133] is used as the based base. The same method is used to process the solar PV and wind generation data.

The hyperparameters used in the proposed learning algorithms are summa-
Figure 20. A modified IEEE 34 Bus Test Feeder [133].

rized in Table I. To learn the local policy for each agent, a neural network with two layers of 128 ReLU neurons is used. Additionally, a network with the same architecture is used to approximate the value function for each agent. The parameters of the networks $\theta^i$, $\forall i \in \mathcal{N}$ and $\phi^i$, $\forall i \in \mathcal{N}$ are orthogonally initialized. The test system environment is constructed in PandaPower [155] and the experiments are carried out on a personal computer with an Intel Core i7-6700X Processor 3.40GHz $\times$ 8. The code is written in Python 3.7.6 using the deep learning package TensorFlow 2.2.0 [62], and DRL packages OpenAI Gym [61] and Baselines-tf2 [117].

Training Performance

To validate the performance of the proposed method, several benchmark methods are compared, including 1) a centralized single-agent DRL based method, denoted by ‘Central’, in which a central controller that knows the global state of the distribution system is assumed and applied to train a centralized policy using the TRPO algorithm; 2) an independent learning method, denoted by ‘Independent’, in which the agents do not share any information with the others, and learn their
Table 4. Hyper-parameters of the MATRPO algorithm for the Distributed VVC Problem.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># of agents</td>
<td>8</td>
</tr>
<tr>
<td># of steps in one episode(T)</td>
<td>24</td>
</tr>
<tr>
<td># of episodes in each iteration(N)</td>
<td>80</td>
</tr>
<tr>
<td># of iterates</td>
<td>1000</td>
</tr>
<tr>
<td># of ADMM iterates</td>
<td>100</td>
</tr>
<tr>
<td>Maximum KL-Divergence δ</td>
<td>0.01</td>
</tr>
<tr>
<td>ADMM penalty coefficients</td>
<td>1.0</td>
</tr>
<tr>
<td>GAE parameter (λ)</td>
<td>0.95</td>
</tr>
<tr>
<td>Discount factor(γ)</td>
<td>0.995</td>
</tr>
</tbody>
</table>

local policies only using local observations and rewards via TRPO; 3) multi-agent
deep deterministic policy gradient [70], denoted by ‘MADDPG’, which is a state-of-the-art multi-agent DRL method using the centralized training and decentralized execution framework.

The training rewards obtained using different methods are compared in Figure 21. As shown in the figure, the MATRPO method (green line) outperforms the Independent (orange line) and MADDPG (red line) methods by achieving a higher reward at the end of training. For the MADDPG method, although it quickly learned a good policy at the beginning, the performance did not improve as training continued. This may be because MADDPG can only generate continuous actions, but the VVC problem has mixed integer and discrete actions, and the discretization of the continuous actions may hurt the performance. In addition, for the Independent learning method, it can be observed that it does not perform as well as MATRPO and the Central method because of the non-stationary issue. It can also be observed that the proposed MATRPO can achieve comparable performance to the Central method, meaning that the agents trained by the proposed approach can effectively cooperate with each other in a decentralized manner to
Figure 21. Comparison of episode rewards using different methods during the training process.

Figure 22. Distributions of average bus voltage violation on 366 test days using different methods.

solve the VVC task, as well as a central controller can do.

Test Performance

After the model is well-trained, it is tested on the test dataset. A violin plot showing the distribution of the nodal voltage violations on 366 test days is presented in Fig. 22. As shown in this figure, the proposed MATRPO method results in a smaller mean and standard deviation compared to Independent learning
and MADDPG. This indicates that the MATRPO method can result in fewer nodal voltage violations and is more robust against the uncertainty in the distribution network compared to Independent learning and MADDPG.

Fig. 23 compares the cumulative system line losses on 366 test days using different methods. The total loss values yielded by Central, Independent, MATRPO, and MADDPG are 25.6 MW, 31.0 MW, 22.8 MW, and 60.2 MW, respectively. Compared to Independent and MADDPG, the proposed MATRPO-based method reduces the total system loss by 63.3% and 29.0%, respectively. In addition, the result of the proposed method is close to that of the centralized training method. Although the Central method achieves the best training performance, it requires high-bandwidth and reliable communication links to estimate the system state and transmit control signals, and suffers from scalability issues since all computation tasks have to be carried out in a central controller. These results verify the effectiveness of the proposed model in reducing the system loss while regulating bus voltages under certainty.
5.4 Summary

This chapter presents two smart grid applications based on trust-region-based RL and MARL approaches. The first application is distribution grid operation, and the second application is VVC. For the distribution grid operation problem, a safe RL algorithm, named CPO, has been applied to learn a near-optimal policy through safe exploration of the action space. Simulation studies on IEEE-34 and IEEE-123 bus systems using real-world power system data demonstrate the effectiveness of the method. For VVC, a decentralized MARL method proposed in Chapter 4 has been applied to solve the problem. Simulation studies on a modified IEEE-34 bus system showed that the proposed method can successfully learn a set of high-quality decentralized policies for the agents to effectively reduce the system loss and regulate the bus voltages in the nominal range with minor violations.
CHAPTER 6
Conclusions

6.1 Summary of the Dissertation

This dissertation has addressed the challenges of making RL more efficient, stable, and scalable. To overcome these challenges, the dissertation has proposed novel theory and algorithms based on trust-region methods. Additionally, practical applications of RL in the smart grid have been developed using these approaches.

First, this dissertation has presented a new theoretical result that relates the lower bound of performance to an expected KL-Divergence, effectively bridging the gap between theory and practice of trust-region-based RL methods in the literature. Building upon this result, the dissertation has introduced a closed-form update rule for general stochastic policy optimization with a monotonic improvement guarantee. The policy update rule has been derived using calculus of variation. Additionally, the dissertation has demonstrated that the update rule can be extended to cooperative multi-agent RL when agents take turns performing policy updates. As the proposed update rule is analytical, it can serve as a foundation for future work on novel RL theories and principled RL algorithms utilizing parametric or non-parametric policies.

Using the theoretical result proposed earlier, this dissertation introduces an off-policy trust-region policy optimization algorithm that allows for efficient learning of deep neural network policies while maintaining the stability of trust-region methods. Empirical results obtained from various MuJoCo continuous control tasks demonstrate the superior performance of our algorithm over prior trust-region deep RL approaches such as TRPO, PPO, and Trust-PCL in terms of both final return and sample efficiency. The proposed algorithm also achieves comparable performance with the state-of-the-art off-policy algorithm SAC and outperforms
DDPG on most tasks. These results suggest that trust-region methods are not only effective for stabilizing policy optimization but can also enable the learning of high-quality policies for complex continuous control tasks in a sample-efficient manner.

Furthermore, this dissertation has extended the trust-region policy optimization algorithm to cooperative multi-agent systems. As a result, a fully decentralized MARL algorithm, named MATRPO, has been proposed to optimize distributed policies for networked agents. The proposed method’s distributed policy optimization has been shown to be equivalent to single-agent TRPO when each agent’s local observation is sufficient. Simulation studies on the MPE tasks and SMAC scenarios have demonstrated MATRPO’s effectiveness in learning collaborative policies in a fully decentralized manner. The proposed method’s effectiveness and superiority have been further tested by comparing it with several baseline methods, including centralized training and decentralized execution, independent learning, and fully-decentralized training.

Finally, this dissertation has developed two smart grid applications using trust-region-based RL and MARL approaches. The first application, distribution grid operation, applied a safe RL algorithm called CPO to learn a near-optimal policy through safe exploration of the action space. The effectiveness of this method was demonstrated through simulation studies on IEEE-34 and IEEE-123 bus systems using real-world power system data. The second application, VVC, solved the problem using a decentralized MARL method proposed in Chapter 4. Simulation studies on a modified IEEE-34 bus system have shown that the proposed method successfully learned a set of high-quality decentralized policies for the agents to reduce the system loss and regulate the bus voltages in the nominal range with minor violations.
6.2 Challenges and Opportunities

The work presented in this dissertation opens up important research directions and future opportunities. Two critical future directions that this dissertation would like to highlight are:

**Decision Awareness in RL**

A critical aspect of human intelligence is the ability to understand and predict the implications of an action in the environment, and this should also be a fundamental component of artificially intelligent systems, particularly in RL. A "decision-aware" agent is required that not only learns an optimal policy but also trains models that characterize the environment, aid in decision-making, and estimate expected returns. Furthermore, the agent must acknowledge that modeling all environmental complexities is neither practical nor essential, and instead focus on modeling critical aspects for decision-making and policy transfer. Future research should explore decision-aware RL algorithms and their implications and applications in real-world scenarios. The focus should be on decision-aware objectives, model-based/inner interest-driven procedures, and meta-learning techniques for discovering and training components in RL systems. Additionally, conducting theoretical and empirical analyses of the interactions among these components will enhance decision awareness.

**Privacy-Preserving MARL algorithms**

Coordinating groups and networks of autonomous agents is a promising area for research as it allows for the completion of tasks that a single agent cannot achieve alone. However, developing such multi-agent systems poses numerous research challenges. Despite the previous focus on coordination, planning, and scalability, little attention has been paid to privacy protection. Efficient collab-
oration necessitates the sharing of personal information among team members, raising concerns about privacy leakage. For example, coordinating a large group of self-driving vehicles may require passengers to share real-time position, speed, and destination information for collision or traffic congestion avoidance. Thus, effective and trustworthy learning algorithms are necessary to ensure the protection of personal privacy. In the future, privacy-preserving algorithms, horizontal/vertical federated learning, and their application in large-scale MARL should be explored, with specific emphasis on differential privacy learning and homomorphic encryption techniques for encrypted communication and federated training of collaborative agents.
LIST OF REFERENCES


[113] I. Mordatch and P. Abbeel, Emergence of grounded compositional language in multi-agent populations, 2018, vol. 32, no. 1


Li, D., Zhao, D., Zhang, Q., and Chen, Y., “Reinforcement learning and deep learning based lateral control for autonomous driving [application notes],” *IEEE Computational Intelligence Magazine*, vol. 14, no. 2, pp. 83–98


