Learning a Policy from User Preferences

An Interactive Approach to Multi-Objective Reinforcement Learning

Henwei Zeng

Delft University of Technology

TU Delft

Learning a Policy from User Preferences

An Interactive Approach to Multi-Objective Reinforcement Learning

by

Henwei Zeng

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday September 8, 2025 at 12:45 PM.

Study Programme:	Master Computer Science
Research group:	Interactive Intelligence
Study Programme Track:	Artificial Intelligence Technology
Thesis advisor:	P. Murukannaiah
Daily co-supervisor:	Z. Osika
Project Duration:	November, 2024 - September, 2025
Faculty:	Faculty of Electrical Engineering, Mathematics Computer Science, Delft

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

TU Delft

Preface

This research paper was written as a part of my Computer Science master's thesis project at Delft University of Technology. The paper presents my work over the past 9-10 months, where I developed a novel approach to extend the classical Multi-Objective Reinforcement Learning (MORL) by incorporating an interactive component that allows users to specify their objective preferences during the learning phase of the MORL algorithm. I combined the intuitions of both the state-of-the-art *Prediction-Guided Multi-Objective Reinforcement Learning* (PGMORL) method and the *E-NAUTILUS* method to create the User-Guided Multi-Objective Reinforcement Learning algorithm. The algorithm limits the search space within the user-specified preference values by bounding the space using limits, which allows the algorithm to quickly converge towards a policy/solution that the user prefers.

The reason I chose and worked on this project was that during my first year, I developed a great interest in Deep Reinforcement Learning methods and wanted to focus most of my master's work on becoming familiar with the Deep RL concept and exploring its practical applications. This Thesis project allowed me to work on both Multi-objective algorithms and RL methods, which allowed me to extend my knowledge even further in these CS domains.

During the past 2 years in my master's, I learned a lot from my professors and PhD researchers through their works and lectures. Furthermore, also from my colleagues which I also learned and discussed many new/interesting topics and ways to solve the problems. And for the many hours we have worked together on projects to create something that we can be proud of.

Specifically, I want to express my gratitude to my daily supervisor, Zuzanna, for guiding me over the past year through this Thesis project. Her expertise allowed me to explore many different existing algorithms and methods, and to discuss how to proceed during every stage of this project. Next, I also want to thank Pradeep as the responsible professor, for giving me meaningful insights and his valuable feedback on my work.

Beyond academia, I want to thank my friends from the student association DSEA Ignite for their support and for being an unforgettable part of my student life. I am proud to have served a board year with the association, even during challenging times. I also want to thank my housemates, with whom I have shared the past seven years, from the very first day of my bachelor's to the completion of my master's.

Finally, I am deeply grateful to my family: my father, mother, and two younger siblings, for their constant encouragement, patience, and support throughout my studies.

Without the support, guidance, and encouragement of all of you, I would not have been able to reach this point in my academic journey. For that, I am truly grateful. Thank you!

Henwei Zeng
Delft, September 2025

Summary

Many daily-life problems are complex due to their multi-objective nature. For example, we have an objective to reach a destination by driving a car. The driver's preference can differ for each person, some want to drive safely and arrive at their destination in a shorter amount of time. Others drive fast, but navigate more dangerously through the traffic. Over the past decade, there has been growing research on Multi-Objective Reinforcement Learning (MORL) problems, which simulate the complexities of real-life scenarios. Because there are multiple objectives to be optimized, the majority of the MORL meth- ods focus on providing a dense set of solutions called the Pareto Front as a result. The issues with the current approaches are that generating a large solution set requires high computational costs, and it can still be difficult for the user to find their most preferred solutions from a large solution set. One solution to this is to create an *interactive* method, where the user is asked for their preferred solutions during every iteration of the learning/search process of the algorithm. This allows the algorithm, by using the provided information, to converge towards a single solution that is preferred by the user, rather than providing a large collection of solutions. However, currently, there is limited or no literature on interactive MORL algorithms.

In this work, we dive deeply into the current state-of-the-art MORL works and other Multi-Objective Optimization (MOO) methods. Literature on Classical Multi-Objective Optimization and Evolutionary Multi-Objective Optimization methods was exhaustively studied. We took the ideas from different MORL works and MOO works and combined them to create a novel interactive MORL method.

Our novel algorithm uses a bounding of the search space concept to only search for policies/solutions that are strictly better than the provided feedback from the user. The user can give feedback by selecting their most preferred solution in each iteration from the list of non-dominated solutions. By bounding the search space in this way, the algorithm can slowly converge towards the user's preferred region of solutions, while allowing some flexibility to change their preference since they can also choose for older non-dominated solutions that are outside the current bounded region.

Furthermore, in this work, we also investigated different evaluation methods to properly evaluate the performance and usefulness of interactive algorithms. We came up with several metrics to compare the quality of solutions and the performance of the training process. Our metrics consist of creating an artificial user utility function to compare the final solutions and the number of time steps required to converge to a preferred solution to test the performance in terms of speed.

Our experiments show that interactive methods can converge faster than non-interactive ones when user preferences are skewed toward one objective. A limitation, however, is the risk of getting trapped in local optima, which may slow progress in certain cases. We believe this is a meaningful contribution to the growing field of interactive MORL by offering both methodological insights and empirical evidence, laying a foundation for further research on interactive MORL.

Learning a Policy from User Preferences: An Interactive Approach to Multi-Objective Reinforcement Learning

Henwei Zeng
Delft University of Technology
Delft, The Netherlands

Zuzanna Osika
Delft University of Technology
Delft, The Netherlands

Pradeep Murukannaiah
Delft University of Technology
Delft, The Netherlands

ABSTRACT

Many real-life problems are complex due to their multi-objective nature. Over the past decade, there has been growing research on Multi-Objective Reinforcement Learning (MORL) problems, which simulate the complexities of real-life scenarios. Because there are multiple objectives to be optimized, the majority of the MORL methods focus on providing a dense set of solutions called the Pareto Front as a result. The issues with the current approaches are that generating a large solution set requires high computational costs, and it can still be difficult for the user to find their most preferred solutions from a large solution set. In this research, we propose an *interactive* MORL method where the user is asked for their preferred solution in every iteration from the current solution set, and the algorithm utilizes this information to enhance its learning process to find preference-aligned solutions. This is achieved by bounding the solution space to only search for new policies that outperform the previously user-selected solution within these bounds. We evaluate our method using an artificial user function to simulate preferences, comparing it with non-interactive MORL methods. Metrics to compare the quality of solutions include the number of learning steps required to converge to a preferred solution, the value achieved on the artificial user function. The results demonstrate that the interactive method provides a dense set of solutions in the user's region of interest, and it tends to converge faster towards the user's preferred solution.

KEYWORDS

Reinforcement Learning, Interactive, Multi-Objective Optimization, Preference Learning

1 INTRODUCTION

Many real-world decision-making problems involve balancing multiple, often conflicting objectives. Improving one objective frequently leads to the deterioration of another, and such problems typically require decisions to be made sequentially over time. A powerful framework for tackling these challenges is Multi-Objective Reinforcement Learning (MORL), which extends classical single-objective Reinforcement Learning (RL) to optimize over multiple objectives [20, 35, 41].

Consider navigation optimization in traffic scenarios [50]. When the goal of the driver is to go from one place to their destination, multiple considerations are often involved, such as distance, fuel consumption and traffic density. Each user values those objectives differently, one can prefer a low traffic density in exchange for a longer distance, and another would prefer a low fuel consumption. MORL offers two main approaches to solving such problems. The *a priori* approach incorporates user preferences into the learning process from the start, while the *a posteriori* approach learns a diverse set of policies that approximate the Pareto front, allowing the user to select a preferred solution afterward.

However, both approaches have their limitations. *A priori* methods lack flexibility; if user preferences change, the agent typically needs to be retrained entirely. *A posteriori* methods, on the other hand, are computationally expensive and often overwhelm the user with a large set of solutions to choose from. This results in difficulties in applying MORL to real-world problems.

In contrast, the field of Multi-Objective Optimization (MOO) offers a third approach, which is the *interactive* optimization. Interactive MOO mitigates the limitations of both *a priori* and *a posteriori* methods by incorporating user feedback throughout the optimization process [13, 24, 26]. This allows the algorithm to converge toward a single, user-preferred solution without requiring prior knowledge of the user's preferences or overwhelming the user with a large set of solutions. The benefits of interactive optimization include reduced computational cost and the ability to adapt to evolving preferences, since users can learn about the trade-offs and adjust their input during the search process.

These insights motivate the integration of interactive MOO principles into MORL, which makes them more applicable to complex real-life problems. The interactive MORL literature is still very limited, there is one existing interactive MORL approach that allows the user to specify a target point in the objective space, which the agent attempts to reach by minimizing the difference between its expected return and the target [44]. The target can be updated before and during training, enabling user guidance throughout the learning process.

In this work, we propose an interactive preference-guided MORL framework that builds on the PGMORL algorithm [51] and incorporates ideas from the E-NAUTILUS method [37]. Our approach allows the user to interactively influence the learning process by selecting preferred solutions after each iteration. These preferences are then used to constrain the search space to policies that are strictly better than the selected one, focusing exploration on the region of interest.

To summarize, our contributions are as follows:

- We introduce a novel interactive MORL algorithm with an evolutionary search component. During the learning process, the user selects their most preferred solution, and the algorithm bounds the search space to focus on improving upon this solution. This guides the search toward the user's preferred region and results in a compact set of high-quality policies.
- To evaluate our algorithm, we adopt and adapt several evaluation strategies from the field of evolutionary algorithms (EA) [2, 7, 8, 21] to enable fair and meaningful comparisons with a non-interactive baseline. Specifically, we make use of artificial user utility functions, measure

the number of steps required to reach predefined target values, and compare the final output solutions between methods.

The paper is structured as follows: In Section 2, we will review the preliminaries, explaining key concepts utilized in this work and discussing related research. We will then move to Section 3, where we will go over the existing implementations first, and afterwards we introduce our proposed method and provide a detailed explanation. Next, in Section 4, we will present the specifics of our experimental setups, such as environments and evaluation methods. This will be followed by Section 5, where we will thoroughly analyze the results and provide explanations. In Section 6, we will shortly go over what could be improved and suggest directions for future research on interactive MORL. Finally, in Section 7, we will summarize our findings and conclude our work.

2 BACKGROUND

In the following subsections, a background is given about multi-objective optimization (MOO) and some of the current state-of-the-art MORL methods in how they calculate the Pareto front. Afterwards, classical and evolutionary interactive approaches are explained in more depth.

2.1 Multi-Objective Problem

We define the multi-objective problem as follows:

$$\max f(x) = \max\{f_1(x), f_2(x), \ldots, f_i(x)\} \tag{1}$$

with $i \geq 2$ conflicting objective functions f_i. Usually, we want to maximize all the objective functions simultaneously. x is the objective value and the vector of objective function values $z = f(x) = (f_1(x), ..., f_i(x))^T$ is the *objective vector*, due to the conflicting nature of the functions, it is often not possible, in non-trivial settings, to get a solution where all individual objective function values are optimal in the multi-objective problem [13, 24, 26, 28, 37]. This brings us to the definition of *Pareto Optimality*:

Definition 1 (Pareto Optimality). *A solution x is (globally)* ***Pareto optimal*** *if there does not exist another solution x' such that $f_i(x') \geq f_i(x)$ for all $i = 1, \ldots, k$ and $f_j(x') > f_j(x)$ for at least one index j.*

An objective vector $z^* \in Z$ is ***Pareto optimal*** *if there does not exist another vector $z \in Z$ such that $z_i \geq z_i^*$ for all $i = 1, \ldots, k$ and $z_j > z_j^*$ for at least one index j. Equivalently, z^* is Pareto optimal if the policy corresponding to it is Pareto optimal.*

A set of all Pareto Optimal solutions is called the *Pareto set*, and the image of such a set is called the *Pareto front*.

To formally determine whether one solution is better than another in a multi-objective context, we rely on the concept of *Pareto dominance*. Pareto dominance provides a partial ordering of solutions based on the idea that one solution is strictly better if it performs at least as well across all objectives and strictly better in at least one.

Definition 2 (Pareto Dominance). *Given two objective vectors $z, z' \in Z$, we say that z* ***Pareto dominates*** *z' (denoted $z \succ z'$) if $z_i \geq z'_i$ for all $i = 1, \ldots, k$ and $z_j > z'_j$ for at least one index j.*

Equivalently, a solution x Pareto dominates another solution x' if $f_i(x) \geq f_i(x')$ for all $i = 1, \ldots, k$, and $f_j(x) > f_j(x')$ for some j.

For most multi-objective optimization methods, using and having knowledge of the *ideal* and *nadir vector* can be quite valuable. The *ideal vector*, $z^* = (z_1^*, ..., z_i^*)^T$, contains the most optimal individual objective function values, which is the highest value for a maximization problem. The *nadir vector*, $z^{nad} = (z_1^{nad}, ..., z_i^{nad})^T$, on the other hand, is defined as the objective vector where each value is the worst possible outcome. In practice, calculating the nadir vector can be quite challenging, but it can be approximated using a pay-off table or other more recent approximation methods [4, 12, 24].

2.2 Multi-Objective Reinforcement Learning

Within MORL, we can formulate the multi-objective problem as a Multi-Objective Markov Decision Process (MOMDP) [20]. The MOMDP is represented by the tuple $\langle S, A, T, \gamma, \mu, R \rangle$ where:

- S is the state space.
- A is the continuous or discrete action space.
- $T : S \times A \rightarrow S$ is the transition function, specifying the probability of transitioning from state s to s' given action a.
- $\gamma \in [0, 1)$ is the discount factor
- $\mu : S \rightarrow [0, 1]$ is the probability distribution over the initial states
- $R : S \times A \times S \rightarrow \mathbb{R}^d$ is the vector reward function, over $d \geq 2$ number of objectives.

The main difference between MOMDP and the traditional single-objective MDP is the vector of rewards R, since we get a reward value for each objective for each action that we take in the environment. This makes the problem much more complex, as traditional RL methods cannot learn with multiple reward values during the learning process of the agent, and we cannot make use of the fact that we can maximize a single reward. Within MORL, we can classify the methods into two main categories depending on the output of the algorithm. First, we have a single-policy method, where the output is a single solution. The user's preference must be known before starting the training of the agent(s). A major drawback of this method is that the preference of user might change. Then the policy has to be retrained by adjusting the preferences. For the second method, we have the multi-policy method, where the output is a set of solutions whose corresponding objective vectors form an approximation of the Pareto front. For this method, the user preference is not known a priori, but the preferred policy can be selected after generating the output. This method is slower to train compared to the single-policy method, but the user can easily switch to a different policy when the preference changes, since the algorithm outputs a wide coverage of different policies. Both methods depend on what we call the *user utility*, where the user's preference for different objectives is mapped to a scalar value [20, 32].

2.2.1 Scalarization Functions. One of the solutions to the vectorized reward problem is that we make use of a scalarization function, also known as a utility function in the majority of MORL literature [20], $u : \mathbb{R}^d \rightarrow \mathbb{R}$. The scalarization function maps the vector of rewards of the policy to a single scalar value. This effectively reduces our MOMDP problem to a single-objective problem. Another advantage of the scalarization function is that we can also take the user utility into account, this approach allows us to collect the user's preference a priori and use that information to derive desirable policies [35]. We can split the scalarization

function into two categories, namely *linear* and *non-linear scalarization* The most common linear scalarization function is the weighted sum of the values for each objective function [17]:

$$U(z) = U(f(x)) = \omega^\top f(x) = \sum_{i=1}^{m} \omega_i f_i(x) \tag{2}$$

Where $U(z)$ represents the singular scalarized reward from the objective vector of rewards z, the weights ω can be used to express the preference of the user. However, the drawback of a linear scalarization is that it is unable to approximate a concave Pareto Front, as policies in the concave area usually receive less scalarized reward compared to the policies in convex regions [35, 47]. Therefore, we also have non-linear scalarization functions, such as the *weighted Chebyshev scalarization function* [45], to approximate the true Pareto Front.

2.2.2 Single-Policy Methods. When a single policy & a linear scalarization function method is chosen, meaning with a known user utility, any standard single-objective RL algorithm can be applied to multi-objective problems by transforming MOMDP into MDP with linear scalarization using weights [35]. When the true Pareto front can only be approached by non-linear scalarization functions, one MORL method that can be considered is the Expected Utility Policy Gradient (EUPG) [34]. The EUPG optimizes over the expected value of the utility of the return using a policy gradient method combined with Monte-Carlo simulations. Van Moffaert et al. [45] proposed the non-linear Chebyshev scalarization function, where the weights are applied to the L_∞, which is also known as the Chebyshev metric.

$$\min_{x \in \mathbb{R}^n} L_\infty(x) = \max_{o=1,\ldots,m} \omega_o \left| f_o(x) - z_o^* \right| \tag{3}$$

Where the z_o^* is the utopian point, which is the ideal vector plus some small constant ϵ: $z_o^* = z^* + \epsilon$, this is combined with the single-objective Q-learning algorithm to get the new multi-objective Q-learning algorithm (MO Q-learning).

2.2.3 Multi-Policy Methods. The literature on the multiple policies MORL algorithm is more substantial than the single policy algorithm since it is usually assumed that user do not know their exact preference a priori. Furthermore, most of the recent MORL works focus on linear scalarization functions. Abels et al. [1] proposed a dynamic weights method where the multi-objective Q-network output depends on the importance of each objective. The network is trained on a Diverse Experience Replay (DER) that can make use of the non-stationary weight settings. A similar work by Reymond et al. [33] also utilizes a single conditioned network that can encompass all non-dominated policies. The main advantage of such a network is that it is easily scalable in terms of the number of objectives and is very sample-efficient. Another approach is the work of Xu et al. [51], where they make use of a prediction-guided model to train a population of networks efficiently. The non-dominated policies are stored in a Pareto Archive, and the user can select their preferred policy from the archive after training. Similarly, the work of Parisi et al. [31] is also a population-based evolutionary method, where the algorithm tries to effectively learn the continuous approximation of the Pareto Front using an episodic exploration strategy and importance sampling.

2.2.4 Interactive MORL Methods. A third approach to obtaining the user's preferred policy is the interactive method. Limited research has been conducted in the interactive MORL domain,

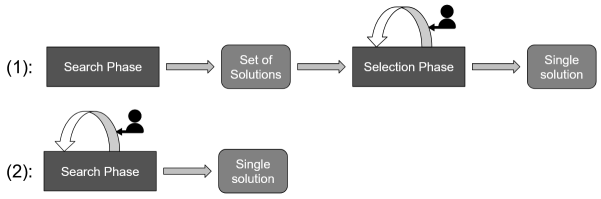


Figure 1: Interactive Method Classifications: (1) Interactive Selection Methods, (2) Interactive Search Methods

where user preferences are incorporated during the learning process of the agent to produce a single satisfactory policy. The advantage of this approach is that it eliminates the need for the user to specify their preferences a priori and avoids the higher computational costs associated with generating multiple policies. One interactive approach is the idea of Vamplew et al. [44], where they extend the Multiple Direction Reinforcement Learning (MDRL) and Q-learning approach to the task of learning multi-objective Pareto Optimal policies. This Q-steering algorithm can then be extended to where the user can interactively specify a target point and the agent *steers* towards the target point by trying to minimize the distance between the average reward and the target. However, some issues were raised during the experiments of the algorithm, for example, when we strongly favour one objective over all others, the algorithm can fail to converge to an optimal policy. Another issue is the high memory requirement of the Q-steering algorithm, which makes it harder to scale to a high-dimensional problem. Other interactive MORL works are based on the multi-objective multi-armed bandits (MOMABs) problem. Roijers et al. [36] used an interactive Thompson Sampling method for MOMABs. Similar to our approach, the algorithm employs a linear weighted-sum scalarization function, as described in Equation 2. By asking pairwise comparison queries to the user, the algorithm can learn the preference of the user, which can be translated to the weights ω.

2.3 Interactive Multi-Objective Optimization

In most MOO methods, the involvement of a *decision maker* (DM) is crucial, as the DM needs to select the most preferred solution/policy. We know three different kinds of categories for MOO, which are a priori, a posteriori and interactive methods. In a priori methods, the preference of DM is known beforehand, the algorithm can search for a single Pareto Optimal solution that fits the preference. In a posteriori methods, however, the algorithm does not know the preference and usually tries to generate the Pareto front and the DM can select their most preferred solution afterwards. And lastly, we have the interactive methods where the DM is actively participating and providing feedback during the search process, at the end, a final Pareto Optimal solution is found according to the DM's preferences. Interactive MOO has many advantages over a priori or a posteriori optimization methods. The DM can slowly learn more about the problem and what kind of solution is possible and can adjust their preference during the search process. Furthermore, compared to the a posteriori method, the DM will not be overwhelmed by the many different solutions, rather they get to choose from a small set of solutions until they find a single solution that fits their preference.

Within interactive methods, we could classify the methods into two different classes [22]:

- Interactive Selection Methods, those methods are usually semi-a posteriori since the algorithm runs in two phases. The first phase is where we generate a Pareto front, and the second phase is where the algorithm guides the DM to their preferred solution in the generated Pareto front.
- Interactive Search Methods, where the algorithm asks for user feedback to steer the search process towards a more relevant and preferred area for the user. Often, evolutionary multi-objective optimization (EMO) methods are classified within this class.

2.3.1 Interactive Selection Methods. Interactive Selection Methods is also what is called the classical interactive MOO in the literature. Usually, a set of solutions is generated and presented to the DM. The DM can then provide feedback or select their preferred solution. Based on the feedback, a more representative set of solutions is generated. This is repeated for several iterations until the DM finds a satisfying solution [24]. The advantage of the Interactive Selection method is that the process of generating the Pareto front and the interactive process are separated, meaning that the time required by the DM to arrive at a preferred solution is generally reduced compared to involving the DM during the search process. Within the interactive MOO field, we can further divide interactive methods into different classifications based on what type of feedback the DM needs to provide [23, 24, 26, 38].

- *Aspiration Levels Method*: In the aspiration levels method [9, 49], the DM provides what their aspiration levels are for each objective function value $f_i(\pi)$ for all i. The aspiration levels represent the minimum value or return that the solution policy needs to reach for the DM to be considered satisfactory. Similar methods that require some reference point or a goal to reach also fall into this category.
- *Classification Method*: In the classification method [4, 27], the DM needs to provide some information on how each objective value needs to be changed to become their desired solution. This is often done by giving feedback on whether the value needs to be improved, remains as it is, or can be worsened. In multi-objective problems, it is simply not always possible to improve objective values while keeping other values the same; therefore, some objective values need to be worsened if you want to improve some other values.
- *Trade-Off Method*: The trade-off method is one of the more classical interactive methods. In this class, we utilize the trade-off between two different objective value functions. The DM usually needs to "trade" by diminishing the value of one objective function to improve the value of another objective function, while all other objective functions remain the same. This scheme is what we call the *subjective trade-off* [18, 29]. Another popular scheme is to show two solutions and ask the DM the desirability of such trade-off, we call this the *objective trade-off* [10, 52].
- *Comparing Solutions Method*: In this method, the DM needs to select their preferred solution from the presented set of solutions [25, 37, 40]. Based on the selected solution, new solutions are generated that are similar to the selected solution. Usually compared to the other methods, the cognitive load from the DM is lower since the DM only needs to select a solution rather than giving (sophisticated) preference information.

2.3.2 Interactive Search Methods. In contrast to the first method, interactive multi-objective meta-heuristics involve the DM with the MOO algorithm during the run, steering the search towards a more desired segment of the Pareto front [22]. Compared to the Interactive Selection method, a notable limitation of this method is its requirement to search through many different solutions and iterations during the search or learning process. This can cause human fatigue due to the necessity for continuous feedback to the algorithm. Certain algorithms have been enhanced to predict human preferences, thereby reducing the need for constant interaction by the decision maker [42]. The majority of interactive multi-objective meta-heuristics are evolutionary MOO methods. One of the very first methods of this kind was proposed by Tanino [43], where the evaluation of the current population is derived from the preference of the DM. The DM can point out unsatisfactory solutions or provide some kind of aspiration level. Many interactive multi-objective evolutionary algorithms (I-MOEA) differ in what part of the algorithm is adjusted using the preference of the DM, it can be either objectives, dominance and crowding distance [5, 19]. In the objectives case, usually the objective values get changed according to DM's preference, which will guide the search process slowly towards the preferred region. An example would be the work of Wagner and Trautmann [46], where they optimize a desirability function of the objectives, which expresses the preference of the user. To adjust dominance, the Pareto dominance is often replaced by a different dominance function that considers the preference information [6, 14, 39]. In those functions, they often use a reference point, supplied by the DM, and derive the dominance from the distance to the reference point while preserving the original Pareto Dominance as much as possible. Finally, crowding distance is a metric used to estimate the proximity of a solution to its neighbouring solutions in the objective space. A high crowding distance means that the solution is in a sparse region, with a low number of neighbouring solutions. An example would be a light beam search method combined with the NSGA algorithm from Deb and Kumar [11], where they search in the neighbourhood of interesting solutions.

3 METHOD

In this section, the core algorithms are introduced, where the inspiration has been drawn from for our method design: PGMORL and the E-Nautilus method [37, 51]. In the last subsection, the adjustments to the PGMORL algorithm, for including interactive optimization with user feedback, are explained in more depth.

3.1 Prediction-Guided Multi-Objective Reinforcement Learning (PGMORL)

The PGMORL method is a multi-policy evolutionary MORL method where the algorithm constructs a set of Pareto-optimal policies that capture the Pareto front [51]. The algorithm consists of 3 different stages:

3.1.1 Warm-up Stage. In the warm-up stage, the algorithm initializes n different starting agents/networks, and those policies are paired with one of the n evenly distributed non-negative weights $\{\omega_i\}(\sum_j \omega_{i,j} = 1, 1 \leq i \leq n)$ for the rewards. Then the agents are trained using the MOPPO algorithm for a fixed amount of k iterations. The warm-up stage is important before running the core evolutionary stage of the algorithm, as the agents need to get out of the low-performance policy region before the model can compare the agents in the next stage.

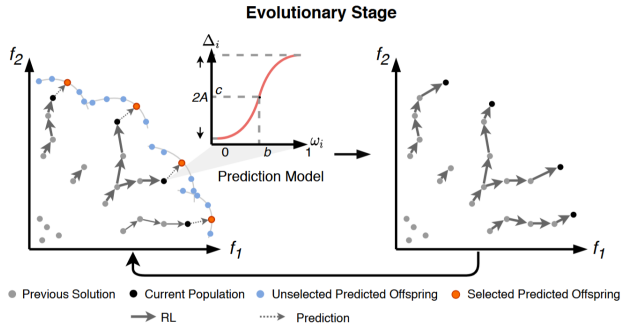


Figure 2: Evolutionary stage of PGMORL Algorithm [51]

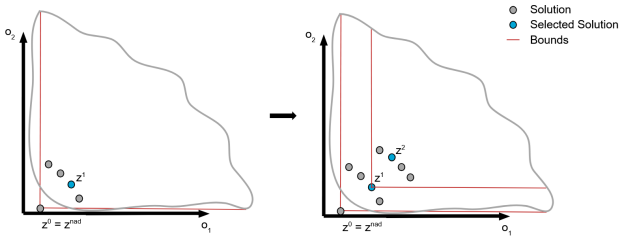


Figure 3: *E-NAUTILUS* [37], example with two iterations, where from the selected solution in z^1, a new iteration is generated with solutions that are strictly better than the selected solution

3.1.2 Evolutionary Stage. In the evolutionary stage, as shown in Figure 2, the algorithm keeps track of the current and past population of the agents along with their performances. Then it uses an analytical model to predict which combination of weights and agents will improve the Pareto front the most based on the past data. The model selects n different policy-weight combinations to train in the next l training iterations, producing new offspring policies. The evolutionary stage is finished after m number of generations. During this stage, a Pareto archive is used, and all non-dominated policies are stored, which will be the output at the end of the stage.

3.1.3 Post-processing Stage. At the final stage, the Pareto front is constructed from the set of final policies. The algorithm groups similar policies into families. A continuous Pareto representation will then be computed by linearly interpolating the policy families.

3.2 E-Nautilus

E-Nautilus is an Interactive Selection MOO method where we require an already generated Pareto front of the solutions [37]. This method takes the human aspects regarding trade-offs and anchoring bias into consideration. The algorithm has 3 stages, a preprocessing stage where we get the Pareto front of the solutions, a decision-making stage where the DM interacts with the algorithm to reach their preferred solution, and finally a post-processing stage where we find the closest solution from the Pareto front to the selected solution of the DM. The main advantage of using these stages is that there is no human interaction needed in the first and last stage, the first stage usually requires the most time, since generating the Pareto front to an MOO problem can take an extensive amount of time. The main functionality of this method lies in the interactive decision-making stage. In short, during the interactive stage, the DM is iteratively shown a set of intermediate points between the nadir point and the ideal point as seen from Figure 3. The DM selects its most preferred point, and new points are generated that are closer to the provided Pareto front, but also dominate the selected point. This ends when the maximum number of pre-specified iterations is reached or the DM has already selected one solution from the Pareto front.

Algorithm 1 Interactive Preference-Guided MORL

Input: Number of agents n, warm-up iterations k, training interval l, max generations m
Initialize: Generate n initial agents with evenly distributed weights ω: $\{\omega_i\}(\sum_j \omega_{i,j} = 1, 1 \leq i \leq n)$
for k **do** ▷ Warm-up stage
  Train $\{(n_i, \omega_i)\}_{i=1}^{n}$ agents using MOPPO
end for
Initialize population $\mathcal{P}$ with trained agents
for generation $g = 1$ to m **do** ▷ Evolutionary stage
  Evaluate all agents in $\mathcal{P}$
  User selects most preferred policy: π^{pref}
  Extract objective vector $z^{pref} = [x_1, x_2, \dots]$ from π^{pref}
  Define bounds based on $\ell(z) = z - \delta \cdot \sqrt{|z|}$
  Filter population: $\mathcal{P}_{filtered} = \{\pi \in \mathcal{P} \mid z(\pi) \succeq \ell(z)\}$
  Use prediction model to select n agents from $\mathcal{P}_{filtered}$
  for l **do**
    Train $\{(n_i, \omega_i)\}_{i=1}^{n}$ selected agents using MOPPO
  end for
  Update $\mathcal{P}$ with new agents
end for
Output: Final set of policies within bounds; user selects final preferred policy

3.3 Interactive Preference-Guided Multi-Objective Reinforcement Learning

In this section, we propose our main contribution. An interactive MORL method that searches for Pareto Optimal policies using the preference information that is provided by the DM. To make it interactive, it is crucial to identify the best moment in the algorithm to request user feedback and to utilize the user input to generate a more preferred policy. Our idea is to modify the PGMORL algorithm to make it interactive by using some core intuitions behind the interactive E-NAUTILUS method.

3.3.1 Overview. The overview is also presented in Algorithm 1. Initially, we opted to maintain the warm-up stage consistent with the original PGMORL algorithm, where we initialize n agents and weights: ω. The agents and their corresponding weights are trained for k warm-up iterations. This phase facilitates initial exploration and learning about the environment, which is critical for the agents. Furthermore, actions from the agents are usually highly noisy. Therefore, interacting with DM usually will not result in anything meaningful. After warming up, we proceed to the evolutionary stage. In this stage, the DM will be actively interacting with the algorithm. During each generation g, the DM can provide their preference information based on the current Pareto front by selecting their preferred policy π^{pref}. And then the information is used to select the next agent-weights combination to train for l iterations. In the next subsection 3.3.2, we will go into more details about this part. After the maximum number of generations m, the evolutionary stage terminates and a final

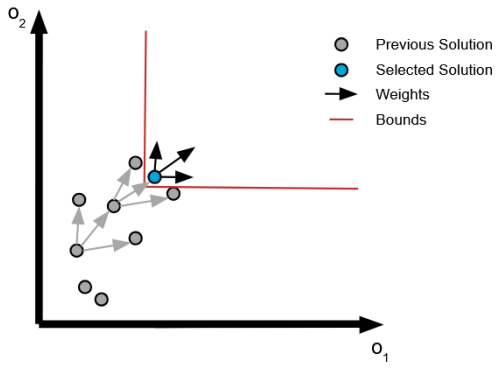


Figure 4: Bounding the search space

set of policies from the Pareto Archive, filtered by the bounds, is shown to the DM. The DM can make their final choice by selecting their most preferred policy in case there is more than one policy. The post-processing stage is removed compared to the original PGMORL algorithm, as there is no need to post-process the set of solutions since the output of the interactive method is only one policy.

3.3.2 Limit search region through bounding the objective vector. At each iteration, before selecting the agent-weights combinations for training, the current Pareto front will be presented to the DM. From the current front, the DM can choose their most preferred solution. The algorithm then uses the preference information extracted from the objective values from the selected solution $z^{pref} = [x_1, x_2, \dots]$. Using the objective vector, we set a lower (or upper, in case of minimization) bound that the objective vector must meet before being considered to become a candidate to train in the next iteration. To allow some exploration, the bounds are relaxed by a small margin $\delta \cdot \sqrt{|z|}$, in case more solutions in the neighbours have more potential to improve compared to the preferred solution. Formally, we define the lower bound ℓ as:

$$\ell(z) = z - \delta \cdot \sqrt{|z|} \tag{4}$$

Finally, the prediction model is only able to select agents that have a strictly better performance than the lower bounds: $z(\pi) \succeq \ell(z)$. This limits the search space to only agents that have similar or better performance than the preferred solution. At every generation, the bounds will slowly become tighter until the search converges to a region that is close to the true Pareto front, as shown in Figure 4.

4 EXPERIMENTAL SETUP

We implement our interactive multi-objective reinforcement learning by extending the PGMORL [51] method available in morl-baselines [16]. The code can be found on our GitHub page.[1] We test our method, the PGMORL and the PGMORL+E-NAUTILUS method, on several MORL environments, which are explained in the next subsection. In the rest of this section, we will describe evaluation methods to properly compare our method against the a posteriori PGMORL as a baseline. In our work, we implement a novel technique that does not yet have established evaluation methods. We are using some ideas from interactive evolutionary and interactive MOO methods to evaluate the performance of our method and the non-interactive method [2, 7, 8, 21].

[1] https://github.com/Henweiz/interactive_MORL

4.1 Environments

We employ two mo-gym environments: `mo-mountaincarcontinuous-v0` and `mo-halfcheetah-v5` [16]. These environments were chosen as they allow seamless integration with the current implementation of PGMORL, and the core of our algorithm is similar to PGMORL.

4.1.1 mo-mountaincarcontinuous-v0. is a continuous and multi-objective version of the classic `mountaincar` environment [30]. The environment is a car that is placed at the bottom of a valley. The goal of the agent is to make the car reach the flag on the top of the right hill. The observation space is the position of the car along the x-axis and the velocity of the car. The action is the acceleration that can be applied to the car in either direction. And finally, there are two reward values for this environment:

(1) Time penalty: The time it takes to reach the destination
(2) Fuel cost:
A negative norm based on the action vector: $-||action||^2$

Therefore, the trade-off in this environment is either, spend a considerable amount of fuel to reach the destination faster. Or an efficient use of fuel, but having a higher time penalty.

4.1.2 mo-halfcheetah-v5. is a Multi-Joint dynamic with Contact (MuJoCo) environment [48]. Where physical contact between the robot and the environment is simulated. The environment consists of 9 different body parts and joints connecting them. The action space of the agent is to control the joints in such a way that it makes the cheetah robot run forward. Furthermore, it also has a large observation space consisting of the position of the body parts and their velocities. The rewards consist of the following:

(1) Forward reward: The agent receives a positive reward when it moves forward in the right direction
(2) Control cost: A negative reward for the agent when it takes large actions

The goal of the environment is to enable the robot cheetah to move as fast as possible in the desired direction while maintaining a low control cost.

4.2 Evaluation methods

To evaluate the effectiveness of our proposed approach, we compare three different methods: our interactive preference-guided MORL, the baseline non-interactive PGMORL method, and a hybrid method that combines PGMORL with the E-NAUTILUS interactive decision-making procedure. These methods are selected to assess how user interaction—either integrated during learning or applied after learning—affects the quality and relevance of the resulting solutions.

- **Interactive Preference-Guided MORL:**
Our proposed method, where the user provides feedback during the learning process to guide the search toward their preferred region of the solution space. The algorithm adaptively restricts the search based on the selected solution to focus exploration.
- **PGMORL:** A non-interactive baseline method that combines evolutionary search and prediction modeling to generate a diverse set of policies approximating the Pareto front.

- **E-NAUTILUS+PGMORL:** A two-phase hybrid method. First, PGMORL is used to generate a Pareto-optimal solution set. Then, E-NAUTILUS is applied to interactively select the user's preferred solution from the final set.

4.2.1 Artificial user selection. For our interactive methodology to evaluate performance between the interactive method and the non-interactive method, it is essential to establish a model of user behaviour within the environment. Employing a human user for this process is usually impractical for the experiment framework, as it proves time-consuming when the algorithm operates over an extended duration, requiring the human user to constantly interact with it. Therefore, we use an artificial user selection method with a predefined utility function to determine its preferred policy [8]:

$$U(x_1, x_2) = \omega_1 \cdot x_1 + \omega_2 \cdot x_2 \tag{5}$$

Where x_1 and x_2 are the average episodic rewards for the objectives across five validation episodes. And ω is the weight assigned to the objective value. However, a limitation of utilizing an artificial user is its lack of 'flexibility' in decision-making when compared to a human decision maker (DM). Human DMs are capable of learning and adapting throughout the agent's training phase, usually resulting in a change of preference, whereas an artificial user consistently opts for the option with the highest utility function value. For our experiments, we have designed three distinct artificial users, each characterized by specific linear utility functions tailored for maximization problems:

- *User A*: with equal preferences: $U(x_1, x_2) = 0.5 \cdot x_1 + 0.5 \cdot x_2$
- *User B*: with preference on objective 1: $U(x_1, x_2) = 0.7 \cdot x_1 + 0.3 \cdot x_2$
- *User C*: with preference on objective 2: $U(x_1, x_2) = 0.3 \cdot x_1 + 0.7 \cdot x_2$

During each interactive decision-making instance, the artificial user selects the solution with the highest utility, as determined by its predefined user utility function. The selected solution is returned as feedback to the interactive method, which then uses it to update the bounds that constrain the search space.

To evaluate the performance of the method over time, we track two metrics based on the artificial user's utility assessments:

- **Maximum User Utility** $U_{max}(g)$: This metric represents the highest utility value in the population at generation g. It is computed by evaluating all individuals using the artificial user's utility function and selecting the one with the highest value. This serves as an indicator of the best solution quality for the user's preferences:

$$U_{\max}(g) = \max_{i \in P(g)} U(x_1^i, x_2^i) \tag{6}$$

- **Average Population Utility** $\bar{U}(g)$: This is the mean utility value of all individuals in the population at generation g. It indicates whether the population as a whole is converging toward the user's preferred region. Higher value means that the population is close to the preferred solution:

$$\bar{U}(g) = \frac{1}{|P(g)|} \sum_{i \in P(g)} U(x_1^i, x_2^i) \tag{7}$$

Where P(g) is the population at evolutionary generation g and $U(x_1^i, x_2^i)$ is the utility value of individual i based on its objective values x_1 and x_2.

4.2.2 Steps until threshold reached. An alternative evaluation metric involves measuring the number of global steps or the number of generations required for the algorithm to reach a predefined utility threshold. This threshold represents a minimum acceptable performance level as determined by the artificial user's utility function. Formally, for a given threshold value τ, we define the stopping time t_τ as the first global step at which the individual with the highest utility in the population exceeds the threshold:

$$t_\tau = \min \{g \mid U_{\max}(g) \geq \tau\} \tag{8}$$

The goal of the interactive method is to require fewer steps to reach this threshold τ compared to the non-interactive baseline, due to the guidance provided by the DM, which helps steer the agents toward regions of the objective space that align more closely with the user's preferences. For evaluation, we define three distinct target thresholds, each corresponding to a specific preference profile:

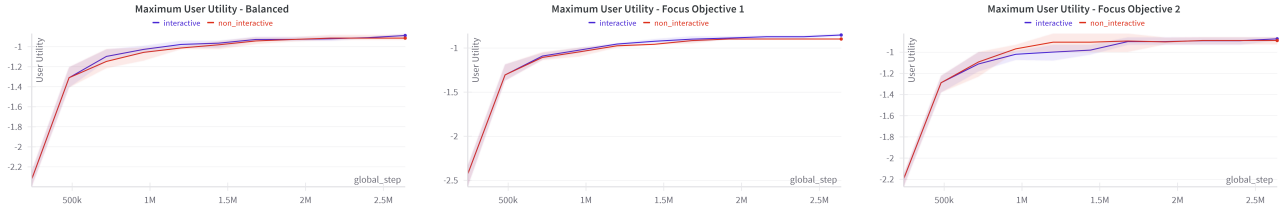
- *Target A*: A balanced threshold τ_c considering equal performance on both objectives x_1 and x_2
- *Target B*: A threshold τ_a focused on maximizing objective x_1
- *Target C*: A threshold τ_b focused on maximizing objective x_2

For each generation, instead of using an artificial user function to select the most preferred policy, we use the Euclidean distance metric $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$ to select the closest solution to the target solution as the artificial DM.

4.2.3 Quality of the Final Policy. The primary objective of the previous comparisons was to evaluate our interactive method against a non-interactive baseline. To further assess its performance, we compare the final solution generated by our method with that selected by E-NAUTILUS, a well-established interactive decision-making approach. Importantly, only E-NAUTILUS operates on a precomputed Pareto front—specifically, the one produced by the base PGMORL implementation—whereas our interactive method generates its own final solution through preference-guided search during training. For both methods, the final solution is selected based on the artificial user's utility function, as described in Subsection 4.2.1. To ensure a fair and robust comparison, we aggregate the outputs from three different random seeds and apply Pareto filtering to remove dominated solutions across these runs. The selected solutions are then evaluated based on their utility values and whether one dominates the other. For E-NAUTILUS, we also must specify the number of solutions shown per iteration as well as the total number of iterations. Although increasing these parameters typically improves selection accuracy, we set both to 3 in our experiments to match the limited size of the PGMORL-generated Pareto fronts in the tested MORL environments.

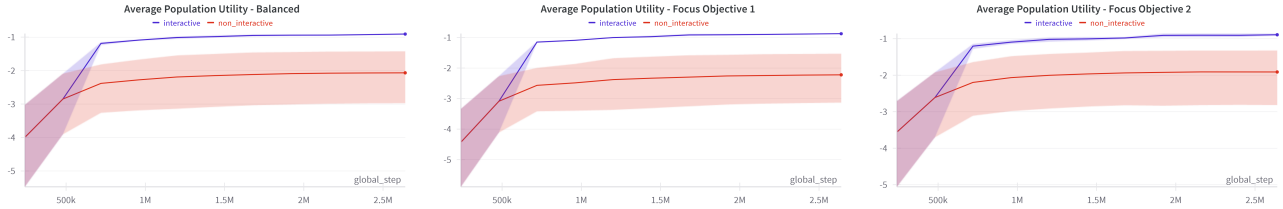
5 RESULTS

The training details and their hyperparameters can be found in Appendix A. The experiments are run on the Delft High Performance Computing Centre (DHPC) [15].



(a) Balanced Preference **(b) Focus Objective 1** **(c) Focus Objective 2**

Figure 5: MountainCar: Maximum User Utility under different preference settings.



(a) Balanced Preference **(b) Focus Objective 1** **(c) Focus Objective 2**

Figure 6: MountainCar: Average Population Utility under different preference settings.

5.1 Artificial Utility Function

In the artificial utility function evaluation, we look at both the best individual performance $U_{max}(g)$ and the population performance $\bar{U}(g)$ over time. We run each experiment for 3 different seeds.

5.1.1 mo-mountaincarcontinuous-v0. The maximum user utility values are shown in Figure 5. Overall, the performance of the two methods is largely comparable. Under the balanced preference setting, the interactive method initially outperforms the non-interactive method, although the latter catches up over time. For objective focus 1, both methods maintain similar performance throughout the experiment. In contrast, under objective focus 2, the interactive method starts with lower performance but quickly reaches a similar policy to the non-interactive approach. Across all three scenarios, the interactive method ultimately achieves higher maximum user utility, continuing to improve over time, whereas the non-interactive method tends to converge earlier.

In Figure 6, we see the average population utility over time. All three plots show similar results, where the interactive method clearly converges to the preferred region of the user by filtering individuals that are not within the bounded limits. For the non-interactive method, the population utility remains roughly the same after the initial improvements. This shows that the non-interactive opts for an even spread of the different solutions rather than converging to a single region in the search space. For both plots, we also see that the averages converge quite quickly, with barely any improvements over time.

Overall, the results indicate that both methods are generally capable of finding solutions aligned with the user's preference within a similar number of training steps. However, as shown in Figure 6, the interactive method tends to discover a higher number of similar solutions within the preferred region. Additionally, Figure 5 reveals that when preference is placed on the second objective (x_2), the interactive method performs slightly worse compared to the other preference settings. We assume this is caused by the environmental dynamics and reward structure. Specifically, the first objective (x_1) represents the time penalty to reach the goal, while the second objective (x_2) corresponds to fuel consumption. Prioritizing x_2 early in training may hinder the agent's ability to reach the goal efficiently, as it learns to minimize fuel usage. This not only increases the cost for x_1 but also inadvertently raises x_2 due to prolonged episode durations and a higher number of actions. In contrast, when focusing on objective 1 or adopting balanced preferences, the interactive method typically outperforms the non-interactive method across different seeds. These observations highlight the importance of initially optimizing for x_1 before shifting focus to x_2.

5.1.2 mo-halfcheetah-v5. Figure 7 displays the results for the Maximum User Utility. For objective focus 1, the non-interactive method clearly outperforms the interactive method. In contrast, for objective focus 2, both methods achieve comparable performance. Under the balanced preference, the interactive method performs better during the initial half of training, but the non-interactive method eventually catches up and surpasses it. Compared to the `mo-mountaincarcontinuous-v0` environment, the interactive
method in `mo-halfcheetah-v5` generally fails to outperform the non-interactive method by the end of training.

Figure 8 presents the Average Population Utility results. The trends observed are generally consistent with those from the other environment and across all preference types. However, in contrast to the `mo-mountaincarcontinuous-v0` environment, the `mo-half-cheetah-v5` environment exhibits the opposite behaviour in terms of variance across seeds. Specifically, the interactive method shows significantly higher variance, while the non-interactive method appears more stable. For objective focus 2, one seed resulted in an empty population at a certain point during training. This likely occurred due to the population selection mechanism, where only the best-performing individuals are retained each generation. If, after applying the preference-based bounds, no individuals meet the criteria, the population may be left empty.

The results indicate that the non-interactive method generally outperforms the interactive approach. We believe this is primarily due to the interactive method becoming trapped in local optima, leading the algorithm to repeatedly select the same policy across multiple generations before any potential improvement can be made. This highlights a limitation of our evaluation setup: the artificial user consistently selects the same solution, whereas a real user might adjust their selection strategy upon noticing a lack of progress. Moreover, we observed that the `mo-halfcheetah-v5` environment poses a significant challenge for training compared to `mo-mountaincarcontinuous-v0`. Even the base PGMORL algorithm struggles to generate a well-formed, continuous Pareto front in this setting. The results for objective 1 suggest that thorough policy exploration is critical in this environment. However, the interactive method with bounding the search space significantly restricts exploration, which in turn slows the performance improvement compared to the more freely exploring non-interactive method.

5.2 Steps to Reach Threshold

For this evaluation, we compared our interactive method and the non-interactive baseline. Each experiment is run for 3 different target values and five different seeds. The average is reported in the tables.

5.2.1 mo-mountaincarcontinuous-v0. For this environment, we decided to go for a maximum of 5 million global steps before timing out the run, and the following three threshold values:

- Target A: [-1, -1], with a focus on balanced objectives
- Target B: [-0.8, -1.2], with a focus on optimizing objective 1
- Target C: [-1.8, -0.5], with a focus on optimizing objective 2

The results of the experiment can be found in Table 1. We can see that for target A, the non-interactive baseline performs slightly better than the interactive method. Both modes were able to reach the target values within the maximum allotted global steps. For targets B and C, we see that the interactive method performs significantly better than the non-interactive method. The interactive method was able to find a policy that dominates the target policy much faster and was able to find it within 5 million global steps, while the non-interactive method failed quite consistently. Moreover, the standard deviation of the interactive method is relatively low for target C, which means it is able to find a preferred point fast and consistently.

We see from these results that the interactive method is able to find policies that are more focused on one objective faster, rather than a balanced policy placed in the middle of the Pareto front. Our intuition is that a balanced policy is easier to find when the exploration space is large, since you can reach it from multiple angles. For example, a policy that performs well for x_1 and then trains with a high weight for x_2 in the following iteration(s) and vice versa. When we are limiting the bounds for the balanced policy, it will only improve the most when the weights are even ($\omega(0.5, 0.5)$).

However, when we are putting more weight on one objective, the interactive method performs better since we do not waste resources on training policies that go in a different direction (high weight on the other objective). Therefore, the algorithm is able to find the preferred solution much faster compared to the non-interactive baseline.

5.2.2 mo-halfcheetah-v5. For this environment, we decided to go for a maximum of 7.5 million global steps before timing out the run and the following three threshold values:

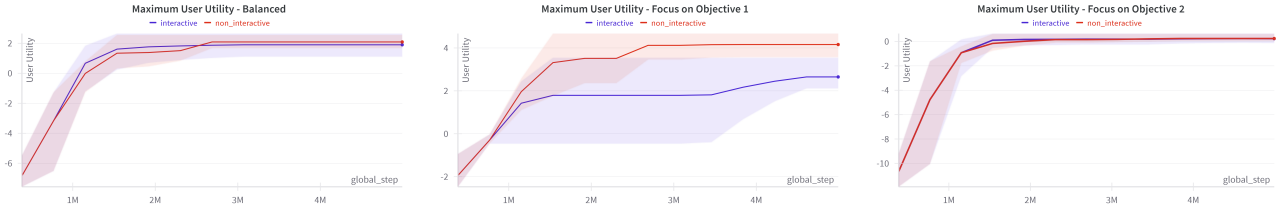
- Target A: [5, -7], with a focus on balanced objectives
- Target B: [7, -20], with a focus on optimizing objective 1
- Target C: [1, -0.5], with a focus on optimizing objective 2

From the results, we can observe that the environment is more difficult compared to the `mo-mountaincarcontinuous-v0` environment, as there are many more failed runs. For both target A and B, the non-interactive baseline performs significantly better compared to the interactive method. We noticed that the issue with the interactive method in this environment is that it gets much more easily stuck in local maxima compared to the non-interactive method, which is also mentioned before in Subsection 5.1.2. The artificial user then selects the same policy for multiple iterations without getting a better-performing solution. The non-interactive has much higher exploration space, so it is possible to reach the target thresholds from multiple angles, similarly to what we discussed in the `mo-mountaincarcontinuous-v0` results section. This is also one of the issues of the PGMORL algorithm design mentioned by the authors Xu et al. [51], as the algorithm can get stuck for a long period of time in local minima/maxima in order to reach a better performance. We also noticed that when the interactive method is able to find a solution, it does so in a similar number of global steps as the baseline method. However, for target C, it is much easier for the interactive method to find a dominating policy compared to the non-interactive counterpart.

5.3 Quality of Final Policies

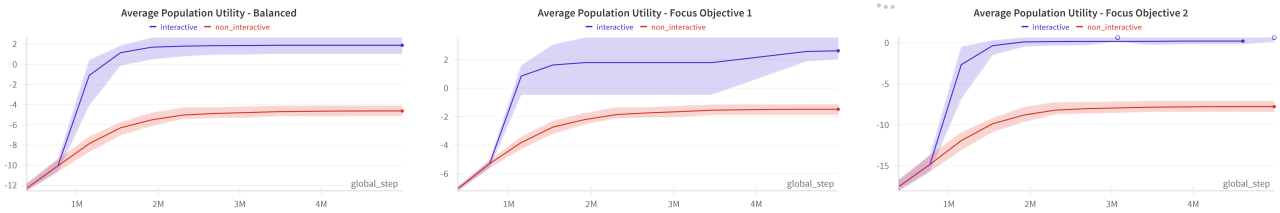
Figure 9 shows the Pareto-filtered results aggregated over three different seeds. The figure includes the final outputs selected by E-NAUTILUS+PGMORL for each of the three artificial user utility functions, as well as the final outputs from our Interactive Preference-Guided MORL approach. To improve visual clarity in cases where solutions overlap, a small jitter is applied, for example, the final solution selected by E-NAUTILUS for objective x_1 is the same as the one chosen for the balanced preference. The interactive method (denoted by triangles) tends to produce final solutions clustered around high values of objective 1 (x_1), while the non-interactive method yields more dispersed solutions across the objective space. We hypothesize that this behaviour arises because utility values are more easily maximized when prioritizing x_1. This is likely due to the higher reward scale associated with x_1 compared to x_2. Additionally, since x_1 corresponds to the time penalty for reaching the goal, optimizing it often results in lower fuel consumption as well, thereby indirectly improving x_2. To test this hypothesis, we conducted a separate experiment in which rewards were normalized before being passed to the user utility function. The results, shown in Figure 10, display a less concentrated spread of solutions. Notably, a few solutions produced by the interactive method are now shifted more toward the x_2 direction. Across all experiments, we observe that the final policies are non-dominated with respect to each other. An interesting finding is that the interactive method consistently offers multiple similar high-quality solutions in the same region in the final output after aggregation across seeds. When utility normalization is applied, more preferred solutions emerge for the artificial user.

The results for the `mo-halfcheetah-v5` environment, shown in Figure 11, differ notably from those of `mo-mountaincarcontinuo-`



(a) Balanced Preference **(b) Focus Objective 1** **(c) Focus Objective 2**

Figure 7: HalfCheetah: Maximum User Utility under different preference settings



(a) Balanced Preference **(b) Focus Objective 1** **(c) Focus Objective 2**

Figure 8: HalfCheetah: Average Population Utility under different preference settings.

Table 1: Comparison between interactive and non-interactive method for `mo-mountaincarcontinuous-v0` environment

Targets $[x_1, x_2]$	Method	Avg Steps ($\cdot 10^6$) ± Std	Avg Time per Run	Timeouts
A: [-1, -1]	Interactive	2,448 (± 0,621)	0:16:56	-
	Non-Interactive	**1,968** (± 0,520)	**0:14:07**	-
B: [-0.8, -1.2]	**Interactive**	**2,928** (± 0,935)	**0:20:31**	-
	Non-Interactive	No results	0:36:17	5/5
C: [-1.8, -0.5]	**Interactive**	**1,584** (± 0,215)	**0:11:28**	-
	Non-Interactive	3,104 (± 1,750)	0:23:18	2/5

Table 2: Comparison between interactive and non-interactive method for `mo-halfcheetah-v5` environment

Targets $[x_1, x_2]$	Method	Avg Steps ($\cdot 10^6$) ± Std	Avg Time per Run	Timeouts
A: [5, -7]	Interactive	3,845 (± 3,340)	0:50:04	2/5
	Non-Interactive	**1,690** (± 0,210)	**0:20:59**	-
B: [7, -20]	Interactive	6,230 (± 1,958)	1:19:03	4/5
	Non-Interactive	**2,652** (± 2,152)	**0:34:37**	**1/5**
C: [1, -0.5]	**Interactive**	**3,072** (± 2,839)	**0:38:34**	-
	Non-Interactive	6,538 (± 2,715)	1:24:54	4/5

`us-v0`. These results are from the non-normalized reward experiment, and we observe that the final policies are more widely spread across the objective space for the interactive method. Interestingly, as in the mountain car environment, the final solutions selected by E-NAUTILUS for the balanced preference and for objective x_1 are the same policy.

In this environment, variation between runs with different seeds is more pronounced. We assume this is often due to whether the algorithm's exploration gets trapped in a local maximum or not. When the artificial user prioritizes a single objective, we observe that E-NAUTILUS tends to have policies that dominate our interactive MORL method on x_1, while our interactive method finds higher-performing solutions for x_2, and also has solutions that are more towards the high-valued x_2 region. For the balanced preference, however, E-NAUTILUS+PGMORL generally dominates the solutions produced by our interactive method.

In summary, neither method consistently dominates the other across all scenarios. However, when the artificial user "properly" selects solutions that closely align with their stated preferences, the interactive method is more likely to produce outcomes that lie within the desired region of the objective space compared to E-NAUTILUS+PGMORL. This highlights the strength of the interactive approach in aligning with specific user preferences. On the other hand, as discussed in previous subsections, the

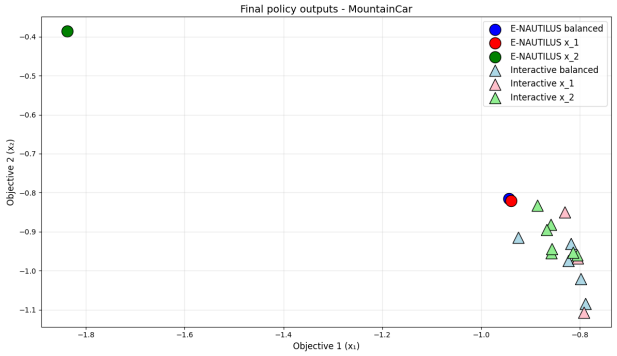


Figure 9: Final Policy Outputs of MountainCar Environment

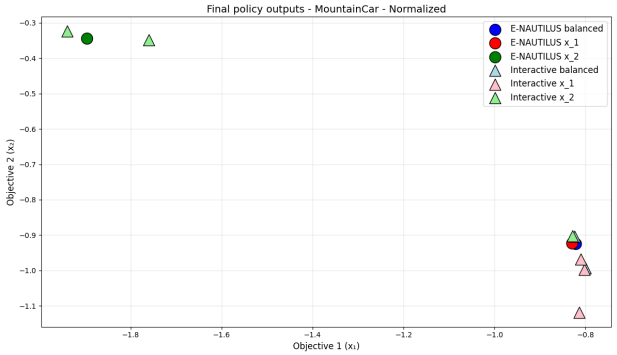


Figure 10: Final Policy Outputs of MountainCar - Normalized

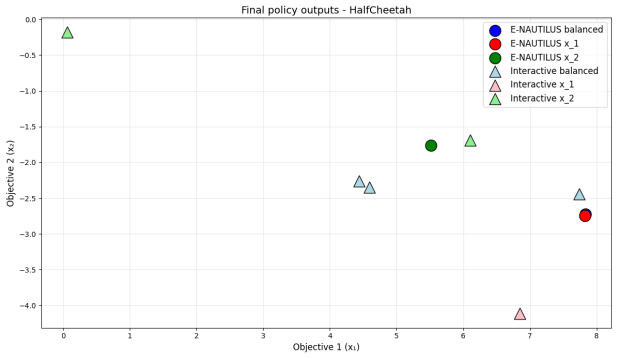


Figure 11: Final Policy Outputs of HalfCheetah Environment

interactive method is more prone to getting trapped in local optima compared to the non-interactive approach, primarily due to its more constrained exploration space, caused by the bounds, which limit its ability to discover diverse solutions.

6 FUTURE WORK

Currently, our integration of interactivity remains relatively simple, and there is significant room for further improvements. For instance, we have only tested linear utility functions and environments with largely linear Pareto fronts. An interesting future direction is to investigate whether interactive preference guidance can enable the discovery of non-linear trade-off solutions, particularly in environments that are typically unreachable using linear scalarization approaches.

Additionally, the current PGMORL implementation supports only two continuous objectives. Extending this to higher-dimensional objective spaces would be valuable for understanding how well the interactive method scales and performs in more complex decision scenarios.

Another limitation we observed is that the algorithm tends to become trapped in local optima, an issue that becomes more pronounced when the search space is constrained by user preferences. To address this, future work could incorporate a dynamic relaxation of the preference bounds, activated when the algorithm detects a prolonged stagnation in policy improvement.

One aspect we were unable to investigate in this work is the impact of human fatigue. Frequent interaction with the algorithm over long training periods can be demanding for DMs. It would be valuable to explore methods for reducing the frequency of required interactions, either through predictive modeling of user preferences or by strategically scheduling queries, in order to minimize the cognitive load on the human user.

6.1 Incorporate Interactivity into MORL algorithms

Our work primarily introduces a method to incorporate user interactivity into MORL algorithms, guiding the search toward preferred regions of the solution space. So far, we have implemented this by applying bounds to the evolutionary search component of the MORL algorithm. We believe this concept is generalizable to any MORL method that relies on population-based optimization and individual selection. Using interactive selection for an individual in the population allows the evolutionary method to generate solutions that are similar to the selected solutions. For other classes of MORL algorithms, different forms of interactivity could be explored, for example, incorporating a target point and/or changing the reward structure, as proposed by Vamplew et al. [44]. Many current multi-policy methods use a selection model that typically chooses the next policy to train based on the maximization of coverage metrics like hypervolume or diversity. To make MORL interactive, the user should be able to intervene during this policy selection step. However, for some algorithms, this can be difficult since user intervention will often conflict with the optimal policy selection algorithm, resulting in slower convergence or reduced hypervolume efficiency. In most cases, non-interactive methods will outperform purely interactive ones in terms of the quality of the final solutions or training durations.

Consider the Generalized Policy Improvement (GPI) MORL from Alegre et al. [3]. The method allows for a sample-efficient learning by identifying the optimal weight vector to train on using GPI. The selection method relies on what they define as *corner weights*. Corner weights are weight vectors where two or more policies share the same value when calculated using the utility function. The corner weight with the most promising improvement is then selected to train. If we want to extend this algorithm to make it interactive, a suggestion is to adjust the selection model so that the user is able to select their preferred corner weight to train, rather than using GPI. It is important to recognize that corner weights are to maximize hypervolume gains; Selecting preferred corner weights might not always help the algorithm converge towards the user's preferred region. Since corner weights tend to be more concentrated in areas where the policies can be improved the most.

7 CONCLUSION

This work provides valuable insights into the novel research area of interactive Multi-Objective Reinforcement Learning (MORL). We argue that interactive MORL is a promising direction for the future of reinforcement learning, especially given that many complex real-world problems are inherently multi-objective and often require human-in-the-loop decision-making. Incorporating user feedback during the learning process can accelerate convergence toward preferred solutions and reduce the cognitive burden of manually selecting solutions from the final output of a posteriori multi-policy MORL algorithms.

In this paper, we introduced the concept of Interactive Preference-Guided MORL, where the search space of the algorithm is dynamically constrained based on the user's previously selected solution. This bounding approach limits the exploration of the policy space to regions of interest, thereby aligning the search more closely with the decision-maker's preferences. We also proposed several evaluation strategies by adapting ideas from existing literature, including the use of artificial user utility functions, measuring the time required to reach a target solution, and comparing final outputs between interactive and non-interactive methods.

Our experimental results show that using a bounded search space leads to a greater concentration of solutions in the user's preferred region and allows the interactive method to find a satisfactory solution more quickly than its non-interactive counterpart. However, the constrained exploration can also make the algorithm more susceptible to getting trapped in local optima. This limitation suggests potential for future work on dynamically relaxing the search bounds when stagnation is detected.

Overall, we believe our work makes a meaningful contribution to the growing field of interactive MORL by offering both methodological insights and empirical evidence, laying a foundation for further research on interactive MORL.

8 ACKNOWLEDGMENTS

I would like to thank my daily supervisor, Zuzanna Osika for the guidance throughout my thesis period and Pradeep Murukannaiah for his valuable feedback on many aspects of this research. Lastly, I would like to express my gratitude to the Delft High Performance Computing Centre (DHPC) [15], for allowing me to use the hardware provided to run the experiments.

A HYPERPARAMETERS

Table 3: Hyperparameters used for each environment.

Experiment	Env	Ref Point	Origin	# Parallel Envs	Pop Size	Warmup	Evol. Iters	Steps/Episode	Global Steps	Gamma
MountainCar	mo-mountaincarcontinuous-v0	[-110, -110]	[-110, -110]	4	6	10	10	1000	2.5M	0.995
Threshold MountainCar	mo-mountaincarcontinuous-v0	[-110, -110]	[-110, -110]	4	6	10	10	1000	5M	0.995
HalfCheetah	mo-halfcheetah-v5	[-100, -100]	[-100, -100]	4	6	40	40	400	5M	0.99
Threshold HalfCheetah	mo-halfcheetah-v5	[-100, -100]	[-100, -100]	4	6	40	40	400	7.5M	0.99

Table 4: Description of hyperparameters used in all experiments.

Parameter	Description
`env`	MO-Gym ID of the multi-objective environment used
`ref_point`	Reference point in the objective space
`origin`	Origin point in the objective space during evaluation (usually same as ref_point)
`parallel_envs`	Number of parallel environments for experience collection
`pop_size`	Number of agent-weight combinations that are trained during each generation
`warmup_iterations`	Initial warm-up iterations to get the policies out of low-performance region
`evolutionary_iterations`	Number of training iterations for each generation
`steps_per_episode`	Maximum number of environment steps per episode
`global_steps`	Total number of environment steps used for training
`gamma`	Discount factor used in return calculation

The rest of the hyperparameters are the default PGMORL hyperparameters, which can be found in our code on GitHub[2]. The random seeds we used for our experiments in Section 5.1 and 5.3 are 1, 42, and 100. For the threshold experiments in Section 5.2, we used 5 different seeds, which are 1, 7, 10, 42, and 100.

[2]https://github.com/Henweiz/interactive_MORL

REFERENCES

[1] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic weights in multi-objective deep reinforcement learning. In *International conference on machine learning*. PMLR, 11–20.

[2] Bekir Afsar, Kaisa Miettinen, and Francisco Ruiz. 2021. Assessing the performance of interactive multiobjective optimization methods: A survey. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–27.

[3] Lucas N Alegre, Ana LC Bazzan, Diederik M Roijers, Ann Nowé, and Bruno C da Silva. 2023. Sample-efficient multi-objective learning via generalized policy improvement prioritization. *arXiv preprint arXiv:2301.07784* (2023).

[4] R_ Benayoun, J De Montgolfier, Jo Tergny, and O Laritchev. 1971. Linear programming with multiple objective functions: Step method (STEM). *Mathematical programming* 1, 1 (1971), 366–375.

[5] Jürgen Branke. 2008. Consideration of partial user preferences in evolutionary multiobjective optimization. *Multiobjective optimization: Interactive and evolutionary approaches* (2008), 157–178.

[6] Jürgen Branke and Kalyanmoy Deb. 2005. Integrating user preferences into evolutionary multi-objective optimization. In *Knowledge incorporation in evolutionary computation*. Springer, 461–477.

[7] Jürgen Branke, S Greco, R Słowiński, and P Zielniewicz. 2010. Interactive evolutionary multiobjective optimization driven by robust ordinal regression. *Bulletin of the Polish Academy of Sciences: Technical Sciences* 3 (2010).

[8] Jürgen Branke, Salvatore Greco, Roman Słowiński, and Piotr Zielniewicz. 2014. Learning value functions in interactive evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 19, 1 (2014), 88–102.

[9] John Telfer Buchanan. 1997. A naive approach for solving MCDM problems: The GUESS method. *Journal of the Operational Research Society* 48, 2 (1997), 202–206.

[10] Vira Chankong and Yacov Y Haimes. 2008. *Multiobjective decision making: theory and methodology*. Courier Dover Publications.

[11] Kalyanmoy Deb and Abhay Kumar. 2007. Light beam search based multi-objective optimization using evolutionary algorithms. In *2007 IEEE congress on evolutionary computation*. IEEE, 2125–2132.

[12] Kalyanmoy Deb and Kaisa Miettinen. 2009. Nadir point estimation using evolutionary approaches: better accuracy and computational speed through focused search. In *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems: Proceedings of the 19th International Conference on Multiple Criteria Decision Making, Auckland, New Zealand, 7th-12th January 2008*. Springer, 339–354.

[13] Kalyanmoy Deb, Karthik Sindhya, and Jussi Hakanen. 2016. *Multi-objective optimization*. CRC Press, 161–200.

[14] Kalyanmoy Deb, Ankur Sinha, Pekka J Korhonen, and Jyrki Wallenius. 2010. An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE Transactions on Evolutionary Computation* 14, 5 (2010), 723–739.

[15] Delft High Performance Computing Centre (DHPC). 2024. DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2.

[16] Florian Felten, Lucas N. Alegre, Ann Nowé, Ana L. C. Bazzan, El Ghazali Talbi, Grégoire Danoy, and Bruno Castro da Silva. 2023. A Toolkit for Reliable Benchmarking and Research in Multi-Objective Reinforcement Learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.

[17] Florian Felten, El-Ghazali Talbi, and Grégoire Danoy. 2024. Multi-Objective Reinforcement Learning based on Decomposition: A taxonomy and framework. *Journal of Artificial Intelligence Research* 79 (2024), 679–723.

[18] Arthur M Geoffrion, James S Dyer, and A Feinberg. 1972. An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management science* 19, 4-part-1 (1972), 357–368.

[19] Dunwei Gong, Xinfang Ji, Jing Sun, and Xiaoyan Sun. 2014. Interactive evolutionary algorithms with decision-maker s preferences for solving interval multi-objective optimization problems. *Neurocomputing* 137 (2014), 241–251.

[20] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, and Fredrik Heintz. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems* 36, 1 (2022), 26.

[21] Sandra Huber, Martin Josef Geiger, and Marc Sevaux. 2015. Simulation of preference information in an interactive reference point-based method for the bi-objective inventory routing problem. *Journal of Multi-Criteria Decision Analysis* 22, 1-2 (2015), 17–35.

[22] Andrzej Jaszkiewicz and Jürgen Branke. 2008. Interactive multiobjective evolutionary algorithms. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer, 179–193.

[23] Mariano Luque, Francisco Ruiz, and Kaisa Miettinen. 2011. Global formulation for interactive multiobjective optimization. *Or Spectrum* 33 (2011), 27–48.

[24] Kaisa Miettinen. 1999. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science Business Media.

[25] Kaisa Miettinen, Petri Eskelinen, Francisco Ruiz, and Mariano Luque. 2010. NAUTILUS method: An interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research* 206, 2 (2010), 426–434.

[26] Kaisa Miettinen, Jussi Hakanen, and Dmitry Podkopaev. 2016. Interactive nonlinear multiobjective optimization methods. *Multiple criteria decision analysis: State of the art surveys* (2016), 927–976.

[27] Kaisa Miettinen and Marko M Mäkelä. 2006. Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research* 170, 3 (2006), 909–922.

[28] Kaisa Miettinen and Marko M Mäkelä. 2002. On scalarizing functions in multiobjective optimization. *OR spectrum* 24 (2002), 193–213.

[29] Kaisa Miettinen, Francisco Ruiz, and Andrzej P Wierzbicki. 2008. Introduction to multiobjective optimization: interactive approaches. In *Multiobjective optimization: interactive and evolutionary approaches*. Springer, 27–57.

[30] Andrew William Moore. 1990. *Efficient Memory-based Learning for Robot Control*. Technical Report. University of Cambridge.

[31] Simone Parisi, Matteo Pirotta, and Jan Peters. 2017. Manifold-based multi-objective policy search with sample reuse. *Neurocomputing* 263 (2017), 3–14.

[32] Roxana Rădulescu, Patrick Mannion, Diederik M Roijers, and Ann Nowé. 2020. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems* 34, 1 (2020), 10.

[33] Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé. 2022. Pareto conditioned networks. *arXiv preprint arXiv:2204.05036* (2022).

[34] Diederik M Roijers, Denis Steckelmacher, and Ann Nowé. 2018. Multi-objective reinforcement learning for the expected utility of the return. In *Proceedings of the Adaptive and Learning Agents workshop at FAIM*, Vol. 2018.

[35] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48 (2013), 67–113.

[36] Diederik M Roijers, Luisa M Zintgraf, and Ann Nowé. 2017. Interactive thompson sampling for multi-objective multi-armed bandits. In *International conference on algorithmic decision theory*. Springer, 18–34.

[37] Ana B Ruiz, Karthik Sindhya, Kaisa Miettinen, Francisco Ruiz, and Mariano Luque. 2015. E-NAUTILUS: A decision support system for complex multiobjective optimization problems based on the NAUTILUS method. *European Journal of Operational Research* 246, 1 (2015), 218–231.

[38] Francisco Ruiz, Mariano Luque, and Kaisa Miettinen. 2012. Improving the computational efficiency in a global formulation (GLIDE) for interactive multiobjective optimization. *Annals of Operations Research* 197 (2012), 47–70.

[39] Lamjed Ben Said, Slim Bechikh, and Khaled Ghédira. 2010. The r-dominance: a new dominance relation for interactive evolutionary multicriteria decision making. *IEEE transactions on Evolutionary Computation* 14, 5 (2010), 801–818.

[40] Ralph E Steuer and Eng-Ung Choo. 1983. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical programming* 26 (1983), 326–344.

[41] Richard S Sutton, Andrew G Barto, et al. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[42] Hideyuki Takagi. 2001. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proc. IEEE* 89, 9 (2001), 1275–1296.

[43] Tetsuzo Tanino. 1993. An interactive multicriteria decision making method by using a genetic algorithm. In *Proc. of International Conference on System Science & System Engineering*. 381–386.

[44] Peter Vamplew, Rustam Issabekov, Richard Dazeley, Cameron Foale, Adam Berry, Tim Moore, and Douglas Creighton. 2017. Steering approaches to Pareto-optimal multiobjective reinforcement learning. *Neurocomputing* 263 (2017), 26–38.

[45] Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. 2013. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*. IEEE, 191–199.

[46] Tobias Wagner and Heike Trautmann. 2010. Integration of preferences in hypervolume-based multiobjective evolutionary algorithms by means of desirability functions. *IEEE Transactions on Evolutionary Computation* 14, 5 (2010), 688–701.

[47] Hongze Wang. 2024. Multi-objective reinforcement learning based on nonlinear scalarization and long-short-term optimization. *Robotic Intelligence and Automation* 44, 3 (2024), 475–487.

[48] Paweł Wawrzyński. 2009. A cat-like robot real-time learning to run. In *International Conference on Adaptive and Natural Computing Algorithms*. Springer, 380–390.

[49] Andrzej P Wierzbicki. 1980. The use of reference objectives in multiobjective optimization. In *Multiple criteria decision making theory and application: Proceedings of the third conference Hagen/Königswinter, West Germany, August 20–24, 1979*. Springer, 468–486.

[50] Xi Xiong, Jianqiang Wang, Fang Zhang, and Keqiang Li. 2016. Combining deep reinforcement learning and safety based control for autonomous driving. *arXiv preprint arXiv:1612.00147* (2016).

[51] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. 2020. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International conference on machine learning*. PMLR, 10607–10616.

[52] Stanley Zionts and Jyrki Wallenius. 1976. An interactive programming method for solving the multiple criteria problem. *Management science* 22, 6 (1976), 652–663.