VIETNAM NATIONAL UNIVERSITY HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Le Bang Giang

APPLY GRAPH NEURAL NETWORK FOR SEARCHING PARETO OPTIMALITY IN MULTI-AGENT REINFORCEMENT LEARNING

MASTER'S THESIS

Hanoi - 2024

VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

LE BANG GIANG

APPLY GRAPH NEURAL NETWORK FOR SEARCHING PARETO OPTIMALITY IN MULTI-AGENT REINFORCEMENT LEARNING

Field: Computer Science
Major: Computer Science
Code: 8480101.01

COMPUTER SCIENCE MASTER'S THESIS

SUPERVISOR: Dr. Ta Viet Cuong

Hanoi - 2024

AUTHORSHIP

I hereby declare that the work contained in this thesis is of my own and has not been previously submitted for a degree or diploma at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no materials previously published or written by another person except where due reference or acknowledgment is made.

Signature:. . . . . .

Full name: Le Bang Giang

SUPERVISOR'S APPROVAL

I hereby approve that the thesis in its current form is ready for committee examination as a requirement for the The Degree of Master in Computer Science at the University of Engineering and Technology - Vietnam National University Hanoi.

Signature:. . . . . .

Full name: Ta Viet Cuong

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who have supported and guided me throughout the course of my research and the preparation of this thesis.

First and foremost, I would like to extend my sincere thanks to my supervisor Dr. Ta Viet Cuong for his invaluable guidance, encouragement, and expert advice throughout this project.

My heartfelt thanks go to my family. Their support has been my foundation, and I am truly thankful for their understanding during the challenges of this journey.

Finally, I would like to thank my colleagues and friends at HMI Laboratory and the IT department for their thoughtful discussions and emotional support. Their presence made this process more enjoyable and less isolating.

This work would not have been possible without the contributions of all these individuals. To each of you, I offer my deepest appreciation.

Le Bang Giang

ABSTRACT

In this thesis, I study the problem of finding Pareto optimal policies in multi-agent reinforcement learning problems with cooperative reward structures. I show that any algorithm where each agent only optimizes their reward is subject to suboptimal convergence. Therefore, to achieve Pareto optimality, agents have to act altruistically by considering the rewards of others. This observation bridges the multi-objective optimization framework and multi-agent reinforcement learning together. I first propose a framework for applying the Multiple Gradient Descent algorithm (MGDA) for learning in multi-agent settings. I further show that standard MGDA is subjected to weak Pareto convergence, a problem that is often overlooked in other learning settings but is prevalent in multi-agent reinforcement learning. To mitigate this issue, I propose MGDA++, an improvement of the existing algorithm to handle the weakly optimal convergence of MGDA properly. Theoretically, I prove that MGDA++ converges to strong Pareto optimal solutions in convex, smooth bi-objective problems. I further apply Graph Neural network to model the interaction between agents to our framework. Empirically, I demonstrate the superiority of our MGDA++ in cooperative settings in the Gridworld and MPE benchmarks. The results highlight that our proposed method can converge efficiently and outperform the other methods in terms of the optimality of the convergent policies.

Keywords: *Multi-Agent Reinforcement Learning, Multi-Objective Optimization, Multiple Gradient Descent, strong Pareto Optimality, Graph Neural Network*

Contents

Authorship i

Supervisor's approval ii

Acknowlegement iii

Abstract iv

List of Figures vi

List of Tables ix

Chapter 1 Introduction 1

1.1. Motivation 1

1.2. Contributions 3

1.3. Outlines 4

Chapter 2 Background and Related works 6

2.1. Multi-Agent Reinforcement Learning 6

2.2. Multi-Objective Optimization 9

2.3. Graph Neural Network for MARL 11

2.4. Related works 12

Chapter 3 A new method for finding Strong Pareto policies 15

3.1. The difficulty of finding the Pareto optimality with current MARL framework 15

3.2. MGDA with Trust region Policy Optimization 17

3.3. The problem of MGDA 18
3.4. MGDA++ Algorithm 19
3.5. Apply Graph Neural Network for handling agent interactions 26

Chapter 4 Experiments **29**

4.1. Experimental setup 29
4.1.1. Benchmarks 29
4.1.2. Baselines 31
4.1.3. Hyperparameter settings 33
4.2. Results 34
4.2.1. Without Graph Neural Network 34
4.2.2. With Graph Neural Network 37
4.2.3. Summary of the experimental findings 40

Conclusion **42**

Publication related to this thesis **45**

References **46**

List of Figures

1.1 A visualization of the cooperative setting considered in this thesis. . . . 2

3.2 Objective landscape in 2D space. 19

3.3 Comparison of MGDA, MGDA with Adam and MGDA++. Left: MGDA gets stuck at Pareto Stationary points, where the learning completely stops. Middle: Changing the optimizer does not help in avoiding the suboptimal convergence. Right: MGDA++ is able to converge to strong Pareto Optimal solutions while avoiding being trapped at Pareto Stationary points. 21

3.4 Comparison of MGDA and MGDA++ convergent points. Left: I test MGDA and MGDA++ on the synthetic problem from Lin et al. [1]. Both algorithms can converge to different Pareto optimal solutions in most usual cases. Right: I initiate the two algorithms with the same starting point in a simple quadratic bi-objective optimization problem, $F_{1,2}(x) = \|x \pm \mathbf{1}\|^2$. While MGDA stops as soon as it reaches the first stationary point, MGDA++ avoids small gradient norm solutions by further taking an additional step into the relative interior of the Pareto Set. In this example, MGDA++ does not converge to balls with radius $\epsilon/2$ around stationary points whose gradient norms of one of the objectives equal 0. For visualization, I plot such a ball with doubled radius $\epsilon = 0.01$ as a blue-dotted empty circle in the figure. 27

4.1 Four scenarios of the Gridworld environment. 31

4.2 Average reward of all agents in the four scenarios. Note that, the averaging operator can be viewed as a particular linearization of the rewards vector. 37

4.3 Results in MPE benchmark, showing the performance of each agent over the training period. 38

4.4 Average reward of all agents in the four scenarios with graph-based methods. 40

4.5 Result on MPE with graph-based methods. 41

List of Tables

4.1 Hyperparameter setting of MGPO++. 34

4.2 Result on Gridworld environment, all reported values are rounded to the first place after decimal. The best values of each agent are highlighted in bold, while the second-best values are underscored. 36

4.3 Result on Gridworld environment with graph-based methods. 39

List of Acronyms

Acronym	Definition
MARL	Multi-Agent Reinforcement Learning
MOO	Multi-Objective Optimization
GCN	Graph Convolutional Network
GNN	Graph Neural Network
MGDA	Multiple Gradient Descent Algorithm
MAPPO	Multi-Agent Proximal Policy Optimization
MGPO	MGDA with MAPPO
MGPO++	MGDA++ with MAPPO
DQN	Deep Q Learning
NE	Nash Equilibrium
PO	Pareto Optimality

Chapter 1

Introduction

1.1. Motivation

Multi-agent Reinforcement Learning (MARL) is the learning framework where multiple different agents learn to make optimal decisions in an environment through the Reinforcement Learning paradigm. MARL research advances rapidly, with the recent development of MARL achieving impressive results on a wide range of learning scenarios, notably in *zero-sum* games [2], where the sum of rewards of all agent equal zero, and *fully cooperative* environments [3, 4] where all agents share the same reward function.

While many notable recent successes of deep MARL come from zero sum and fully cooperative games, many real world problems are not limited by these assumptions on the reward function. Many practical problems can comprise of self-interested agents, where each agent can have different and possibly competing objectives [5]. This setting can partially encompass both cooperative and competitive settings, as some agents can have conflicting goals, and spontaneous cooperative efforts from multiple agents sometimes is required to achieve better common goals.

Terminologically, this setting of reward structure, i.e. other than zero-sum and fully cooperative, is a rather abstract concept, it does not have a formal definition and is not generally agreed upon in the literature. For example, Lowe et al. [5] define such reward structures as mixed cooperative-competitive settings, while Hu et al. [6] denote them as cooperative and/or collaborative settings, with the cooperative one putting more emphasis that it generally devoid of conflicting scenarios and is closer

to the fully cooperative settings than in the collaborative one, which can contain partial competitiveness between agents. Another formal definition that is closely related to cooperativeness is Markov Potential game [7], which takes inspiration from Potential games in Game Theory [8] and encompasses fully cooperative setting as a special case, though this definition requires the value functions of agents to satisfy a certain condition. Throughout this thesis, I generally use the term cooperative settings in the general sense that they are neither zero-sum nor fully cooperative, I also expect that agents should achieve more rewards through cooperation, thus putting more focus on the cooperative side of the reward scale. However, I do not require this to be strictly satisfied to validate the proposed theory, but rather see it as an intuitive context in the design of the proposed algorithm.

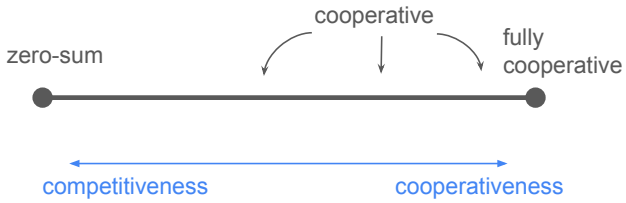


Figure 1.1: A visualization of the cooperative setting considered in this thesis.

Unfortunately, MARL with cooperative reward structures is an under-explored area of research. In these problems, there might be multiple different but optimal policies, in the sense that none of which is better than the other. This concept is known as the Pareto optimality in multi-objective optimization. It is desirable that we can find such Pareto optimal policies, as they are equivalent to the optimal policies in the classical setting of single agent problems [9]. While the current MARL methods are known to find Nash equilibria [7], such solutions can be suboptimal [10].

In this thesis, I first show that in general cooperative environments, agents need to explicitly consider the optimization of other agents to achieve Pareto optimality. Such behaviors are known as altruistic learning in RL literature [11]; altruistic learners learn to act for the benefit of other agents even though those actions do not bring about any personal gain for the actors. To learn altruistically, one agent needs to optimize not only the reward of itself but also the rewards of other agents, which involves some form of multi-objective optimization. As a result, we connect the multi-objective framework to the MARL domain. The application of Multi-Objective Optimization algorithms (MOO) into the MARL picture follows naturally.

Multiple Gradient Descent Algorithm (MGDA) [12] is one of the most popular gradient-based multi-objective methods. MGDA can find arbitrary solutions in the Pareto optimal Set using first-order gradient descent. However, MGDA is known to only converge to weak Pareto optimal solutions [13]. While this problem is not significant in other learning settings, I show that MARL problems can have many weak Pareto Stationary points, which can reduce the efficacy of MGDA. To this end, I identify the effect of diminishing gradient norms as a root cause to the weak Pareto convergence issue of MGDA and propose an improved version of MGDA++ based on this observation. I demonstrate both theoretically and empirically that MGDA++ can converge to strong Pareto optimal solutions. In the process of theoretical development, I also propose a novel *sufficient* condition for a solution to be strong Pareto optimal with convex objective functions.

Theoretical results from Zhao et al. [14] suggest that in fully cooperative environments, agents need to communicate their actions for Policy Gradient methods to converge to global optimal policies. On the other hand, without this communication scheme, the best we currently have are Nash Equilibrium convergences [7, 15]. As a result, while acting altruistically is a necessary condition for learning Pareto policies, it alone is not enough to guarantee Pareto solutions. Furthermore, while the theoretical formulation of MARL settings usually assumes that all agents can have access to the true underlying state representations (in the sense that they satisfy Markov properties [16]) of the environment, which is usually not the case in actual problems. Partial observability [17], non-stationarity [18] and miscoordination [15] by perceiving other changing agents as a static part of the environment introduce additional uncertainty to the learning process. Therefore, I next apply graph neural network to model the interaction between agents. I employ a standard Convolutional Graph Neural Network (GNN) [19] in the modeling of agents features embedding with the proposed methods. This incoporation of graph modeling results in better aggregation of the information between neighboring agents that cannot be achieved by individual agents learning schemes alone.

1.2. Contributions

To summarize, my contributions in this thesis are:

- **Connect MOO with MARL**. I show that to achieve Pareto optimality in MARL,

agents need to consider the objective of other agents, I connect the multi-objective optimization with MARL problems, and propose to apply MGDA to the MARL problems.

- **Novel MOO algorithm for finding Strong Pareto solutions**. I propose MGDA++, an extension of MGDA that converges to strong Pareto Solutions with bi-objective problems in convex, smooth settings both theoretically and empirically. To my knowledge, this strong Pareto convergence result in the convex setting with gradient descent is the first in the literature. I also propose a new *sufficient* condition for a solution to be Strong Pareto optimal in convex settings, while the condition provided in Desideri [12] is only a *necessary* condition.

- **Apply GNN to handle agents' interactions**. I examine the practicality of graph neural networks within the proposed framework of MARL with MOO and altruistic learning to handle the dynamic changes of different agents during the training process.

- **Empirical demonstration on two benchmarks**. I demonstrate the effectiveness of MGDA++ with trust-region methods and graph neural networks through several cooperative scenarios in the Gridworld and MPE benchmarks. The proposed method is able to achieve better convergence solutions across different agents in comparison with a variety of other baselines.

1.3. Outlines

This thesis is structured as follows:

- Chapter 1: Introduction. This chapter presents the research problems, outlines the context and motivations behind the study, and emphasizes the key contributions of the thesis.

- Chapter 2: Background and Related Works. This chapter introduces the problem setups for multi-agent reinforcement learning (MARL), multi-objective optimization (MOO), and Graph-Neural Network modeling with MARL, along with the key methods within each field. Additionally, it presents related works that provide context and outlook of the current state of the literature.

- Chapter 3: A new method for finding Strong Pareto policies. This chapter presents a novel framework that integrates multi-agent reinforcement learning (MARL) with multi-objective optimization (MOO) and Graph Neural Networks. I start by arguing that framing the MARL problem as a MOO problem is essential for achieving Pareto optimality. Building on this, I adapt the MOO algorithm MGDA for MARL, revealing a limitation known as Weak Pareto Convergence. To address this, I propose MGDA++, a new MOO algorithm designed to ensure Strong Pareto Solutions in convex bi-objective scenarios. Finally, I demonstrate how Graph Neural Networks can be applied within this MOO-MARL framework.

- Chapter 4: Experiments. This chapter provides the experimental details with other baselines, the experiments are aimed at examining the effectiveness of MGDA++ and Graph Neural Network separately in two MARL benchmarks.

- Chapter 5: Conclusion. This chapter concludes the thesis by summarizing the key contributions and future directions for extending this work.

Chapter 2

Background and Related works

In this section, I provide the necessary context and foundation for understanding the research topic in the thesis. Key concepts, notations, theories, and previous studies relevant to the subject are introduced and outlined, laying stages for later sections that follow.

2.1. Multi-Agent Reinforcement Learning

Markov game. The problem of Multi-Agent Reinforcement Learning is defined as a Markov game, determined by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, \gamma, \rho \rangle$. I denote $\mathcal{N}$ the set of all N agents, $\mathcal{S}$ the state space, $\mathcal{A}$ the joint action space of each i agent's action space $\mathcal{A}_i$, i.e $\mathcal{A} = (\mathcal{A}_1 \times \cdots \times \mathcal{A}_N)$, ρ the initial state distribution, and $\gamma \in [0, 1)$ the discount factor. In the cooperative setting, all the agents have a different reward function $r^i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. At each timestep t, all agents take actions $\mathbf{a}_t = (a_t^1, a_t^2, \ldots, a_t^N)$ according to their respective policy π^i, then the environment transitions to the next state s_{t+1} according to the dynamic kernel P. The total expected discounted cumulative returns of each agent are defined as $J_i(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}} \sum_t \gamma^t r_t^i$. Unlike the previous works in the cooperative settings [5], I do not assume each agent to necessarily maximize its returns but to find equilibriums such that no better joint policies exist, such equilibriums are called Pareto optimal solutions, formally defined in the next section.

Optimality concepts in MARL. In multiagent games, two optimality concepts prevail: Pareto optimality (PO) and Nash Equilibrium (NE). In NE, each agent assumes that the strategies of all other agents remain unchanged, leading to an equilibrium where no agent

has the incentive to alter their course of action. Conversely, PO involves the notion of collective welfare, wherein each agent endeavors to attain outcomes beneficial to all. PO states are those where no individual agent can increase their gains without diminishing those of others.

Fully-cooperative vs. cooperative settings. The simplest approach to enforce cooperation within a group is through reward aggregation, where the rewards of individuals are replaced by the sum (or average) of the agents' rewards in the whole group [5, 20]. In this way, all agents share a common objective, which is the reward of the whole group, the loss of the group is the loss of every agent. This setting generally goes under the name *fully cooperative* settings in the literature [3, 4]. Rewards summation, however, introduces the problem of *credit assignment* [21], where the information on the contributions of each agent is lost through the scalarization operations.

On the other hand, in the *cooperative* setting where the rewards of each agent are treated differently as is, the rewards of each agent are usually optimized independently as separated single-agent RL problems [22]. In this manner, it is not important whether the rewards of each agent are the same or not. This learning approach is often known as *independent learning*, with examples such as IQL, IPPO, and MADDPG. In this approach, each agent selfishly optimizes only its own reward signal, and therefore cooperative behaviors are generally difficult to emerge since the cooperation can only emerge implicitly [23]. Surprisingly, both approaches can suffer from the problem of suboptimal Nash Equilibrium convergence [7, 9, 10].

MAPPO [4]. In the fully cooperative setting, one of the most widely used MARL algorithms is MAPPO. MAPPO closely follows the implementation of PPO [24] in the single-agent setting; the agents share a single policy network π and a value function V. The value function serves as a baseline to reduce the variance of the estimation of Monte Carlo estimate and can access additional information, such as true underlying states or the aggregations of the agents' observations. In particular, the update of MAPPO inherits the trust region mechanism of PPO by using a clipping objective to prevent the

new policy from deviating too far away from the old policy.

$$L^{\text{MAPPO}}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) = \sum_i^n \mathbb{E}_{s,\mathbf{a}\sim\pi_{\theta_{\text{old}}}}\left[\min\left(\frac{\pi_\theta(\mathbf{a}^i|s)}{\pi_{\theta_{\text{old}}}(\mathbf{a}^i|s)}A_{\pi_{\theta_{\text{old}}}}(s,\mathbf{a}), \text{clip}\left(\frac{\pi_\theta(\mathbf{a}^i|s)}{\pi_{\theta_{\text{old}}}(\mathbf{a}^i|s)}, 1\pm\epsilon\right)A_{\pi_{\theta_{\text{old}}}}(s,\mathbf{a})\right)\right], \tag{2.1}$$

where $A_{\pi_{\theta_{\text{old}}}}(s,\mathbf{a})$ denote the estimate of the advantage function, calculated by subtracting the Q value estimate with the value function V learned above. Also, note that the input of $A_{\pi_{\theta_{\text{old}}}}$ includes the actions and states of all agents, while $\pi_\theta(\mathbf{a}^i|s)$ only access to the states (which is usually agents' local observations) and actions of the current agents, this explains why MAPPO is a *Centralized Training with Decentralized Execution (CTDE)* [21] approach. MAPPO is designed to work in fully cooperative settings, meaning the rewards are the same across all agents.

IQL [25, 26]. Independent Q Learning is an extension of Deep Q learning (DQN) of Deepmind [27] to the Multi-agent domains. IQL optimizes the Q function of each agent independently as if they were separate Q learning problems. For each agent, their Q value function is learned through the TD error as

$$L_{\text{TD}} = \mathbb{E}\left[\left(r + \gamma \text{sg}\left(\max_{a'} Q_{\theta'}(s',a')\right) - Q_\theta(s,a)\right)^2\right] \tag{2.2}$$

with $sg(\cdot)$ denotes the stop-gradient operator, and θ' represents the parameters of the target Q-network, a stabilized copy of the learning Q-network that updates more slowly to improve convergence.

IPPO [22]. IPPO decomposes MARL problems into decentralized policies learning using PPO. In contrast to MAPPO, which uses a centralized value function that inputs the full states of the environments, IPPO's critic only operates on local observations of the agents. Thus, IPPO is both a decentralized training and a decentralized execution algorithm. While there are skeptics about the theoretical limits of IPPO [22] on training stability and convergence, later works alleviate this doubt by showing that both IPPO and MAPPO require a centralized trust region constraint for all agents to guarantee monotonic improvements [28].

2.2. Multi-Objective Optimization

Notations. Let $[n]$ to denote the set of the first n positive intergers, $[n] = \{1, \dots, n\}$, all the norms $\|\cdot\|$ used in this thesis is assumed to be L2 norm and the inner product $\langle \cdot, \cdot \rangle$ is the dot product. Also, $|S|$ denotes the cardinality of a finite set S. For two vectors x and y, I define x^i the i^{th} element of the vector x, and the partial order $x \preceq y$ and $x \prec y$ denote the elementwise inequality $x^i \leq y^i$ and $x^i < y^i$, respectively.

Problem setting and definitions. Consider the optimization of the multi-objective function $F : \mathbb{R}^m \to \mathbb{R}^n$ as

$$\min_{x \in \mathbb{R}^m} F(x) = [F_1(x), F_2(x), \dots, F_n(x)]. \tag{2.3}$$

In contrast to single-objective optimization, solving the Multi-Objective Optimization problem requires making trade-offs between objectives, as it's typically impossible for all objectives to be optimal simultaneously. Similar to single-objective optimization, the concept of stationarity extends to multi-objective settings.

Definition 2.1. *[Pareto stationary] x^* is a Pareto stationary point if*

$$range(JF(x^*)) \cap -\mathbb{R}^n_{++} = \emptyset$$

with $JF(x)$ is the Jacobian matrix of $F(\cdot)$ at x and $\mathbb{R}_{++}$ is the set of positive numbers.

Similarly, the concept of Pareto optimality is analogous to the global optimality in the single-objective domain.

Definition 2.2. *x^* is called a Pareto (optimal) solution if there does not exist any other x such that $F(x) \preceq F(x^*)$ and $F(x) \neq F(x^*)$.*

Definition 2.3. *x^* is called a weak Pareto (optimal) solution if there does not exist any other x such that $F(x) \prec F(x^*)$.*

Definition 2.4. *x^* is called an ε-Pareto (optimal) solution if there exists a Pareto optimal solution x such that $F(x^*) \preceq F(x) + \varepsilon$.*

The Definition of Pareto optimality in Definition 2.2 is sometimes known as the strong Pareto optimality to distinguish it from the weak Pareto one defined in Definition 2.3. Weak and strong Pareto points are equivalent under certain conditions from the

objective function F, e.g. strong quasiconvex and continuity [29]. In this work, I consider the convex objective functions where weak and strong PO solutions are different (Assumption 1). Weak PO can be arbitrarily worse than strong Pareto solutions.

Steepest gradient descent. In steepest descent method [30], the objective is to find a common update vector d such that it minimizes all the objectives' first-order approximation with an L2 norm regularization,

$$\min_{d} \max_{i=1,\dots,n} \langle \nabla F_i(x), d \rangle + \frac{1}{2}\|d\|^2. \tag{2.4}$$

This problem can be formulated as a linear program as

$$\begin{aligned} \text{minimize} \quad & \alpha + \frac{1}{2}\|d\|^2 \\ \text{s.t.} \quad & \langle \nabla F_i(x), d \rangle - \alpha \leq 0, \quad \forall i = 1, \dots, n. \end{aligned} \tag{2.5}$$

MGDA. The dual problem of (2.5) is the following problem,

$$\begin{aligned} \text{minimize} \quad & \left\| \sum_i^n \lambda_i \nabla F_i(x) \right\|^2 \\ \text{s.t.} \quad & \textstyle\sum_i \lambda_i = 1 \\ & \lambda_i \geq 0 \qquad\qquad \forall i = 1, \dots, n. \end{aligned} \tag{2.6}$$

The formulation in (2.6) is considered in [12], where it gets the name MGDA, and is further developed by [31] to work with large-scale neural networks. MGDA is known to converge to arbitrary Pareto stationary points. Geometrically, the unique solution in (2.6) is the smallest norm vector in the convex hull of all the gradient vectors.

Stationary points and Weak Pareto solutions. The definition of stationarity in definition 2.1 has a deep connection with the properties of Weak Pareto solutions in definition 2.3. Intuitively, if the range of the Jacobian is in the interior R^n_{++}, then there exists a direction vector d whose projections onto all gradient vectors are positive, such update vectors would improve all the objectives with sufficiently small update step sizes, and thus cannot be a weakly Pareto optimal point. Specifically, the vector d satisfies $JF(x)d > 0$, the positive inequality arises due to the interior R^n_{++}, and the existence of d is guaranteed by the nonemptiness of the range of Jacobian. The matrix multiplication between JF and d can be seen as the projection operator of d onto all gradient vectors. As a result, Pareto Stationarity, with which many MOO algorithms with gradient descent converge to, is not a suitable criterion for finding Strong Pareto solutions. In the next chapter, I present a new sufficient condition for a solution to be strong Pareto in the convex objective settings. This new criterion is crucial in developing new algorithms with a Strong Pareto solution guarantee.

2.3. Graph Neural Network for MARL

Multi-agent environments can be represented as a dynamic graph, where agents are represented by nodes, and each node i has an associated set of neighbors B_i including itself. This set of neighboring nodes can depend on several factors such as proximity or observability and is usually defined a priori to the training. The graph is dynamic, meaning its structure can change over time as agents' interactions evolve throughout an episode.

DGN [19]. DGN applies a Convolutional Graph Neural Network to learn the Q functions. At each timestep, DGN receives the observation o^i of each agent i, to which it encodes to a latent vector h^i using a feature encoder, which can be MLP or CNN networks. The convolutional layers aggregate the observational feature vectors of neighboring agents, and output latent feature $h^{i'}$. Each convolutional layer has a receptive field, meaning that it can only access the latent of neighboring nodes. DGN has several stacking convolutional layers to gradually increase effective receptive fields as a later layer can communicate information with nodes several hopes away from their communication ranges. At the last graph layer, the features of the preceding Convolutional layers are concatenated and fed into a Q network.

The Convolutional kernel in DGN is implemented by multi-head dot-product attention architecture. More specifically, with each agent i, their observational and latent encoding is projected to query, key, and value vector representation by each independent attention head. For example, the attention weights between the agent i and j are as follows

$$\alpha_{ij}^m = \frac{\exp(\tau\langle W_Q^m h_i, W_K^m h_j\rangle)}{\sum_k^{B_i} \exp(\tau\langle W_Q^m h_i, W_K^m h_k\rangle)} \tag{2.7}$$

and the output of the convolutional layer

$$h^{i'} = \mathrm{MLP}\left(\mathrm{concat}_m \left[\sum_{j_{B_i}} a_{ij}^m W_V^m h_j\right]\right) \tag{2.8}$$

with W_Q^m, Q_K^m, and W_V^m the weights of query, key, and value of the attention head m, respectively. The concatenation of all attention heads goes through a single-layer MLP network to output the latent of the layer $h^{i'}$.

2.4. Related works

Multi-agent Reinforcement learning for finding Pareto Optimal Policies. While the concept of Pareto optimality is irrelevant in zero-sum games. In fully cooperative environments, there exist global, joint optimal policies that are better than any other policies as in the single agent setting [16]. The Pareto front in such cases thus comes down to containing a single optimal value function, leading to the equivalence of weak and strong Pareto solutions [14]. Dominant approaches in this setting include value decomposition [3] and policy-gradient methods [4].

Early works in cooperative multi-agent settings involve training independent learners for each agent [25]. It was observed that independent learning could converge to Nash Equilibriums through Policy Gradient methods [7]. With the actor-critic architecture, Lowe et al. [5] introduces centralized training to cooperative-competitive environments. More recently, Zhao et al. [14] proposed a method for achieving global convergence to optimal policies in fully cooperative domains, and the highlighted suboptimality of current MARL methods by Papoudakis et al. [10]. This has sparked interest in exploring Pareto optimal policies using multi-agent reinforcement learning techniques.

To address this, Christianos et al. [9] proposes Pareto actor-critic, which employs a modified advantage estimation for directing the joint policy to Pareto equilibrium. However, this approach suffers from exponential computational costs and assumes a non-conflict nature of the agents' rewards. Furthermore, it is restricted to single-reward domains, specifically fully cooperative environments.

Multi-Objective/Multitask learning. While there are works that can find strong ε-Pareto optimal solutions using gradient-free approaches [32], I focus on the gradient-descent-based methods. Desideri [12] introduces MGDA as steepest gradient descent [30] that converge to Pareto Stationary points. Subsequent future works include Sener et al. [31] that improves MGDA to be suitable for learning in high dimensional gradient in neural networks and Lin et al. [1] for finding diverse Pareto solutions.

Regarding the optimality concept, many works do not distinguish strong and weak Pareto optimal solutions. For example, [1, 33] define weak and strong Pareto optimality as similar concepts. Other works assume additional conditions so that weak and strong Pareto optimal solutions coincide [34]. Pareto Stationary solutions are de facto in the

theoretical results of MOO methods with gradient-based approaches [13, 35].

On the other hand, some other works explore the approaches of leveraging user preference directions to the Pareto optimality search [36, 37], such methods require additional knowledge of the preference vectors. Linearization approaches are easier to optimize [38], but they are potentially incapable of fully exploring the Pareto front [39]. Other researches focus on finding the optimal solution on the average objectives while mitigating conflict between gradients [40, 41], which does not account for a range of Pareto optimal solutions. The proposed algorithm in this thesis preserves the desired property of MGDA that can still explore most of the Pareto Front while improving on the strong Pareto convergence.

Graph modeling in MARL. When the environment requires coordination between agents, independently trained RL agents are shown to not perform well, in both value-based learning [42] and policy gradient [5]. DGN [19] directly applies a graph convolutional architecture with multi-head attention kernel to a popular RL algorithm DQN [19]. Nayak et al. [43] propose InfoMARL, a MARL method that aggregates information of neighboring agents using a graph neural network specifically for collision-free navigation tasks. HAMA [44] uses a hierarchical graph attention network to learn the hierarchical relationship between individual agents and groups of agents. Similarly, a hierarchical graph network is employed in LSC [45] to model the hierarchical communication structure of agents, effectively organizing message information generation and propagation in inter- and intra-groups. More recently, Hu et al. [46] propose CommFormer that learns the communication graph instead of relying on a hand-crafted one.

Modeling communication and interaction of agents in MARL is a diverse and relatively new field of research [47]. The connection between agents can be modeled under various types of communication constraints and modeling [48], embedded under the question of what and to whom to communicate with [49, 50]. This can be seen as a question of how to design model architectures, training schemes, and information processing procedures to effectively handle and combine an exponentially large dimension of all the agents' local inputs in the environment. Graph neural networks emerge as an effective method of modeling such connections. While the aforementioned GNN methods are designed specifically for particular problem settings and thus a prior inductive knowledge of the environments is usually appreciated. In this thesis, I focus on the application of pure Convolutional Neural Networks into my framework of

MOO-MARL. This design decision is expected to avoid the unnecessary complexity of Graph-based methods prematurely in the evaluation of their effectiveness in cooperative settings.

Chapter 3

A new method for finding Strong Pareto policies

In this chapter, I present my novel framework of MARL combined with MOO and Graph Neural networks. First, I discuss the current problem of the MARL framework in the cooperative setting, showing the need to cast the problem of Multi-agent learning to a MOO problem as a necessary requirement for achieving Pareto optimality. I further propose a method that extends a MOO algorithm MGDA to the optimization of MARL. This extension exposes a weakness of MGDA that is often overlooked in other learning settings; the Weak Pareto Convergence. As a result, I propose a novel MOO algorithm MGDA++, which is based on the observation that the gradient of converged objectives can hinder the update vectors of other ones in MGDA. MGDA++ guarantees to obtain Strong Pareto Solutions in convex bi-objective settings. In the last section, I show how to apply Graph Neural Network to the MOO-MARL framework.

3.1. The difficulty of finding the Pareto optimality with current MARL framework

One of the challenges in cooperative multiagent reinforcement learning with policy gradients (PG) is that the solutions found by these algorithms are usually suboptimal. For example, [9] provides a matrix game example where the PG converges to a Pareto-dominated policy. Similar results are also reported empirically in [10], who showed that

many recent MARL algorithms are unable to converge to the Pareto-optimal equilibrium.

Theoretical insights by Leonardos et al. [7] establish that PG methods converge toward NE in Markov Potential Games. The convergence relies on the independent learning assumption, which states that every agent learns, and acts, using only their local information. This suffices for NE, since agents are perceived as static from others' perspectives, rendering local information adequate. On the other hand, Zhao et al. [14] show that MAPPO converges to globally optimal policies in fully cooperative MARL, given that agents can observe their peers' actions. Without action-sharing, PG methods are only able to converge to NE solutions [15]. The action-sharing mechanism can be seen as a type of communication, in which agents share their strategy and planning. Given their close connections, a natural question arises:

> *Can policy gradient methods converge to Pareto optimal policies in cooperative games, given arbitrary communication?*

Unfortunately, the answer to this question is negative; in certain instances, PG methods fail to converge to PO, even when agents freely exchange their actions. The matrix game example in Figure 3.1 illustrates this argument.

Given a matrix game in a normal form as in Figure 3.1, where each agent's actions solely influence the rewards of the other. As a result, any policy profile in this matrix game is a Nash Equilibrium. To see why PG does not converge to PO, we first find the gradient of an agent by using the Policy Gradient theorem [16],

$$\nabla_{\pi^i} J_i = \sum_s d(s) \sum_a A^{\pi^i}(s,a) \nabla \pi^i, \qquad (3.1)$$

where $d(s)$ denotes the unnormalized discounted marginal state distribution. But since the reward of the agent i does not depend on its actions, the advantage function A^{π^i} becomes constant, as a result

$$\nabla_{\pi^i} J_i = \sum_s d(s) A^{\pi^i} \sum_a \nabla \pi^i = 0. \qquad (3.2)$$

		Player 2	
		A	B
Player 1	A	1,2	0,2
	B	1,0	0,0

Figure 3.1: Example of the matrix game, the tuple in each table cell contains the reward of agent 1 and 2, respectively. There is one Pareto Optimal solution at (A,A). Note that in this example, *any* policy profile, stochastic or deterministic, is a Nash Equilibrium.

This is logical, because if an agent has no control over the reward they receive - a constant function of action, then the gradient - the rate of change of the return over the actions - of a constant function is zero. Therefore, in such situations, Policy Gradient fails to learn.

Note that knowing the action that the other agent will take has no benefit in this case, again because the agent cannot utilize this information to improve its decision which does not affect its reward; an agent has no way to dictate other agents to act upon their own interest. Furthermore, a similar argument applies to the sharing parameter setting. Because the gradient is zero, no learning happens.

Moreover, we can extend the above observation to value-based algorithms such as IQL and MADDPG [5], when the policy of one agent is stable, the learned values of an agent is a constant function, and if the choice of the optimal action in Q function to break ties is arbitrary, then the algorithm is also prone to converging to suboptimal policies. I formalize the above idea to a broad class of MARL algorithms as follows.

Any MARL algorithm where each agent only optimizes their objective is susceptible to converging to Pareto Dominated policies.

The problem is that each agent selfishly optimizes only their own reward, so no learning can happen if their actions and rewards are uncorrelated. This issue can be fixed straightforwardly: during training, agents should transition from self-centered reward optimization to considering the objectives of other agents as well. As a result, I have connected the multi-objective ideas to the training of multi-agent RL in cooperative settings. Furthermore, note that although the example presented in Figure 3.1 seems contrive, more complex scenarios involve the interaction of agents where one agent can help other agents to improve their performance, while greedy agents optimizing only their objective can bring worse outcomes to the whole.

3.2. MGDA with Trust region Policy Optimization

In this section, I discuss how MAPPO [4] can be combined with the MGDA algorithm to find Pareto optimal policies. MAPPO is an algorithm designed for fully cooperative settings, where all agents share a common reward function. As a result, for MAPPO to work in multi-reward settings, I first extend the value function to have a multi-head structure, where each head is responsible for predicting the baseline for the

corresponding agent. This structure is akin to the multi-head critic in multitask learning [40]. As a result, the objective for optimizing the policy i with respect to the reward j with PPO is as follows

$$L_{\text{PPO}}^{i,j} = \min\left(\frac{\pi_{\text{new}}^{i}}{\pi_{\text{old}}^{i}}\hat{A}_{j}^{\pi_{\text{old}}^{i}}, \text{clip}\left(\frac{\pi_{\text{new}}^{i}}{\pi_{\text{old}}^{i}}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{j}^{\pi_{\text{old}}^{i}}\right), \tag{3.3}$$

where $\hat{A}_{j}^{\pi_{\text{old}}^{i}}$ is the estimated advantage of the policy i to the reward of agent j, with the baselines from the multi-head value function for variance reduction. As a result, for the agent i, the set of gradient $\{\nabla L_{\text{PPO}}^{i,j}\}_{j=1}^{n}$ w.r.t. other agents' reward can be passed into the MGDA problem in (2.6) (and similarly MGDA++ presented in the next section) to find a descent direction that improves on the returns of all agents. The algorithm can then proceed with the usual descent methods [51].

3.3. The problem of MGDA

It is known that MGDA can converge to Pareto stationary solutions [12]. However, as illustrated in Figure 3.3, a stationary point can be a weak Pareto Optimal point and thus can be very suboptimal compared to the strong Pareto one. This is a distinct difference from the single objective optimization; in the convex case, stationary points are global optimal solutions, whereas in MOO, stationary points only ensure weak optimality, as demonstrated by the following lemma [35].

Lemma 1 (Theorem 3.3, [35])**.** *When the objective functions are convex, any Pareto stationary points $\hat{x}$ are weak Pareto optimal solutions.*

As a result, MGDA is only able to converge to weak Pareto optimal solutions [13]. Due to the optimization of the search directions in MGDA, the algorithm stops at any weak Pareto points. This stopping criterion can cause the algorithm to terminate prematurely or worse, not learn at all just by adding additional spurious objectives. I illustrate this phenomenon through the following corollary.

Corollary 2. *Given any optimization problem with the objective function $F : \mathbb{R}^m \to \mathbb{R}^n$, by adding a dummy objective $F_{n+1}(x) = c, \forall x$ with c a constant number, then all the points x in the new problem are weakly Optimal. As a result, the MGDA algorithm is not working in the modified problem.*

I further show the weak optimal convergence of MGDA through a toy experiment in Figure 3.3 with the bi-objective convex landscape in Figure 3.2, defined as

$$F_i(x) = \|y\|^2, \quad \text{with } y^j = \begin{cases} x^j & \text{if } j \neq i \\ \max(0, |x^j| - 5) & \text{if } j = i \end{cases} \tag{3.4}$$

The weak Pareto optimal convergence of MGDA can be attributed to the heavy influence of the diminishing-norm gradients from the already converged objectives to the common descent direction. To elaborate, as gradients approach zero, the weights vector in the sum of the problem (2.6) becomes highly concentrated on the corresponding entries. While the impact of small gradients is also noted in [12], attributing to the slow convergence rate of practical MGDA. I further associate it with the weak Pareto convergence problem. Furthermore, while the recommended approach to normalize the gradient norms all to be equal can somewhat mitigate the weak convergence issue of MGDA indirectly, its efficacy remains largely empirical, lacking solid theoretical grounding within MGDA's framework. Rather, it stands as an implementation detail for improving convergence speed in practice.

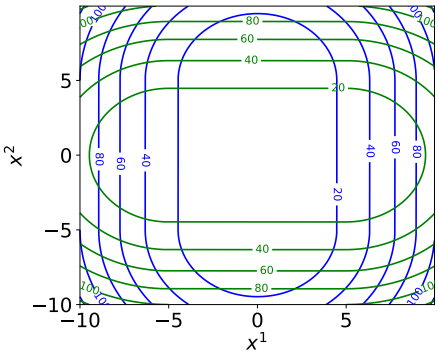


Figure 3.2: Objective landscape in 2D space.

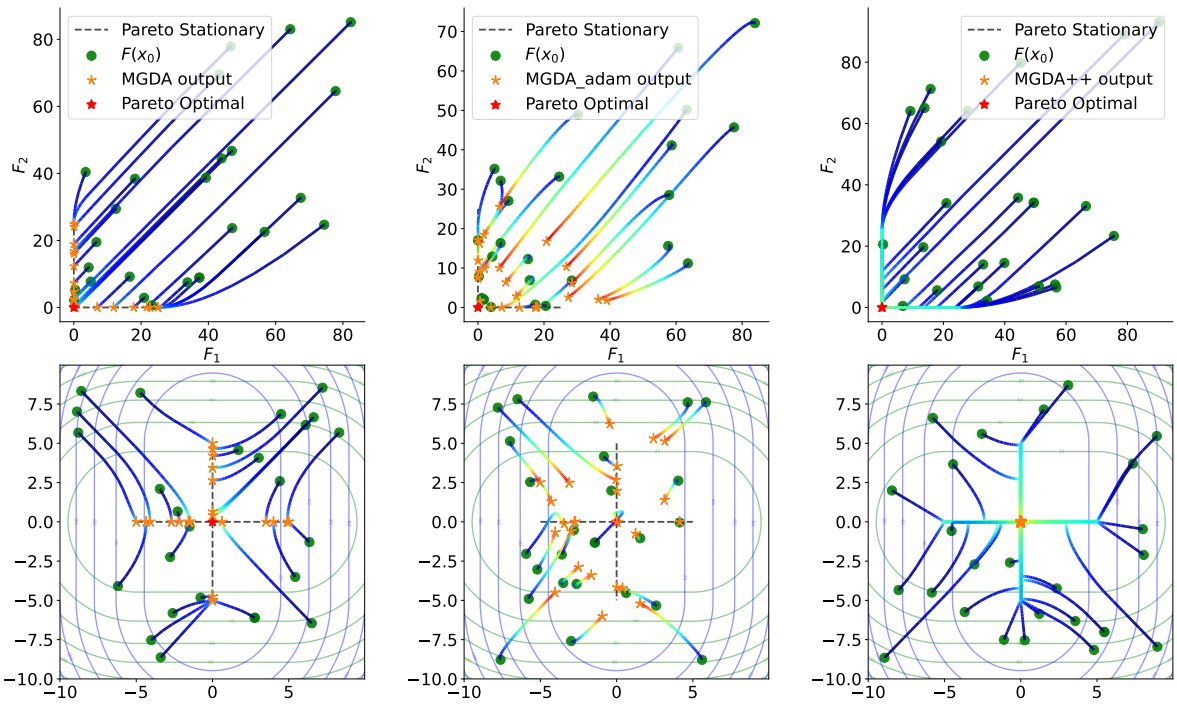
3.4. MGDA++ Algorithm

In Lemma 3, I establish a crucial criterion for strong Pareto optimality when the convex sum of *non-zero* gradients equals zero. This insight suggests that a specific rooting of the problem of MGDA might be due to the diminishing gradient norm of some specific objectives that impede the optimization of others, and hints at the potential enhancement to MGDA by carefully treating the objectives that already converge. This motivates us to propose an improved version of MGDA, which removes the impact of small gradient norms on the optimization of other non-converging objectives by considering in (2.6) only a subset of gradients whose norm is greater than a certain threshold. The new algorithm, denoted as MGDA++, is described in the Algorithm 1.

Algorithm 1 MGDA++ Algorithm

Input: $\epsilon > 0$, initial solution x_0

for $k = 0, 1, \ldots$ **do**
  $S_k \leftarrow \emptyset$
  for $i = 1, \ldots, n$ **do**
    Calculate $\nabla F_i(x_k)$
    if $\|\nabla F_i(x_k)\| > \epsilon$ **then**
      $S_k = S_k \cup \{i\}$
    end if
  end for
  if $S_k = \emptyset$ **then**
    Stop
  end if
  Find $\{\lambda_i\}_{i \in S_k}$ by solving (2.6) on the subset of gradients $\{\nabla F_i(x_k)\}_{i \in S_k}$
  $d_k \leftarrow \sum_{i \in S_k} \lambda_i \nabla F_i(x_k)$
  Choose step size t_k
  $x_{k+1} \leftarrow x_k - t_k d_k$
end for



(a) MGDA (b) MGDA w/ Adam (c) MGDA++

Figure 3.3: Comparison of MGDA, MGDA with Adam and MGDA++. Left: MGDA gets stuck at Pareto Stationary points, where the learning completely stops. Middle: Changing the optimizer does not help in avoiding the suboptimal convergence. Right: MGDA++ is able to converge to strong Pareto Optimal solutions while avoiding being trapped at Pareto Stationary points.

Below, I provide the theoretical analysis of the convergence property of my proposed MGDA++. Within my analysis, I make the following assumptions,

Assumption 1. *All objective functions F_i are convex and L-smooth.*

Assumption 2. *With a given point x_0, the sum of objectives function F has a bounded level set $\Gamma = \{x | \sum_i F_i(x) \leq \sum_i F_i(x_0)\}$. Furthermore, the optimal set of each objective function F_i is non-empty and is denoted by X_i^*, the optimal value is denoted similarly as F_i^*.*

First, I introduce a sufficient condition for a solution to be Pareto optimal when the objective functions are convex. It is important to distinguish this condition from the widely acknowledged necessary conditions for Pareto optimality [12]. Note that this and the next proposition hold true for a general number of tasks.

Lemma 3. *Under assumption 1, if there exists a convex combination of the subset of all non-zero gradient vectors*

$$\sum_{i \in S} \lambda_i \nabla F_i(x) = 0; \quad \lambda_i > 0, \quad \|\nabla F_i(x)\| > 0 \quad \forall i \in S \tag{3.5}$$

with $S \subseteq [n], S \neq \emptyset$, then x is Pareto optimal.

Proof. We only need to prove that x is Pareto optimal on the subset of the S objectives, since objectives with zero gradients are already optimal. By contradiction, assume that x is a point that satisfies condition 3.5 and there exist some x^* such that $F_S(x^*) \preceq F_S(x)$ and $F_S(x^*) \neq F_S(x)$. I will show that all the gradient vectors of the objectives in S lie in one half-plane whose orthogonal vector is $x^* - x$.

Let $v = x^* - x$, then $v \neq 0$. Since all function F_i are convex, then,

$$F_i(x + tv) \leq (1 - t)F_i(x) + tF_i(x^*) \tag{3.6}$$

with some $t \in (0, 1]$. Divide both sides with t and take the limit toward 0, we have

$$\lim_{t \to 0} \frac{F_i(x + tv) - F_i(x)}{t} \leq F_i(x^*) - F_i(x) \leq 0, \quad \forall i \in S \tag{3.7}$$

All the directional derivatives of F_i along the vector v are non-positive, which means that they all belong to the same half-space perpendicular to the vector v. Furthermore, since $F_S(x^*) \neq F_S(x)$, some gradient vectors are strictly inside the half-space.

It is easy to see that any convex sum of non-zero vectors in a half-space, with some strictly inside, cannot equal zero, so this contradicts the condition in 3.5. As a result, no such x^* exists, and thus x is Pareto optimal by definition. □

Next, I show that an arbitrary approximation of Pareto optimal solutions can be obtained if the gradients of all objectives are small enough,

Proposition 4. *Under assumptions 1 and 2, then for any $\varepsilon > 0$, it is possible to choose an $0 < \epsilon \leq \sqrt{2L\varepsilon}$ such that if any x satisfies $\|\nabla F_i(x)\| < \epsilon$, then $F_i(x) \leq F_i^* + \varepsilon, \forall i \in [n]$. Such x are necessarily ε-Pareto optimal solutions.*

Before presenting the proof, I first remind two basic properties of convex, smooth function. For a convex, L-smooth function f and two $x, y \in \text{dom}(f)$, then the followings hold true

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|^2 \tag{3.8}$$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2L} \|\nabla f(x) - \nabla f(y)\|^2 \tag{3.9}$$

These properties are well known and the derivation can be found in optimization textbooks (e.g. Theorem 5.8 [52]).

Proof of Proposition 4. From (3.9), for each $i \in [n]$, let $x_i^* \in X^*$ and plug x_i^* and x into y and x respectively

$$F_i(x_i^*) \geq F_i(x) + \langle \nabla F_i(x), x_i^* - x \rangle + \frac{1}{2L} \|\nabla F_i(x)\|^2 \tag{3.10}$$

In order for $F_i(x_i^*) \geq F_i(x) - \varepsilon$ for any x, define $B = \|x_i^* - x\|$, we need that

$$\langle \nabla F_i(x), x_i^* - x \rangle + \frac{1}{2L} \|\nabla F_i(x)\|^2 \tag{3.11}$$

$$\geq \frac{1}{2L} \|\nabla F_i(x)\|^2 - \|\nabla F_i(x)\| \|x_i^* - x\| \qquad \text{(Hölder's inequality)}$$

$$= \frac{1}{2L} \|\nabla F_i(x)\|^2 - B \|\nabla F_i(x)\| \geq -\varepsilon \tag{3.12}$$

which is a quadratic function of $\|\nabla F_i(x)\|$. If $\varepsilon \geq B^2 L/2$, then it is satisfied with any value of $\|\nabla F_i(x)\|$, solving this with $\varepsilon \leq B^2 L/2$ yields

$$\begin{cases} \|\nabla F_i(x)\| \geq L \left(B + \sqrt{B^2 - \frac{2}{L}\varepsilon} \right) & (3.13) \\ \|\nabla F_i(x)\| \leq L \left(B - \sqrt{B^2 - \frac{2}{L}\varepsilon} \right) \leq \sqrt{2L\varepsilon} & (3.14) \end{cases}$$

we then show that (3.13) is an invalid choice of $\|\nabla F_i\|$ and would never happen under assumption 1.

Plug x and x_i^* into y and x in both (3.9) and (3.8), we get

$$\frac{1}{2L} \|\nabla F_i(x)\|^2 \leq F_i(x) - F_i(x_i^*) \leq \frac{L}{2} \|x - x_i^*\|^2 \tag{3.15}$$

$$\Rightarrow \|\nabla F_i(x)\| \leq L \|x - x_i^*\| = LB \tag{3.16}$$

this invalidates the choice (3.13), then we are left with the case (3.14). As a result, by choosing $\epsilon \leq \sqrt{2L\varepsilon}$, then any x with $\|\nabla F_i(x)\| \leq \epsilon$ will satisfy that $F_i^* \geq F_i(x) - \varepsilon$, this concludes the proof. □

Now we are ready to present my main theoretical result of the convergence of MGDA++ with two tasks.

Theorem 5. *Under assumptions 1 and 2, for any $\varepsilon > 0$, with $n = 2$ and by choosing $\epsilon < \sqrt{2L\varepsilon}$ and with appropriate choices of each update steps t_k as*

$$t_k = \begin{cases} \max\left(\frac{|S_k|\|d_k\|^2 + \left\langle \sum_{i\in\overline{S}_k} \nabla F_i(x_k), d_k \right\rangle}{nL\|d_k\|^2}, 0\right) & \text{if } \|d_k\| > 0 \\ 0 & \text{if } \|d_k\| = 0 \end{cases}, \tag{3.17}$$

with $\overline{S}_k$ the complement of S_k, then each convergent subsequence of MGDA++ converges to either Pareto optimal or ε-Pareto optimal solutions.

Proof of Theorem 5. At the iteration k, let S_k the set of indices for which $\|\nabla F_s(x_k)\| > \epsilon, \forall s \in S_k$ and $\overline{S}_k$ its complement ($|S_k| \leq 2$). We also assume that the set S_k is non-empty, or else all the gradient vectors have norms less than ϵ and the algorithm would terminate. Then, let d_k the solution to the problem (2.4) with the set of S_k objectives, we have the following observations

$$F_i(x_{k+1}) \leq F_i(x_k) + \langle \nabla F_i(x_k), x_{k+1} - x_k \rangle + \frac{L}{2}\|x_k - x_{k+1}\|^2, \quad i \in \overline{S}_k \tag{3.18}$$

$$F_j(x_{k+1}) \leq F_j(x_k) - t_k\|d_k\|^2 + \frac{L}{2}t_k^2\|d_k\|^2, \quad j \in S_k \tag{3.19}$$

with $x_{k+1} - x_k = -t_k d_k$, summing over all indices, we get

$$(F_1 + F_2)(x_{k+1}) \leq (F_1 + F_2)(x_k) + Lt_k^2\|d_k\|^2 - t_k|S_k|\|d_k\|^2 - t_k\left\langle \sum_{i\in\overline{S}_k} \nabla F_i(x_k), d_k \right\rangle \tag{3.20}$$

As a result, if we choose

$$t_k = \begin{cases} \max\left(\frac{|S_k|\|d_k\|^2 + \left\langle \sum_{i\in\overline{S}_k} \nabla F_i(x_k), d_k \right\rangle}{2L\|d_k\|^2}, 0\right) & \text{if } \|d_k\| > 0 \\ 0 & \text{if } \|d_k\| = 0 \end{cases} \tag{3.21}$$

then the sum of all the objectives is guaranteed to non-increase after each iteration. Let $f_k = (F_1 + F_2)(x_k)$, $f_k \geq \sum_i F_i^*$, the sequence $f_0, f_1, \ldots$ is monotonically non-increasing and bounded below, so it converges to some f^*, since $\{x_k\}$ is bounded by assumption 2, there exists a subsequence that converges to some $\widetilde{x}$,

1. If at $\widetilde{x}$, all the gradient vector norms are less than ϵ, then the algorithm terminates at $\widetilde{x}$, and by choosing small enough ϵ, by proposition 4, the algorithm converges to arbitrary ε-Pareto optimal solutions.

2. Otherwise, let $\widetilde{S}$ the set of indices for which $\|\nabla F_i(\widetilde{x})\| > \epsilon, \forall i \in \widetilde{S}$, then $\widetilde{S}$ is nonempty, we also define $\widetilde{d}$ as the solution of the problem 2.4 at $\widetilde{x}$. We consider two possible cases.

   a) If $\|\widetilde{d}\| = 0$, then since $\widetilde{d}$ is the optimal solution of 2.4, then $0 = \widetilde{d} = \sum_{i \in S^*} c_i \nabla F_i(\widetilde{x}), \sum_i c_i = 1, c_i \geq 0$ on the subset $\widetilde{S}$ of non-zero gradient vectors, and according to Lemma 3, $\widetilde{x}$ is Pareto optimal.

   b) If $\|\widetilde{d}\| > 0$, then the corresponding update step $\widetilde{t} = 0$ or else by updating $x' = \widetilde{x} - \widetilde{t}\widetilde{d}$, then $(F_1 + F_2)(x') < f^*$. Since $\widetilde{t} = 0$, we have that

$$|\widetilde{S}|\|\widetilde{d}\|^2 + \Big\langle \sum_{i \in \overline{\widetilde{S}}} \nabla F_i(\widetilde{x}), \widetilde{d} \Big\rangle \leq 0 \tag{3.22}$$

   and because $\|\widetilde{d}\| > 0$, so $|\widetilde{S}| = 1$. Let j be the only index in $\overline{\widetilde{S}}$, then

$$\|\widetilde{d}\|^2 + \langle \nabla F_j(\widetilde{x}), \widetilde{d} \rangle \leq 0 \tag{3.23}$$

$$\Rightarrow \|\widetilde{d}\|^2 - \|\nabla F_j(\widetilde{x})\|\|\widetilde{d}\| \leq 0 \tag{3.24}$$

$$\Rightarrow \|\widetilde{d}\| \leq \|\nabla F_j(\widetilde{x})\|. \tag{3.25}$$

   However, since $|\widetilde{S}| = 1$, let k the only index in $\widetilde{S}$, then $\widetilde{d} = \nabla F_k(\widetilde{x})$. By definition of $\widetilde{S}$, $\|\widetilde{d}\| = \|\nabla F_k(\widetilde{x})\| > \epsilon$, but $\|\nabla F_j(\widetilde{x})\| \leq \epsilon$, which contradicts with (3.25). As a result, this case cannot happen.

In all the possible cases, I have shown that the algorithm converges to either Pareto or ε-Pareto optimal solutions, this concludes the proof. □

Remark 1 (Update step sizes)**.** *At some iteration, if $\overline{S}_k = \emptyset$ (i.e. no gradient vectors are dropped), then the choice of the update steps $t_k = \frac{1}{L}$, which is the optimal update step for convex, smooth functions. In this case, MGDA++ returns to the normal MGDA with a constant step size. Empirically, I find that MGDA++ works fine with constant step sizes in practice.*

Remark 2 (Challenges when generalizing to general $n > 2$)**.** *The convergence analysis of MGDA++ is hard for several reasons: first, each objective is not monotonically decreasing, because when the gradients become small, they are removed from consideration of steepest decent direction. Secondly, gradient norms do not monotonically decrease with arbitrary update step sizes. Finally, the algorithm does*

not converge to a single optimal solution x^* nor objective vector F^*. In the proof of Theorem 5, I specifically rely on the monotonicity of the sum of all objective functions by carefully selecting step sizes. The proof generally fails to generalize to other numbers of n, as when $|S_k| > 1$, the update vector d can have a small norm $< \epsilon$ even when all other gradient vectors in S_k have norm $> \epsilon$, which cannot ensure the monotonicity of $\{f\}_k$.

Remark 3 (Convergent solutions)**.** *MGDA++ guarantees strong optimality by avoiding potentially suboptimal solutions with small gradient norms, illustrated as the case 2b) in the proof of theorem 5. However, some of these potential solutions are actually Pareto optimal. As a result, MGDA++ only recovers a subset of the Pareto set where the optimality is ascertained. I demonstrate this phenomenon in Figure (3.4). I believe this is inevitable without any look-ahead search while keeping the optimal convergence guarantee.*

Theorem 5 establishes strong Pareto optimal convergence with convex objective functions. To the best of my knowledge, even within the limited setting of bi-objective optimization, this is the first strong Pareto convergence instance of gradient descent methods with convex functions in the literature.

3.5. Apply Graph Neural Network for handling agent interactions

I next examine an application of Convolutional Graph Neural network into my algorithm in section 3.2.. More specifically, I base my implementation of GCN for MARL on DGN [19]. The observations of each agent are first encoded by a feature encoder f_e and then fed into the convolutional layer, which is a multi-head attention network. With each attention head, I project the feature encoding of the previous layer into query, key, and value vectors. The value feature vector is then the weighted sum of the value vectors, whose weights are calculated as a softmax distribution of the dot product of the corresponding key and query vectors between neighboring agents in the adjacency graph. The output of a convolutional layer is a concatenation of all the attended values vectors, projected into another space with a smaller dimension by using a single-layer MLP. Several design choices such as network architecture are deferred to Hyperparameters section. The pseudocode of the Graph network in our method is

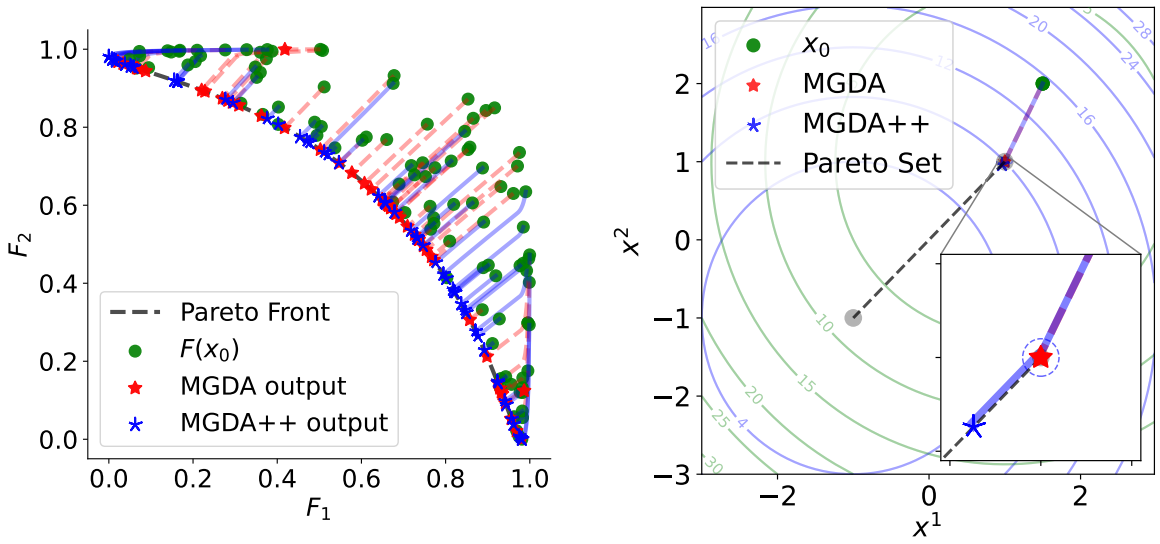


Figure 3.4: Comparison of MGDA and MGDA++ convergent points. Left: I test MGDA and MGDA++ on the synthetic problem from Lin et al. [1]. Both algorithms can converge to different Pareto optimal solutions in most usual cases. Right: I initiate the two algorithms with the same starting point in a simple quadratic bi-objective optimization problem, $F_{1,2}(x) = \|x \pm \mathbf{1}\|^2$. While MGDA stops as soon as it reaches the first stationary point, MGDA++ avoids small gradient norm solutions by further taking an additional step into the relative interior of the Pareto Set. In this example, MGDA++ does not converge to balls with radius $\epsilon/2$ around stationary points whose gradient norms of one of the objectives equal 0. For visualization, I plot such a ball with doubled radius $\epsilon = 0.01$ as a blue-dotted empty circle in the figure.

detailed in Algorithm 2.

Here, I highlight the main differences between my Convolutional Graph approach and DGN are that 1) DGN follows the single-agent optimization approach discussed in section 3.1., while our method follows the MOO optimization framework with altruistic learning, and 2) DGN employ a value-based, off-policy RL algorithm DQN, while I base my algorithm on the on-policy, trust region method MAPPO [4] (section 3.2.).

Algorithm 2 Graph Neural Policy for multi-agent RL

Input: Observations $\{o_i\}_i^N$ of all agents, number of agents N, encoder network f_e, weight matrices of query, key and value $(W_{Qk}^m, W_{Kk}^m, W_{Vk}^m)$, number of Graph Convolutional layer K (i.e hops), number of multi-head attention M, Adjacency matrix B

Output: Actions $\{a_i\}_i^N$

Encode the observations by the encoder to get latent $z_i = f_e(o_i), \forall i \leq N$
Let $h_0^i := z_i, \forall i \leq N$
for $k = 0, \ldots, K-1$ **do**
    for $i = 1, \ldots, N$ **do**
        Calculate attention weights $\{\alpha_{ij}^m\}_j^{B_i}$ according to Eq (2.7) $\forall m \leq M$
        Calculate the value vectors $V_i^m = \sum_j^{B_i} a_{ij}^m W_{Vk}^m h_k^j, \quad \forall m \leq M$
        Calculate $h_{k+1}^i = MLP(\text{concat}[V_i^m, \forall m \leq M])$ (Eq 2.8).
    end for
end for
for $i = 1, \ldots, N$ **do**
    // Find logit (discrete) or mean and std (gaussian)
    $h^i = MLP(\text{concat}[h_k^i, k \leq K])$
    // Sample from distribution (e.g. Categorical, Gaussian, or Squashed Gaussian)
    $a_i \sim \text{dist}(h_i)$
end for

Chapter 4

Experiments

In this chapter, I present the experimental setup for evaluating the performance of the proposed method with other baselines. The experiments are designed to answer the following questions: 1) Does altruistic learning actually help in finding Pareto solutions, 2) Does MGDA++ improve on the strong Pareto optimality compared to the baselines, and 3) How does incorporating graph neural network into the altruistic learning and multi-objective optimization framework impact the performance of the overall algorithms.

4.1. Experimental setup

4.1.1. Benchmarks

For my experimental analysis, I evaluate my methods on two MARL benchmarks: Gridworld and MPE.

Gridworld. I consider the grid world environments, inspired by the altruistic learning environments setup [11], where each agent can navigate in a grid with four directions. Agents receive a big reward ($+10$) when reaching the goal position (depicted as an apple) with matching colors for the first time in each episode. Conversely, collisions with walls or other agents result in a small penalty (-0.1). Furthermore, there are doors that agents cannot pass through, which can be opened as long as some agent occupies the key position with the same color as the doors. I investigate four variations of the environment (Figure 4.1):

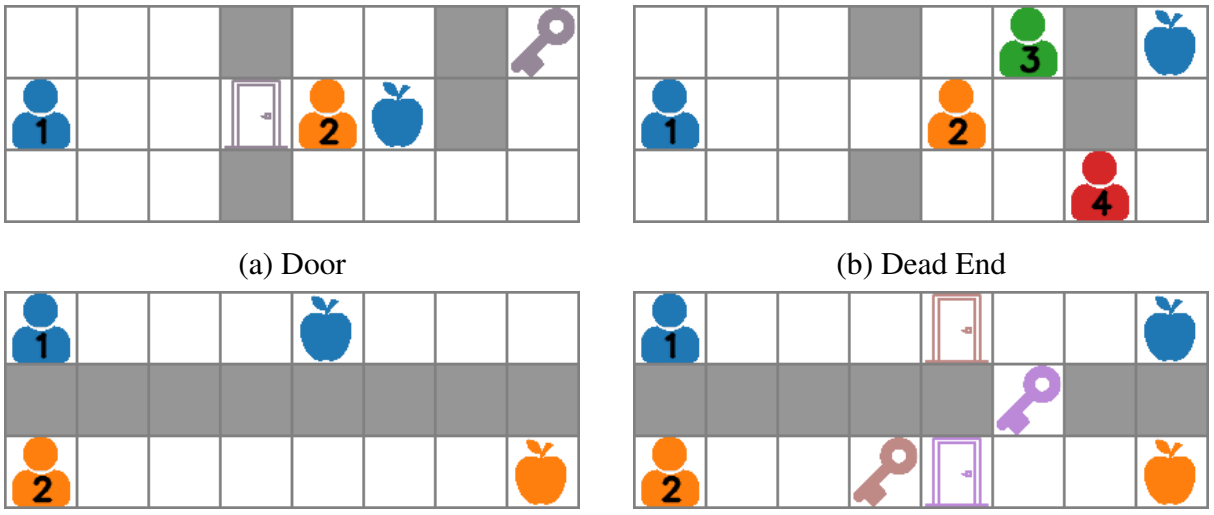
a) **Door** [11]. This environment requires the second agent to open the door for the first one so that it can reach its goal. However, this action does not gain any benefit for that agent. In this environment, selfish learners are expected to fail, while other altruistic learning paradigms will succeed. This environment illustrates the advantage of altruistic behaviors.

b) **Dead End** [11]. In this scenario, the first agent must navigate to the goal located at the far end of the map, but there are several agents that can block the path of the first agent. This problem illustrates the optimization landscape of MARL that can be complex and dynamical due to the interactions and exploration of other agents. This is a distinction from other settings, where one agent's learning can interfere with the others' optimization problems.

c) **Two Corridors**. Two agents are located at separate corridors, each with their respective goals located on the other side. This scenario does not require any cooperation between agents. However, their tasks vary in difficulty. While the first agent traverses only half the corridor, the second agent explores to the full end. The asymmetric level of exploration means one agent will converge faster than the other one. Once the easier task agent converges, every policy from the harder task agent becomes weak Pareto optimal. This example illustrates the prevalence of weakly optimal solutions in MARL with diverging convergent rates, particularly in tasks with varying difficulty levels.

d) **Two Rooms**. This environment studies the implicit cooperation. Two agents are trapped in two different rooms. While both agents cannot open their doors themselves, the first one can open the door of the second. Upon escaping, the second agent can choose to reciprocate by opening the first agent's door or pursuing its own objectives, potentially betraying its benefactor. However, opening the second agent's door is always optimal from the first agent's perspective, as it is the only way to mutual escape. This optimal behavior is not explicitly incentivized in the reward scheme, as helping the other does not directly contribute to finding the goal.

Multiple Particle Environment (MPE) [5]. MPE features a set of scenarios with a variety of reward structures ranging from competitive to cooperative. In each scenario, each agent is represented as a particle in a 2D plane that can move, communicate, and



(a) Door (b) Dead End

(c) Two Corridors (d) Two Rooms

Figure 4.1: Four scenarios of the Gridworld environment.

interact with other agents. The goals of agents are usually to navigate to some landmarks or avoid other adversary agents. Similar to other works, I consider only the subset of cooperative MPE environments for my evaluation [4, 53], namely Spread and Reference. I omit the Speaker scenario because it focuses on the heterogeneous-agent setting, i.e. different agents have different observation and action spaces [53].

1. **Reference**. This scenario features two agents and three landmarks of different colors. Each agent aims to approach their designated target landmark. However, this target is only known to the other agent. As a result, they can only reach their desired landmark through communication with the other. Both agents are capable of speaking and listening at the same time.

2. **Spread**. In this scenario, there are three agents with three landmarks. Agents need to cover all the landmarks while avoiding collisions to achieve high rewards, which are based on the distance between the closest agent to each landmark.

4.1.2. Baselines

To evaluate the effectiveness of my methods and to analyze the impact of different components within my framework, I divide the empirical assessments into two distinct sets of experiments. The first set examines the performance of MGDA++ in MARL context, with baselines for comparisons are MAPPO [4], IPPO [22], and IQL [26].

The second set of experiments investigates the benefits of integrating a graph neural network into my learning framework. Here, the baselines utilize the same GNN architecture, comparing against DGN [19] and methods from the first section with an added Convolutional Graph Neural Network. Specifically, for non-graph-based approaches, I include the following baselines

1. **Multi-head MAPPO**. I extend MAPPO [4] to work in cooperative environments with different reward functions for each agent. As described in section 3.2., the algorithm is similar to MAPPO, the differences are that the output of the value function is multi-headed, and each head corresponds to predicting the expected returns of the respective agents. The value network is shared between agents and can access global information during training, adhering to the Centralized training decentralized execution (CTDE) paradigm.

2. **IPPO** [22] decomposes the learning of all agents in a completely decentralized manner, where each of them is a single, independent PPO learner. The learning of IPPO disregards the dynamic of all other agents, attributing them to the non-stationarity of the environment. However, with appropriate setups, IPPO can perform competitively with other centralized training methods [28].

3. **IQL** [26] is the extension of the DQN architecture to multiagent environments. Similar to IPPO, IQL also learns decentrally with independent Q networks and separate replay buffers for each agent. Compared to other methods, IQL is the only off-policy, value-based method that I consider in my work.

4. **MGDA w/ MAPPO (MGPO)**. I use the multi-head architecture of MAPPO, but instead of individually optimizing the objective of each agent separately, I use MGDA to find a common direction for all the agents as discussed in section 3.2..

5. **MGDA++ w/ MAPPO (MGPO++)**. This setting is similar to MGPO, but I use my proposed MGDA++ method in place of MGDA.

For the graph-based methods, I use a Convolutional Neural Network as a graph-base network for combining the features of neighboring agents, I base my implementation of GCN on DGN [19], and extend it to MAPPO, MGPO, and MGPO++. The compared baselines are as follows

1. **DGN** [19] employs a Convolutional Graph Neural network with DQN [27], where each agent is represented as a node in the graph and edges between agents represent the connections between them (locality or observability). DGN uses multi-head attention as the convolution kernel to aggregate information of neighboring nodes for node feature representations. DGN has several layers of convolutions to gradually increase the receptive fields for later representations.

2. **GCN-MAPPO**. I extend the MAPPO multi-head in the previous baseline with a Graph Convolutional Network, resulting in GCN-MAPPO. With this change, agents can process information about other agents in the vicinity through the node feature aggregation of the convolution operator. The optimization is the same as in MAPPO, i.e. all agents only optimize their own rewards. The architecture of GCN also uses multi-head attention as in DGN for fair comparisons.

3. **GCN-MGPO**. Similar to GCN-MAPPO, except that the agents now optimize for the objectives of the other agents as well. The optimization of GCN-MGPO follows that of MGPO in the previous setting baseline, i.e. using MGDA for the optimization, with the graph modeling of GCN as in GCN-MAPPO.

4. **GCN-MGPO++**. Similar to GCN-MGPO, but I replace MGDA with my proposed method MGDA++.

4.1.3. Hyperparameter settings

For all the baselines, I use the default hyperparameters reported in the original papers. For consistency, all methods utilize three-layer MLPs with hidden sizes of 64. Hyperparameters of MAPPO, IPPO, and IQL follow the default settings of their repositories. MGPO and MGPO++ adopt MAPPO's hyperparameters. In MGDA++, ϵ is set to 0.05, except in Dead End where it is 0.1. In each scenario, I train all the methods for 500k environment steps in Gridworld and 5M steps in MPE. To ensure fair comparisons and a better analysis of the optimization methods, I avoid parameter sharing in the actor networks for on-policy methods and in the value networks for value-based methods. All experiments are repeated with three different seeds to improve evaluation reliability. With graph-based methods, GCN is used in all methods with the same architecture reported in DGN [19]. For simplicity, I assume that the graphs considered in all experiments are densely connected; i.e. all agents can communicate

with all other agents. I argue that this is not problematic in my test cases because of a relatively small number of agents in each benchmark. Due to computational constraints, I set both the number of convolutional graph layers and the number of attention heads to 1 in all of the graph-based methods. The encoder is a two-layer MLP in MPE and a 3-layer CNN network in Gridworld. All the other MLP networks described in Algorithm 2 is a single-layer MLP. The activation functions are ReLU in all architectures. Refer to table 4.1 for a more complete hyperparameter setting.

Table 4.1: Hyperparameter setting of MGPO++.

Hyperparameters	**Values**	**Hyperparameters**	**Values**
activation	ReLU	use proper time limits	True
use feature normalization	True	training threads	10, 200
learning rate	5e-4, 7e-4	gain	0.01
hidden size	64	data chunk length	10
optim eps	1e-5	weight decay	0
value loss coef	1	use max grad norm	True
max grad norm	10.0	use GAE	True
GAE lambda	0.95	use huber loss	True
ppo epoch	5, 10	huber delta	10.0
epsilon in MGDA++	0.05, 0.1	num graph convolutional layers	1

4.2. Results

4.2.1. Without Graph Neural Network

Gridworld. In Gridworld environment, we observe the superiority of my method to all other baselines (Table 4.2 and Figure 4.2). In the Door scenario, MAPPO with multi-objective optimization is the only method that can converge to the optimal solution for

the first agent, while all other algorithms with single objectives do not learn to cooperate. In this case, we already see the advantage of MGDA++ compared to MGDA. While MGDA helps in finding the optimal policy for the first agent, the second agent is yet to converge at the time of convergence of the first one, after which this agent is unable to update due to the dominance of the diminishingly small gradient from the first agent, analyzed in the previous section.

A similar result is observed in the Dead End scenario. Notably, MGPO does not learn except for the first agent, this highlights the fact that MGDA can only learn the optimal policy for only one agent. In MGPO++, we observe that all agents can learn good solutions without being hindered once some agents converge. I note that there are runs where the first agent is able to reach the goal, illustrating that the algorithm can find several local Pareto solutions, depending on the exploration of agents. This also demonstrates that MGDA++ can converge to strong Pareto solutions with a higher number of objectives ($n > 2$).

In the Two Corridors scenario, all the single objective methods converge as this scenario does not need cooperation. However, MGPO does not converge for the second agent, which is the harder task agent, since after the first agent converges, every policy from the second agent is weakly optimal. Hence, the algorithm stops after the first agent reaches optimal policy, even though the second agent is still learning.

In the last scenario, only MGDA++ is able to reach the optimal policies for both agents. While MGDA can find high rewards for the first agent, the second agent is trapped after the first agent's convergence. The difference between MAPPO and IPPO is that I set the entropy coefficient of MAPPO to 0. While the first agent can find high rewards initially in MAPPO, this is only the artifact of the exploration of the second agent. As a result, when the entropy level of both agents decreases over time, the second agent's performance drops. On the other hand, the performance with IQL is unstable, possibly because of the non-stationarity perceived by the first agent due to the exploration effect of the second one. While this problem is also present in the on-policy methods, it is particularly persistent in the off-policy method because the collected data from the past in the replay buffer differs from the current policy.

MPE. The results on MPE are presented in Figure 4.3. In the Reference scenario,

Scenario	agent	MGPO++	MGPO	MAPPO	IQL	IPPO
Door	1	**10.0 ± 0.0**	9.9 ± 0.0	0.0 ± 0.0	−0.0 ± 0.0	−0.0 ± 0.0
	2	**−0.0 ± 0.0**	−6.9 ± 4.9	**−0.0 ± 0.0**	−0.1 ± 0.0	**−0.0 ± 0.0**
Dead End	1	**3.3 ± 4.7**	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 0.4	0.0 ± 0.0
	2	**−0.0 ± 0.0**	−14.4 ± 0.8	0.0 ± 0.0	−0.0 ± 0.0	0.0 ± 0.0
	3	**−0.0 ± 0.0**	−13.9 ± 2.2	**−0.0 ± 0.0**	−0.1 ± 0.0	**−0.0 ± 0.0**
	4	−0.0 ± 0.0	−14.7 ± 2.2	**0.0 ± 0.0**	−0.1 ± 0.0	−0.0 ± 0.0
Two Corridors	1	**10.0 ± 0.0**	**10.0 ± 0.0**	**10.0 ± 0.0**	9.9 ± 0.0	9.9 ± 0.0
	2	**10.0 ± 0.0**	−1.1 ± 1.6	**10.0 ± 0.0**	9.9 ± 0.0	9.9 ± 0.1
Two Rooms	1	**10.0 ± 0.0**	**10.0 ± 0.0**	−0.0 ± 0.0	−5.6 ± 8.0	6.5 ± 3.6
	2	**9.8 ± 0.2**	−17.7 ± 0.9	0.0 ± 0.0	−0.1 ± 0.0	−0.0 ± 0.0

Table 4.2: Result on Gridworld environment, all reported values are rounded to the first place after decimal. The best values of each agent are highlighted in bold, while the second-best values are underscored.

we observe that MAPPO cannot learn because in this environment, the goal of one agent is only known by the other, and they have to explicitly communicate to reach their respective goal. However, agents are *not* rewarded for communicating the correct landmark locations for the other agent. As a result, the performance of MAPPO agents decreases over time due to the drop in the policy's entropy level, which is relatively high at the beginning of the training process. Agents can still find their target landmarks by chance, but this phenomenon reduces over time as the policy becomes more deterministic as the training progresses. This scenario is similar to the Two Rooms scenario in the previous experiments, and the performance behaviors of all the methods follow a similar pattern. In particular, with MGPO, the second agent does not learn after the first agent converges. This freeze of the second agent also has negative feedback on the performance of the first agent, preventing it from further finding better policies. In contrast, with MGPO++, we observe a small increase in the rewards of both agents at around 3.5M timesteps, possibly because the second agent improves enough to cooperate better with the first agent. This reward jump is completely absent in MGPO.

In the Spread scenario, I do not find much difference across methods (and even between agents), which might be due to the reward landscape of agents being quite

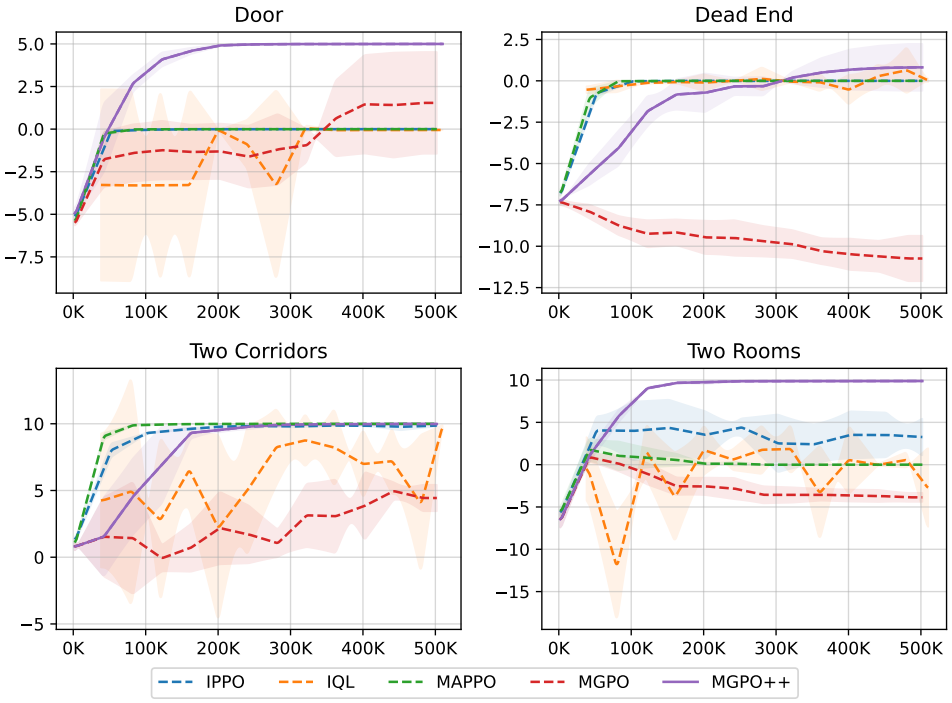


Figure 4.2: Average reward of all agents in the four scenarios. Note that, the averaging operator can be viewed as a particular linearization of the rewards vector.

similar, resulting in an almost fully cooperative situation. In the fully cooperative setting, strong and weak Pareto solutions coincide, and the update directions found by both MGDA and MGDA++ return to the usual gradient vectors. As a result, all of the Policy Gradient-based methods converge at the same rate.

4.2.2. With Graph Neural Network

In this section, I present the experimental results on both MPE and Gridworld with graph-based methods.

Gridworld. We observe a similar pattern to the relative performance of different methods in the previous setting without graph neural networks. Only Altruistic learning algorithms can find good cooperative behaviors between the two agents in the Door scenario. GCN-MGPO performs similarly to the previous setting, with only the first agent converging. In this scenario, GCN-MGPO++ follows up second.

Dead End renders to be a hard-exploration problem where almost all of the considered methods cannot learn good policies in both graph and non-graph settings. With GCN-MGPO, while being able to converge for the first agent, the remaining agents

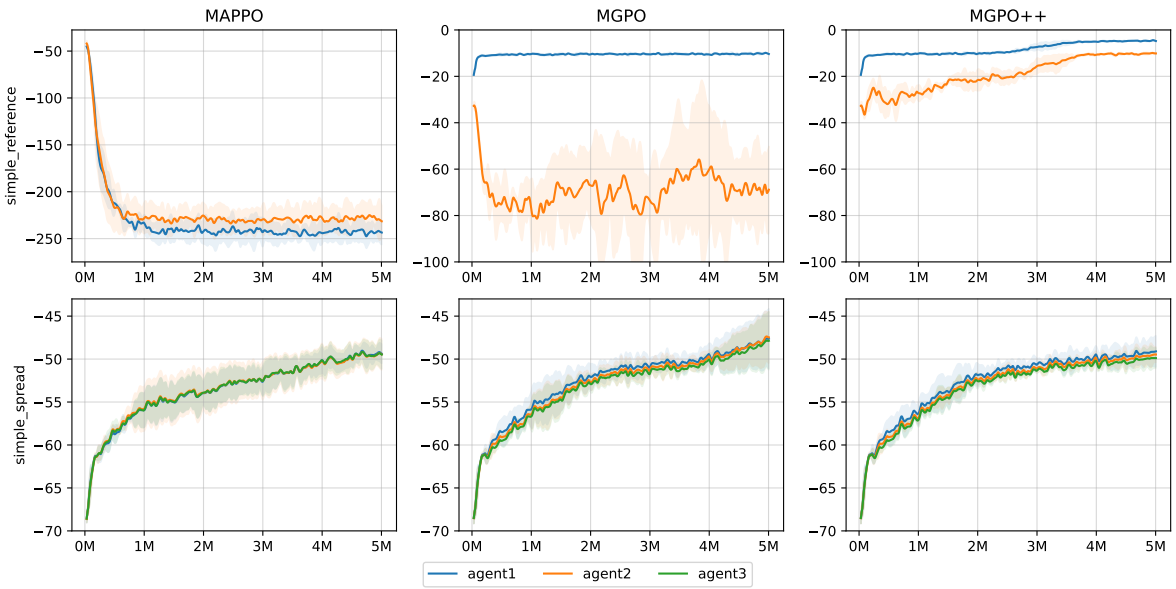


Figure 4.3: Results in MPE benchmark, showing the performance of each agent over the training period.

suffer.

In Two Corridors, all methods converge except for the second agent of GCN-MGPO, which suffers from the weak Pareto convergence problem. In particular, we observe that DGN does not perform well in this scenario, even when this is a very simple environment with no cooperation required. This suggests that DGN might not be a stable approach in terms of graph modeling of agent interactions; DGN's communication signals aggregated from the neighboring agent graphs turn out to be noisy and can interfere with the learning of the other agent, even when their actual interactions are independent.

GCN-MGPO++ remains superior in Two-Rooms, with both agents learning to meaningfully cooperate with each other. Similarly, GCN-MGPO learns the optimal policy for the first agent, but this optimality hinders the second agent learning process. Finally, both GCN-MAPPO and DGN fail in this scenario.

MPE. Experimental results in MPE remain quite similar to the previous section (Table 4.3). First of all, Spread appears to be too close to the fully cooperative setting, and thus all Policy Gradient-based methods admit very similar learning curves. DGN greatly falls behind the other algorithms, and completely stops improving after around 500k

Scenario	agent	GCN-MGPO++	GCN-MGPO	GCN-MAPPO	DGN
Door	1	$\underline{3.3 \pm 4.7}$	$\mathbf{10.0 \pm 0.0}$	-0.2 ± 0.2	-0.3 ± 0.7
	2	$\mathbf{0.0 \pm 0.0}$	-9.8 ± 0.3	$\mathbf{-0.0 \pm 0.0}$	$\underline{-0.1 \pm 0.0}$
Dead End	1	$\underline{0.0 \pm 0.0}$	$\mathbf{3.3 \pm 4.7}$	$\underline{-0.0 \pm 0.0}$	-0.8 ± 0.9
	2	$\mathbf{0.0 \pm 0.0}$	-17.1 ± 3.6	$\mathbf{-0.0 \pm 0.0}$	$\underline{-0.1 \pm 0.0}$
	3	$\mathbf{0.0 \pm 0.0}$	-10.2 ± 8.0	$\underline{-0.1 \pm 0.1}$	$\underline{-0.1 \pm 0.0}$
	4	$\mathbf{0.0 \pm 0.0}$	-13.1 ± 9.1	$\underline{-0.8 \pm 1.1}$	$\mathbf{-0.1 \pm 0.0}$
Two Corridors	1	$\mathbf{10.0 \pm 0.0}$	$\mathbf{10.0 \pm 0.0}$	$\mathbf{10.0 \pm 0.0}$	$\underline{8.6 \pm 0.6}$
	2	$\mathbf{9.9 \pm 0.1}$	-6.3 ± 5.3	$\mathbf{9.8 \pm 0.1}$	$\underline{5.6 \pm 1.3}$
Two Rooms	1	$\mathbf{10.0 \pm 0.0}$	$\mathbf{10.0 \pm 0.0}$	$\underline{3.3 \pm 4.7}$	$\underline{3.3 \pm 1.2}$
	2	$\mathbf{3.3 \pm 4.7}$	-9.2 ± 8.0	$\underline{-0.0 \pm 0.0}$	-1.6 ± 0.4

Table 4.3: Result on Gridworld environment with graph-based methods.

environment steps. In this scenario, we observe that the asymptotic performances of graph-based methods are slightly behind when no graph neural networks are used.

Reference is a scenario where applying Graph Neural Networks is expected to improve the performance of non-altruistic learning methods, since the agents have to *communicate* to solve their tasks, and the problem of the single agent optimization approach is that selfish agents have no reward incentives to do so. Indeed, we observe a notable improvement in GCN-MAPPO in this setting. More specifically, when compared to the non-graph-based approach, where MAPPO dwindles around the total rewards of -230 for the whole training process, the rewards are improved to around -100 when applied to graph neural networks. The explanation for this improvement is that instead of relying on the other agent to communicate, GNN can directly aggregate that piece of information that should not have been available to the agents themselves. This information aggregation scheme of graph modeling network, while shown to improve the performance, only solves the problem of communication, it does not solve the problem of optimization; agents still have no way to dictate others to act for their benefit, as indicated in the section 3.1.. As a result, both DGN and GCN-MAPPO fall short compared to MGPO and MGPO++. The improvement of graph neural networks to MGPO and MGPO++ remains negligible since both address the communication problem through *optimizations* rather than leveraging *information sharing* as in GCN.

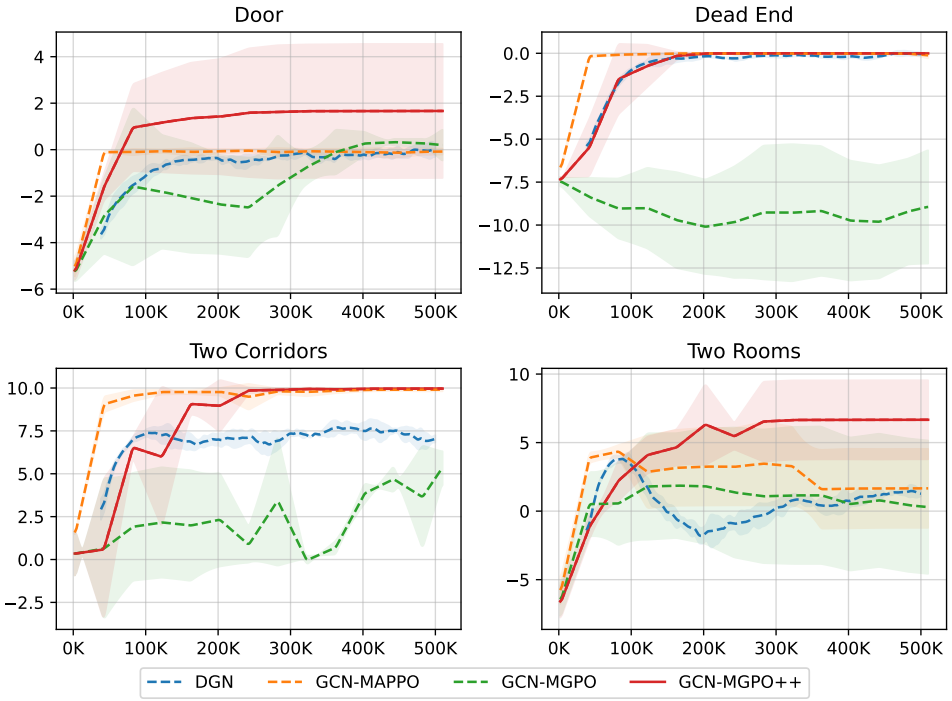


Figure 4.4: Average reward of all agents in the four scenarios with graph-based methods.

4.2.3. Summary of the experimental findings

In this section, I give a summary of my empirical results presented in the previous sections.

The proposed MGDA++ outperforms all other baselines from both the traditional optimization with independent learning in MARL and the multi-objective optimization approach using MGDA. This result is consistent in both Gridworld and MPE benchmarks. Compared to MGDA, the algorithm can avoid suboptimal convergence to Weak Pareto optimal policies. Compared to the independent learning approach, the algorithm successfully elicits cooperation between agents, even when their reward functions do not explicitly encode those behaviors. On the other hand, MGPO constantly fails to find satisfactory policies for all agents in most of the tested scenarios, this suggests that Weak Pareto points are very common in cooperative MARL problems.

Unfortunately, applying graph neural networks to the proposed framework yields limited benefits. While in MGPO++ the performance remains almost the same in MPE, it slightly declines in two out of 4 scenarios in Gridworld. Among all tested approaches, MAPPO shows the most significant benefit from incorporating GCN, though this advantage is only seen biggest in reference since this environment requires

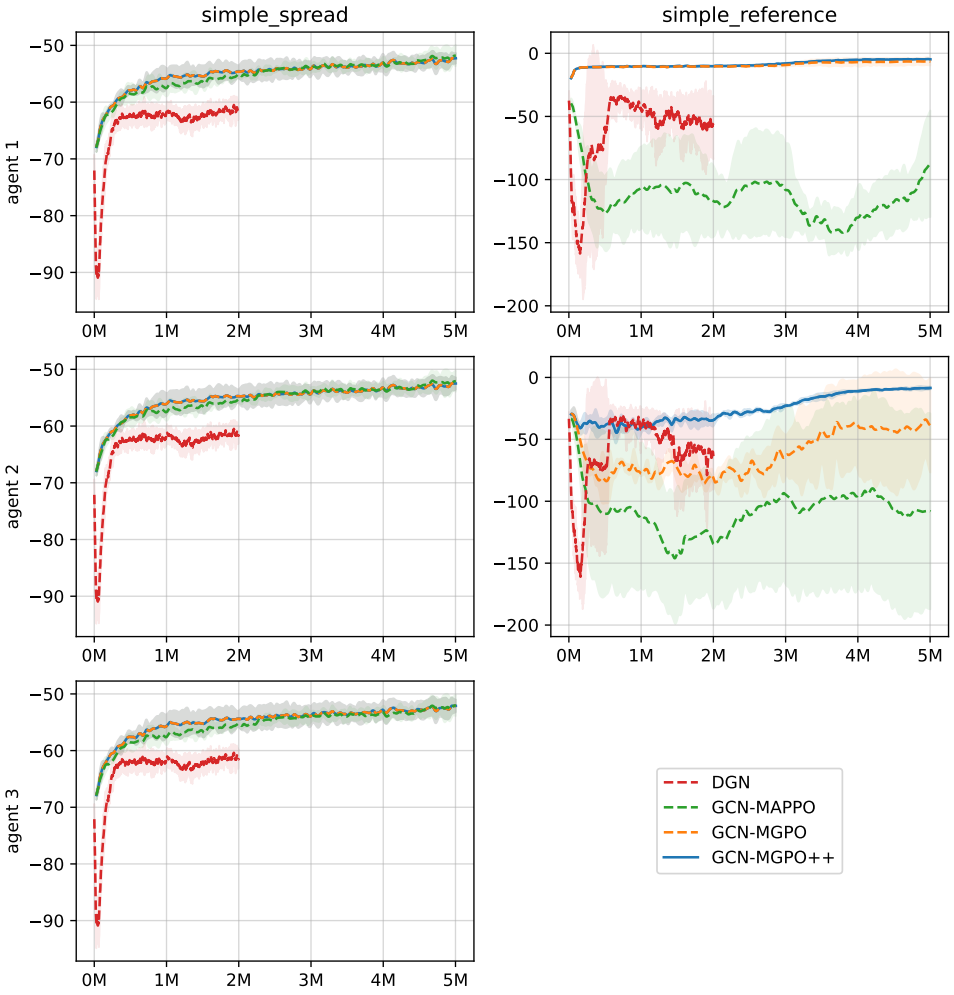


Figure 4.5: Result on MPE with graph-based methods.

agents to communicate. MGPO also seems to benefit from GCN in reference, but the improvement is modest. Still, GCN-MGPO++ continues to outperform other baselines as seen in Figure 4.4, and the problem of Weak Pareto convergence in MGPO persists even with graph-based method (depicted in Figure 4.5).

The impact of agents' communication on the learning of MARL methods has been scrutinized in previous works, with some reporting a possibility of worse outcomes. For example, Nayak et al. [43] remark that more information might not necessarily be better for performance, and introducing communication can increase the sample complexity of MARL algorithms as the increasing state-space dimensionality. This observation is corroborated by Jiang et al. [54]; redundant communication will negatively impact performance. Several promising avenues for reducing the adverse effect of communication are communication pruning [54], communication temporal regularization [19], or some form of information bottleneck [55] in the communication

protocols such as Information Aggregation in InforMARL [43]. Exploring these strategies is left for future work.

Conclusion

In this thesis, I explore the possibility of training multi-agents in a cooperative scenario. By viewing the vector-based reward as a multi-objective task, we can apply the well-known MGDA algorithm to the MARL domains. However, the standard MGDA only guarantees convergence to a weak Pareto optimal solution. Therefore, I introduce MGDA++, which filters out the small norm objective gradients to enhance convergence properties. I also provide theoretical analysis for MGDA++ in bi-objective problems with convex and smooth settings. I further apply a graph neural network to model the interaction between agents in this framework. The combination of MGDA++ and trust region MARL with GCN is evaluated in different scenarios in the Gridworld and MPE benchmarks. The results highlight that this proposed method can direct the agents' policy to strong Pareto optimality, outperforming various baselines.

Future works: Based on the result of this thesis, it can be extended with several promising directions as follows

- *Generalize to $n > 2$ tasks.* One possible direction is to extend the theoretical analysis to a general number of objectives. Furthermore, it would be valuable to evaluate MGDA++ on a broader range of testing scenarios beyond MARL benchmarks, such as Multi-Task learning [31] and Multi-Task RL [40].

- *More complex Benchmarks.* While the evaluation mainly focuses on toy examples like Gridworld and the small-scale MPE benchmark, testing MGPO++ in more challenging tasks would offer greater insights. From a practical perspective, experimenting with more complex scenarios and large-scale MARL benchmarks would help further validate the effectiveness of combining MGDA++ with the trust region framework introduced in this thesis.

- *Improve the Graph Neural Network Component.* Future work could investigate

improved architectures for the GNN component, focusing on minimizing its limitations in scenarios where communication is of limited relevance. Additionally, a benchmark on communication MARL that can demonstrate the advantage of the proposed framework with graph neural networks could be used to highlight this work.

Parts of this thesis will be presented under the title *"Toward Finding Strong Pareto Optimal Policies in Multi-Agent Reinforcement Learning"* in ACML Journal track, held in Hanoi, VietNam on December 2024.

Publications related to this thesis

1. Le, Bang Giang, and Viet Cuong Ta. ”Toward Finding Strong Pareto Optimal Policies in Multi-Agent Reinforcement Learning.” arXiv preprint arXiv:2410.19372 (2024). (To be presented at ACML 2024 Journal Track)

2. Le, Bang-Giang, Thi-Linh Hoang, Hai-Dang Kieu, and Viet-Cuong Ta. ”Structural and Compact Latent Representation Learning on Sparse Reward Environments.” In Asian Conference on Intelligent Information and Database Systems, pp. 40-51. Singapore: Springer Nature Singapore, 2023.

3. Le, Bang Giang, and Viet Cuong Ta. ”Distill Knowledge in Multi-task Reinforcement Learning with Optimal-Transport Regularization.” In 2022 14th International Conference on Knowledge and Systems Engineering (KSE), pp. 1-6. IEEE, 2022.

4. Le, Bang Giang, and Viet Cuong Ta. ”On the Effectiveness of Regularization Methods for Soft Actor-Critic in Discrete-Action Domains” (Accepted with minor revision to IEEE Transactions on Systems, Man and Cybernetics: Systems).

5. Le, Bang Giang, and Viet Cuong Ta. ”Low Variance Trust Region Optimization with Independent Actors and Sequential Updates in Cooperative Multi-agent Reinforcement Learning” (Major revision, submited to Journal of Autonomous Agents and Multi-Agent Systems).

Bibliography

[1] X. Lin, H.-L. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong, "Pareto multi-task learning," *Advances in neural information processing systems*, vol. 32, 2019.

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[3] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[4] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24611–24624, 2022.

[5] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[6] S. Hu, Y. Zhong, M. Gao, W. Wang, H. Dong, X. Liang, Z. Li, X. Chang, and Y. Yang, "Marllib: A scalable and efficient multi-agent reinforcement learning library," *Journal of Machine Learning Research*, vol. 24, no. 315, pp. 1–23, 2023.

[7] S. Leonardos, W. Overman, I. Panageas, and G. Piliouras, "Global convergence of multi-agent policy gradient in markov potential games," *arXiv preprint arXiv:2106.01969*, 2021.

[8] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[9] F. Christianos, G. Papoudakis, and S. V. Albrecht, "Pareto actor-critic for equilibrium selection in multi-agent reinforcement learning," *arXiv preprint arXiv:2209.14344*, 2022.

[10] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," *arXiv preprint arXiv:2006.07869*, 2020.

[11] T. Franzmeyer, M. Malinowski, and J. F. Henriques, "Learning altruistic behaviours in reinforcement learning without external rewards," 2022.

[12] J.-A. Désidéri, "Multiple-gradient descent algorithm (mgda) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5-6, pp. 313–318, 2012.

[13] J. Fliege, A. I. F. Vaz, and L. N. Vicente, "Complexity of gradient descent for multiobjective optimization," *Optimization Methods and Software*, vol. 34, no. 5, pp. 949–959, 2019.

[14] Y. Zhao, Z. Yang, Z. Wang, and J. D. Lee, "Local optimization achieves global optimality in multi-agent reinforcement learning," *arXiv preprint arXiv:2305.04819*, 2023.

[15] J. G. Kuba, R. Chen, M. Wen, Y. Wen, F. Sun, J. Wang, and Y. Yang, "Trust region policy optimisation in multi-agent reinforcement learning," *arXiv preprint arXiv:2109.11251*, 2021.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[17] S. Morad, R. Kortvelesy, M. Bettini, S. Liwicki, and A. Prorok, "Popgym: Benchmarking partially observable reinforcement learning," *arXiv preprint arXiv:2303.01859*, 2023.

[18] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. De Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," *arXiv preprint arXiv:1707.09183*, 2017.

[19] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," *arXiv preprint arXiv:1810.09202*, 2018.

[20] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043*, 2019.

[21] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[22] C. S. De Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. Torr, M. Sun, and S. Whiteson, "Is independent learning all you need in the starcraft multi-agent challenge?," *arXiv preprint arXiv:2011.09533*, 2020.

[23] L. Zheng, J. Yang, H. Cai, M. Zhou, W. Zhang, J. Wang, and Y. Yu, "Magent: A many-agent reinforcement learning platform for artificial collective intelligence," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

[26] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[27] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[28] M. Sun, S. Devlin, J. Beck, K. Hofmann, and S. Whiteson, "Trust region bounds for decentralized ppo under non-stationarity," *arXiv preprint arXiv:2202.00082*, 2022.

[29] S. Cato, "When is weak pareto equivalent to strong pareto?," *Economics Letters*, vol. 222, p. 110953, 2023.

[30] J. Fliege and B. F. Svaiter, "Steepest descent methods for multicriteria optimization," *Mathematical methods of operations research*, vol. 51, pp. 479–494, 2000.

[31] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[32] Y. G. Evtushenko and M. Posypkin, "A deterministic algorithm for global multi-objective optimization," *Optimization Methods and Software*, vol. 29, no. 5, pp. 1005–1019, 2014.

[33] S. Zhou, W. Zhang, J. Jiang, W. Zhong, J. Gu, and W. Zhu, "On the convergence of stochastic multi-objective gradient manipulation and beyond," *Advances in Neural Information Processing Systems*, vol. 35, pp. 38103–38115, 2022.

[34] A. Roy, G. So, and Y.-A. Ma, "Optimization on pareto sets: On a theory of multi-objective optimization," *arXiv preprint arXiv:2308.02145*, 2023.

[35] L. Zeng, Y. Dai, and Y. Huang, "Convergence rate of gradient descent method for multi-objective optimization," *Journal of Computational Mathematics*, vol. 37, no. 5, p. 689, 2019.

[36] D. Mahapatra and V. Rajan, "Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization," in *International Conference on Machine Learning*, pp. 6597–6607, PMLR, 2020.

[37] M. Momma, C. Dong, and J. Liu, "A multi-objective/multi-task learning framework induced by pareto stationarity," in *International Conference on Machine Learning*, pp. 15895–15907, PMLR, 2022.

[38] P. Xiao, H. Ban, and K. Ji, "Direction-oriented multi-objective learning: Simple and provable stochastic algorithms," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[39] Y. Hu, R. Xian, Q. Wu, Q. Fan, L. Yin, and H. Zhao, "Revisiting scalarization in multi-task learning: A theoretical perspective," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[40] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

[41] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18878–18890, 2021.

[42] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[43] S. Nayak, K. Choi, W. Ding, S. Dolan, K. Gopalakrishnan, and H. Balakrishnan, "Scalable multi-agent reinforcement learning through intelligent information aggregation," in *International Conference on Machine Learning*, pp. 25817–25833, PMLR, 2023.

[44] H. Ryu, H. Shin, and J. Park, "Multi-agent actor-critic with hierarchical graph attention network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 7236–7243, 2020.

[45] J. Sheng, X. Wang, B. Jin, J. Yan, W. Li, T.-H. Chang, J. Wang, and H. Zha, "Learning structured communication for multi-agent reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 2, p. 50, 2022.

[46] S. Hu, L. Shen, Y. Zhang, and D. Tao, "Learning multi-agent communication from graph modeling perspective," *arXiv preprint arXiv:2405.08550*, 2024.

[47] C. Zhu, M. Dastani, and S. Wang, "A survey of multi-agent deep reinforcement learning with communication," *Autonomous Agents and Multi-Agent Systems*, vol. 38, no. 1, p. 4, 2024.

[48] L. Yu, Y. Qiu, Q. Wang, X. Zhang, and J. Wang, "Low entropy communication in multi-agent reinforcement learning," in *ICC 2023-IEEE International Conference on Communications*, pp. 5173–5178, IEEE, 2023.

[49] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[50] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," in *International Conference on machine learning*, pp. 1538–1546, PMLR, 2019.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[52] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[53] Y. Zhong, J. G. Kuba, X. Feng, S. Hu, J. Ji, and Y. Yang, "Heterogeneous-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 25, no. 1-67, p. 1, 2024.

[54] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," *Advances in neural information processing systems*, vol. 31, 2018.

[55] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.