Multi-expert Preference Alignment in Reinforcement Learning

by

Litian Li

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday February 6, 2024 at 13:30 PM.

Student number:	5598419	
Project duration:	March 6, 2023 – February 6, 2024	
Thesis committee:	Dr. L. C. Siebert, ,	TU Delft, supervisor
	Dr. Matthijs. Spaan,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

I was attracted to this project from the beginning, as the concept aligns with what I have always wanted to pursue since deciding to major in computer science. I am broadly interested in the development of autonomous agents that can behave like human beings, possessing the ability to perceive surrounding information, infer and reason from what they perceive, and act according to the environment and its requirements, or even based on their own motivation. Reinforcement Learning is partly a solution to this puzzle, serving as an abstract concept that simulates an agent receiving environmental information and learning behavior through interaction with the environment. Adaptation to changes in the environment is more closely aligned with real-world scenarios, where the agent would be required to adapt and align with multiple requirements in order to accommodate different preferences, either from the current context, individual preferences, or domain knowledge.

This project has broadly and deeply immersed me in the related concepts, knowledge, and research in corresponding areas. Throughout the entire process, I have always delved into numerous challenging concepts and state-of-the-art solutions for different related research fields. Confusion often arises when I am distracted by potential research directions, solutions, and their underlying assumptions. Determining the relationships among all these elements and gaining a deeper understanding of them, as well as how they relate to my research, presents quite a challenge. However, the excitement and pleasure always follow once I achieve a better understanding every time.

Lastly, I would like to express my sincere appreciation to my supervisor, Dr. L. C. Siebert. I am grateful to him for accepting me to undertake this project as my thesis, and for being so kind, patient, and guiding when I faced difficulties and frustration throughout the project. I am also thankful for his tolerance of my many shortcomings, providing ample space for me to grow through the project. Furthermore, he consistently offered critical insights and guidance, which greatly assisted me in researching on the right track.

Litian Li
Delft, January 2024

Abstract

This project explores adaptation to preference shifts in Multi-objective Reinforcement Learning (MORL), with a focus on how Reinforcement Learning (RL) agents can align with the preferences of multiple experts. This alignment can occur across various scenarios featuring distinct preferences of experts or within a single scenario that experiences a shift in preferences. Unlike traditional RL, which requires retraining policies every time an individual expert's preference is introduced—resulting in high computational complexity and impracticality—this project proposes a single-policy RL algorithm named Generalized Preference-based PPO (GPB PPO). This algorithm integrates environmental information and experts' preference requirements throughout the decision-making process. By exposing the agent to diverse preference scenarios during training, it learns a policy conditional on preference and can generalize to any given preference. This method eliminates the need for explicit retraining and additional adaptation when preferences shift. The generalization and adaptation capabilities of GPB PPO are further evaluated under both stationary and non-stationary environments.

Contents

Preface i

Summary ii

1 Introduction 1
- 1.1 Introduction . . . 1
- 1.2 Research Questions . . . 2
- 1.3 Thesis outline . . . 3

2 Background 4
- 2.1 Reinforcement Learning . . . 4
- 2.2 Deep Reinforcement Learning . . . 7
- 2.3 State Augmentation . . . 8
- 2.4 Multi-objective RL . . . 9
- 2.5 Proximal Policy Optimization . . . 9
- 2.6 Non-Stationary Environments in MORL . . . 10

3 Related Work 11
- 3.1 General RL Adaptation between tasks . . . 11
- 3.2 Adaptation specifically on preference over multi-objective RL Approach . . . 12

4 Method: Generalized Preference-Based PPO 13
- 4.1 State Augmentation with processed preference information . . . 13
- 4.2 Neural Network Architecture. . . . 13
- 4.3 Preference Sampling with Parallel Environment . . . 14
- 4.4 Policy Evaluation in GPB PPO . . . 14
- 4.5 GPB PPO algorithm training procedure . . . 16

5 Experiment 18
- 5.1 Environment Description . . . 18
- 5.2 Evaluation Metrics . . . 18
  - 5.2.1 Generalization . . . 18
  - 5.2.2 Performance Comparing to Standard PPO . . . 19
  - 5.2.3 Preference Sampling Granularity Influence . . . 20
  - 5.2.4 Performance in time-dependent preference changing condition . . . 20
  - 5.2.5 Performance in region-dependent preference changing condition . . . 20
- 5.3 Evaluation Procedure . . . 20
  - 5.3.1 Procedure to evaluate generalization . . . 20
  - 5.3.2 Procesure to evaluate performance comparison between GPB PPO and Standard PPO . . . 21
  - 5.3.3 Procedure to evaluate the preference sampling granularity . . . 21
  - 5.3.4 Procedure to evaluate the performance under time-dependent preference change 21
  - 5.3.5 Procedure to evaluate the performance in region-dependent preference changing condition . . . 22
- 5.4 Experiment 1: Result Analysis . . . 23
  - 5.4.1 Result: Solution coverage . . . 24
  - 5.4.2 Result: Performance Comparing to Standard PPO . . . 25
  - 5.4.3 Result: Preference Sampling Granularity Influence . . . 25
  - 5.4.4 Result: Performance on Time-Dependent Preference Changing Environment . . 27
  - 5.4.5 Result: Performance on Region-Dependent Preference Changing Environment . 27
- 5.5 Experiment 2: Result Analysis . . . 30
  - 5.5.1 Result: Solution Coverage . . . 30

5.5.2 Result: Performance Comparing to Standard PPO 31
5.5.3 Result: Preference Sampling Granularity Influence 31
5.5.4 Result: Performance In Time-preferences Changing Environment 32
5.5.5 Result: Performance on Region-Dependent Preference Changing Environment . 34

6 Discussion and Conclusion **37**
6.1 Discussion 37
6.2 Conclusion 38

7 Future Work **39**

References **40**

A Implementation Details **42**
A.1 Network Implementation details 42
A.2 Training Hyperparameters 42
A.3 Details of State Augmentation 43

1 Introduction

1.1. Introduction

With the rapid development of Reinforcement Learning (RL) across various fields and their corresponding applications, more complex considerations have arisen. These include whether, in a complex task scenario, an agent can adjust its preferences for trading off among different competing objectives if needed. These considerations could be task-specific requirements in various situations or pertain to safety concerns. For example, in a warehouse, there are various tasks, such as delivering goods and assisting workers with unloading items. The importance of these tasks may vary by date or according to different regions of the warehouse. Furthermore, an autonomous driving agent might have different preferences for prioritizing driving styles, such as being extra careful and minimizing steering on rainy days, or maintaining a certain speed on the highway.

The importance of agents being able to behave appropriately in various situations has been extensively researched across multiple fields. Padakandla et al. [14] explore the challenges that RL agents face under different operating conditions and discuss the development of methods for handling environments that change dynamically. Additionally, Li et al. [9] highlight the necessity for agents to adapt in urban driving scenarios. They demonstrate that agents must adjust to different preferences, including adhering to traffic rules, adapting driving styles to varying road conditions, and ensuring passenger comfort while aiming to reach the destination as quickly as possible. From another perspective, the concept of human alignment, as discussed by Gabriel et al. [6], is another instance where agents should behave appropriately in different contexts. This is due to the variability in human preferences or ethics under different conditions. Moreover, Cavalcante Siebert et al. [3] discuss the concept of meaningful human control. In the real world meaningful control might vary in different situations, a control that is meaningful in one condition may differ in others. From this perspective, the importance of an agent's ability to behave appropriately under various situations is further highlighted. Furthermore, the alignment problem, as described by Vamplew et al. [26], can be formulated as a multi-objective problem where the importance assigned to different objectives reflects varying human preferences.

To enable an agent to behave appropriately in various scenarios, the concept of adaptability has been introduced in the field of RL. Various approaches have been researched to address the challenges of adaptation, which traditional RL methods cannot solve. The most well-known approach is gradient-based meta-RL, as seen in the works of Finn et al. [5], Al-Shedivat et al. [24], and Nagabandi et al. ([13]). These approaches utilize the gradient information from the meta-policy's gradient descent into potential task-specific policies, enabling the meta-policy parameters to be quickly adapted to task-specific policies. Additionally, several other important approaches in meta-RL have been developed, such as RNN-based meta-RL by Duan et al. [4], and context-based meta-RL by Rakelly et al. [19] and Luo et al. [10]. The former focuses on learning the latent information of task-specific environments and encapsulating this information into the RNN layer, while context-based RL aims to infer contextual latent variables, making the behavior of the context/task conditional on these inferred variables, In different environment dynamics, like road conditions, as uphill, sandy terrain, and waterlogged surfaces,

agents may not need to re-learn specific policy for each type of road. Instead, they can collect a small amount of experience and quickly update their policy based on the characteristics of meta-policy.

In the context specific to multi-objective RL, as discussed in Roijers et al. [20]. This implies assigning different weights to various objectives and prioritizing them accordingly. Traditional methods for solving multi-objective RL are based on two categories. The first, scalarization methods [20], involve combining multiple objectives into a single objective by assigning weights to each and aggregating them into a single scalar value. This approach aims to learn a policy given the preferences using traditional RL techniques. The second category, as explored by Pirotta et al. (2015) [17], seeks to find a set of Pareto front solutions in the preference space. However, these methods often encounter scalability and computational intensity issues. Thus some researchers are introduced based on leveraging the advantages of Universal Value Function framework proposed by Schaul et al. [21], which can learn policies based not only on the state but also on the given preferences, they have been developed to address adaptability in specific multi-objective problems. These approaches, similar in learning policies conditional on both state and preferences, include the work of Yang and Sun et al. [29] and Buet-Golfouse and Pahwa [2]. They introduced algorithms that learn policy conditional on all possible preferences during training, which can generalize well across the entire preference space. This is achieved by learning policies based on state and preference-conditional Q-values, where each potential preference has its corresponding Q-value. Additionally, Abels et al. [1] developed a similar approach but with a focus on online adaptation. All of these approaches rely on the preference-based Q-value vector in an off-policy manner.

In this paper, we explore how a RL agent can handle multiple different stationary environments with various preference conditions, as well as non-stationary environments where preferences shift. The concepts of stationary and non-stationary environments, as defined by Padakandla (2021) [14], refer to shifts in environmental dynamics. Our research focuses on shifts in the reward function. A stationary environment implies a Markov process with a fixed reward function and specified preferences, whereas a non-stationary environment suggests that the preferences for the reward function might shift, leading to a Markov process with multiple different reward functions. More specifically, we investigate how an agent can handle multiple different stationary scenarios (across multiple episodes) without retraining a new policy for each new preference, and how it can adapt in a non-stationary scenario (within a single episode) when the reward function changes. For the non-stationary scenario, we treat it as consisting of multiple different stationary scenarios, each with a specific reward function, allowing the agent to adapt from one scenario to the next. In a realistic setting, the agent may face the challenge of having insufficient time to gather experience for adapting to new preferences. In line with the work of Yang et al. (2019) [29], Buet et al. (2023) [2], and Abels et al. (2019) [1], we aim to narrow the gap in how an agent can maintain a single policy that generalizes to any encountered preference in the preference space. This approach seeks to minimize intensive re-training and ensure effective adaptation in changing scenarios.

In the contribution, we propose a generalized Proximal Policy Optimization (PPO) [22] in an on-policy manner, different to the off-policy manner by [29, 2, 1] which is capable of approximating the preference space. This enables the agent to align with the preferences of multiple different experts either in a stationary or non-stationary scenario. We utilize the advantages of on-policy PPO, characterized by parallel environment training, by sampling all possible preferences during training. The state information, augmented with the processed preference, is taken as input. This approach allows for learning a single policy that can adapt to any given preference in the preference space.

1.2. Research Questions

Multiple preferences across different episodes of stationary conditions or within episodes of non-stationary condition pose the challenge to RL that whether the RL approach can simplify such as having a policy can generalize and perform well on any given preferences, as well as adapt well when the preference shift in a non-stationary scenario. These challenges bring up the following research questions:

1. **How can preference information be incorporated into Decision-Making?**

2. **How such a preference-based approach be able to generalize across the preference space?**

3. **How does such a preference-based approach perform on a given preference?**

4. **How can this approach make Trade-Off in a preference shift scenario?**

1.3. Thesis outline

The thesis is structured into several chapters, beginning with a Background Information chapter that introduces foundational knowledge necessary to understand this project, such as insights into RL and DRL, multi-objective RL, state augmentation, and PPO. This is followed by a Related Work chapter that specifically describes the research closely related to the problem this project aims to solve. Subsequently, a Methods chapter details the approach proposed by this project. Next, the Experiments chapter discusses the experimental setup and corresponding results. The thesis concludes with a chapter on Discussion, Conclusion, and Future Work.

2

Background

This chapter will describe the background knowledge related to this project, building a foundation for understanding it. It begins with a basic description of Reinforcement Learning, Deep Reinforcement Learning, Multi-Objective Reinforcement Learning, State Augmentation, and Non-Stationary Environments.

2.1. Reinforcement Learning

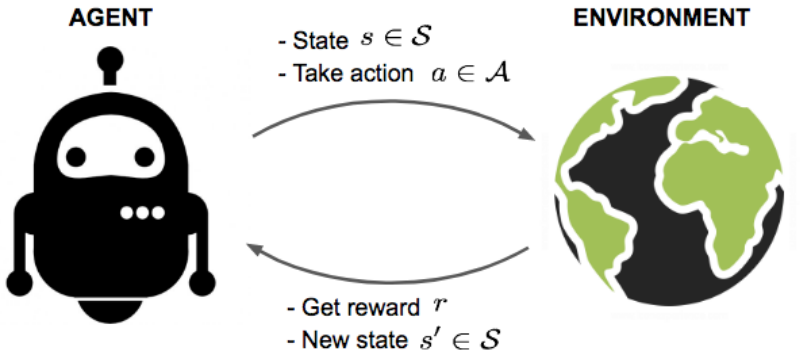

Figure 2.1: Reinforcement Learning [28]

RL is one type of machine learning [25], where an agent learns to make decisions by interacting with an environment. The core principle of RL is trial and error, where the agent performs actions based on the state information and receives feedback in the form of rewards as positive feedback or penalties as negative feedback from the environment. The goal of an RL agent is to maximize the total cumulative reward over time, as a higher total cumulative reward implies a better policy.

In the setting of RL, there are some key components, which include:

- Agent: The learner or decision-maker that interacts with the environment.
- Environment: The system the agent interacts with, and from which it gets state and reward information.
- State: A representation of the current situation of the environment as perceived by the agent.
- Action: What the agent can do to influence the state of the environment, such as moving or collecting items, which affects the perceived state information.
- Reward: Feedback from the environment in response to the actions taken by the agent, guiding the agent towards the goal of maximizing the final return, a reward function is represented as $R(s, a)$.

- Policy: A policy π is a strategy used by the agent, defined as a function $\pi : S \to A$ in deterministic policies, or $\pi : S \times A \to [0, 1]$ in stochastic policies, mapping states to actions or probabilities of taking each action, respectively.

Policy

In reinforcement learning, a policy defines the behavior of an agent in a given environment. Formally, a policy π is a mapping from states S to actions A. Depending on the nature of the policy, it can be deterministic or stochastic.

A deterministic policy is a function $\pi : S \to A$ that maps each state $s \in S$ to a specific action $a \in A$. In mathematical terms, a deterministic policy for state s can be defined as:

$$a = \pi(s)$$

This means that for a given state s, the policy π always selects the same action a.

On the other hand, a stochastic policy assigns a probability distribution over actions for each state. It is represented as $\pi(a|s)$, which is the probability of taking action a when in state s. For a sophisticated stochastic policy, the choice of action can depend on a variety of factors and can be represented as:

$$\pi(a|s) = P(A_t = a|S_t = s)$$

where A_t is the action taken at time t and S_t is the state at time t. Sophisticated policies can adapt to the environment's dynamics and are particularly useful in complex, non-deterministic environments.

Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) provide a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker.

An MDP is defined by a tuple (S, A, P, R, γ), where: - S is a set of states, - A is a set of actions, - P is a state transition probability matrix, - R is a reward function, and - γ is a discount factor, $0 \leq \gamma \leq 1$.

The state transition probability matrix P defines the probability of transitioning from state s to state s' after taking action a, denoted as $P(s'|s, a)$.

The reward function R defines the immediate reward received after transitioning from state s to state s' due to action a, often expressed as $R(s, a, s')$.

The objective in an MDP is to find a policy π, which is a function from states S to actions A, that maximizes the cumulative reward. The value of a state s under policy π, denoted as $V^\pi(s)$, is the expected return starting from s and following by π . It is given by:

$$V^\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, \pi\right]$$

Here, R_{t+k+1} is the reward received at time $t + k + 1$, and γ^k is the discount factor applied k steps into the future.

The goal is to find the optimal policy π^* that yields the highest value for each state. The optimal policy satisfies the Bellman optimality equation, which in the value function form is:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} P(s'|s, a) \left[R(s, a, s') + \gamma V^*(s')\right]$$

This equation recursively defines the value of a state as the maximum expected return achievable, considering the immediate reward and the discounted value of the next state.

Value Function

In reinforcement learning, value functions play a crucial role in determining how good it is for an agent to be in a certain state or to perform a certain action in that state. There are two main types: the state value function and the action value function.

The state value function, $V^{\pi}(s)$ evaluates how good the agent is the state, for a policy π is the expected return when starting from state s and following π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

The action value function, or Q-function, $Q^{\pi}(s, a)$ evaluates how good the agent take an action in the state, evaluates the expected return of taking action a in state s under policy π:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Optimal Policy

The optimal value function is to find the best policy in reinforcement learning. It is defined as the maximum expected return that can be obtained from any given state, and it is achieved by following the optimal policy. The optimal state value function, denoted $V^*(s)$, and the optimal action value function, denoted $Q^*(s, a)$, are calculated as the maximum over all policies:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$
$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

The optimal policy π^* is the one that consistently chooses the best action as indicated by the optimal action value function. It is formally expressed as the action that maximizes the expected return in the state value function and the action value function:

$$\pi^* = \arg\max_{a} V^*(s)$$
$$\pi^* = \arg\max_{a} Q^*(s, a)$$

The existence of an optimal policy π^* guarantees that for all states s and actions a, the value functions under the policy π^* will be equal to the optimal value functions:

$$V^{\pi^*}(s) = V^*(s)$$
$$Q^{\pi^*}(s, a) = Q^*(s, a)$$

These equations collectively form the foundation for reinforcement algorithms that seek to approximate the optimal policy and value functions.

Bellman equations

The Bellman equations are fundamental to understanding the principles of dynamic programming and reinforcement learning. They provide recursive decompositions for the value functions, both for the state value function and the action value function.

For the state value function, the Bellman equation describes the relationship between the value of a state and the values of subsequent states. It is given by:

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} P(s'|s, a)[R(s, a, s') + \gamma V^{\pi}(s')]$$

Here, $V^{\pi}(s)$ is the expected return when starting in state s and following policy π, $\pi(a|s)$ is the probability of taking action a in state s under policy π, $P(s'|s, a)$ is the probability of reaching state s' from state s

after taking action a, $R(s, a, s')$ is the reward received after transitioning from state s to state s', and γ is the discount factor.

For the action value function, the Bellman equation is as follows:

$$Q^{\pi}(s, a) = \sum_{s' \in S} P(s'|s, a)[R(s, a, s') + \gamma \sum_{a' \in A} \pi(a'|s')Q^{\pi}(s', a')]$$

In this equation, $Q^{\pi}(s, a)$ represents the expected return of taking action a in state s and then following policy π, and $\pi(a'|s')$ is the probability of taking action a' in state s' under policy π.

The Bellman optimality equations, which are used to find the optimal policy, are derived from these equations by taking the maximum over all actions:

$$V^{*}(s) = \max_{a \in A} \sum_{s' \in S} P(s'|s, a)[R(s, a, s') + \gamma V^{*}(s')]$$

$$Q^{*}(s, a) = \sum_{s' \in S} P(s'|s, a)[R(s, a, s') + \gamma \max_{a' \in A} Q^{*}(s', a')]$$

These optimality equations express the value of a state or action as the best possible outcome from the state onward, considering both the immediate reward and the discounted future rewards.

2.2. Deep Reinforcement Learning

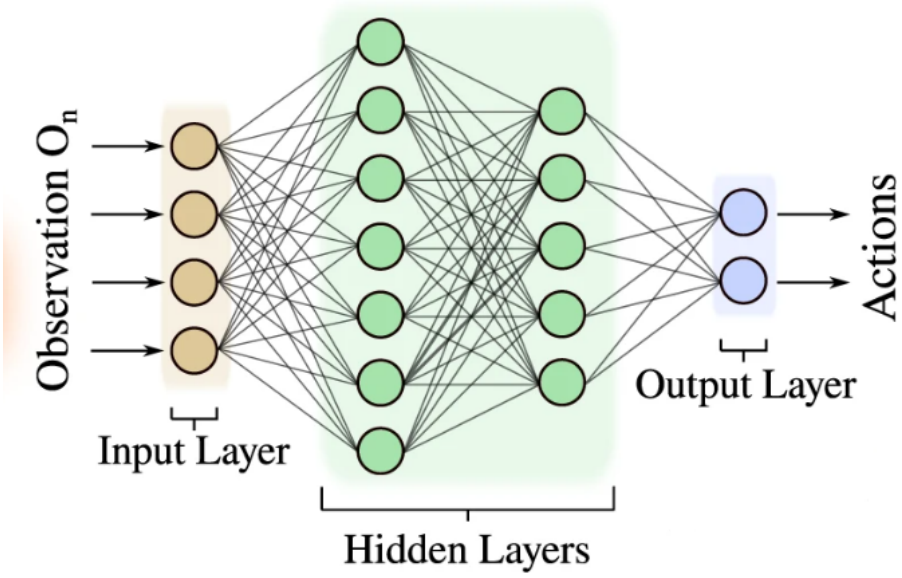

Figure 2.2: Deep Neural Network [12]

Deep Reinforcement Learning (DRL) [11] combines neural networks with a reinforcement learning architecture, enabling agents to learn optimal policies from high-dimensional state information. The core idea is to use a deep neural network to approximate the high-dimensional policy and value functions that are essential for RL.

Function approximation in DRL is the process of estimating the value function or policy function, which can be too complex to represent explicitly when dealing with large state or action spaces. Deep neural networks, due to their universal function approximation [21] properties, are particularly suitable

for this task. The approximation involves parameterizing the value function $V(s;\theta_v)$ or the action-value function $Q(s,a;\theta_q)$ with weights of neural network θ_v and θ_q respectively.

The value function in DRL is approximated using a deep neural network known as a Q-network. The network takes a state s as input and outputs the estimated value of each action. The parameters θ_q of the network are learned by minimizing the loss function, which is typically the mean-squared error between the predicted Q-values and the target Q-values, defined as:

$$L(\theta_q) = \mathbb{E}\left[\left(Q(s,a;\theta_q) - (r + \gamma \max_{a'} Q(s',a';\theta_q^-))\right)^2\right]$$

where θ_q^- represents the parameters of a target network that is periodically updated with θ_q to stabilize training.

Policy function approximation in DRL is often realized through policy gradient methods, where a neural network, parameterized by θ_p, directly maps states to actions. The policy network is trained to maximize an objective function, usually the expected return, by adjusting the parameters θ_p in the direction of the gradient of the expected return. This gradient, given an action probability $\pi(a|s;\theta_p)$, can be estimated as:

$$\nabla_{\theta_p} J(\theta_p) = \mathbb{E}\left[\nabla_{\theta_p} \log \pi(a|s;\theta_p) \cdot Q(s,a;\theta_q)\right]$$

The policy is updated by applying gradient ascent on θ_p using this estimated gradient.

Deep Reinforcement Learning thus leverages the representational power of deep neural networks for function approximation, allowing agents to operate in environments with previously high dimensional states and action space. This advancement has led to significant breakthroughs in applying RL to real-world problems and more complex scenarios.

2.3. State Augmentation

State augmentation is a technique in Deep Reinforcement Learning (DRL) that enhances the state representation to provide the agent with additional context or information, which may not be directly available in the raw state. This technique can improve learning efficiency and policy performance, guiding the agent to a better direction of learning a optimal policy. There are some related approaches to state information shaping, as mentioned in Verma [27] and Kamalaruban [8].

By augmenting the state space, we expand the state s to an augmented state s', which concatenates the original state with extra features or information x:

$$s_{augment} = [s, x]$$

The augmented information x could include past states and actions, additional domain knowledge, or more abstract features extracted from the environment.

In the context of value function approximation, the augmented state is used as input to the neural network, leading to an augmented Q-function $Q(s',a;\theta_q)$. The learning objective remains the minimization of the difference between the predicted Q-values and the target Q-values, now both conditioned on the augmented state:

$$L(\theta_q) = \mathbb{E}\left[\left(Q(s_{augment},a;\theta_q) - (r + \gamma \max_{a'} Q(s'_{\text{augment}},a';\theta_q^-))\right)^2\right]$$

Similarly, for policy function approximation, the augmented state feeds into the policy network, resulting in a policy $\pi(a|s';\theta_p)$ that is more informed and potentially capable of capturing dependencies that span across timesteps:

$$\nabla_{\theta_p} J(\theta_p) = \mathbb{E}\left[\nabla_{\theta_p} \log \pi(a|s_{augment};\theta_p) \cdot Q(s_{augment},a;\theta_q)\right]$$

State augmentation is thus a powerful tool in DRL, allowing the agent to make more informed decisions by considering additional contextual information and can be particularly beneficial in complex environments where the agent needs to remember past experiences or when the environment is only partially observed.

2.4. Multi-objective RL

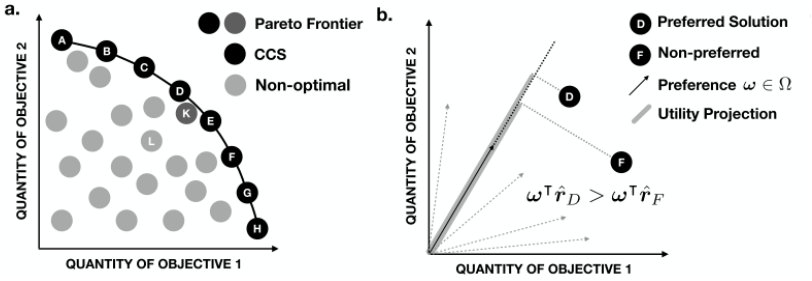

Figure 2.3: Competing Objectives [29]

Multi-Objective Reinforcement Learning (MORL) is how an agent learns policies of balancing multiple competing objectives. Unlike standard reinforcement learning with one reward signal, in MORL, the agent works with a reward vector. Each element of this vector represents a different goal the agent tries to achieve.

In MORL, at each step t, the agent receives a reward vector $\mathbf{r}_t$ with N different rewards, one for each objective:

$$\mathbf{r}_t = [r_t^1, r_t^2, \ldots, r_t^N]$$

The aim is to find a policy π that maximizes a multi-dimensional value function $\mathbf{V}^\pi(s)$, which reflects the expected total rewards for all objectives:

$$\mathbf{V}^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1} \mid S_t = s \right]$$

Incorporating the concept of a linear scalarization strategy (CSS), we assume a linear relationship between the objectives. This means we can combine the multiple objectives into a single one by giving each objective a weight. The combined reward at time t then becomes:

$$r_t^{\text{combined}} = \sum_{i=1}^{N} w_i r_t^i$$

where w_i is the weight for the i-th objective, reflecting its relative importance.

The optimal policy in a CSS-based MORL problem is the one that maximizes this combined reward. The policy is considered Pareto optimal if no other policy can improve any one objective without making at least one other objective worse. For policy π, it must hold that there is no other policy π' such that:

$$\mathbf{V}^{\pi'}(s) \geq \mathbf{V}^\pi(s)$$

with at least one strictly better objective.

MORL is useful for complex problems like managing finances or controlling robots, where you have to look at several things at once and make the best decision considering all of them.

2.5. Proximal Policy Optimization

Proximal Policy Optimization (PPO) [22] is an advanced on-policy gradient method used in reinforcement learning that addresses the inefficiencies of earlier algorithms like Trust Region Policy Optimization (TRPO) [23]. PPO seeks to take the biggest possible step to improve the policy while ensuring the

step is not so large that it causes performance to collapse and makes the training more stable.

The key idea behind PPO is to limit the policy update at each iteration, making sure that the new policy is not too far from the old policy. This is achieved by using a clipped objective function, which prevents large updates to the policy. The objective function that PPO maximizes is:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}\left[\mathsf{min}(r_t(\theta)\hat{A}_t, \mathsf{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

where $r_t(\theta)$ is the probability ratio of the new policy to the old policy, $\hat{A}_t$ is an estimator of the advantage function at time t, and ϵ is a hyperparameter that defines the clipping range.

PPO uses Generalized Advantage Estimation (GAE) for calculating the advantage function, which provides a balance between bias and variance by mixing n-step returns with value function baselines. GAE is defined as:

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V$$

where $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the temporal difference error, γ is the discount factor, λ is the GAE parameter that interpolates between using many steps (high variance, low bias) and one step (low variance, high bias), and $V(s)$ is the value function for state s.

By optimizing this clipped objective and using Generalized Advantage Estimation (GAE) for stable advantage estimation, PPO leads to more stable and sample-efficient learning compared to previous approaches.

2.6. Non-Stationary Environments in MORL

Non-stationary environments [14, 15] in Reinforcement Learning (RL) are those in which the underlying dynamics, which include the element of the environments, transition probabilities, and reward functions, change over time. These environments pose a significant challenge for RL agents because the policies and value functions that were optimal at one point in time may become sub-optimal or even worse as the environment dynamic changes.

A common factor that can lead to non-stationary in MORL is the change in preferences over the different objectives. In many real-world scenarios, the importance or preference given to each objective is not fixed; it may change due to human requirements, external factors, or some domain knowledge of the decision-making process.

For instance, consider an agent balancing between two objectives, such as speed and keeping quiet. The relative importance of these objectives might shift, favoring keeping quiet over speed in school/hospital areas but prioritizing speed in highways. These shifting preferences can be formalized as a time-dependent weighting vector:

$$\mathbf{w}(t) = [w_1(t), w_2(t), \ldots, w_N(t)]$$

where $w_i(t)$ denotes the weight of the i-th objective at time t. The combined reward function then becomes a function of time:

$$r_t^{\mathsf{combined}}(s,a,t) = \sum_{i=1}^{N} w_i(t) r_t^i(s,a)$$

This temporal variability in the reward structure means that an agent must constantly adapt its policy to remain aligned with the changing preferences. The challenge is increased that the agent may not be aware of the timing or nature of these shifts, requiring the agent to detect changes and learn within this non-stationary context actively. However, this project assumes that the preference is given without the need for detection.

3

Related Work

In this chapter, we describe the field of related and recent research how an RL agent can efficiently adapt from one task to another. We then focus further on the problem of multiple preferences in Multi-objective RL, and the corresponding adaptation from one preference to another.

3.1. General RL Adaptation between tasks

In the field of RL, the most well-known and popular approach solving the adaptation from one task to another task (while each task has its own environment dynamics including both state information and reward function), is meta-learning, and especially the methods of Gradient-Based Meta RL, most build upon the idea from Model-Agnostic Meta-Learning (MAML) [5], that enables an agent to rapidly adapt to new tasks with minimal training. This meta-learning technique involves training a meta-policy in such a way that it can be quickly fine-tuned with a small number of gradient steps for a variety of tasks. Specifically, MAML optimizes the initial parameters of the meta-policy using gradient descent, ensuring that the policy is positioned in the parameter space where it can effectively adapt to new tasks with only a few gradient updates. Building upon the idea of MAML in reinforcement learning, Nagabandi et al. [13] present a meta-reinforcement learning approach specifically designed for dynamic and real-world environments. Their method utilizes model-based algorithms, enabling efficient learning and quick adaptation in environments with changing dynamics, thus highlighting the real-world applicability of meta-RL. Meanwhile, Al-Shedivat et al. [24] explore continuous adaptation in non-stationary and adversarial environments through meta-learning. Their research is notable for its focus on adaptability in environments characterized not only by change but also by competition with other opponents or agents, employing a model-agnostic methodology. Both approaches complement the MAML framework by demonstrating the versatility of meta-learning strategies in a range of challenging reinforcement learning scenarios, from dynamic physical environments to competitive and evolving settings.

In addition to gradient-based meta RL, the second well-known approach is context-based meta RL. Rakelly et al. [19] introduced an efficient off-policy meta-RL framework that utilizes probabilistic context variables. This framework leverages these context variables to adaptively adjust the policy in response to the dynamics of newly encountered environments. The context variables are inferred from the current environment by accumulating collected experiences, using a Variational Autoencoder (VAE). Actions are determined based on both the state information and these inferred context variables, which are specific to the current environmental dynamics. After exploring various tasks, each with its own stationary environmental dynamic, the framework learns a meta-policy with a set of initial policy parameters conditional on the context variables. This meta-policy can quickly adapt to new tasks based on the inferred task-specific context variables. This approach differs from MAML, which uses gradient information to update the meta-policy parameters; instead, it updates the meta-policy using the inferred variables. Building upon this context-based meta RL idea, Luo et al. [10] further advanced the concept by developing a context-sensitive policy that adapts to sudden environmental changes. Their work emphasizes the practical application of context awareness in dynamic settings, showcasing the evolution

and applicability of context-dependent strategies in modern RL challenges.

Moreover, RNN-based meta-RL represents another influential approach in the field. A notable example is RL^2 : Fast Reinforcement Learning via Slow Reinforcement Learning, presented by Duan et al. (2016) [4]. This innovative method encapsulates the learning process within the framework of a Recurrent Neural Network (RNN). Known as RL^2, this approach enables an agent to learn the Markov Decision Process (MDP) itself as part of the learning process. This means it treats the dynamics of each environment as a distinct MDP. The corresponding information for each MDP is learned by the RNN layers, allowing the agent to adapt to the specific environment dynamics based on the latent information in the underlying RNN. The system achieves fast adaptation to new tasks, demonstrating significant improvements in learning efficiency and adaptability across various RL tasks.

3.2. Adaptation specifically on preference over multi-objective RL Approach

While the aforementioned research focuses on adaptation involving shifts in environmental dynamics, these shifts encompass various possible elements that differ from one task to another. This includes not only changes in the reward function but also shifts in other objective distributions within the environmental dynamics. Therefore, having well-defined domain knowledge beforehand is infeasible, and it is more practical to have a meta-policy that can quickly adapt from one environmental scenario to another. Specifically, for adaptation in multi-objective scenarios with different preferences, the different environmental scenarios are well-defined, characterized by varying preferences over different objectives, with different weights assigned to each. Consequently, learning a policy that can approximate the optimal policy for any given preference becomes feasible. The research specifically for adaptation on MORL has leveraged the power of Universal value function approximators [21], to achieve that policy is conditional on both state and also the preference, with a Q-value vector for different preference settings, enabling the agent can have a single policy network that can represent different preference given it as additional input. This idea is similar as taking the preference as a specific context, resulting in a Contextual markov decision processes [7], each action is based on the state along with the context. Abels et al. [1] introduce a dynamic preference-conditional Q-network in an online learning manner. This method learns policies across various encountered preferences and maintains a diverse buffer, with each buffer storing experiences corresponding to different sets of preferences in the preference space. The algorithm updates based on the encountered preferences and also actively samples experiences it has previously encountered, selecting those experiences in the diverse buffer that are close to the current preference. Yang et al. [29] further developed a generalized approach based on preference Q-value. That can approximate the optimal policy for any preference within the preference space. This is achieved by sampling all preferences during training iterations. Additionally, they introduce the 'Envelope Update' mechanism, which involves sampling experiences with different preferences to update the policy. This enables the agent to quickly adapt to new preferences by utilizing experiences gathered under various other preferences. Similar to Yang et al. [29], Buet-Golfouse and Pahwa [2] developed a robust algorithm for handling any given preferences. They overcame the problem of high-dimensionality and computational complexity by using actions derived from the scalarized Q-network during both learning and evaluation phases. This methodology results in the algorithm outperforming its competitors with fewer iterations.

4

Method: Generalized Preference-Based PPO

In this chapter, I will describe the details of the proposed method: GPB PPO. This includes state augmentation, how the preference is concatenated with state information, the neural network design, which takes both state and preference information as input, preference sampling during training, and how the policy is evaluated and updated. Finally, I will discuss the training procedure.

4.1. State Augmentation with processed preference information

The design of how to incorporate additional preference information into the RL algorithm is a key element in developing the Generalized Preference-Based PPO. This additional information should be distinct enough to be well recognized during training without significantly increasing computational complexity, and it should also facilitate effective learning. The design of how state information is augmented by addition preference, the detail can be seen in appendix A.3

To enhance the signal of the additional preference information, the preference is processed in a form similar to the state information and then concatenated together as an augmented state, serving as the input for the neural network. More specifically, in our experiment, the state information consists of multiple grid layers, with each layer's size corresponding to the grid environment and representing different objectives distributed within the environment. One grid layer of the state information represents the agent's position, indicated by a floating value of 1.0 in the corresponding entry, while the other entries remain at 0.0 in this layer. Similarly, a value of 1.0 indicates the existence, and 0.0 indicates the non-existence of the corresponding object in a specific layer; there are additional layers representing the distributed positions of different objectives. For a two-objective problem with the preference [0.7, 0.3], we add two grid layers: one layer with all entries set to a value of 0.7, and another with all entries set to 0.3. These two layers are concatenated with the original state information, resulting in new state information with two additional layers for the two-objective preference information. If it is a three-objective problem, then we similarly add three layers representing the preference information, with each layer assigned the corresponding weight for the respective objective in the preference vector.

The motivation behind processing the preference in this way and augmenting it with the state information is that it can provide a strong signal that is well recognized during training, thereby making the learning of a conditional policy possible in the first place.

4.2. Neural Network Architecture.

The design of the Neural Network Architecture is crucial as it needs to handle the additional preference information which is stated in the figure 4.1. The choice of CNN is due to our experiment being based

in a grid-world environment, where multiple layers represent different objects within the environment. Convolutional neural networks (CNNs) are specialized in handling this kind of spatial information. The number of CNN layers depends on the environment's complexity, followed by a Fully Connected (FC) layer that compresses the processed state information into a lower-dimensional format. Additionally, the network accepts the augmented state information along with the processed preference information. The preference vector is directly concatenated with the processed state information from the FC layer, serving as the input for two separate networks: the actor and the critic network. The CNNs act as a common shared network to process the state information. Both the actor and the critic network take the combined input of processed state information and preference vector. Each of these two separate networks consists of two FC layers. The output of the actor network is the action probability in the action space, while the output of the critic network is distinct; it outputs an n-dimensional value, with each value corresponding to each objective.

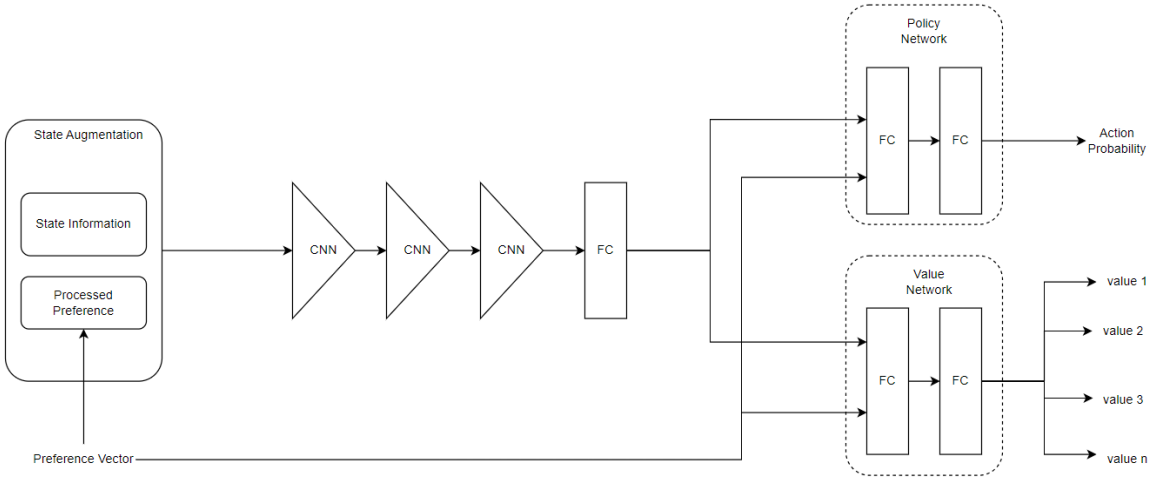

Figure 4.1: Network Architecture

4.3. Preference Sampling with Parallel Environment

The strategy of preference sampling directly affects how well GPB PPO can generalize to any given preference, so it must be carefully designed, the detail of it can be viewed in the flowchart figure 4.2. The advantage of the parallel training capability of the PPO algorithm is well utilized in my training process. We sample preferences after collecting a certain number of episodes, instead of using fixed preferences throughout the entire training process. This approach creates a parallel environment training method where the number of preferences sampled during training matches the number of created parallel environments, with each preference assigned to a corresponding environment. Agents interact within each environment using the assigned preference, collecting experiences that are stored in a central reply buffer, thus enabling the buffer to accumulate experiences under diverse preferences. Furthermore, the frequency of preference sampling is set to every 4 episodes, meaning that after collecting 4 episodes from each environment, a new set of preferences is sampled and assigned to the environments. This ensures the buffer contains experiences with regularly updated preferences, allowing the method to explore as many preferences as possible and the neural network to keep updating the policy conditioned on diverse preference parameters. Additionally, preferences are sampled from a uniform distribution as floating-value scaled, n-dimensional vectors for an n-objective problem, with the weights assigned to each objective summing up to 1 in a preference vector.

4.4. Policy Evaluation in GPB PPO

The evaluation and policy update is similar to standard PPO, but with a little bit different. The transition stored in the replay buffer is a tuple (state, action, vector rewarad, preference) instead of (state, action,

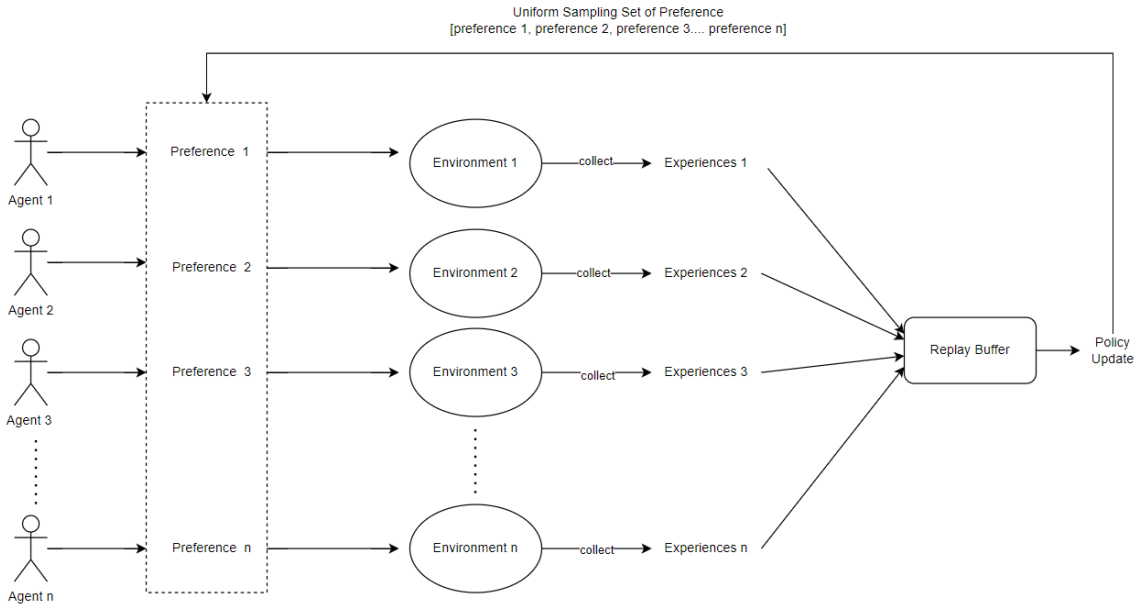

Figure 4.2: Preference Sampling And Parallel Environment

scalarized reward) as it in common approach in MORL. After a number of episodes collected, the policy update is conducted. First the scalarized reward for each time step is calculated as vector reward * preference, and then being normalized. And then one episode is evaluated as how standard PPO did, but the estimated value

Policy evaluation is a critical step in reinforcement learning where we assess how good a policy is at achieving the set objectives. In our multi-objective setup, each policy action results in a vector of rewards corresponding to different objectives. To evaluate the policy effectively, we combine these rewards with a preference vector, which gives us the flexibility to prioritize certain objectives over others during training.

In our experiments, we calculate the combined reward for each objective by taking the dot product of the reward vector and the preference weights. For example, if we have three objectives, the combined reward R_t at time t is calculated as follows:

$$R_t = \sum_{i=1}^{n} r_{t,i} \cdot w_i$$

where $r_{t,i}$ is the reward for objective i at time t, and w_i is the weight assigned to that objective. Similarly, the value network outputs a vector of values, and we compute the weighted sum of these values to estimate the state-value function V as:

$$V(s_t) = \sum_{i=1}^{n} V_i(s_t) \cdot w_i$$

where $V_i(s_t)$ is the value estimate for objective i at state s_t.

During policy evaluation, we use Generalized Advantage Estimation (GAE) to calculate the advantage function A, which helps us understand how much better it is to take a specific action compared to the average. The advantage is computed using the value function estimates and the rewards. It is then normalized to ensure stable training.

We also employ a clipping mechanism to ensure that the policy updates are not too large, which could destabilize the learning process. The clipped loss is defined as the minimum between the product of the likelihood ratio and the advantage, and a clipped version of this product using an epsilon parameter ϵ:

$$L^{CLIP}(\theta) = -\min(\rho_t(\theta) \cdot A_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon) \cdot A_t)$$

where $\rho_t(\theta)$ is the likelihood ratio of the new policy to the old policy, and A_t is the advantage at time t.

The value loss is then computed as the mean squared error between the estimated and the target values:

$$L^{VALUE}(\theta_v) = \frac{1}{T}\sum_{t=1}^{T}(V_{target}(s_t) - V(s_t))^2$$

where $V_{target}(s_t)$ is the target value for state s_t.

Through this evaluation process, we update our policy by optimizing the clipped losses and value losses, ensuring that our agent's policy can adapt to the preferences and achieve a balance between the different objectives.

4.5. GPB PPO algorithm training procedure

The details of the training procedure in the GPB PPO algorithm are stated in Algorithm 1. First, the network parameters are initialized orthogonally [18], and N number of parallel environments are created, along with the corresponding sampling function. This function samples N preferences (equal to the number of parallel environments) under a uniform distribution with a floating-point scale. Additionally, the experience buffer is initialized as empty. During every episode, there is a set of N sampled preferences and corresponding N parallel environments. For each environment env_i, a preference n_i is assigned. The assigned preference is processed into the proper form to augment the state information, resulting in an augmented state. In every step, for each environment, the agent interacts with the environment based on the augmented state and the preference (the preference is augmented with the state, and also serves as extra input for both the actor and critic network). Then, the tuples $(s_t, a_t, s_{t+1}, \mathbf{r}_t, w_n)$ from all environments are stored into the central experience buffer. After collecting a certain number of episodes, the policy is evaluated based on the experiences in the buffer under diverse preferences. For each episode, the weighted sum return and the weighted sum advantage values are computed. The critic value is computed using the n-dimensional values output from the critic network and the corresponding preference. Losses are computed, and then the policy is updated. Subsequently, the buffer is cleared, and a new set of N preferences is sampled.

Algorithm 1 Generalized Preference-Based PPO

Orthogonally initialize actor and critic networks
Initialize N parallel environments
Initialize replay buffer $\mathcal{B} \leftarrow \varnothing$
Initialize a preference sampling distribution $\mathcal{D}_w$
for each episode $i = 1, 2, \ldots, M$ **do**
    Sample N linear preferences $w \sim \mathcal{D}_w$
    Assign preference w_n to environment env_n for $n = 1, 2, \ldots, N$
    for time step $t = 0, 1, 2, \ldots, T$ **do**
        Augment state $s_{aug_t} \leftarrow$ concatenate$(s_t, w_{processed_n})$
        Execute action a_t according to policy $\pi(a_t | s_{aug_t}, w_n)$
        Observe the next state s_{t+1} and vectorized reward $\mathbf{r}_t$
        Store $(s_t, a_t, s_{t+1}, \mathbf{r}_t, w_n)$ in $\mathcal{B}$
    end for
    if an update is due **then**
        for all episodes in $\mathcal{B}$ **do**
            Compute the batch loss
            Compute the value loss
            Perform gradient descent to update policies
        end for
        Clear the buffer $\mathcal{B} \leftarrow \varnothing$
    end if
end for

5 Experiment

In the experimental chapter, I have designed two experiments to assess the perspective of generalization, performance and adaptation of GPB PPO. These experiments are all conducted in a grid world environment, characterized by discrete state and action spaces, the environment setting is similar to [16]. These experiments shared similarities in terms of environment design, and evaluation procedures.

5.1. Environment Description

The environments of both experiments are based on a grid world environment, the second one is a bit more complex than the first one. For both environments, the goal of the agent is to collect as many items as possible to maximize the return within the maximal time step, collecting different types of items representing different objectives, and the trade-off depending on the given preference. The agent has the following action space as shown in 5.1

$$A = \{\text{move up}, \text{move down}, \text{move left}, \text{move right}, \\ \text{collect-up}, \text{collect-down}, \text{collect-left}, \text{collect-right}\} \tag{5.1}$$

For Experiment 1, the grid world size is 8 * 8, there are two objectives: Objective 1 is to collect as many boxes as possible, while Objective 2 is to collect as many water bottles as possible. At the beginning of an episode, there are 6 boxes and 6 water bottles, respectively. The termination condition is reached at 25 time steps. After every episode, the environment is reset with a random distribution of the agent and all the boxes and water bottles. The environment is shown in Figure 5.1.

For Experiment 2, the grid world size is 10 * 10, there are three objectives: Objectives 1 and 2 are the same as in Experiment 1, and the additional Objective 3 is to collect as many pieces of wood as possible. At the beginning of an episode, the number of each of these items is 6. The termination condition is reached at 40 time steps. After every episode, the environment is reset with a random distribution of the agent, all the boxes, water bottles, and wood. The environment is shown in Figure 5.2

The comparison of two experiment environments is list in table 5.1

5.2. Evaluation Metrics

5.2.1. Generalization

The first metric to be evaluated is the generalization of GPB PPO. This measures whether it can perform well with any given preference and make proper trade-offs between competing objectives. Thus, we generate a set of continuous preferences in the preference space and evaluate the solution by GPB PPO on each of them. We visualize all of these solutions in a multi-dimensional space, where each dimension represents the reward for each objective. This assesses whether it can generalize well across

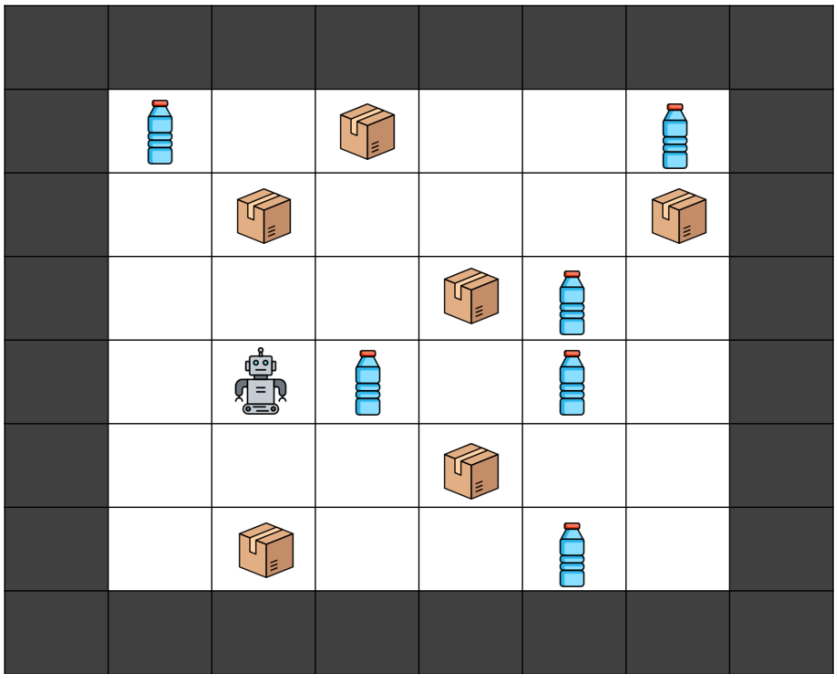

Figure 5.1: Environment of Experiment 1

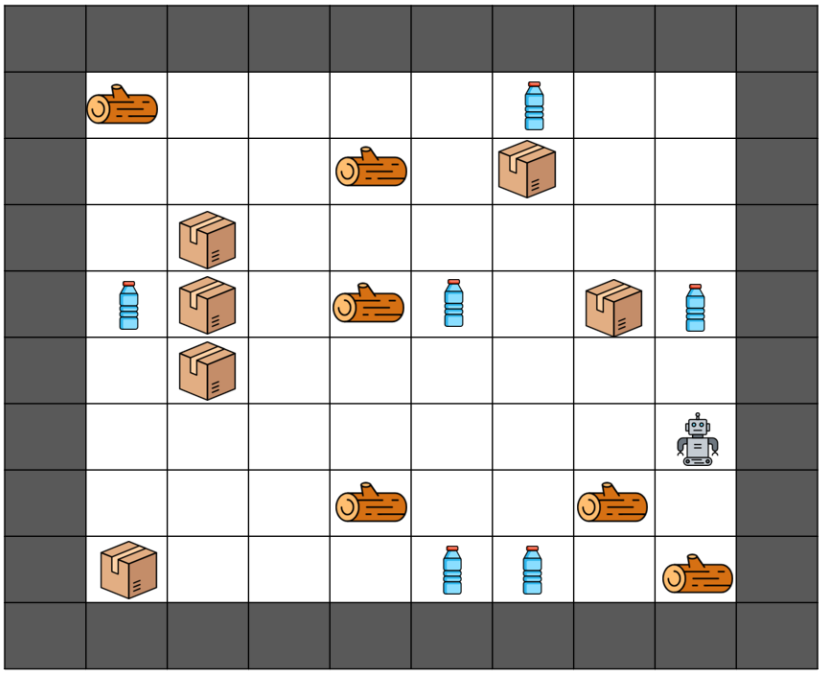

Figure 5.2: Environment of Experiment 2

different trade-offs among objectives, forming a Pareto front with linear preferences.

5.2.2. Performance Comparing to Standard PPO

The second metric evaluates the performance of GPB PPO compared to Standard PPO. Unlike Standard PPO, GPB PPO can adapt to align with any preference by directly taking this preference as input. In contrast, training all policies for all preferences using Standard PPO is infeasible. Therefore, we select a set of preferences and train policies based on these selected preferences using Standard PPO.

Environment similarities and differences		
Environment Elements	Environment 1	Environment 2
Number of objectives	2	3
Number of items for each objective	6	6
Grid World Environment Size	8x8	10x10
Terminal Condition	25 time steps	40 time steps
Number of actions	8	8
Randomly distributed (Environment Reset)	Agent and all items	Agent and all items

Table 5.1: Comparison of two different environments

We then compare the performance of GPB PPO and Standard PPO on these preferences in terms of returns and standard deviations.

5.2.3. Preference Sampling Granularity Influence

The granularity of preference sampling in GPB PPO plays a crucial role, as it affects the training from beginning to end. To evaluate the impact of sampling granularity across various preferences, policies were trained with different preference sample granularities. Subsequently, the return on various preferences generated by these policies was evaluated.

5.2.4. Performance in time-dependent preference changing condition

To further test the adaptability of GPB-PPO in scenarios where preferences change, we investigated its performance in situations where preferences vary within a single episode and change at certain time steps. The cumulative reward for each objective and the return corresponding to the change in time step were evaluated. This was also compared to the performance of Standard PPO.

5.2.5. Performance in region-dependent preference changing condition

In addition to evaluating time-dependent preference changes, another scenario involving region-dependent preference change conditions is also evaluated. The grid environment is divided into several different regions, each favoring a specific objective. The agent can make advance adjustments for the upcoming preferences to enable smoother adaptation. The performance metrics recorded include the number of transitions between different regions, the total objective reward, the return, and region-specific objective rewards, as well as the different item collection ratios corresponding to different objectives in each region. This evaluation, from a different perspective, aims to assess behavior based on adaptation in decision-making that considers upcoming preferences.

5.3. Evaluation Procedure

Both experiments follow the same evaluation procedure to assess GPB PPO using the metrics mentioned in 5.2. And this section will go through the evaluation procedure of both experiment on each of the metrics one by one.

5.3.1. Procedure to evaluate generalization

GPB PPO policy is trained under a floating-point scale of preference sampling for both experiments.

For Experiment 1, we evaluated the GPB PPO policy on a set of continuous preferences in a two-dimensional preference space, using minimal steps of 0.01. This resulted in a total of 101 preferences. Examples include [1,0], [0.99,0.01], [0.98,0.02], [0.97,0.03], ..., [0.02,0.98], [0.01,0.99], [0,1]. We ran the GPB PPO trained model on each of these preferences for 100 episodes, and the corresponding rewards for objectives were recorded. These were then plotted in a 2-D space to visualize how the model trades off between two objectives across these preferences.

For Experiment 2, the set of continuous preferences in a three-dimensional preference space consisted of a total of 5131 preferences, with minimal steps of 0.01. Examples include [1,0,0], [0.99,0.01,0], [0.98,0.01,0.01], [0.97,0.02,0.01], ..., [0,0,1]. We ran the GPB PPO trained model on each of these preferences for 100 episodes as well, recording the corresponding rewards for objectives. These results were plotted in three 2-D spaces, each using two of the three dimensions as the x-axis and y-axis. Additionally, we projected these results in a 3-D space.

5.3.2. Procesure to evaluate performance comparison between GPB PPO and Standard PPO

The GPB PPO policies is trained under a floating-point scale of preference sampling for both experiments. To compare its performance to Standard PPO, a set of preferences is selected for each experiment based on two criteria: first, the degree of clarity with which one objective is prioritized over others, and second, different granularities at 0.1, 0.05, and 0.01. Following these criteria:

For Experiment 1, the preferences selected are [1,0], [0.9, 0.1], [0.7, 0.3], [0.53, 0.47], [0.5, 0.5], [0.45, 0.55], [0.33, 0.67], [0.2, 0.8], [0.15, 0.85], and [0,1].

For Experiment 2, the preferences selected are [1,0,0], [0.0,0.25,0.75], [0.1,0.2,0.7], [0.66,0.32,0.02], [0.04,0.92,0.04], [0.5,0.0,0.5], [0.25,0.5,0.25], and [0.33,0.33,0.34].

Then, for each experiment, we ran the corresponding GPB PPO trained model on the respective set of preferences, each for 1000 episodes, and recorded the return and standard deviation.

5.3.3. Procedure to evaluate the preference sampling granularity

For each of the experiments, three policies with three different preference sampling granularity (floating point scale, 0.05, 0.01) were trained.

For Experiment 1, each of the three trained GPB PPO models was evaluated on a set of continuous preferences within a two-dimensional preference space, using a minimal step of 0.01, resulting in a total of 101 preferences. Each preference was tested with 100 episodes, with the return and standard deviation being recorded.

For Experiment 2, each of the three trained GPB PPO models was evaluated on a set of continuous preferences within a three-dimensional preference space, also using a minimal step of 0.01, which resulted in a total of 5131 preferences. Each preference was again tested with 100 episodes, and the return and standard deviation were recorded.

5.3.4. Procedure to evaluate the performance under time-dependent preference change

To evaluate performance during changes in preferences over time, the preference is altered every certain number of time steps.

For Experiment 1, three different preference changing frequencies were conducted: every 15 time steps, every 10 time steps, and every 6 time steps. This resulted in the experienced preferences within an episode being 2, 3, and 5, respectively. The maximum time step of an episode in this test is 30, which is 5 steps more than the 25 used during training.

In Experiment 2, three different preference-changing frequencies were tested: every 20 time steps, every 10 time steps, and every 8 time steps. This resulted in the number of experienced preferences within an episode being 2, 4, and 5, respectively. The maximal time step in this test is 40, which is the same as it was during training.

For each experiment, the corresponding cumulative objective reward and return, along with the changes over time, were recorded. The details of preference change frequency and corresponding total number of experienced preference is in table 5.2

Time-depedent preference changing		
	Preference Changing Frequency	Number of preference experienced
Experiment 1	every 15/10/6 time step	2/3/5 preferences
Experiment 2	every 20/10/8 time step	2/4/5 preferences

Table 5.2: Time-dependent preference changing of two experiment

5.3.5. Procedure to evaluate the performance in region-dependent preference changing condition

In Experiment 1, the environment is divided into two regions, the green region and the blue region (refer to Figure 5.3). The green region favors Objective 1: collecting as many boxes as possible, while the blue region favors Objective 2: collecting as many water bottles as possible. The agent can take the upcoming preference information of the corresponding region into its current decision-making process by estimating the upcoming trajectory and determining which region it will move through. This is achieved by using the learned policy for rollout, which utilizes the current policy to execute the trajectory.

More specifically, I have established four different conditions to account for upcoming preference information by looking N steps ahead using rollouts guided by the current policy. These conditions are look-aheads of 1, 5, 10, and 15 steps. For example, in the 1-step look-ahead scenario, the agent decides based on the preference at its current position. In the 5-step scenario, the agent takes into account the region information for the next 5 steps.

For details on how preferences are determined, refer to Figure 5.4. The agent rolls out 5 trajectories and analyses the region information within these trajectories to determine the appropriate preference for the current step, and make the corresponding preference-conditional action.

Initially, the preference is set to [0.5, 0.5]. The agent then uses the policy based on this preference at the current step to rollout a number of trajectories, analyze the upcoming region information, and estimate the new preference, and the old preference is updated by this estimated preference, and then the agent takes action based on this new preference. This process guides the interaction with the environment until the maximum of 30 steps is reached.

For Experiment 2, the environment is divided into three regions: the green, blue, and yellow regions (refer to Figure 5.5). The green and blue regions favor the same objectives as in Experiment 1, but with the addition of the yellow region, which favors Objective 3: collecting as much wood as possible.

four different conditions to account for upcoming preference information by looking N steps ahead using rollouts guided by the current policy. These conditions are look-ahead of 1, 5, 10, and 15 steps, same as experiment 1.

Initially, the preference is set to [0.33, 0.33, 0.34], since all objectives are considered equally important. The process then follows the same steps as in Experiment 1 to estimate the new preference, update the preference, and take action based on the updated preference. The agent continues to interact with the environment until a maximum of 40 steps is reached. The only difference is in how to estimate the preference since the number of objectives and the number of regions are now three, but the estimation procedure remains the same. For details, see Figure 5.6.

Lastly, for both experiments, testing is conducted by running the trained PPO model for 200 episodes in the respective regional versions of the environment.

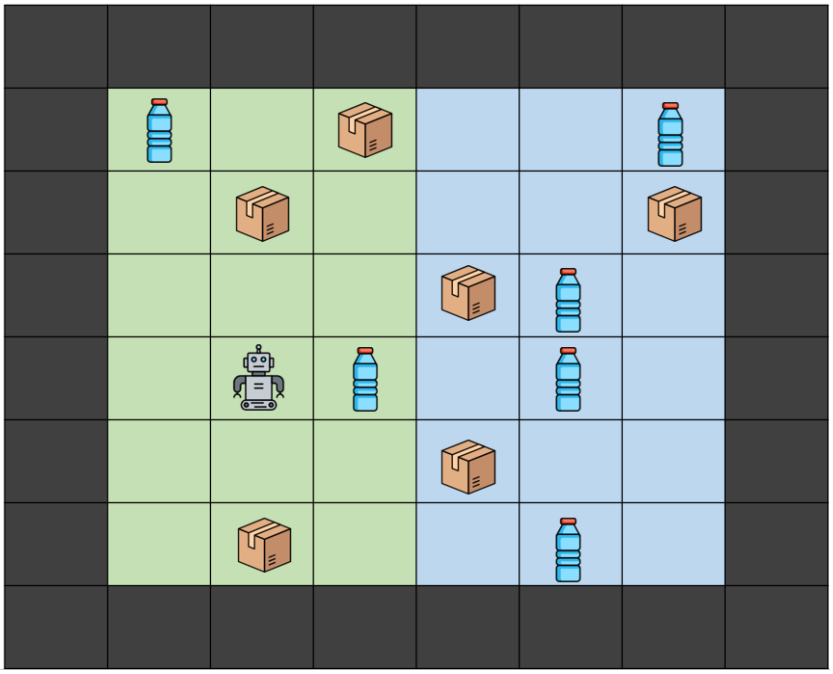

Figure 5.3: Experiment 1: Two regions

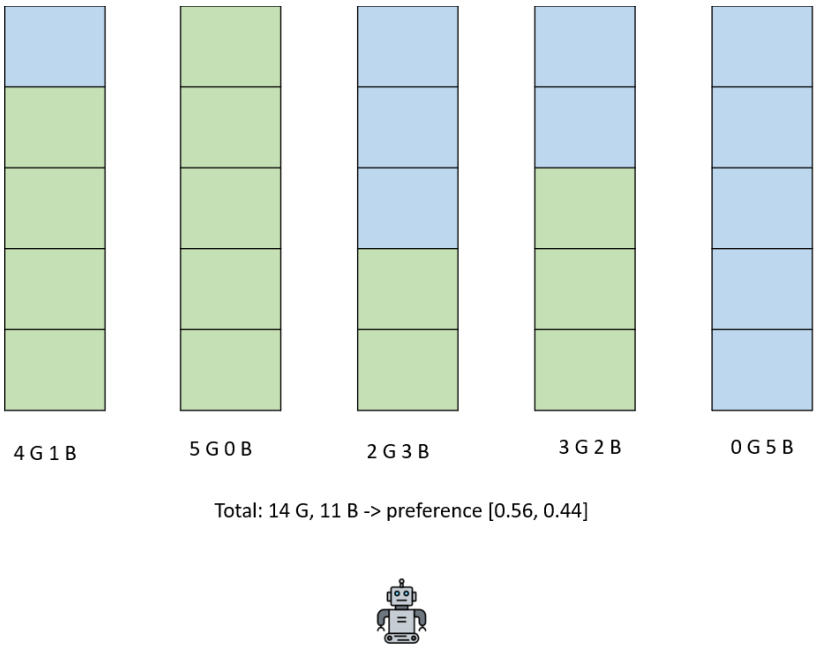

Figure 5.4: Experiment 1 rollout example

5.4. Experiment 1: Result Analysis

In this section, the results corresponding to all the evaluation metrics (see Section 5.2) of Experiment 1 will be described and analyzed.

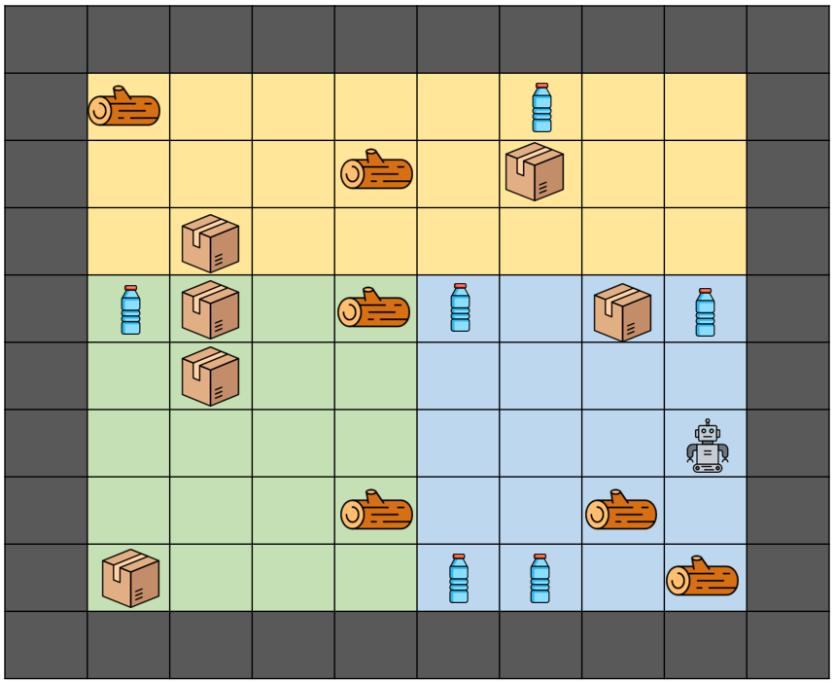

Figure 5.5: Experiment 2: Three regions

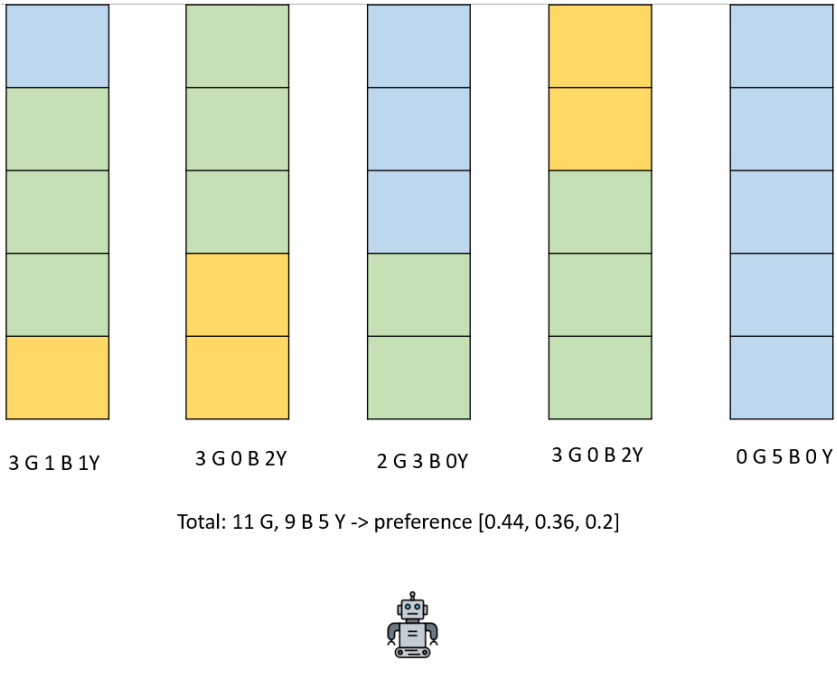

Figure 5.6: Experiment 2: rollout example

5.4.1. Result: Solution coverage

Figure 5.7 shows the solution distribution for a total of 101 preferences in a 2-D visualization. These solution points form a front, with dense points gathering at the trade-off point where the two objectives have similar importance. Moreover, the reward for objective 2 decreases as the reward for objective 1 increases, and vice versa. These points indicate a set of non-dominated, optimal solutions where improvements in one objective can only be achieved at the cost of another.

These results reflect that GPB PPO learns a set of solutions for a given set of possible linear pref-

erences, forming a Pareto front. This indicates that GPB PPO can indeed generalize well across any preferences in the preference space.

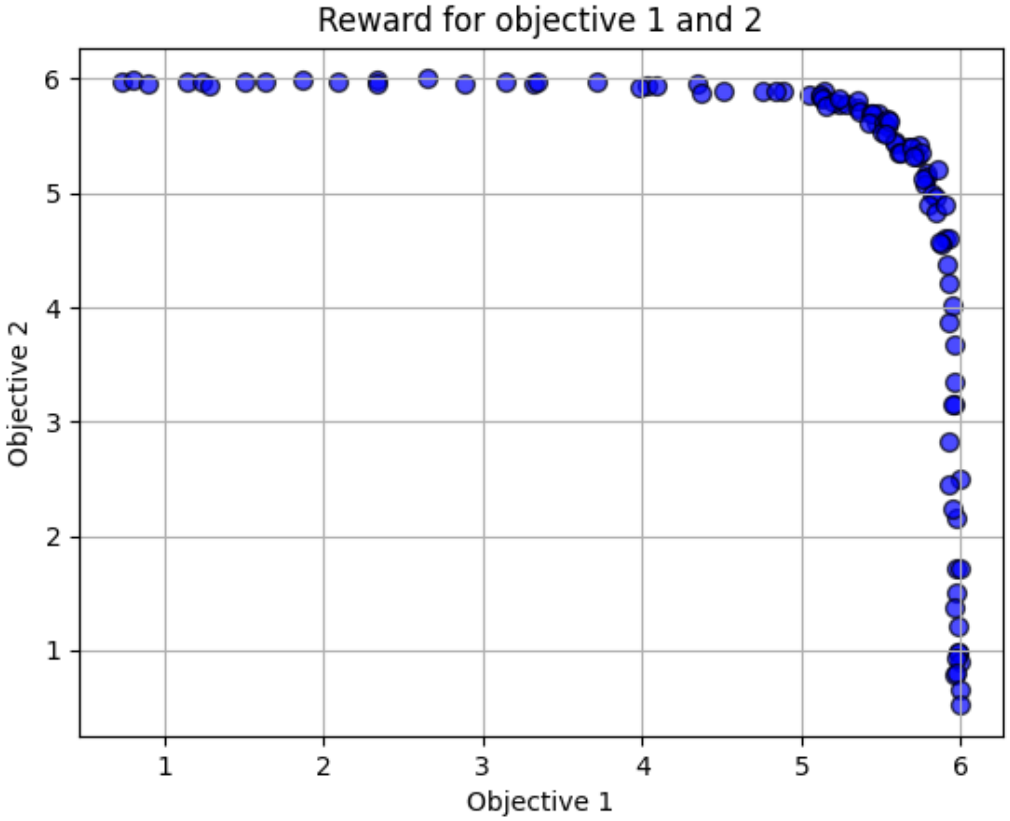

Figure 5.7: Experiment 1: Objective Reward Distribution in the Pareto Front

5.4.2. Result: Performance Comparing to Standard PPO

The comparative analysis depicted in figure 5.8 and table 5.3 compare the performance of return and the standard deviation of GPB PPO model and standard PPO model on a set of selected preferences.

The first observation is that for preferences where one objective is more clearly prioritized over another, the standard deviation is relatively lower for both models. This is typical, as seen in the standard deviations of [1,0], [0,1], and [0.9,0.1], which are relatively low. Additionally, the standard deviation of Standard PPO on preferences with a clearer prioritization of one objective is slightly lower than that of GPB PPO. However, this differs for preferences where both objectives are similar in importance, such as [0.45,0.55], [0.53,0.47], and [0.5, 0.5], that the standard deviation of standard PPO is slightly higher than GPB PPO.

The second observation is that generally, the average return of Standard PPO is slightly higher than that of GPB PPO. However, for preferences like [0.53, 0.47] and [0.5, 0.5], where both objectives are similar in importance, the average return of GPB PPO is slightly higher than that of Standard PPO.

5.4.3. Result: Preference Sampling Granularity Influence

Figure 5.9 presents a comparison of three GPB PPO models with three different preference sampling granularities (floating value scale, 0.05, 0.1). The blue line depicts the performance of the model trained

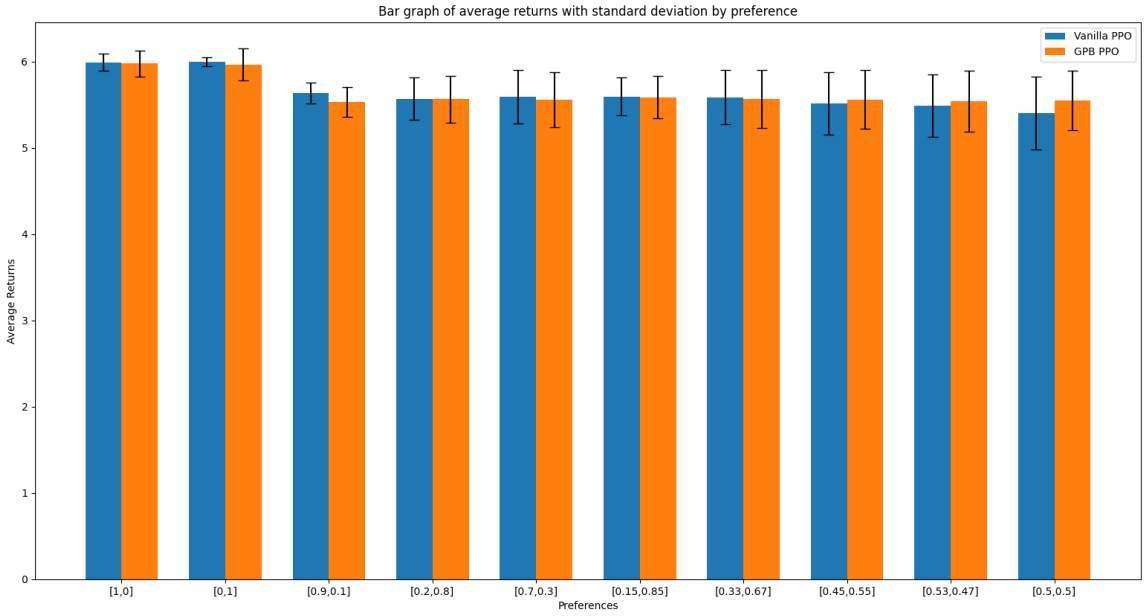

Figure 5.8: Experiment 1: Return Mean Comparison On Specific Preference (Bar)

Preferences	Baseline PPO	GPB PPO
[1,0]	5.99 (±0.1)	5.98 (±0.15)
[0,1]	6 (±0.05)	5.97(±0.18)
[0.9,0.1]	5.63 (±0.12)	5.53 (±0.17)
[0.2,0.8]	5.57 (±0.25)	5.56 (±0.27)
[0.7,0.3]	5.59 (±0.31)	5.56 (±0.31)
[0.15,0.85]	5.6 (±0.21)	5.59 (±0.25)
[0.33,0.67]	5.59 (±0.31)	5.57 (±0.34)
[0.45,0.55]	5.51 (±0.36)	5.56 (±0.34)
[0.53,0.47]	5.49 (±0.36)	5.54 (±0.35)
[0.5,0.5]	5.4 (±0.42)	5.55 (±0.35)

Table 5.3: Experiment 1: Return Mean Comparison On Specific Preference (Table)

under uniform sampling at a floating value scale, the yellow line indicates the performance of the model trained with 0.05 granularity, and the green line denotes the performance of the model trained with 0.1 granularity; the shaded areas represent their respective standard deviations.

As can be observed, the three GPB PPO models exhibit similar performance across this range of preferences. While some models perform slightly better than others in specific preference intervals, the differences between them are not immediately apparent.

More detailed information is shown in Tables 5.4 and 5.5, which describe the performance of three models on a total of 101 preferences, as well as the pairwise comparisons between each pair of models. It can be seen that the model sampled with a floating-point scale significantly outperforms the model sampled with 0.05 granularity, but it does not significantly outperform the model sampled with 0.1 granularity. Additionally, the model sampled with 0.1 granularity significantly outperforms the model sampled with 0.05 granularity.

Figure 5.9: Experiment 1: GPB PPO models under three different sampling granularity. return mean across total 101 preference with minimal 0.01 step

GPB PPO Models	N	Mean	Std Deviation	Minimal	Maximum
Model 1	101	5.60	0.12	5.39	6.00
Model 2	101	5.58	0.12	5.37	6.00
Model 3	101	5.61	0.12	5.42	6.00

Table 5.4: Experiment 1: Performance Descriptives of three models on 101 preferences. Model 1:floating point scale, Model 2: 0.05 granularity, Model 3: 0.1 granularity

5.4.4. Result: Performance on Time-Dependent Preference Changing Environment

The figures 5.10 detail the cumulative rewards of objectives over time for both GPB PPO and standard PPO. The green solid line represents the reward for objective 1 of GPB PPO, and the green dotted line represents the reward for objective 1 of standard PPO. Meanwhile, the orange solid line represents the reward for objective 2 of GPB PPO, and the orange dotted line represents the reward for objective 2 of standard PPO. Each row of subplots in the figure represents a different change frequency of preferences. The obvious observation is that both policies follow very similar trends in the growth of cumulative rewards for objectives, in line with the assigned preferences in the corresponding time intervals.

Figure 5.11 shows the cumulative return over time for both GPB PPO and standard PPO. The green solid line represents the cumulative return of GPB PPO, and the orange dotted line represents the cumulative return of standard PPO. Each row of subplots also indicates three different preference change frequencies. The same observation is noted: they exhibit similar growth trends, and these two lines are closely aligned with the assigned preference in the corresponding time intervals.

5.4.5. Result: Performance on Region-Dependent Preference Changing Environment

From Tables 5.6, 5.7, 5.8, and 5.9, we can see that four different rollout lengths conditions were used to estimate the current preference upon which the agent can base its action. The most straightforward observation is that, along with the increase in rollout length, there is a resulting finer-grained preference. This comes with a greater total number of different preferences being estimated, which would lead to

Pairwise Comparisons	Mean Difference	Std. Error	95% Confidence Interval	Significance (p-value)
Model 1 and Model 2	0.0242	0.0052	[0.0139,0.0345]	< 0.001
Model 1 and Model 3	-0.0073	0.0052	[-0.0178,0.0028]	0.155
Model 2 and Model 3	-0.0315	0.0052	[-0.0420,-0.0214]	< 0.001

Table 5.5: Experiment 1: Pairwise Comparisons. Model 1:floating point scale, Model 2: 0.05 granularity, Model 3: 0.1 granularity

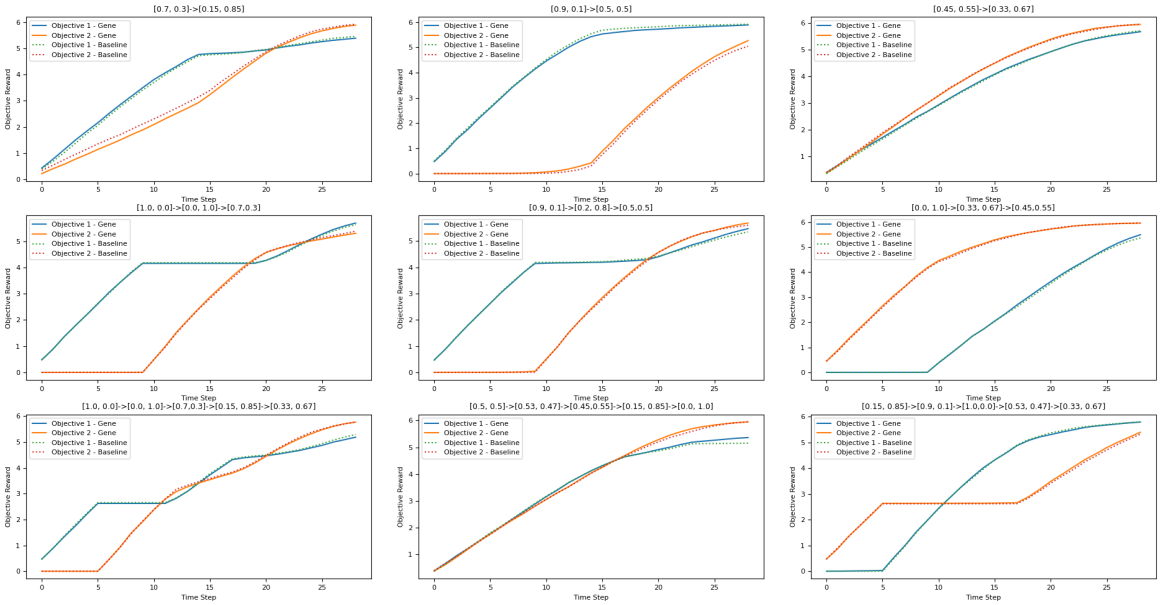

Figure 5.10: Experiment 1: Time-Dependent Preference Change: Preference Changes Every 15/10/6 Time Steps, Comparison Between GPB PPO and Baseline PPO on Objective Rewards (Each row represents different preference change frequency, with three different preference sequences)

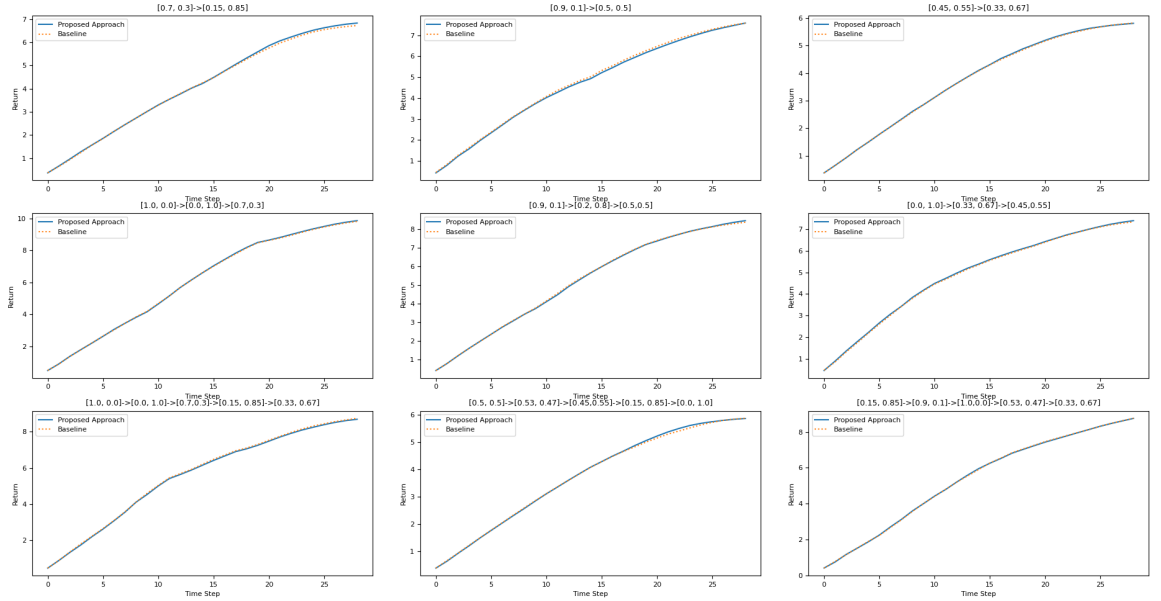

Figure 5.11: Experiment 1: Time-Dependent Preference Change: Preference Changes Every 15/10/6 Time Steps, Comparison Between GPB PPO and Baseline PPO on return (Each row represents different preference change frequency, with three different preference sequences)

a smoother policy adaptation and a reduction in the number of transitions between different regions.

Once the agent takes more steps ahead into its decision-making and adjusts its preferences more frequently, it reduces the conflicts between the previous policy and the newly adapted policy, thus achieving a better overall outcome. For example, even in the green region which favors the first objective with a [1,0] preference, a smoother adaptation occasionally leads the agent to also take the [0,1] preference, which is favored in the blue region, into consideration for upcoming steps. Nevertheless, it still obtains a higher reward for Objective 1 as more steps are taken into consideration.

On the other hand, when more steps are considered in the lookahead, more rewards are collected for Objective 1, even in the blue region, which favors Objective 2, and vice versa. Another observation is that with more steps looked ahead and smoother adaptation, the standard deviations of objective rewards, region-specific rewards, and returns all decrease.

	Obj 1	Obj 2
Objective Reward	3.3 ($\pm$1.63)	2.7 ($\pm$1.92)
Return	5.83 ($\pm$2.70)	
Green	2.64 ($\pm$1.44)	0.11 ($\pm$0.46)
Green Ratio	82.79 %	3.52%
Blue	0.08 ($\pm$0.38)	2.1 ($\pm$1.64)
Blue Ratio	2.83%	63.93%
Number of Transition	14.55 ($\pm$7.73)	
Number of Preferences	2	

Table 5.6: Experiment 1: 1-steps look ahead

	Obj 1	Obj 2
Objective Reward	4.28 ($\pm$1.66)	4.16 ($\pm$1.67)
Return	6.09 ($\pm$2.28)	
Green	2.65 ($\pm$1.34)	1.19 ($\pm$1.09)
Green Ratio	85.19 %	42.12%
Blue	1.13 ($\pm$1.09)	2.54 ($\pm$1.51)
Blue Ratio	39.75%	77.31%
Number of Transition	6.13 ($\pm$2.28)	
Number of Preferences	26	

Table 5.7: Experiment 1: 5-steps look ahead

	Obj 1	Obj 2
Objective Reward	5.1 ($\pm$1.22)	4.74 ($\pm$1.53)
Return	6.29 ($\pm$1.77)	
Green	2.70 ($\pm$1.27)	1.82 ($\pm$1.13)
Green Ratio	86.48 %	63.31%
Blue	1.85 ($\pm$1.23)	2.51 ($\pm$1.40)
Blue Ratio	64.67%	79.65%
Number of Transition	3.63 ($\pm$4.13)	
Number of Preferences	51	

Table 5.8: Experiment 1: 10-steps look ahead

	Obj 1	Obj 2
Objective Reward	5.56 ($\pm$0.68)	5.46 ($\pm$0.80)
Return	6.50 ($\pm$1.10)	
Green	2.71 ($\pm$1.18)	2.50 ($\pm$1.31)
Green Ratio	88.1 %	84.42%
Blue	2.44 ($\pm$1.22)	2.51 ($\pm$1.23)
Blue Ratio	83.04%	82.19%
Number of Transition	2.67 ($\pm$1.67)	
Number of Preferences	76	

Table 5.9: Experiment 1: 15-steps look ahead

5.5. Experiment 2: Result Analysis

In this section, the results corresponding to all the evaluation metrics (see Section 5.2) of Experiment 2 will be described and analyzed.

5.5.1. Result: Solution Coverage

Figures 5.12 and 5.13 offer a visual exploration of the Pareto front as derived from a three-objective optimization experiment using the GPB PPO, with total 5131 points while each associated with one preference in the preferences space with minimal 0.01 step.

In Figure 5.12. The x-axis and y-axis represent the objective reward for two of three objectives while the variation in brightness of the color represents the change in the third objective reward.

The observation is that the distribution of data points reveals a distinct front in the two-dimensional projections. Along the boundary where the colors change, there appears to be a trade-off between the objectives represented by the x-axis and y-axis. As one increases, the other decreases, forming a Pareto front for these two objectives. This is most evident in the first subplot of the figure.

And as the reward for these two objectives represented by the x and y axes increases, the color becomes darker. The extreme case is observed in the top-right area for all subplots, where, as the rewards for both objectives increase, the reward for the third objective decreases, as indicated by the darkening color.

Figure 5.13 provides a clearer view of the trade-offs in a landscape with three objectives, presenting the Pareto front in a three-dimensional objective space. Here, the color transition from dark to light represents the varying rewards of the third objective. The observation in this spatial reveals that the rewards for objectives 1 or 2 are relatively low at the lightest point. Conversely, in the area with the darkest points, the rewards for objectives 1 and 2 are relatively high. Moreover, the distribution of data points clusters in a manner that narrows toward the edges, indicating a grouping of optimal solutions. This pattern illustrates the concept of trade-offs in multi-objective optimization: improving the outcome for one objective typically requires a compromise in others.

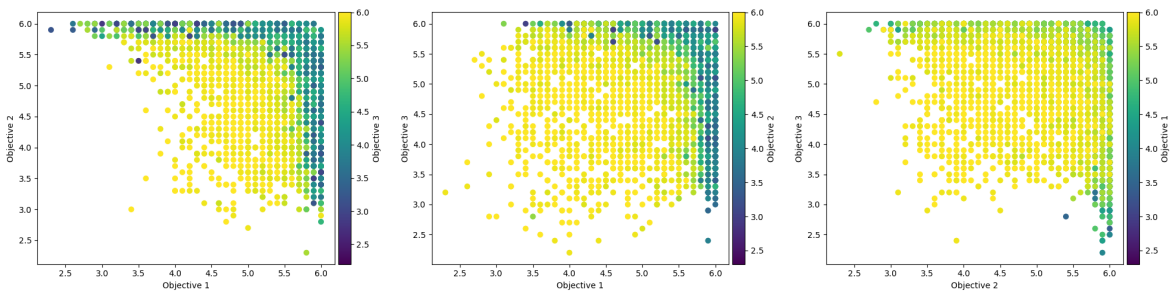

Figure 5.12: Experiment 2: Three objectives Pareto-front 2D visualization

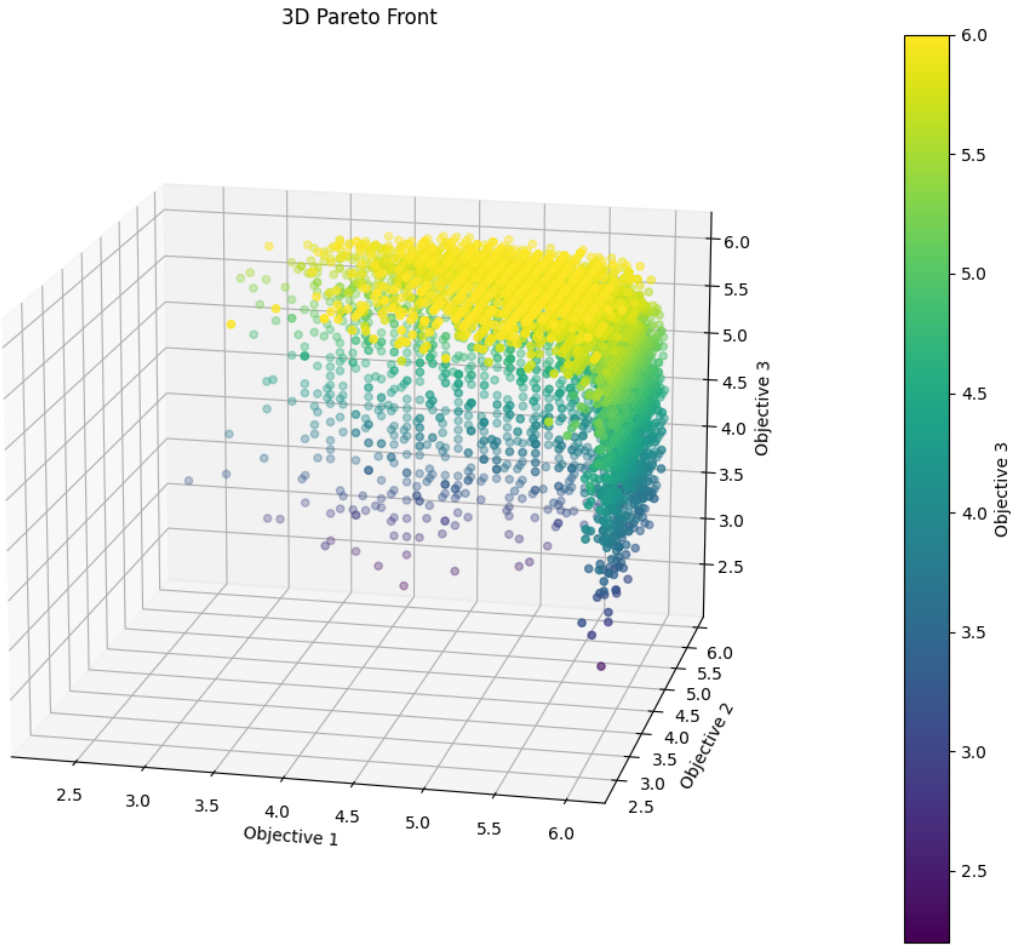

Figure 5.13: Experiment 2: Three objectives Pareto-front 3D visualization

5.5.2. Result: Performance Comparing to Standard PPO

The figures 5.14 and 5.10 provide a comparative analysis of average returns between the GPB PPO and the baseline PPO across a set of preferences in a three-objective scenario.

Compared to the observations in Experiment 1, the performance of GPB PPO is more comparable to that of Standard PPO. GPB PPO performs slightly better on three of the eight selected preferences than Standard PPO, but for the remaining preferences, Standard PPO still outperforms GPB PPO by a small margin. Moreover, the observations here also follow a similar pattern to this metric in Experiment 1. For those preferences with a clearer priority over objectives, such as [1,0,0] and [0.04,0.92,0.04], the standard deviation is relatively lower for both GPB PPO and Standard PPO. In addition, the standard deviation for these preferences is relatively lower for Standard PPO than for GPB PPO.

5.5.3. Result: Preference Sampling Granularity Influence

This figure 5.15 displays the comparative performance of three GPB PPO models, each trained using different sampling granularities. The x-axis represents the index of a total of 5131 preferences with minimal steps of 0.01, and the y-axis represents the average return. The blue line represents the model trained with floating-point scale sampling granularity, followed by the yellow line for the model trained with 0.05 granularity, and the green line for the model trained with 0.1 granularity.

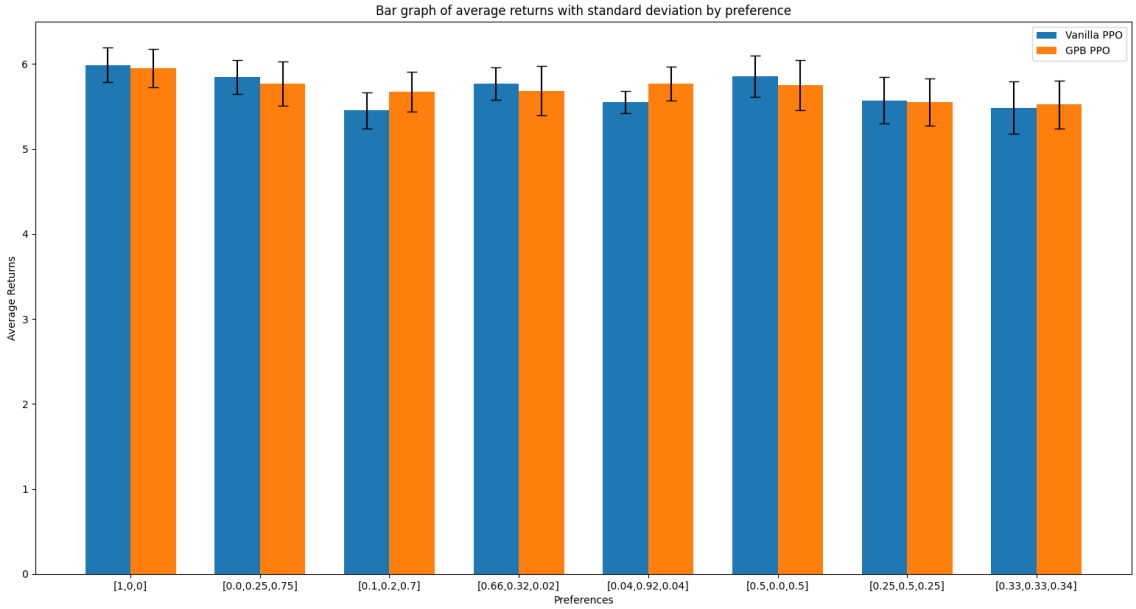

Figure 5.14: Experiment 2: Comparison between GPB PPO and stand PPO set of preferences (Bar)

Preferences	Baseline PPO	GPB PPO
[1,0,0]	5.99 (±0.02)	5.95 (±0.22)
[0,0.25,0.75]	5.85 (±0.2)	5.77 (±0.26)
[0.1,0.2,0.7]	5.45 (±0.21)	5.67 (±0.23)
[0.66,0.32,0.02]	5.77 (±0.19)	5.68 (±0.29)
[0.04,0.92,0.04]	5.55 (±0.13)	5.77 (±0.2)
[0.5,0,0.5]	5.85 (±0.24)	5.75 (±0.30)
[0.25,0.5,0.25]	5.57 (±0.27)	5.55 (±0.28)
[0.33,0.33,0.34]	5.48 (±0.3)	5.52 (±0.28)

Table 5.10: Experiment 2: Comparison between GPB PPO and stand PPO set of preferences (Table)

Based on observation, in most of the preference space, the GPB PPO model trained with a floating-point granularity performs better than the other two models, as indicated by the blue line consistently being above the other two lines. Furthermore, in a small segment of the preference space, the GPB PPO model trained with 0.05 granularity slightly outperforms the model trained with 0.1 granularity. This is evidenced by the yellow line being slightly higher than the green line around the index around 1000 and also at the most right side of the x-axis, where the yellow line is again marginally above the green line.

The tables 5.11 and 5.12 provide more detailed information about the performance of these three models, as well as pairwise comparisons between them. They indicate that there are significant differences; the GPB PPO model with floating point scale granularity significantly outperforms the other two across the total 5131 preferences. Furthermore, the model with 0.05 granularity significantly outperforms the model with 0.1 granularity on these 5131 preferences.

5.5.4. Result: Performance In Time-preferences Changing Environment

Figure 5.16 displays the change of cumulated objectives rewards for three objectives over time, as evaluated by both GPB PPO and Baseline PPO. The solid lines represent the performance of GPB PPO, whereas the dotted lines indicate the performance of Baseline PPO. The sub-plots are organized by rows, each representing different frequencies of preference changes: every 20, 10, and 8 time steps, respectively.

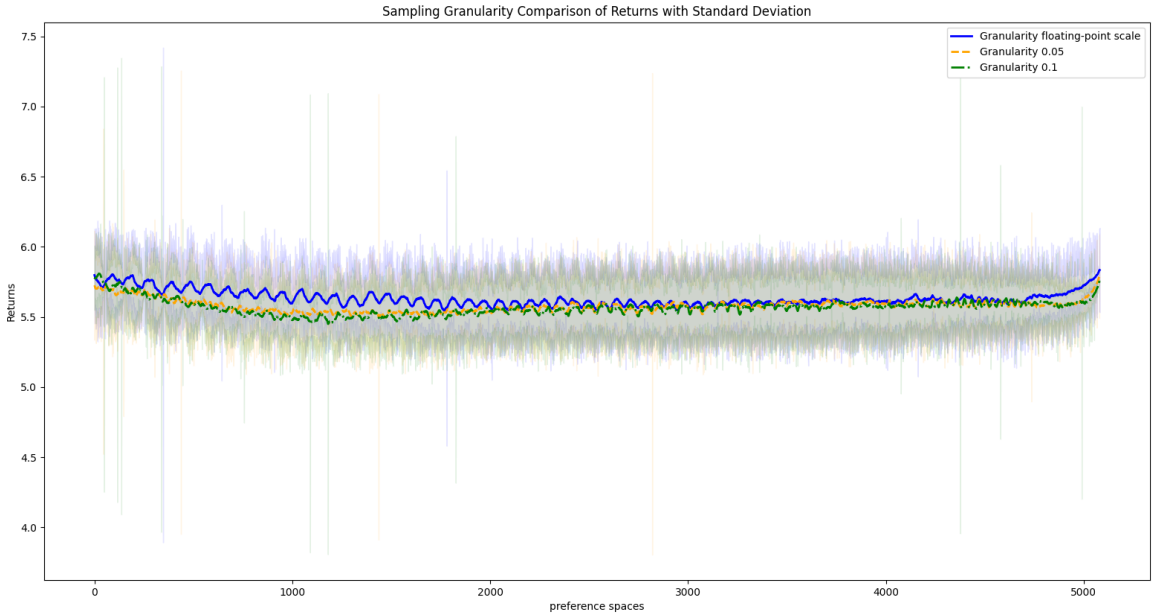

Figure 5.15: Experiment 2: GPB PPO models under three different sampling granularities, evaluation on Return Mean Across total 5131 Preference Space With minimal 0.01

GPB PPO Models	N	Mean	Std Deviation	Minimal	Maximum
Model 1	5131	5.63	0.11	5.25	6.00
Model 2	5131	5.58	0.11	4.90	6.00
Model 3	5131	5.57	0.12	4.87	6.00

Table 5.11: Experiment 2: Performance Descriptives of three models on 5131 preferences. Model 1:floating point scale, Model 2: 0.05 granularity, Model 3: 0.1 granularity

It is observable that when a specific objective is given clear priority under a specific preference within the corresponding time interval, both GPB PPO and Baseline PPO exhibit nearly identical trend patterns, especially for those preferences that clearly signal prioritizing one objective over others, it means that both Standard PPO and GPB PPO agree on the policy of maximizing that particular objective with clear priority. An example can be seen in the leftmost subplot in the first row of Figure 5.16, in the first 20 time steps, where the preference is [1,0,0], the cumulative reward for objective 1 for both models — represented by the solid blue line and the dotted blue line — follow similar growth trends. However, the dotted blue line is slightly higher than the solid blue line, indicating that Standard PPO performs marginally better than GPB PPO under the preference [1,0,0], meanwhile, the cumulative rewards for objectives 2 and 3 diverge for both models, as there is no consideration given to earning rewards for objectives 2 and 3 under this preference. However, once the preference changes in the second time interval, from 20 to 40 time steps, with the new preference [0.25, 0.5, 0.25], the second objective (represented by yellow lines) becomes slightly more prioritized than the others. Both the solid and dotted yellow lines follow an increasing trend, indicating that both Standard PPO and GPB PPO models agree on prioritizing the second objective. Nonetheless, the slope of the Standard PPO is greater than that of GPB PPO, suggesting that the performance of Standard PPO is slightly higher than GPB PPO for this preference.

The same interpretation applies to the other sub-plot as well: when one objective is more clearly prioritized than the others, the cumulative reward for that objective in both models follows a similar growth trend. Generally, the performance of Standard PPO is slightly higher than that of GPB PPO, as indicated by the dotted line being above the solid line for that objective. This also implies that both models have a similar policy on how to earn rewards for the clearly prioritized objective. In contrast, under the

Pairwise Comparisons	Mean Difference	Std. Error	95% Confidence Interval	Significance (p-value)
Model 1 and Model 2	0.0528	0.002	[0.0483,0.0527]	< 0.001
Model 1 and Model 3	0.0647	0.002	[0.0602,0.0692]	< 0.001
Model 2 and Model 3	0.0120	0.002	[0.007,0.0164]	< 0.001

Table 5.12: Experiment 2: Pairwise Comparisons. Model 1:floating point scale, Model 2: 0.05 granularity, Model 3: 0.1 granularity

condition where there is a highly prioritized objective, both models exhibit quite different understandings of the policy needed to earn rewards for the non-prioritized objectives. The lower the priority of an objective, the more the policies of both models diverge in earning rewards for them. This is indicated in any subplot, where the lines corresponding to the lower-priority objectives are more divergent.

Furthermore, an interesting observation is where there is no relatively highly prioritized objective compared to others. For example, in the third subplot of the second row, during the first time interval under the preference [0.33, 0.33, 0.34], all the cumulative rewards for all objectives by both models are very close together. This suggests that both models agree on all objectives being equally important in contributing to the returns. Then, once the preference changes to [0.1, 0.2, 0.7], where the third objective is relatively more prioritized than the first and second, the green lines follow very similar trends, with the dotted line slightly above the solid one. This indicates that both Standard PPO and GPB PPO models concur on the policy to earn as much reward for the third objective as possible, but Standard PPO performs a bit better. Regarding the remaining objectives, the policies of both models diverge since the first objective, with a weight of 0.1, and the second objective, with a weight of 0.2, are not clearly distinguishable in terms of priority. Thus, both models might not have a clear consensus on their actions to earn rewards for these two objectives

Figure 5.17. Despite the divergence in the pattern of objective rewards when objectives have similar weights but one is relatively more prioritized, the overall trend in cumulative returns for both GPB PPO and Baseline PPO exhibits a similar pattern. It is noteworthy that Baseline PPO generally achieves slightly higher returns than GPB PPO across all sub-plots, as indicated by the orange dotted line consistently being higher than the blue solid line to varying degrees, and this difference in performance accumulates over time.

However, this still reflects that GPB PPO performs well in adapting to a time-dependent, preference-changing environment, albeit slightly less effectively than the Standard PPO.

5.5.5. Result: Performance on Region-Dependent Preference Changing Environment

From Tables 5.13, 5.14, 5.15, and 5.16, we can see that four different rollout lengths conditions were used to estimate the current preference upon which the agent can perform a preference-conditional action.

The most straightforward observation is that, along with the increase in rollout length, there is a resulting finer-grained preference. This comes with a greater total number of different preferences being estimated during agent interaction with the environment, which would lead to a smoother policy adaptation and a reduction in the number of transitions between different regions. The number of transitions significantly dropped from 21.07, 8.15, 7.06 to around 5.93, there is less conflict of moving between different regions.

Another observation is that with a greater number of steps looked ahead, the total return does not necessarily continue to increase, but more objectives are collected across the entire environment. With fewer steps looked ahead, more region-specific rewards are collected, as well as the corresponding collection ratios. For example, in the 1-step look-ahead condition, 76.36 % of Objective 1 is collected

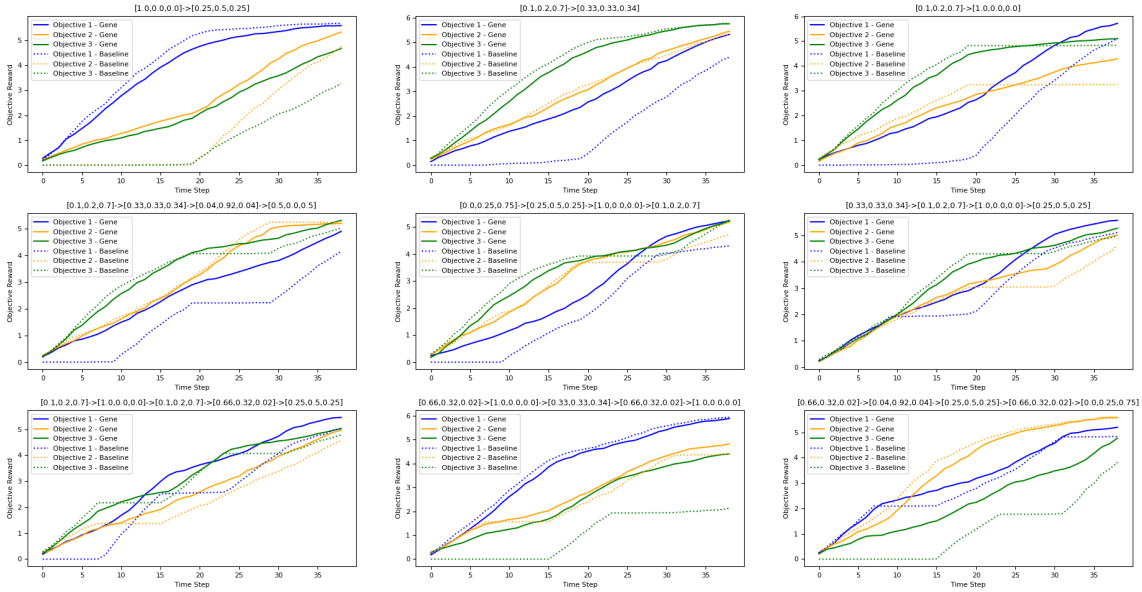

Figure 5.16: Experiment2: Cumulative reward for objectives with Time-Dependent Preference Change comparison between GPB PPO and standard PPO: (each row represents the preference change frequency, they are 20/10/8 time step respectively))

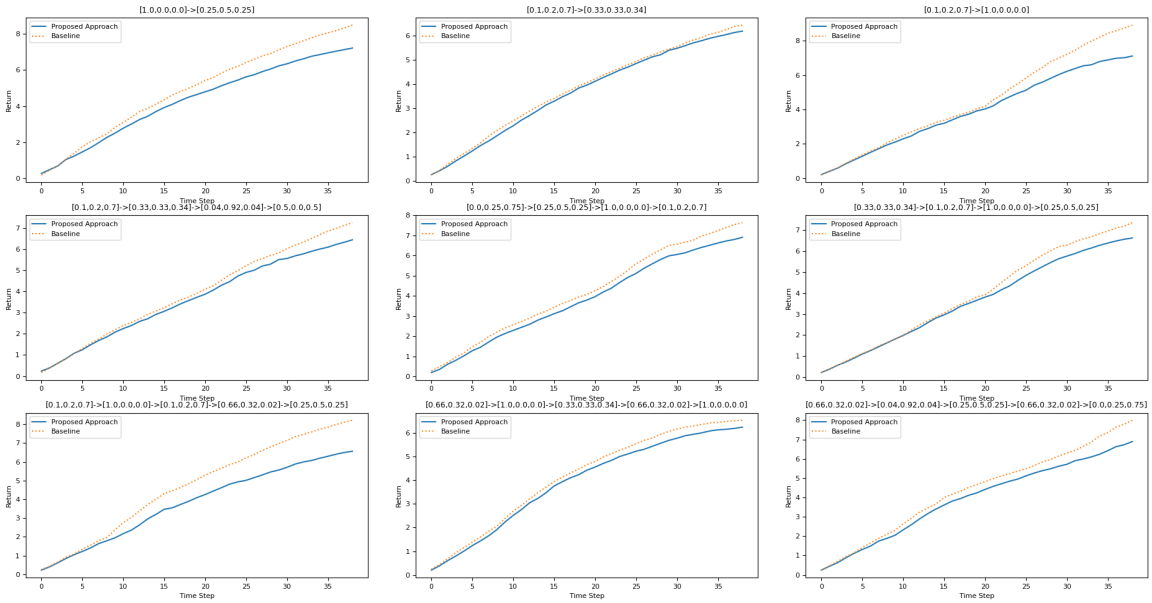

Figure 5.17: Experiment2: Cumulative return with Time-Dependent Preference Change comparison between GPB PPO and standard PPO: (each row represents the preference change frequency, they are 20/10/8 time step respectively

in the green region, 76.56 % of Objective 2 is collected in the blue region, and 73.71 % of Objective 3 is collected in the yellow region. The collection of remaining objectives in these regions is lower. However, with more steps taken into consideration, the collection ratio of each objective tends to become more uniform across each region.

	Obj 1	Obj 2	Obj 3
Objective Reward	3.75 (±1.69)	3.85 (±1.80)	3.72 (±1.76)
Return	5.79 (±2.85)		
Green	1.60 (±1.23)	0.89 (±0.87)	0.84 (±0.92)
Green Ratio	76.36%	45.92%	46.72%
Blue	0.64 (±0.86)	1.50 (±1.23)	0.62 (±0.83)
Blue Ratio	33.35%	76.56%	33.13%
Yellow	0.91 (±0.94)	0.91 (±0.91)	1.77 (±1.28)
Yellow Ratio	42.16%	42.58%	73.71%
Number of Transition	21.07 (±12.50)		
Number of Preferences	3		

Table 5.13: Experiment 2: 1-steps look ahead

	Obj 1	Obj 2	Obj 3
Objective Reward	4.47 (±1.31)	4.8 (±1.18)	4.48 (±1.32)
Return	6.32 (±2.19)		
Green	1.76 (±1.18)	1.27 (±1.09)	1.10 (±0.97)
Green Ratio	88.38%	68.20%	65.81%
Blue	0.95 (±0.95)	1.6 (±1.1)	0.85 (±0.93)
Blue Ratio	51.89%	86.62%	43.08%
Yellow	1.11 (±1.07)	1.32 (±1.07)	1.94 (±1.32)
Yellow Ratio	48.70%	56.30%	79.54%
Number of Transition	8.15 (±4.40)		
Number of Preferences	290		

Table 5.14: Experiment 2: 5-steps look ahead

	Obj 1	Obj 2	Obj 3
Objective Reward	4.75 (±1.29)	5.09 (±1.23)	4.89 (±1.33)
Return	6.06 (±1.76)		
Green	1.69 (±1.09)	1.55 (±1.10)	1.41 (±1.03)
Green Ratio	87.32%	76.45%	72.83%
Blue	1.09 (±1.07)	1.49 (±1.07)	1.12 (±0.96)
Blue Ratio	60.61%	80.77%	64.00%
Yellow	1.42 (±1.10)	1.45 (±1.05)	1.78 (±1.09)
Yellow Ratio	61.27%	64.38%	79.17%
Number of Transition	7.06 (±5.68)		
Number of Preferences	1070		

Table 5.15: Experiment 2: 10-steps look ahead

	Obj 1	Obj 2	Obj 3
Objective Reward	5.2 (±1.20)	5.32 (±1.24)	5.09 (±1.18)
Return	6.13 (±1.43)		
Green	1.58 (±1.06)	1.43 (±1.12)	1.43 (±1.12)
Green Ratio	82.66%	76.99%	71.59%
Blue	1.33 (±1.02)	1.69 (±1.10)	1.32 (±1.03)
Blue Ratio	73.40%	86.84%	74.11%
Yellow	1.72 (±1.17)	1.59 (±1.11)	1.76 (±1.19)
Yellow Ratio	72.50%	72.69%	77.19%
Number of Transition	5.93 (±3.54)		
Number of Preferences	2233		

Table 5.16: Experiment 2: 15-steps look ahead

6

Discussion and Conclusion

This chapter will discuss the results of the experiments presented in Chapter 5, answer the corresponding research questions, and conclude with an evaluation of our developed approach. Finally, future work will be mentioned.

6.1. Discussion

To answer **Research Question 1: How can preference information be incorporated into decision-making?** it's important to note that decision-making considers not only the state information but also extra domain knowledge or requirements for how to trade off the objectives, specified by assigning preference. The underlying idea of this project aligns with the research in [29, 2, 1], where the algorithm's network takes both state and preference information as input. This concept is similar to Contextual Markov Decision Processes [7], where the agent's actions are based on the state but also consider the current context, which in this case is the preference. A slight difference is that the method in this project does not rely heavily on a preference-based Q-value function to provide the extra preference signal, as [29, 2, 1] did. Instead, it relies on augmenting the state information with the processed preference.

To answer **Research Question 2: how such a preference-based approach can generalize across the preference space**, thus enabling the agent to have the ability to generalize well under different preference conditions, the idea is to expose the agent to as diverse a range of preferences as possible during the training procedure. There are also similarities with [29, 2, 1] in terms of sampling the preferences during training. However, the difference in my approach lies in utilizing the characteristic parallel environment training in PPO, which enables continuous sampling of a set of preferences during training iterations. By assigning different preferences to different environments, the policy is updated through diverse experiences under various preferences. Continually sampling preferences allows the agent to explore as many preferences as possible. In addition to the number of different experiences encountered during training, applying gradient descent with different preference settings on common state information might help the agent understand the relationship between different preferences, thereby improving its ability to generalize.

To answer Research Question 3: How does such an approach perform on a given preference? This project conducted two experiments. It is observed that GPB PPO exhibits comparable performance on a given preference when contrasted with Standard PPO. Although for most of the selected preferences, GPB PPO tends to slightly underperform in comparison to Standard PPO. This could be due to a potential trade-off between generalizing across all preferences within the preference space and optimizing performance for a specific preference. The possible reason is that the algorithm's network has an additional number of parameters related to diverse preference information, which introduces more complexity into policy learning. The increased number of sampled preferences during training may introduce instability, as opposed to Standard PPO's fixed reward signal throughout the entire training procedure, which offers more stability. Nonetheless, GPB PPO still maintains quite good and comparable performance to Standard PPO on specific given preferences, especially considering its

well-generalized capability.

To answer **How this approach can trade-off in a dynamic preference-changing non-stationary scenario**: the setting of this paper posits that dynamic preference changes only occur during the testing phase, as opposed to the training phase like in [1], where preference changes during training. However, similar to [29, 2], training results in a single policy that can adapt to any given preference without an extra adaptation phase. Therefore, through experiments on time-dependent preference changes in testing, and results by altering the preference within an episode, GPB PPO has achieved a cumulative return comparable to Standard PPO, both exhibiting a similar increasing trend in the time-dependent preference-changing scenario. More specifically, in terms of how to trade off between different objectives, GPB PPO and Standard PPO have similar strategies for earning rewards for the relatively higher prioritized objectives. However, for the remaining objectives with comparatively lower priority, GPB PPO and Standard PPO have quite different approaches to trade-offs. In summary, this GPB PPO approach has made a well-balanced trade-off among objectives, achieving a cumulative return that is comparable to Standard PPO according to the changing preferences over time.

From the experiment involving region-dependent preference changes during testing, the agent adjusts its preferences and makes trade-offs by using the trained policy with the current preference to rollout a set of trajectories. It analyzes the upcoming situation and then estimates the new preference, facilitating a smooth policy adaptation with a more fine-grained set of preferences. As can be seen during evaluation, the number of transitions between different regions significantly decreases, implying that there is less conflict between policies. This allows the agent to take not only the current situation but also the upcoming situation into account, preparing it to align with the upcoming preferences. This setting is suitable for real-world scenarios, such as an autonomous agent moving from a highway to a school or hospital area. As it moves toward a school or hospital, the agent gradually adjusts its behavior to prepare for alignment with the preferences specific to a school or hospital, thereby avoiding potential risks caused by drastic policy adaptations. The property of GPB PPO facilitates this setting of more intensive and frequent preference changes since GPB PPO does not require extra adaptation and can immediately act on new preferences by having the preferences as input.

6.2. Conclusion

In this project, the aim is to develop a method that allows the agent to align with multiple different experts' preferences without retraining when there is a need to adapt to a new preference. The proposed approach, GPB PPO, incorporates preference information into sequential decision-making, effectively enabling the algorithm to learn a preference-conditioned policy. This results in a single policy capable of representing diverse preferences, as opposed to multiple distinct policies corresponding to each possible preference. GPB PPO has demonstrated the ability to generalize in adapting to preferences, offering performance comparable to that of standard PPO. This holds true across multiple different preference conditions and in environments with changing preference dynamics, while also supporting more frequent and intensive adaptations.

7

Future Work

Even though GPB PPO has demonstrated good generalization across preferences and performed well, there are still some limitations. The performance may be slightly affected by the diversity of preferences during training, and the parameter space in the neural network of this approach remains large. This is due to the additional preference information being incorporated into the state information, which increases the computational complexity. This issue might be addressed by mapping the augmented state into a low-dimensional, abstract state representation. Furthermore, the generalization of this approach is limited to different preferences over objectives, assuming that other elements in the environment remain unchanged during training. However, this is not always the case in more challenging real-world situations. There might be other elements in the environmental dynamics, in addition to preference variations, that introduce more variability into the policy. An online adaptation mechanism would be more flexible in dealing with these unexpected environmental dynamics shifts, not just preference shifts. Since these differences are not easily defined or covered during training, unlike the preferences. GPB PPO could serve as a foundation upon which to build an online adaptation mechanism. The knowledge from GPB PPO about how to balance different objectives can be effectively utilized, thereby reducing the difficulty of adaptation in an online manner. This would allow the online adaptation to focus more on fine-tuning the policy for shifts in other environmental dynamics. The last limitation to be addressed in future work is that preferences are not always specified explicitly in new situations. Therefore, it would be beneficial to develop a framework that incorporates both GPB PPO, online adaptation, and includes an additional component for detecting environmental changes. This would enable effective inference of underlying preferences, especially in continuous tasks where preferences and other environmental dynamics change continuously, in more challenging scenarios closer to real-world settings.

References

[1] Axel Abels et al. “Dynamic weights in multi-objective deep reinforcement learning”. In: *International conference on machine learning*. PMLR. 2019, pp. 11–20.

[2] Francois Buet-Golfouse and Parth Pahwa. “Robust Multi-Objective Reinforcement Learning with Dynamic Preferences”. In: *Asian Conference on Machine Learning*. PMLR. 2023, pp. 96–111.

[3] Luciano Cavalcante Siebert et al. “Meaningful human control: Actionable properties for AI system development”. In: *AI and Ethics* 3.1 (2023), pp. 241–255.

[4] Yan Duan et al. “Rl ^2: Fast reinforcement learning via slow reinforcement learning”. In: *arXiv preprint arXiv:1611.02779* (2016).

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[6] Iason Gabriel. “Artificial intelligence, values, and alignment”. In: *Minds and machines* 30.3 (2020), pp. 411–437.

[7] Assaf Hallak, Dotan Di Castro, and Shie Mannor. “Contextual markov decision processes”. In: *arXiv preprint arXiv:1502.02259* (2015).

[8] Parameswaran Kamalaruban et al. “Environment Shaping in Reinforcement Learning using State Abstraction”. In: *arXiv preprint arXiv:2006.13160* (2020).

[9] Changjian Li and Krzysztof Czarnecki. “Urban driving with multi-objective deep reinforcement learning”. In: *arXiv preprint arXiv:1811.08586* (2018).

[10] Fan-Ming Luo et al. “Adapt to environment sudden changes by learning a context sensitive policy”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 7. 2022, pp. 7637–7646.

[11] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[12] Lorenzo Moro et al. “Quantum compiling by deep reinforcement learning”. In: *Communications Physics* 4.1 (2021), p. 178.

[13] Anusha Nagabandi et al. “Learning to adapt in dynamic, real-world environments through meta-reinforcement learning”. In: *arXiv preprint arXiv:1803.11347* (2018).

[14] Sindhu Padakandla. “A survey of reinforcement learning algorithms for dynamically varying environments”. In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–25.

[15] Sindhu Padakandla, Prabuchandran KJ, and Shalabh Bhatnagar. “Reinforcement learning algorithm for non-stationary environments”. In: *Applied Intelligence* 50 (2020), pp. 3590–3606.

[16] Markus Peschl et al. “MORAL: Aligning AI with human norms through multi-objective reinforced active learning”. In: *arXiv preprint arXiv:2201.00012* (2021).

[17] Matteo Pirotta, Simone Parisi, and Marcello Restelli. “Multi-objective reinforcement learning with continuous pareto frontier approximation”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 29. 1. 2015.

[18] *PPO Implementation Details*. `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. Accessed: [insert date here]. 2022.

[19] Kate Rakelly et al. “Efficient off-policy meta-reinforcement learning via probabilistic context variables”. In: *International conference on machine learning*. PMLR. 2019, pp. 5331–5340.

[20] Diederik M Roijers et al. “A survey of multi-objective sequential decision-making”. In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 67–113.

[21] Tom Schaul et al. “Universal value function approximators”. In: *International conference on machine learning*. PMLR. 2015, pp. 1312–1320.

[22] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[23] John Schulman et al. “Trust region policy optimization”. In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[24] Maruan Al-Shedivat et al. “Continuous adaptation via meta-learning in nonstationary and competitive environments”. In: *arXiv preprint arXiv:1710.03641* (2017).

[25] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[26] Peter Vamplew et al. “Human-aligned artificial intelligence is a multiobjective problem”. In: *Ethics and Information Technology* 20 (2018), pp. 27–40.

[27] Mudit Verma and Subbarao Kambhampati. “A State Augmentation based approach to Reinforcement Learning from Human Preferences”. In: *arXiv preprint arXiv:2302.08734* (2023).

[28] Lilian Weng. *A (Long) Overview of Reinforcement Learning*. `https://lilianweng.github.io/posts/2018-02-19-rl-overview/`. Accessed: [insert date here]. 2018.

[29] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. “A generalized algorithm for multi-objective reinforcement learning and policy adaptation”. In: *Advances in neural information processing systems* 32 (2019).

A

Implementation Details

A.1. Network Implementation details

The Network Implementation Details is stated in figure A.1

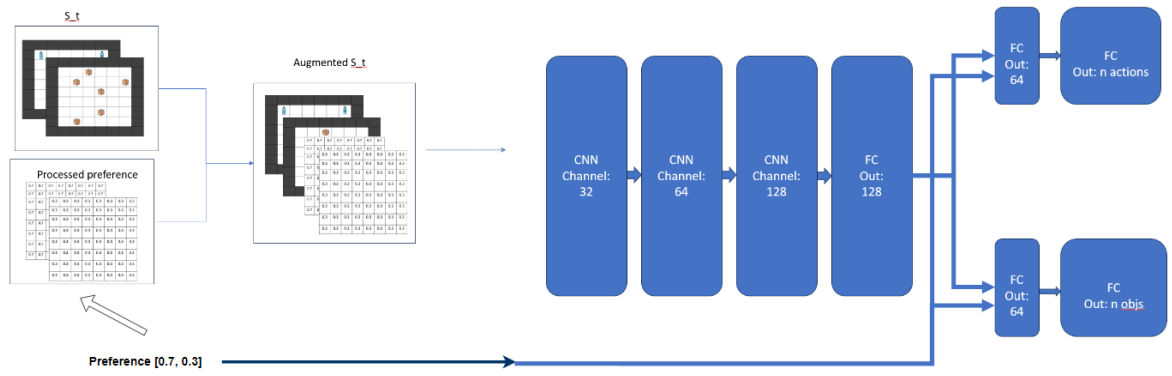

Figure A.1: Details of Network Layer Parameters and input

A.2. Training Hyperparameters

The training hyperparameters of both GBP PPO and the corresponding training process are listed in the table. A.1

GBP PPO Training Hyperparameters	
Training Steps	9e6
Batch Size	32 time steps
Number of Parallel Environments	16
Learning Rate	3e-4 (linearly decreases with training)
Entropy Coefficient	0.01
Discount Factor (gamma)	0.995
Clip Parameter (epsilon)	0.1
Number of PPO Epochs	5
GAE Lambda	0.98
Gradient clipping	0.5

Table A.1: GBP PPO Training Hyperparameters

A.3. Details of State Augmentation

The state representation in Experiment 1 consists of 4 layers. The first layer contains movable and unmovable cells in the grid. The second layer contains information about the agent's position. The third layer contains the positions of all items corresponding to Objective 1, and the fourth layer contains the positions of all items corresponding to Objective 2. The augmented state includes an additional two layers representing the preference for two objectives, with each layer containing the weight value of a specific objective. For further details, see Figure A.2.

Similarly, the state representation in Experiment 2 consists of 5 layers. The first four layers contain the same information as in Experiment 1, but with an additional layer representing the positions of items for Objective 3. The augmented state includes an additional three layers representing the preference for three objectives, with each layer containing the weight value of a specific objective. For further details, see Figure A.3.

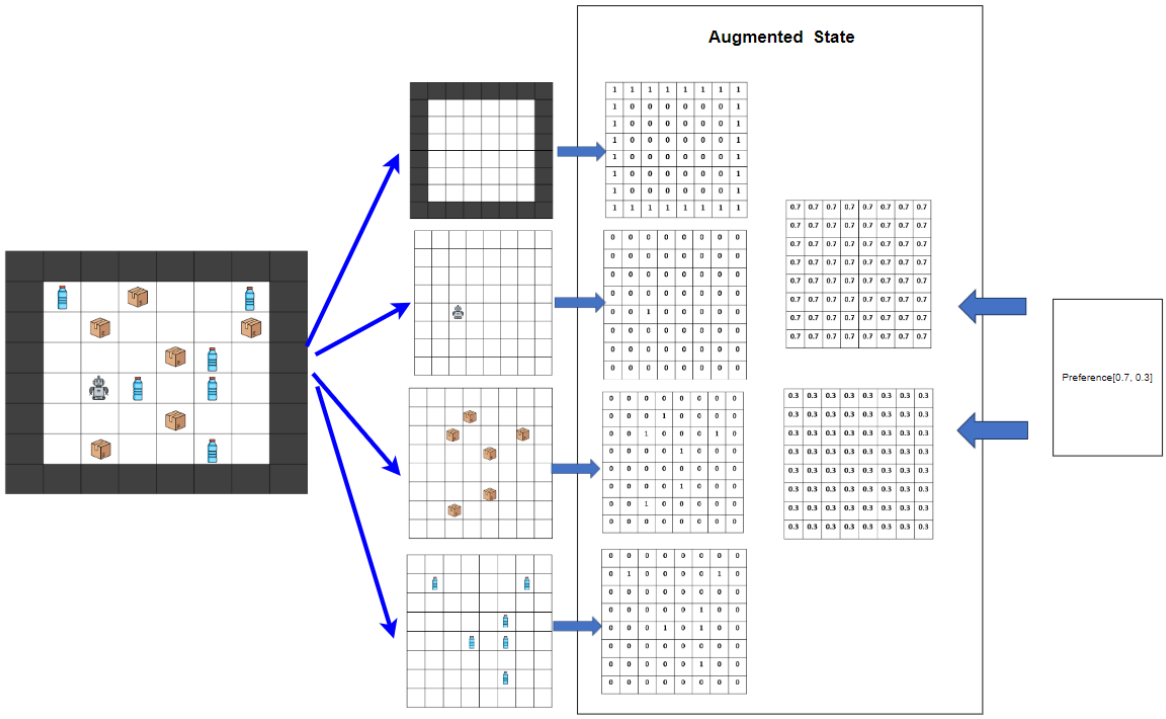

Figure A.2: Augment State of Experiment 1

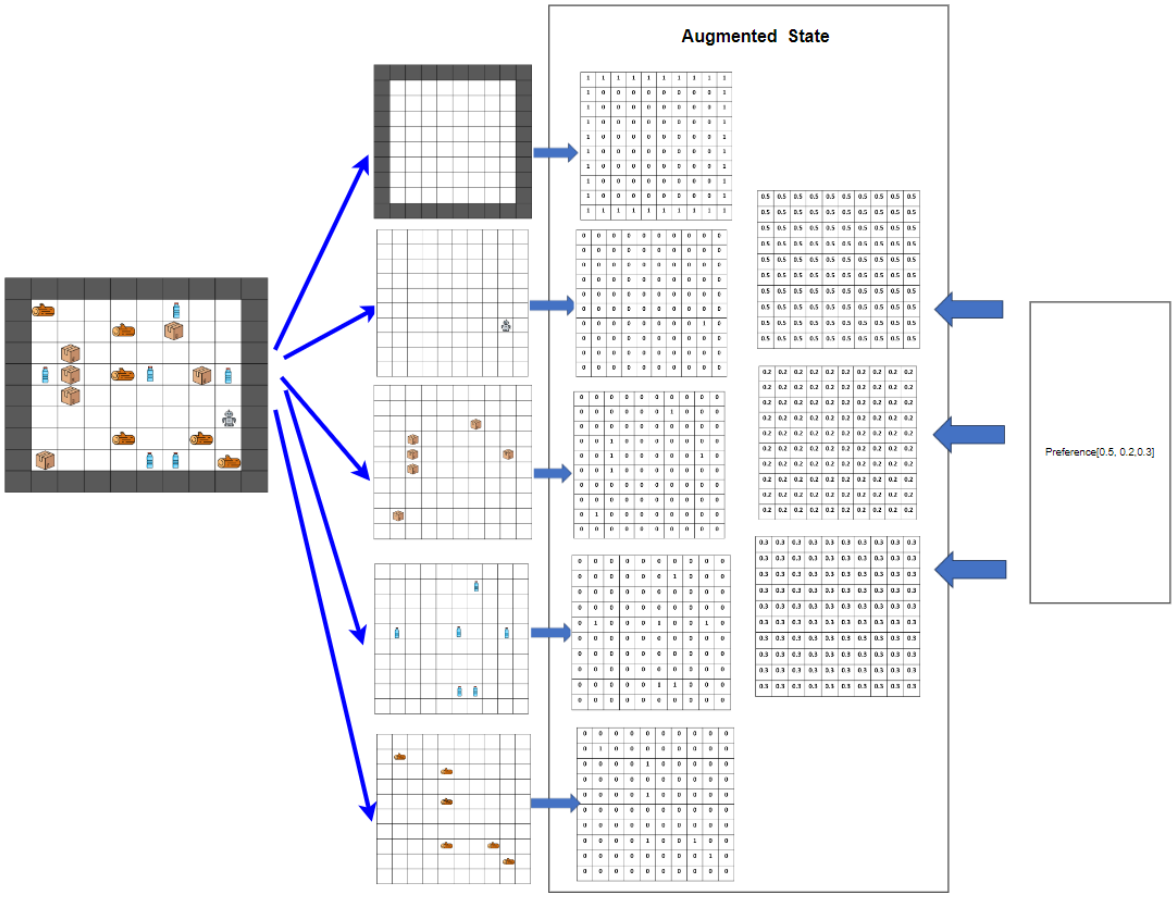

Figure A.3: Augment State of Experiment 2