# Multi-Objective Reinforcement Learning

*Pranav Gupta*


4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2024

# Abstract

Multi-Objective Reinforcement Learning (MORL) addresses situations with two or more objectives, making it crucial for real-world decision-making tasks. However, evaluating algorithms in benchmarking environments often reveals challenges, including reward function hindering agent learning. This study focuses on the Multi-Objective Mountain Car problem, a crucial benchmarking environment plagued by the local optima problem, leading to the agent behaving unintendedly. The reward shaping techniques are successfully implemented and two distinct reward functions, Time-Speed and Movement-Speed, are proposed and thoroughly analyzed. Evaluation includes estimating and examining the Pareto graphs and assessing the objective balancing and correlation analysis during training to understand their impact on the agent's behaviour. The experiments conducted reveal valuable insights applicable to the wider MORL community. It indicates that reward shaping is a viable strategy for multi-objectivizing reinforcement learning problems and that weights to the different objectives through scalarization can significantly influence agent performance. Ultimately, a condensed three-objective reward function, Time-Movement-Speed, is proposed to test the generality of MORL systems. Future work includes finding the non-convex front of the reward functions and adding stochasticity to the state transitions to improve generalizability.

# Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Pranav Gupta*)

# Acknowledgements

I extend a special thank you to my supervisor, Dr. Michael Herrmann for his invaluable guidance and support throughout this journey. I express gratitude to Ebtehal Alotaibi for her insightful suggestions. Heartfelt thank you to my parents for their unwavering love and encouragement, and without whom I could not have made these 4 years possible. Lastly, I would like to thank my friends for always reminding me of the bigger picture.

# Table of Contents


**1 Introduction** **1**
- 1.1 Motivation . . . 1
- 1.2 Contribution . . . 2
- 1.3 Outline . . . 2

**2 Background** **3**
- 2.1 Benchmarking . . . 3
- 2.2 Sparse Rewards . . . 4
- 2.3 Designing of a Reward Signal . . . 5
- 2.4 Evaluating Multi-Objective Reinforcement Learning . . . 5
- 2.5 Pareto Optimality . . . 6
- 2.6 Related Work . . . 6

**3 Methodology** **8**
- 3.1 Multi-Objective Mountain Car . . . 8
  - 3.1.1 Environment Details . . . 8
  - 3.1.2 Sparse Rewards . . . 10
- 3.2 Local Optima Problem . . . 10
- 3.3 Libraries and Evaluation Frameworks . . . 13
  - 3.3.1 Stable Baselines3 . . . 13
  - 3.3.2 Multi-Objective Gymnasium . . . 13
  - 3.3.3 Google Colaboratory . . . 13
- 3.4 Deep Q-Network . . . 13
- 3.5 Scalarization . . . 14
- 3.6 Pareto Optimality . . . 15
- 3.7 Designing of a Reward Signal . . . 15

**4 Experiments** **16**
- 4.1 Experiment Conditions . . . 16
  - 4.1.1 Hyper-parameter Explanation . . . 16
- 4.2 Time-Speed . . . 17
  - 4.2.1 Configuration . . . 17
  - 4.2.2 Pareto Graph . . . 18
  - 4.2.3 Effect of Reward Shaping . . . 19
  - 4.2.4 Optimal vs Non-Optimal Solutions . . . 20
  - 4.2.5 Multi-Objective Optimization . . . 23

4.2.6 Longer Training for Finding Threshold . . . 24
4.2.7 Trajectory Based Analysis . . . 24
4.2.8 Learning and Summary . . . 25
4.2.9 Potential Limitations and Future Developments . . . 26
4.3 Movement-Speed . . . 27
4.3.1 Rationale behind the Reward Function . . . 27
4.3.2 Configuration . . . 27
4.3.3 Multi-Objective Optimization and Reward Shaping . . . 27
4.3.4 Training Comparison . . . 28
4.3.5 Trajectory Based Analysis . . . 30
4.3.6 Learnings and Summary . . . 32
4.3.7 Potential Limitations . . . 32

**5 Results and Discussion** **33**
5.1 Comparison between the Reward Functions . . . 33
5.2 Three Objective Case . . . 33
5.3 General Trends . . . 34
5.4 Critical Analysis and Limitations . . . 35

**6 Conclusion** **37**
6.1 Future Work . . . 38

**Bibliography** **39**

# Chapter 1

# Introduction

## 1.1 Motivation

"Multi-Objective Reinforcement Learning" (MORL) is a branch of reinforcement learning where the objectives can be complementary, conflicting and even independent of each other. Owing to the great promise it has shown in artificial problem sets, MORL has been applied to real-world use cases. The reason being most real-world decision-making tasks require complex trade-offs between various conflicting objectives (Hayes et al., 2022). Though, before the algorithms designed for multi-objective problems can be applied to these real-world use cases, their performance is evaluated on benchmarking environments. However, the environment may sometimes possess reward functions which impede learning or induce unintended behaviour. Hence, analyzing the agent within this context would not provide any valuable insight into how the agent would truly behave in that environment. In such situations, the reward function may require a re-design to remedy the problem, which is a trial-and-error process (Sutton & Barto, 2018). The designer should have a good understanding of the environment and find the source of the problem due to which the agent is not behaving as intended. To ensure that the changes made do not lead to any other unintended consequences, the changed reward function is thoroughly evaluated. The Multi-Objective Mountain Car is an environment which suffers from the agent behaving unintendedly owing to the reward function. The agent converges sub-optimally to the local optima. The environment is part of the Multi-Objective Gymnasium, which is a very popular framework for evaluating algorithms (Felten et al., 2023). Therefore, it is of utmost importance that a viable reward function be found for the environment. Additionally, redesigning the reward enables us to draw conclusions which benefit the wider MORL community. Knox et al. (2023) mention that reward design is considered universally a challenging task, especially for individuals without substantial experience in the domain of reinforcement learning. Therefore, we decided to take up the challenge and provide alternative reward functions for the Multi-Objective Mountain Car.

## 1.2 Contribution

This work highlights the characteristics of the Multi-Objective Mountain Car problem and utilises them to make changes to the reward function. The local optima problem, mentioned above, arises due to a lack of positive rewards. Therefore, the positive rewards are propagated into the agent's behaviour through a technique called Reward Shaping. We analyse the effect of reward shaping and draw conclusions for the general multi-objective reinforcement learning domain. Since the Mountain Car is originally a single objective problem, we aim to conclude whether reward shaping is a viable technique to multi-objectivize problems. Scalarization provides priorities to different objectives to propagate preference the researcher may have, to the agent. Therefore, the evaluation is conducted to measure the change in performance when the priorities are changed, in order, to draw conclusions on the scalarization technique being used in MORL. We aim to show a Pareto Front (optimal set of solutions) for the reward functions proposed. We offer evaluations at the training and trajectory (agent's performance in a singular episode of the environment) levels to ensure that the agent is behaving as intended and we compare the reward functions to show their advantages and disadvantages. All evaluations have been completed by training on the Deep Q-Network under identical hyper-parameters to ensure the results are comparable. Ultimately, the aim is to able to contribute to the MORL domain at a general level, and the reward re-designing task provides us with the tools for it.

## 1.3 Outline

In Chapter 2, we look into the literature surrounding key concepts and characteristics relevant to this work. Furthermore, we discuss the research that has been conducted to tackle problems with our chosen environment.

In Chapter 3, we introduce the Multi-Objective Mountain Car environment and the local optima problem, detail the algorithms and techniques employed in designing and evaluating the suggested reward functions.

In Chapter 4, we explain the rationale behind each proposed reward function. Additionally, we showcase the estimated Pareto Front (if applicable) for the reward function and analyze both the training and evaluation stages.

In Chapter 5, we present a comparative analysis of the reward function characteristics, along with broader trends in the field of multi-objective reinforcement learning. Moreover, a three-objective case of the problem is explored to improve generality.

Lastly, Chapter 6 offers a summary of the work and its achievements. It also outlines suggestions for future research directions.

# Chapter 2

# Background

Reinforcement Learning (RL) is a subset of machine learning, which comprises of different ways of enhancing performance through trial-and-error experiences, it has its roots in experimental psychology (Barto, 1997). Deep Mind had a significant impact in the RL domain when their paper was the first to successfully learn control policies from high dimensional sensory information, more precisely visual information, using reinforcement learning. There was no prior knowledge that the model had been given about the inputs or the environments (Mnih et al., 2015). The policies were developed for Atari games, an artificial domain which has environments designed to be challenging for human players (Mnih et al., 2015). The RL field has shown promise across numerous artificial domains and is beginning show success in real world problems and scenarios (Dulac-Arnold et al., 2021). These are currently, but not limited to, routing optimization (Almasan et al., 2022), big data-based decision-making in financial industries (Singh et al., 2022), and autonomous vehicles (Pérez-Gil et al., 2022) amongst many others.

## 2.1 Benchmarking

To evaluate the progress of the field, the researchers need good benchmarks to facilitate comparison and enable thorough performance testing (Brockman et al., 2016). Research in many parts of reinforcement learning is yet to be standardized, leading to many researchers having to experiment with self-designed environments leading to non-reproducible research (Felten et al., 2023). This has led to researchers often questioning the viability and results of many important and influential papers (Agarwal et al., 2021). Good benchmarks can accelerate the development of various up and coming fields of reinforcement learning like Multi-Objective Deep Reinforcement Learning which can utilized for solving problems with complex multi-objectives (T. T. Nguyen et al., 2020).

In a multi-objective setting, the goals can be complementary, conflicting and even independent from each other. It is usually formalized as a Multi-Objective Markov Decision Process (MOMDP), though it has complex forms, Multi-Objective Partially Observable Markov Decision Processes and Multi-Objective Multi-Agent systems (Hayes et al., 2022). Multi-objectivization of a problem can lead to both the speed of convergence and the quality of the policy being improved, in comparison to its single

objective case (Brys et al., 2014). The applications include water management, where there are significant competing objectives related to socioeconomic factors (Castelletti et al., 2008), military purchasing, where the time and cost alongside the raw material required in manufacturing different weapons and military vehicles are included in the various objectives a government might have (M. T. Nguyen & Cao, 2019) and cyber security, where network security agents have to balance many goals when choosing what steps to take to defend against attackers. To enable result replication and improve the standard of conducting and evaluating research, multi-frameworks have been suggested that incorporate benchmarks to enable comparisons between algorithms being developed in this space (Felten et al., 2023), (Zhu et al., 2023). The frameworks generally have baselines, which are the most prevalent and were suggested by researchers on a hypothesis. For example, the suggestion of converting the benchmarking problem, the Mountain Car problem to its multi-objective counterpart was to test the generality of Multi-Objective Reinforcement Learning systems (Vamplew et al., 2010). However, no evaluation was conducted or results mentioned with any current benchmarking algorithms. It is a limitation since the performance of benchmark environments is usually analyzed by researchers by running a variety of algorithms with comparable hyperparameters and identical hardware resources provided to each (Wang et al., 2020), (Duan et al., 2016). Other ways of understanding the effect of the environment settings have been plotting graphical fronts (Barrett & Narayanan, 2008) and objective reward over time (Gábor et al., 1998). However, the lack of numerical measurements and over-reliance on graphical representations has caused a challenge in comparing performance accurately (Vamplew et al., 2010). The paper did mention that upon better understanding the factors affecting the performance, the benchmarking algorithms suggested could be further developed to include a vast range of characteristics not considered yet. There is a focus on developing frameworks and agents which can be generalized through a change in the reward function (Badia et al., 2020).

## 2.2 Sparse Rewards

A Sparse Reward signal is a reward function where most of the rewards received by the agent are non-positive. In such situations, it may be impossible for the agent to detect progress and “wander aimlessly" for the majority of the time due to the agent being unable to connect the actions being taken to a distant future positive reward. This condition was labelled as the “plateau problem" (Minsky, 1995). It has been a long-standing problem in the field (Arjona-Medina et al., 2019) and in many cases, the agent cannot find that positive future reward and therefore unable to reach the goal state or perform the required task (Hare, 2019). Sutton and Barto (2018) mention that designers of experiments often find it tempting to reward the agent for achieving sub goals, but that might lead the agent to behave differently than as intended and not achieve the goal at all. Reward shaping is a method in which the reward signal is changed by adding additional training rewards to guide the agent’s learning process (Ng et al., 1999). It is equivalent to the agent being provided a more supported environment. (Gupta et al., 2022) showed that the technique can prune significant portions of the state space and give a task related direction to the agent’s optimism. Reward shaping has been able to speed up the reinforcement learning process due to the use of additional domain-based

heuristic knowledge in the reward function. It can be utilized to multi-objectivize single objective problems to lead the agent to identify good actions along with the faster convergence (Brys et al., 2014). The research in recent years has focused on developing and analysing learning algorithms and it is felt that it is time to change focus to how reward signals are treated (Eschmann, 2021).

## 2.3 Designing of a Reward Signal

The task of designing reward signals, also known as reward functions, is part of the general task of designing performance metrics for optimization problems; therefore, its importance goes beyond reinforcement learning (Knox et al., 2023). Sutton and Barto (2018) mention that practically, designing a reward function is an informal trial-and-error process, which requires the environment developer to constantly make changes. Changes are made in scenarios where the agent is failing to learn, learning too slowly or learning unintended experiences. An important thing to look out for is that agents can discover ways to make the environments deliver rewards in unintended ways and this is a critical challenge in the domain Sutton and Barto (2018). Such a situation can have a considerable impact in society and generally arise due to poor design of experiment conditions (Amodei et al., 2016). Scenarios in which unintended and harmful behaviour is observed by the agent when the designer has outlined the wrong objective function is called accidents. One of the reasons for an “accident" to arise is due to “negative side effects". It is a condition in which the designer of the experiment is focused on achieving a specific task in the environment but ignore certain characteristics of the environment and therefore, expressing indifference over what aspects maybe harmful to change. Another important factor to consider is "Safe Exploration", which tries to prevents the agent from taking actions which could have negative or irreparable consequences (Amodei et al., 2016).

## 2.4 Evaluating Multi-Objective Reinforcement Learning

The application of multi-objective reinforcement learning in the real world has led to different approaches being tried by researchers to the problem of this nature. A common approach is to cast the different objections in a single, scalar, addition-based reward function which can combine the important aspects. The generated policy is evaluated and the reward function is redesigned if the actions adopted by the agent are unsatisfactory (Hayes et al., 2022). To understand the non-linear relation between the multi-objective settings, the Chebyshev scalarization was adopted (Van Moffaert et al., 2013). Scalarizations have a limitation in that the reward function is decided beforehand and therefore, there is a bit of guesswork involved in how the policy would behave (Hayes et al., 2022). Though, the scalarization function provide a single score, calculated after aggregating the various objectives in the multi-objective problem, which can be easily ordered and compared to understand the agent’s performance (Van Moffaert et al., 2013).

## 2.5 Pareto Optimality

Multi-Objective Optimization Problems lack a unique solution and instead have a set of solutions which involve trade-offs and ultimately, balancing the different objectives (Ngatchou et al., 2005). The set of optimal solutions is known as the Pareto Optimal set, therefore it is a set of “non-inferior” solutions, which together define a boundary in the objective space, beyond which either objective cannot be improved unless the another objective is sacrificed (Alhammadi & Romagnoli, 2004). It allows the decision maker to make an informed choice by having a wide range of solutions to choose from and therefore understanding the consequences of the decisions taken, on the different objectives. Additionally, the pareto front can help in understanding how the optimization of one objective can affect the performance of the second objective setting. Such sets have many applications in the real world such as waste-water management (Alvarez-Vázquez et al., 2008), structural design in engineering (Gobbi et al., 2014), energy markets (Lee et al., 2024).

The Pareto Front can also be defined as a set of non dominated policies, such that for each policy in the set, there is no other policy which achieves a reward equally or higher in each objective (Hayes et al., 2022). For stochastic policies, the Pareto Front can have infinitely many policies, though it is guaranteed to be convex (Roijers & Whiteson, 2017). Contrastingly, deterministic policies, which are preferred for interpretabilty and safety reasons, can have irregular shapes (Röpke et al., 2024). Pareto Front and Pareto Set are terminologies part of the Pareto optimality and maybe used interchangeably in this work. They refer to the set of non-dominated solutions, which are superior to all other solutions in the search space (Akbari et al., 2014).

## 2.6 Related Work

Bechtle et al. (2019) worked on overcoming the local optima problem by leveraging the potential of curiosity in a new framework, which supports iterative risk-seeking by using a linear quadratic system and tries to reduce the uncertainty in a dynamic model. The paper showcases the need for a change in how the environment is structured regarding the reward provided. However, the solution was only tested in a single objective setting and it is unclear if it can generalize well to the multi objective setting.

Experiments conducted by Cheng (2022) introduced a fourth step to the action space of the Multi-Objective Mountain Car, though it was mentioned as "no matchset" since it did not have a consistent result when taken. The lack of clarity over this action makes it very hard to replicate the results for a different framework, and therefore, the lack of reproducibility of the solution is a very important limitation. Additionally, the continuous transitions of the state variables, Speed and position, are discretized and provided integer values instead, which means assumptions were made over the velocity and position gained for each action, instead of the value being calculated by the inner dynamics of the problem.

T. T. Nguyen et al. (2020) evaluated the Multi-Objective Mountain Car environment and showed the rewards received by each objective in various linear scalarizations, though

no observations were made on whether the agent was able to reach the goal state. The paper was evaluating a framework that they had created for running the problem, similar to the Farama Foundations Multi-Objective Gymnasium (Felten et al., 2023), and the scalarizations provided were quite a few, namely six. Therefore, conclusions drawn are focused on the effectiveness of the framework rather than the environment.

# Chapter 3

# Methodology

We introduce the Multi-Objective Mountain Car environment and its characteristics, the Deep Q-Network algorithm used in this work, the MO-Gymanisum framework and libraries used for evaluation, and hyperparameter conditions under which the experiments have been run. Additionally, we highlight the importance of the Pareto optimality in multi-objective reinforcement learning and how it was achieved in this work.

## 3.1 Multi-Objective Mountain Car

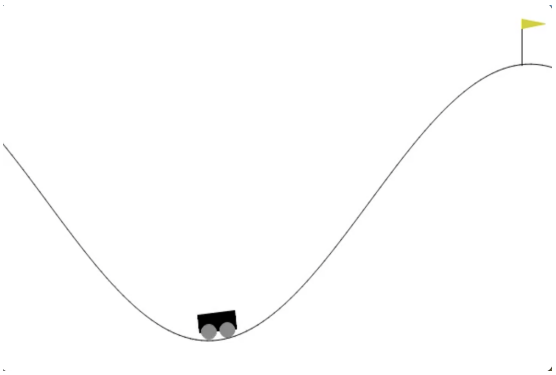


Figure 3.1: A frame from the Mountain Car environment from the MORL Gymnasium created by Felten et al. (2023). The car is stuck at the bottom of the valley and needs to find a trajectory to reach the goal state, indicated by the yellow flag on the mountain on the right. The left side of the environment has a virtual backboard to prevent the car from going too far back.

### 3.1.1 Environment Details

The Multi-Objective Mountain Car environment is a Markov Decision Process where the agent, the Mountain Car, is stuck at the bottom of a sinusoidal valley and does not possess an engine strong enough to will it to the top of the hill, the goal state. Therefore,

it needs to move backwards to gain adequate potential energy. The initial positioning of the Mountain Car is stochastic, but the result of each action is deterministic. The agent must achieve the goal state while balancing multi-conflicting objectives which are configured through the reward function.

**Reward Function and Objectives**:

- **Time:** The agent receives a reward of -1 for each time step it takes to reach the goal state. Not provided on reaching the goal state.
- **Forward Action:** The agent receives a reward of -1 for each forward action at a time step.
- **Backward Action:** The agent receives a reward of -1 for each backward action at a time step.

The agent has to minimize the negative reward received from all 3 objectives while trying to reach the goal state. The configuration of the environment and its transition dynamics are adapted from Moore (Moore, 1990).

**Start State**: The starting position is a uniform random value between [-0.6,-0.4] and the starting Speed is 0

**Goal State**: The flag at the top of the subsequent hill is the agent's goal state, as can be seen in Figure 3.1

**Length of episode**: The agent has 200 timesteps to reach the goal state. If it is unable to do so, the environment is reset, and the agent starts from a start state.

**Swing:** One back-and-forth motion of the car is characterized as a swing. The swing can either be forwards-backwards or vice-versa.

**Environment Solved**: When the reward accumulated for the time objective is lower than -200 (maximum length of an episode), the agent is considered to have reached the goal state.

**Range of State Variables:** The agent's position is within $[-1.2, 0.6]$ and the agent's velocity is within $[-0.07, 0.07]$ at any time step $t$. The agent's state variable values are clipped if they go beyond the specified range. There is a special case when the agent tries to have a position lower than the lower bound (-1.2), while the position is clipped, the velocity is reinitialized to 0. The reason being there is a virtual backboard in the environment to prevent the agent from backwards at position -1.2, and this backboard is inelastic, therefore the agent loses all its velocity when it hits the backboard.

**Transition Dynamics:** At the end of each action, the position and speed of the agent is calculated using

$$v_{t+1} = v_t + (action_t - 1) \cdot \mathrm{f} - \cos(3 \cdot p_t) \cdot \mathrm{g},$$

and

$$p_{t+1} = p_t + v_{t+1},$$

where $v_t$ and $p_t$ are the speed and position at time $t$, respectively. Furthermore, we have constants

$$\mathrm{f\ (force)} = 0.001, \quad \text{and} \quad \mathrm{g\ (gravity)} = 0.0025.$$

Finally, since the agent can perform 3 different actions at any point in time, we have $action_t$, which can have the following values:

- 0 = Move Backwards;
- 1 = zero-throttle, refereed also as Static; and
- 2 = Move Forwards.

The agent receives its position and speed at each time step $t$ as an observation.

### 3.1.2 Sparse Rewards

The Mountain Car environment suffers from the sparse rewards problem, introduced in Section 2.2. According to the reward function highlighted in Section 3.1.1, the agent only receives negative reinforcement until it reaches the goal state. There are no positive rewards awarded, though if the agent reaches the goal, the episode is cut short. Therefore, it receives a lower negative reward compared to if it did not reach the terminal state. Consequently, care was taken to augment the reward function to support the goal of reaching the top of the hill. The support has been added by implementing reward shaping, the reason for it has been highlighted in Section 3.2.

## 3.2 Local Optima Problem

The Mountain Car environment conditions negatively reward moving backwards and forwards but there is no reward metric for the zero-throttle action. This leads to the agent succumbing to taking the zero-throttle action at each time step, therefore, remaining stagnant at the bottom of the sinusoidal valley and letting the episode terminate. It can be observed in Figure 3.2 that the backward and forward actions converge to 0 as the number of time steps in the training elapse due to the increasing number of them being zero-throttle. Taking the above factors into consideration, the agent's preference to remain stagnant and accumulate the negative time rewards leads it to a local optimum situation in terms of cumulative rewards. It is common for poorly crafted reward functions to be exploited by reinforcement learning agents, leading them to get stuck in a local optimum situation (Devidze et al., 2022).

We tried to see if the local optima problem persists in a simpler environment, therefore tried the two-objective environment setting. The backward and forward actions were condensed into a single objective called movement. The objectives of the multi-objective setting became time and movement.

The zero-throttle action was given no reward metric to retain the original reward function as far as possible.

**Rationale**: The single objective case only has the time reward function and the DQN algorithm which was used for evaluation can reach the goal state, illustrated in Figure 3.3. Therefore, if we assign the majority weight to time in the scalarization, the agent might be able to reach the top with movement as a second objective.

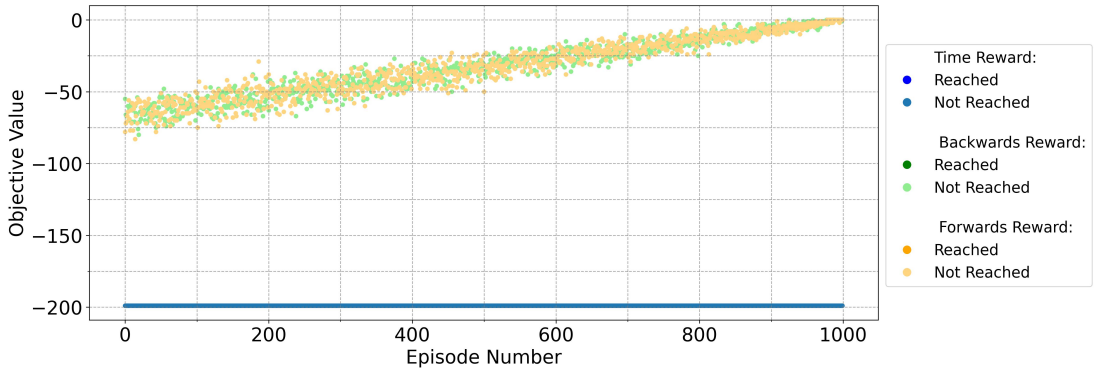



Figure 3.2: The training process of the (0.3, 0.3, 0.4) scalarization for the Time-Backwards-Forwards reward function. The figure shows that the agent slowly and gradually succumbs to the local optima, where all 200 moves are zero-throttle. The rewards received by the forwards and backward objectives are quite similar throughout the training process.

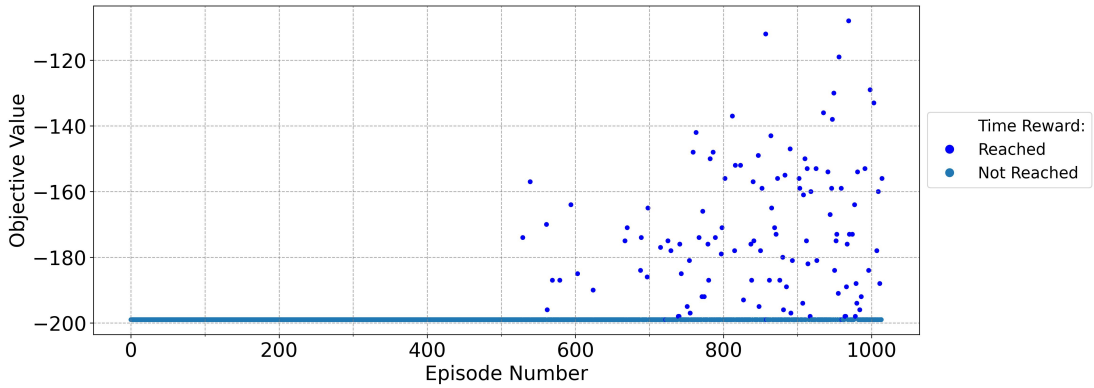



Figure 3.3: The training process of the single objective Mountain Car problem. The agent reaches the goal state in a few episodes, though there are no trends or patterns observed.

Figures 3.4 and 3.5 are with a (0.95, 0.05) scalarization, which means that the priority for the time reward is 19 times as that received for movement. Despite such a skewed scalarization favouring time, it can be observed from Figure 3.4, that at the start of training, the agent tries to explore due to a high epsilon ε and accumulates a high negative total reward for movement, though that slowly and gradually it goes to 0 as the number of training episodes increase. The movement objective moving towards 0 means that the agent is taking fewer steps moving forward or backward and taking more zero-throttle steps instead. The time objective remains at -200, which signifies that the goal state was never reached while training. In Figure 3.5 we observe a circular graph, which is an evaluated episode after training has ceased. Due to the stochastic nature of the problem, the car may not be in the centre at the start. Therefore, the gravity acting on it leads to the circular graph. The action is zero-throttle at all time steps because the agent has succumbed to the local optima state.

**Reasoning with (0.95, 0.05) scalarization**: The reward received at each time step is

-0.95, therefore if the agent takes the zero-throttle step, it only receives a reward of -0.95. While, if the agent adopts the backward or forward movements, the agent receives an additional -0.05 reward, therefore overall receiving a reward of -1. The current reward setup hinders the agent from adopting more exploratory moves to reach the top, due to the zero-throttle step being the course of action at every time step.

**Summary and Learning:** The local optima problem persisted in the Time-Movement reward function and prevented the agent from learning beyond the zero-throttle action and making progress towards the goal state. We can conclude that the reward function in the multi-objective case is not suitable to reach the goal state and therefore needs to be altered to serve as a viable multi-objective benchmarking environment.

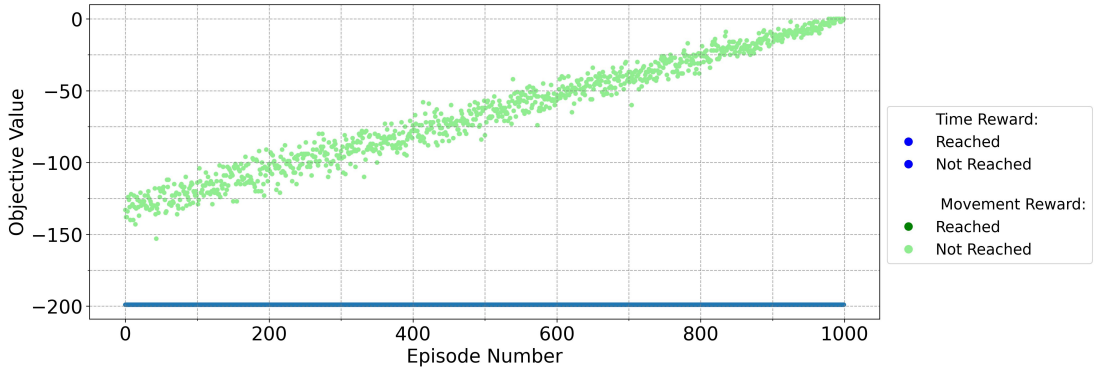



Figure 3.4: The figure shows the training process of the (0.95,0.05) scalarization for the Time-Movement reward function. Like Figure 3.2, the agent succumbs to the local optima at the end of the training process.

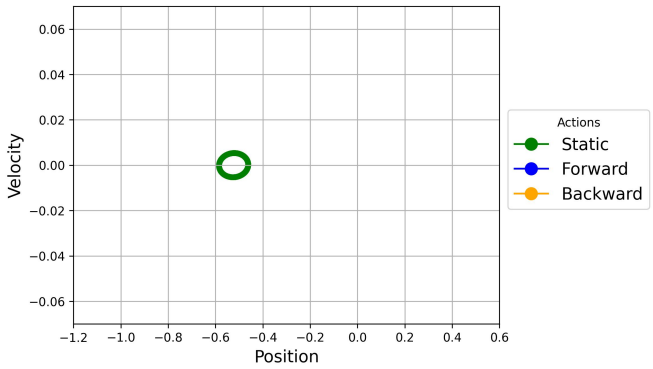



Figure 3.5: Trajectory of the agent trained on (0.95, 0.05) scalarization of the Time-Movement reward function. We observe that all moves taken are the zero-throttle move, also known as static.

**Note:** The graphs shown are for 200,000 training steps to offer a consistent comparison to the graphs in the subsequent reward functions, though the result of the training remains the same even if the agent is trained for 1,000,000 time steps.

## 3.3 Libraries and Evaluation Frameworks

### 3.3.1 Stable Baselines3

Stable Baselines3 is an open-source library in Python consisting of implementations of baseline reinforcement learning algorithms in the PyTorch package (Raffin et al., 2021). The algorithms were designed to make it easier for researchers to replace, refine and test new ideas and it has been used for this purpose in this work. The library requires Python 3.8+ and supports PyTorch $\geq$ 1.13. The library was chosen due to being one of the few algorithm implementation libraries for reinforcement learning with maintained and well-documented code. The Deep Q Network implementation from the library was utilized for the evaluation.

### 3.3.2 Multi-Objective Gymnasium

The Multi-Objective Gymnasium (MO-Gym) is an open-source Python library maintained by the Farama Foundation, it provides the capability to develop and evaluate reinforcement learning algorithms on a variety of multi-objective environments (Felten et al., 2023). The environments are usually important bench-marking environments which are utilized to compare the performances of newly developed algorithms. It offers rendering of the episodes to visualize the behaviour of the agent, and callbacks to analyse the training process and policy updation at every step. The MO-Gym architecture was modified to implement the different multi-objective settings and evaluate them.

### 3.3.3 Google Colaboratory

The experiments were run on the hosted Jupyter Notebook service provided by Google with a T4 GPU. It offers pre-installed packages and allows cloning of GitHub repositories so that the local versions of commonly utilized libraries can be used, as was the case with the MO-Gym in my implementation.

## 3.4 Deep Q-Network

The Deep Q-Network (DQN) is an off-policy architecture, designed by DeepMind, which approximates the state-value function in a Q-Learning framework with a convolutional neural network (Mnih et al., 2013). The model takes raw pixel data and generates a value function to estimate future rewards. The algorithm was initially applied to Atari games and showed unprecedented performance (Mnih et al., 2013). It was chosen as the evaluation algorithm due to T. T. Nguyen et al. (2020) finding that linear deep Q-learning is more dominant than its non-linear counterpart for the Multi-Objective Mountain Car problem. The architecture can compute the Q-Values for possible actions in a state with only a single forward pass through the neural network.

**Characteristics:**

- **Experience Replay**: The agent's experiences

$$e = (\text{state}, \text{action}, \text{reward}, \text{next state}, \text{episode done})$$

**Algorithm 1** Deep Q-Network (DQN) with Experience Replay (Mnih et al., 2013)

Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
**for** episode = 1, M **do**
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    **for** t = 1, T **do**
        With probability $\varepsilon$ select a random action $a_t$
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters $\theta$
    **end for**
**end for**

at every time step pooled over many episodes and stored in a replay memory buffer. The experiences are sampled randomly from a mini-batch and used to train the neural network. The advantage is that it breaks the temporal correlation between consecutive experiences and therefore, makes the network updates more stable. It has a finite memory size so past transitions are erased after the memory buffer size is reached.

- **Epsilon Greedy Algorithm**: The algorithm proposed by Watkins, is a well-known reinforcement learning algorithm (Watkins, 1989). It uses the value of $\varepsilon$ to decide whether the agent should explore the environment or exploit the knowledge it has gained:

$$a_t = \begin{cases} \text{random action with probability } \varepsilon & \text{(exploration)} \\ \text{argmax}_a Q(s_t, a) & \text{(exploitation)}, \end{cases}$$

where:

$$\begin{aligned} a_t &: \text{chosen action at time } t, \\ \varepsilon &: \text{exploration rate, and} \\ Q(s_t, a) &: \text{estimated Q-value for state } s_t \text{ and action } a. \end{aligned}$$

## 3.5 Scalarization

Scalarization is a technique in multi-objective optimization in which the multi-objective function is converted to a single solution using weights. The weights are decided before

the optimization process, therefore enabling the objectives to be given priority and different solutions to be found depending on the weights provided (Gunantara, 2018). The goal of the reinforcement learning agent is to maximize or minimize the total reward function which is given by the following equation, depending on the requirement of the multi-objective problem:

$$F(x) = \sum_{i=1}^{k} w_i \cdot f_i(x). \tag{3.1}$$

## 3.6 Pareto Optimality

The Multi-Objective Mountain Car problem is a multi-objective optimization problem, which means that the optimization can be presented as a Pareto set or a Pareto front, as highlighted in Section 2.5. The problem of finding the Local non-convex Pareto Optimal set of similar problems has been proven to be NP-Hard (Natura et al., 2022). Therefore, the method utilized in this work to generate the Pareto front is the weight aggregation method, it converts the multi-objective problem to a single objective equation by assigning priority to the different objectives and the agent is trained on various weight settings. The method can estimate the non-dominated Pareto front, as highlighted in Section 2.5. The decision maker can then use scalarization to apply their preference to the problem based on their domain-specific knowledge and vary the results while having the guarantee of all of them being optimal.

The equation for a two-objective setup is given by

$$F(x,y) = w_i \cdot x + w_j \cdot y, \tag{3.2}$$

with the constraints

$$w_i, w_j \geq 0 \quad \text{and} \quad w_i = 1 - w_j.$$

## 3.7 Designing of a Reward Signal

We only make changes to the reward function of the environment to prevent introducing side-effects, defined in the Section 2.3. Dulac-Arnold et al. (2021) mentioned that learning from a poorly specified reward function is a critical obstacle which has prevented the application of reinforcement learning in real-world challenges. Since reinforcement learning models will be deployed in more complex, large scale and open-domain problems, it is an important concern to address poorly designed reward functions even in bench-marking environments. The reason being that these environments show researchers the performance of their algorithms before their applied to real-world problems, justified in Section 2.1. Additionally, they can indicate if the mitigations in place to avoid accidents, introduced in Section 2.3 and other similar issues that are being encountered by the algorithm have the intended result and hopefully prevent them it in a real-world situation, which could have catastrophic consequences. The things to consider when designing a reward function for an environment for the agent to run in are highlighted in Section 2.3 and the work aims to introduce different settings for the Multi-Objective Mountain Car.

# Chapter 4

# Experiments

We propose different reward functions to replace the existing configuration and assess the practicality of each by completing the analysis at the training and evaluation levels. Additionally, we highlight the motivation behind each reward function analyze and investigate any unexpected observations while finding potential reasons for it. We complete a comprehensive analysis to provide MORL researchers with an understanding of the characteristics of the reward functions and show the effects of reward shaping and scalarization.

## 4.1 Experiment Conditions

The Multi-Objective Mountain Car environment offered by the MO-Gym (introduced in Section 3.3.2) has a continuous state space, which means that the agent can take uncountable infinite positions in the environment and is represented by a range of real numbers. The action space is discrete, which means the actions that the agent can take are finite and countable. The action consists of backwards, forwards and zero-throttle actions and the acceleration applied in the backwards and forwards movements is constant at every step.

The hyper-parameter tuning is adapted from T. T. Nguyen et al. (2020), they conducted experiments on the setting of the environment, specified in Section 3.1.1, to test the framework developed by them.

### 4.1.1 Hyper-parameter Explanation

Table 4.1 highlights the hyperparameter settings utilized in the execution of the DQN algorithm, mentioned in Section 3.4.

The following hyperparameters may not be commonly used in reinforcement learning, therefore brief details about them have been provided here:

**Replay Buffer Size**: The total number of transitions/experiences that can be stored in the buffer before the oldest experience is removed from it (First-In-First-Out)

**Number of Optimization Steps per Batch**: The number of times the model parameters are updated in a batch by applying the optimization algorithm.

**Maximum Value for Gradient Clipping**: The value to which the gradient is set once it exceeds this value, to prevent the “exploding gradients" problem which is frequently observed in neural networks

| Hyper-parameter | Value |
|---|---|
| Learning Rate | 0.0001 |
| Replay Buffer Size | 20,000 |
| Training Steps | 200,000 |
| Number of steps before Training Starts | 1000 |
| Initial Epsilon | 1 |
| Final Epsilon | 0 |
| Gamma | 0.9 |
| Number of steps between updating the policy | 4 |
| Number of Optimization steps per batch | 1 |
| Maximum Value for the gradient clipping | 5 |

Table 4.1: The hyper-parameter configuration of the DQN algorithm on which the agents for the reward functions were trained. They were adapted from the experiments conducted in T. T. Nguyen et al. (2020).

## 4.2 Time-Speed

The local optima is observed when the agent does not adopt moves other than the zero-throttle, as highlighted in Section 3.2. Therefore, if the agent tries to maximize its speed at any time step $t$, then that can force it to leave the bottom of the valley and encourage it to take higher positions. The high position will enable the agent to have a larger swing radius and therefore, gain a higher speed reward. Swinging a larger radius each time could enable the agent to ultimately reach the goal state. The positive reward propagation, implemented by reward shaping, introduced in Section 2.3, would provide the agent with a clearer idea of where to find the goal state.

### 4.2.1 Configuration

The speed objective is implemented by taking the absolute value of the velocity variable at any time step $t$. The time objective defined in Section 3.1.1 is the second objective. Speed is considered instead of velocity to prevent a misleading reward setting. The rationale behind this lies in the reward structure, where moving backwards, incurs increasingly negative rewards, while moving forwards results in greater positive rewards. Rewarding the forward action might seem intuitive at first, but it can inadvertently encourage the agent to favour moving forwards exclusively, neglecting the necessary backward actions crucial for building momentum to reach the goal state. The trajectory of the agent trained on the Time-Velocity reward function, illustrated in Figure 4.1, highlights that it moves forwards exclusively, as was expected. Additionally, it leads the

agent to the local optima problem, the training process has been graphed in Figure 4.1, which we are trying to mitigate.

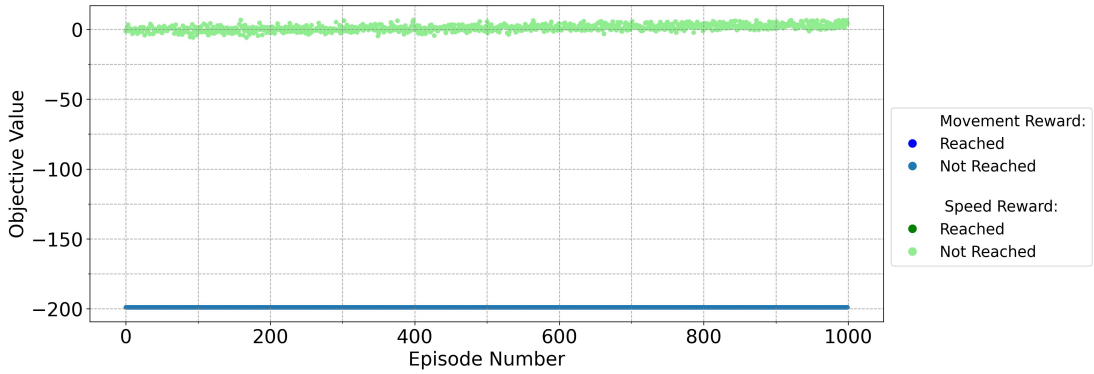



Figure 4.1: Accumulated objective reward per training episode for (0.9,0.1) scalarization of the Time-Velocity reward function. The agent is unable to learn and therefore, unable to reach the goal state.

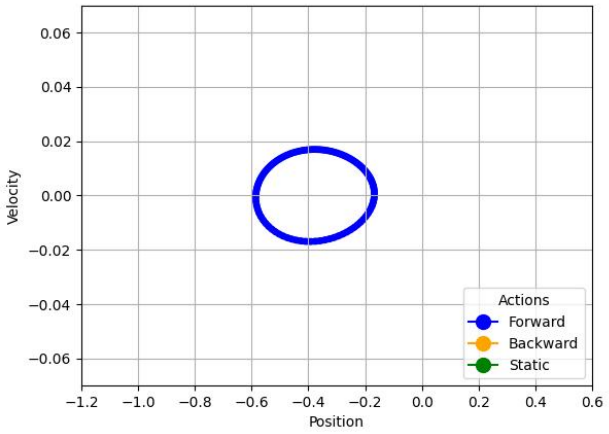



Figure 4.2: Sample trajectory for the (0.9,0.1) scalarization of the Time-Velocity reward function. The agent adopts the forwards action at every time step $t$, though since its engine is not strong enough to scale the mountain, it repetitively descends, resulting in the circular trajectory shown in the Figure.

#### Amplification of Speed

Speed at time step $t$ has a magnitude in the range [-0.07, 0.07], as outlined in Section 3.1.1, while the time reward is -1 at every time step; therefore, to make the two rewards comparable in their magnitude the speed reward is multiplied by an amplifying coefficient. This makes the evaluations easier to understand and analyse with the different weight scalarizations.

### 4.2.2 Pareto Graph

We present a Pareto graph, in Figure 4.3, for the Time-Speed reward function, the significance of which was outlined in Section 2.5. The Pareto graph is estimated by

| Scalarized Weights | Time Reward | Speed Reward | Total Reward |
|---|---|---|---|
| (0.0, 1.0) | -123.44 | 52.63 | 52.63 |
| (0.1, 0.9) | -130.25 | 57.23 | 38.48 |
| (0.2, 0.8) | -118.45 | 49.33 | 15.78 |
| (0.3, 0.7) | -119.3 | 49.42 | -1.20 |
| (0.4, 0.6) | -115.13 | 41.49 | -21.16 |
| (0.5, 0.5) | -134.17 | 54.78 | -39.70 |
| (0.6, 0.4) | -139.56 | 39.46 | -67.95 |
| (0.7, 0.3) | -111.37 | 40.32 | -65.86 |
| (0.8, 0.2) | -118.42 | 43.40 | -86.05 |
| (0.9, 0.1) | -149.19 | 44.95 | -129.77 |
| (1.0, 0.0) | -178.10 | 58.75 | -178.10 |

Table 4.2: Average reward each objective receives aggregated over 100 episodes over different scalarization combinations. The magnitude of the speed reward accumulated at this amplification is lower than that of the negative reward provided by the time objective because the positive reward has been added to solve the problem of sparse rewards, though not overpower the original setting. Though, the positive reward is kept this high, to maintain consistency with the Movement-Speed configuration highlighted in part 4.3.2.

running 11 different weight combinations for 200,000 training steps, each incremented by 0.1 for one objective and decremented by 0.1 for the other.

The equations are given by:

$$w_1 = x, \quad \text{and} \quad w_2 = 1 - x, \quad \forall x \in \{0, 0.1, 0.2, \ldots, 0.9, 1\}.$$

Each point on the graph has been calculated by averaging over 100 different evaluation episodes. The y-axis is the average speed reward received by the agent, and the x-axis is the average time reward received over the 100 evaluation episodes. Table 4.2 tabulates the values seen in the Figure 4.3, and introduces an extra piece of information, which is the total reward received after the scalarization has been applied.

### 4.2.3 Effect of Reward Shaping

We can observe in Figure 4.3 and Table 4.2, that the agent can make its way to the goal state in each of the scalarizations considered since none of the time rewards exceed -180, and the agent would have to have received a time reward -200 if it had been unable to reach the goal state. Therefore, we can conclude that the agent does not converge to a zero-throttle-only action.

The optimal point (0.1, 0.9), due to its inclusion in the Pareto Front, is a great example of how reward shaping has changed the agent's learning process. The training process in which the agent converges to a local optima has been shown in Figure 3.4, and we can observe that the time objective never has a value less negative than -200. In comparison, in Figure 4.5, we observe that the time objective receives rewards which are less negative than -200, with many episodes even recording a time objective reward

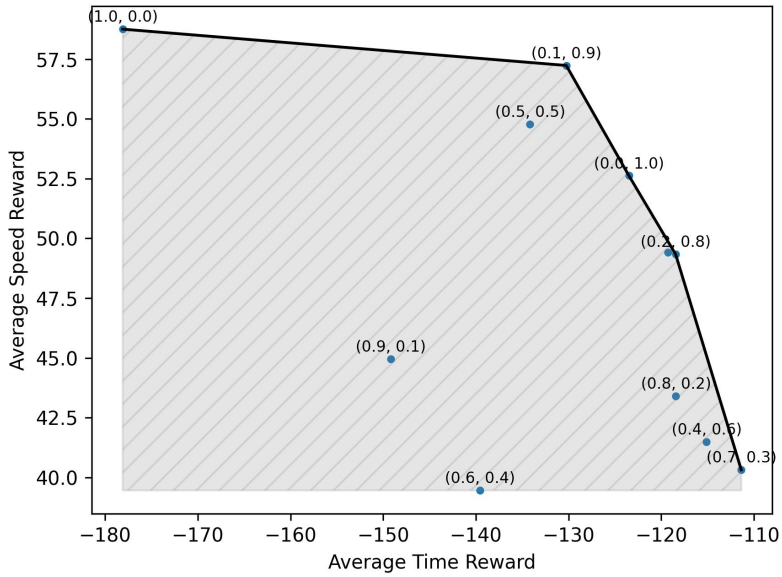



Figure 4.3: Pareto Graph of the Time-Speed reward function. Each blue point is a scalarization combination evaluated to see if it is part of the Pareto Front. The black line indicates the Pareto front, therefore each point on the black line is part of the Pareto front. Every point in the shaded area is a sub-optimal point dominated by one or more points on the Pareto front. The average objective reward values were calculated by running 100 different evaluation episodes. The white area above the Pareto front is the region where the agent could not go due to physical limitations. Point (0.3, 0.7) has not been labelled due to aesthetic reasons since it is very close to (0.2, 0.8) and is dominated by it as well.

close to -100. The value of the total positive reward awarded to the agent in an episode is the reward value of the speed objective. Figure 4.5 shows that the speed objective reward starts to get optimized by the agent around the $200^{th}$ episode. Additionally, as training progresses beyond, the increase in the positive rewards helps the agent reach the goal state consistently, as indicated by the dark green points. Consistently reaching the goal state enables the agent to find more efficient solutions with regards to the time taken from training episode 800 onwards.

We conclude that based on the observations outlined above, the agent behaves in accordance with the description provided in Section 4.2. However, there were a few unexpected findings such as the agent optimizing the time objective after consistently reaching the goal state and the emergence of two distinct sets of solutions, which we will investigate further in Section 4.2.5.

### 4.2.4 Optimal vs Non-Optimal Solutions

The Pareto plot, shown in Figure 4.3, shows multiple points which are in the shaded area, which means they are not part of the Pareto front and are dominated by one or more points on the Pareto front, which is indicated by the black line. Therefore, implying that not all scalarization combinations lead to optimal solutions, despite being trained on

identical hyperparameters of the DQN algorithm. Additionally, it allows us to conclude that the solution space is fractured.

To investigate the reason for sub-optimality, we plot Figure 4.4, which shows the total reward accumulated by each objective at every episode of the training process of (0.9, 0.1), one of the sub-optimal points. The (0.9, 0.1) scalarization assigns a priority 9 times higher to the time objective than to the speed objective. From the figure, we can conclude that there is a lack of convergence of the algorithm since the solutions that the agent finds are distributed and inconsistent. In fact, towards the end of the training process, we can observe light green points for the speed objective and light blue points for the time objective points which indicate that the agent did not reach its goal state. The scattering of the successful episodes, indicated by dark green and dark blue could explain the reasoning for a suboptimal solution, the agent does not have a policy which is dependable and instead varies quite a bit in different episodic runs.

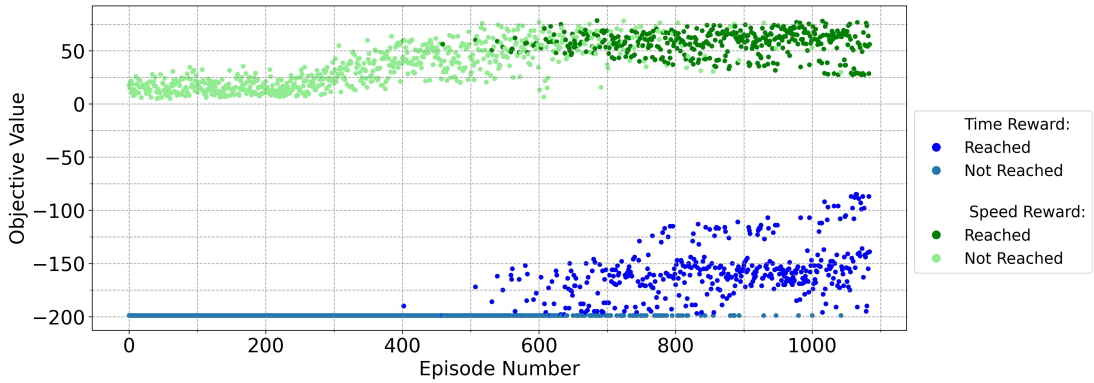



Figure 4.4: Reward accumulated by each objective while training on the (0.9, 0.1) scalarization of the Time-Speed reward function. The dark blue and the dark green points indicate the episodes in which the agent is able to reach the goal state, while the light blue and the light green indicate the episodes in which the agent is unable to reach the goal state. Each point for the speed reward has a corresponding point for the time reward.

The reason for the difference in the results and the emergence of optimal and non-optimal points was initially thought to be the significantly lower average reward of the speed than the average reward for the time objective. The justification being the fact that in Figure 4.6, where speed reward only goes as high as about 80, while the time reward can be as low as -200. With a lower priority provided to speed in the scalarization formula, the agent's emphasis would naturally be lower on the speed objective as is the case in (0.1, 0.9). Though, the reasoning does not hold for all optimal-suboptimal point pairs in the Figure 4.3. While (0.6, 0.4) is a suboptimal solution, (0.7, 0.3) is considered one of the optimal ones, due to its inclusion in the Pareto Front. The performance gap is visible in Table 4.2, with the (0.6, 0.4) taking almost 30 more time steps to reach the goal state, but averaging similar speed rewards. A considerable difference in performance is observed despite the (0.6, 0.4) assigning a higher priority to the speed objective. Therefore, we can't conclude that the difference in the magnitude of the speed and time rewards does always have an impact on the the scalarization achieving

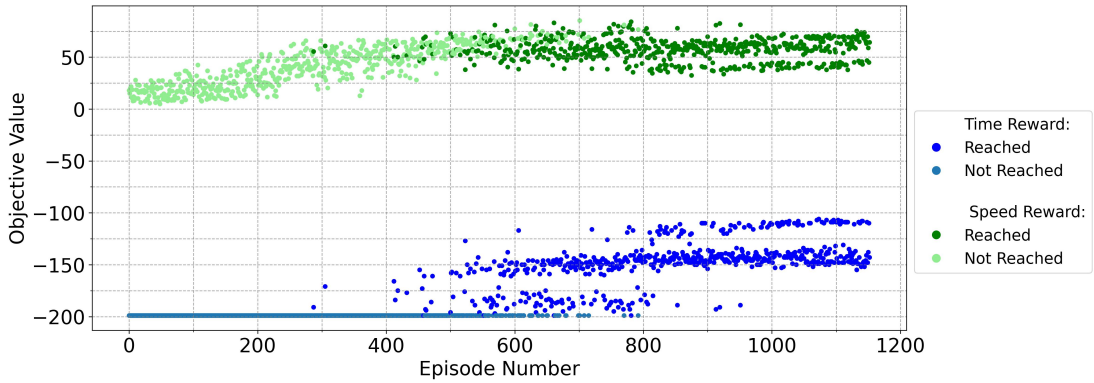
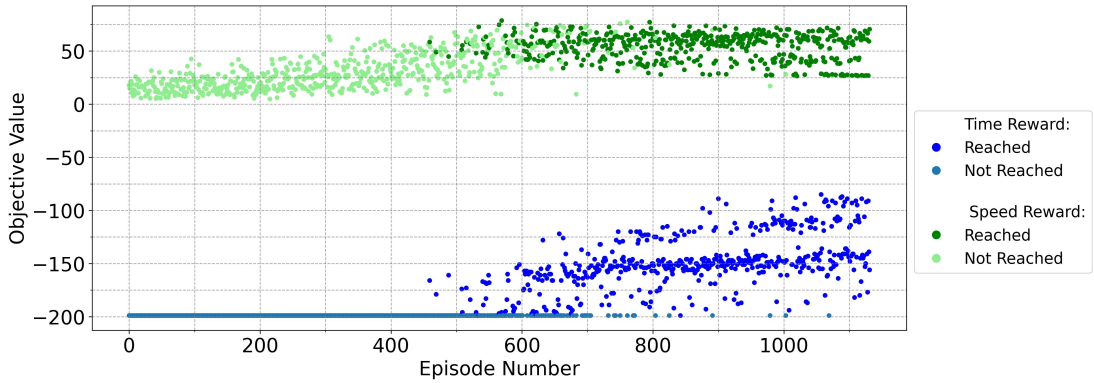


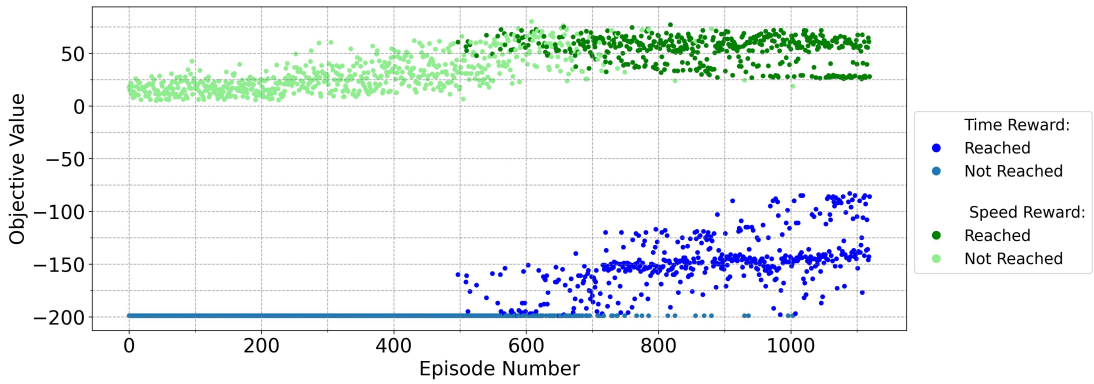

Figure 4.5: Reward accumulated by each objective while training on the (0.1, 0.9) scalarization of the Time-Speed reward function. The dark blue and the dark green points indicate the episodes in which the agent can reach the goal state, while the light blue and the light green indicate the episodes in which the agent is unable to reach the goal state. Each point for the speed reward has a corresponding point for the time reward.


Figure 4.6: Objective Reward Correlation for every training episode of the (0.1,0.9) scalarization. Green points indicate episodes in which the agent reached the goal state, and the red points indicate the ones in which the agent did not reach the goal state.

optimal performance or not. However, it is intriguing that comparable scalarizations, (0.6, 0.4) and (0.7, 0.3), yield such vastly different outcomes. The rewards accumulated by each objectives at every training episodes, shown in Figure 4.7, are quite similar, for both the scalarization combinations. However, upon plotting the evaluation trajectories, the (0.6, 0.4) is highly inconsistent with the routes taken by the agent to reach the goal state. The inconsistency is observed due to the variability in the frequency of swings and the extent of backward movement. Due to the impracticality of plotting all trajectory types, the variability has not been illustrated in a figure. The variable number of swings required to escape could be due to agent having difficulty in escaping the lower portion of the valley, while the inconsistent backward movement could be because the agent has not gauged how far back it may need to move to swing to the goal state. The uncertainty costs the agent in the (0.6, 0.4) case, leading to a sub-optimal solution.




(a) Reward accumulated by each objective while training on the (0.6, 0.4) scalarization of the Time-Speed reward function.




(b) Reward accumulated by each objective while training on the (0.7, 0.3) scalarization of the Time-Speed reward function.

Figure 4.7: Reward accumulated by each objective during training for (0.6, 0.4) and (0.7, 0.3) scalarizations respectively, of the Time-Speed reward function. The dark blue and the dark green points indicate the episodes in which the agent can reach the goal state, while the light blue and the light green indicate the episodes in which the agent cannot reach the goal state. Each point for the speed reward has a corresponding point for the time reward. The training process is quite similar but their performance changes, as can be seen in Table 4.2.

### 4.2.5 Multi-Objective Optimization

We reshape the training process of (0.1, 0.9), an optimal point on the Pareto front, to understand how the agent balances the two objectives by plotting the speed rewards on the x-axis and the time rewards on the y-axis in Figure 4.6. Each red point in the figure represents an episode in which the agent did not reach the goal state. There are a few episodes in which the agent reached the goal state just as the episode's time was elapsing. Though, the most interesting part of the graph is that, there are two collections of points (sets) which can quite clearly be observed. The two sets will be referred to as set A and Set B from now on. Set A averages a -120 reward for the time objective and about a 40 reward for the speed objective, and set B, which is more populated, has a higher speed reward while the time reward is relatively more negative. The episodes in set B have a higher average speed reward than the episodes in set A, indicating that

the duration of the agent's presence has a direct correlation to the total speed reward it can accumulate. The figure shows that episodes in which the agent reaches a goal state have a trend, as the time reward approaches 0, the speed reward accumulated decreases. The speed reward threshold, as seen in the figure, is around 32; all speed reward values below it result in the agent being unable to reach the goal state.

### 4.2.6 Longer Training for Finding Threshold

In Figure 4.6, we observed that the threshold of the accumulated speed reward was around 32, therefore, we ran the training process for longer to see if the threshold decreases. The training process for (0.1, 0.9) has been graphed in Figure 4.8, while we observe that set A and B have become gotten further reinforced by the additional training episodes, a third set has started to appear above set A. The quickest solution requires only around 80 steps, and the lowest accumulated speed reward for a successful episode has reduced to around 27. Therefore, we can conclude that training an optimal point can lead to superior performance and more efficient trajectories being found by the agent

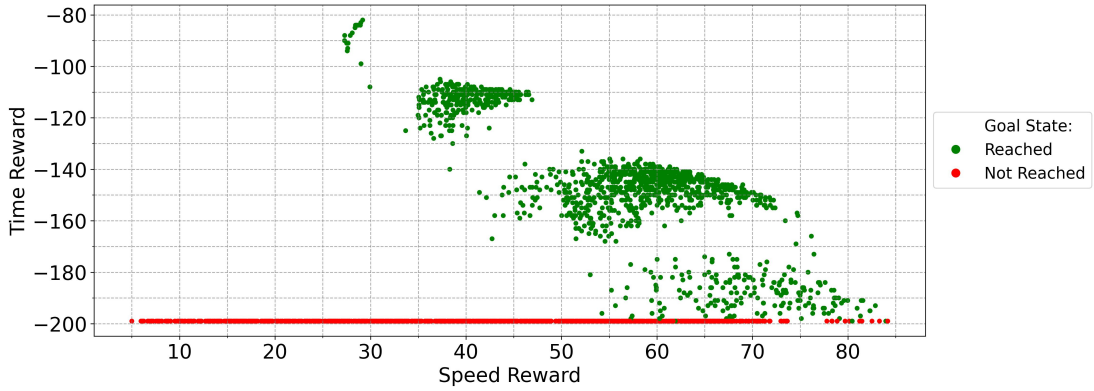



Figure 4.8: Objective Reward Correlation for every training episode of the (0.1, 0.9) scalarization. The agent was trained for 400,000 time steps. Green points indicate episodes in which the agent reached the goal state, and the red points indicate the ones in which the agent did not reach the goal state.

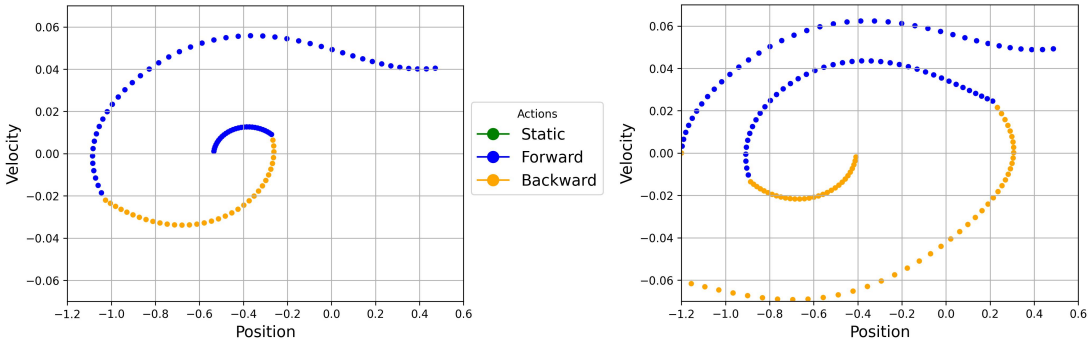
### 4.2.7 Trajectory Based Analysis

In Section 4.2.5, we highlighted the emergence of two different sets of solutions, and after uniformly sampling trajectories from the two sets, we found out that the stochasticity of the starting position of the Mountain Car was the reason for distinct sets of solutions. As outlined in Section 3.1.1, the starting position is uniformly sampled from [-0.4,-0.6]. Consequently, certain starting points prompt the agent to initially move forward, while in other cases, the agent moves backwards first, changing the trajectory it adopts.

The trajectories in Figure 4.9 have been sampled uniformly from the sets present in Figure 4.8. The trajectory in Figure 4.9b illustrates the scenario where the agent moves backwards first. It is evident that the agent takes more time to reach the goal state,

travels further backwards, receives a higher speed reward and takes more swings than the trajectory in Figure 4.9a, where the agent moves forwards first. Additionally, the trajectory in Figure 4.9b hits the backboard while moving backwards, and therefore loses all its velocity as was mentioned in Section 3.1 unlike the trajectory in Figure 4.9a, in which the agent always remains in bounds.

The trajectory in Figure 4.9b was sampled from set B, which is considerably more populated than set A, indicating that the agent has a preference for moving initially moving backwards. The reason for this preference is most likely the starting position, a larger portion of the starting position range prompts the agent to move backwards instead of forwards. Moving backwards initially requires the agent to take an additional swing, as shown in Figure 4.9b, indicating that a single swing consisting of moving backwards and then pushing forwards may not be enough to reach the goal state. Figure 4.9a helps us conclude that a swing and a half is enough to reach the goal state, and additional swings may increase the time required to reach the goal state. From the trajectories illustrated in Figure 4.9, we observe that the agent uses the acceleration in the opposite direction to counteract the velocity gained in the current direction. Lastly, both trajectories do not take even a singular zero-throttle move. This showcases the agent's preference for moving rather than staying still; this could be due to the reward function not penalising the backward or forward actions.




(a) Trajectory starting with forward steps, takes 108 steps to reach goal state.

(b) Trajectory starting with backward steps, takes 152 steps to reach goal state.

Figure 4.9: Trajectories sampled from sets A and B, defined in Section 4.2.5. The sets are found in the training process of the optimal point (0.1, 0.9).

### 4.2.8 Learning and Summary

From the experiments conducted on the proposed reward function of Time-Speed, we can conclude that:

- The Time-Speed reward function addresses the local optima problem.
- The lower the time reward, the longer the agent has to accumulate speed rewards. However, that does not always result in scalarizations with a lower time reward having a higher speed reward, and those are the points which are dominated and do not become part of the Pareto Front.

- Scalarization offers a way to assign different priorities to the objectives, resulting in diverse agent behaviour for each scalarization combination.
- Training Pareto points for longer could lead to the agent achieving superior performance.
- The stochastic starting position leads to fewer trajectories initiated with forward moves and a greater number of trajectories starting with backward moves, highlighting the agent's preferences.
- Trajectories beginning with backward moves take longer and have more number of swings than the ones starting with forward moves.
- The agent does not take any zero-throttle moves.
- The agent can find a significantly larger number of solutions with reward shaping compared to the single objective version, which was illustrated in Figure 3.3.

### 4.2.9 Potential Limitations and Future Developments

While the Time-Speed reward function has the required effect of overcoming the local optima problem and shows that the reward shaping introduced through the speed objective has been effective, there are a few limitations of the suggested reward function.

- In the trajectories, where the agent started backwards, the agent hits the backboard. Such a situation could lead to severe repercussions in the real world, where there may not be a safety net, like the virtual backboard in this environment. Can the issue be treated without changing the environment's characteristics and instead changing the reward function to capture this limitation of the environment?
- The agent uses acceleration in the opposite direction to counteract the velocity in the current direction. This may be an excessive use of energy, but it may be overcome using zero-throttle actions. Could this idea lead to the agent finding more energy-efficient trajectories?
- There are inconsistencies in the number of swings required to find the goal state in the trajectories. Can the agent move backwards initially and reach the goal state in a single swing?

## 4.3 Movement-Speed

The agent's behaviour in the Time-Speed reward signal enabled it to reach the goal state, though had a few challenges, namely infrequent usage of the zero-throttle step and the agent hitting the backboard in many trajectories. Therefore, to overcome the aforementioned challenges and provide MORL researchers with another reward function for the Multi-Objective Mountain Car problem. We condense the backward and forward action objectives, introduced in Section 3.1.1, into a single objective called movement. The suggested reward function has a multi-objective setting of time and movement.

### 4.3.1 Rationale behind the Reward Function

In the Time-Speed reward function, for the points on the Pareto front, the greater the time taken, the higher the speed reward. Therefore, the agent had to find a balance to ensure neither objective was overlooked or overoptimized. The Movement-Speed reward function offers a different paradigm, where succumbing to a local optima is a very likely possibility due to the reward function having a lack of reward signal for the zero-throttle move. The agent, therefore, has to ensure that the moves that it select help it increase its speed objective reward while overcoming the tendency to optimize only the movement objective and remain at the bottom of the valley. Movement in the Mountain Car environment refers to accelerating in the backwards or forwards direction, therefore representing the expenditure of energy by the agent. Consequently, the agent's objective is inherently tied to minimizing the energy spent by the agent.

### 4.3.2 Configuration

The backward and forward objectives, introduced in Section 3.1.1 are condensed into a single objective called the movement objective. Therefore a -1 reward is provided for every backwards or forwards move. The speed objective from Time-Speed is maintained. To ensure consistency between the environments proposed, the amplification coefficient of speed is kept constant. The agent's objective is to minimize the movement reward and maximize the speed reward to get the highest total reward possible. Due to the reward function being a more challenging problem and none of the scalarizations we experimented with converging, the training was increased to 400,000 steps.

### 4.3.3 Multi-Objective Optimization and Reward Shaping

In Section 3.2, it was shown that providing very low priority to the movement attribute still resulted in a local optima. The training process on the equivalent scalarization for the Movement-Speed reward function has been illustrated in Figure 4.10. While the 0.05 weight is provided to the movement objective, the 0.95 weight is provided to the speed objective instead of the time objective. The movement objective is optimized initially, showcasing a tendency to converge to the local optima. However, around the $750^{th}$ episode, rewards accumulated for the speed objective start to increase, and we observe a change in trend in the rewards accumulated for the movement objective. The optimization of the speed objective leads the agent to find the goal state around the

$1000^{th}$ training episode, indicated by the dark green points for the speed objective and dark blue for the movement objective. Therefore, we can conclude that the reward-shaping implemented through the speed objective enables the agent to reach the goal state.

We can observe that with increased movement, the agent can maximize the reward accumulated for the speed objective. However, that does not necessarily result in the agent reaching the goal state, as indicated by the light green points in Figure 4.10. The agent may or may not reach the goal state if the movement objective reward accumulated is more negative than -150. However, once the agent reaches the goal state, the agent can consistently do so when the movement objective is less negative than -150, despite the rewards accumulated for the speed objective not being as high as in the first case. This observation showcases the importance of balancing the two objectives to reach the goal state consistently. Furthermore, it emphasizes the importance of finding "good" steps, which ultimately assist the agent in reaching the goal, since moving more does not guarantee reaching the goal state. The agent grasps the importance of multi-objective optimization, and we observe a lack of episodes with the movement objective reward being more negative than -150 beyond the $2000^{th}$ episode. Towards the end of the training process, we observe two sets of solutions. Figure 4.11, which plots the accumulated speed objective reward for each training episode of the x-axis and the accumulated time objective reward for each training episode of the y-axis, shows the sets of solutions more distinctly. Set A is the set of solutions for which the accumulated movement reward for the episode ranges between -100 and -80. Set B is the set of solutions that distinctly has the largest congregation of green points in the Figure. The sets of solutions were also observed in the training process for the Time-Speed reward function.

We can observe a clear demarcation between the episodes in which the agent reaches the goals state and those where it doesn't in Figure 4.11. This indicates that the speed reward accumulated by the agent needs to be above a threshold for the reward provided by the movement for the agent to reach the goal state. Having an accumulated speed objective reward higher than the threshold does not guarantee that the agent will reach the goal state, but there is a very high chance that it will. However, the Figure shows that when the accumulated movement objective reward is lower than -140, and accumulated speed rewards are greater than 60, it is quite variable as to whether the agent can reach the goal state.

### 4.3.4 Training Comparison

To understand the effect of assigning a higher priority to the movement objective on the training process and ultimate convergence, we plot the training process for the (0.07, 0.93) scalarization case. The training process is illustrated in Figure 4.12 and trained for an identical number of time steps and under identical algorithmic conditions as the (0.05, 0.95) combination. The agent optimizes the movement objective for a longer duration before it starts to optimize the speed objective in (0.07, 0.93) than in (0.05, 0.95). Additionally, in the (0.07, 0.93) training process, we observe that the agent deviates from optimising the movement objective, which subsequently leads to a distribution of

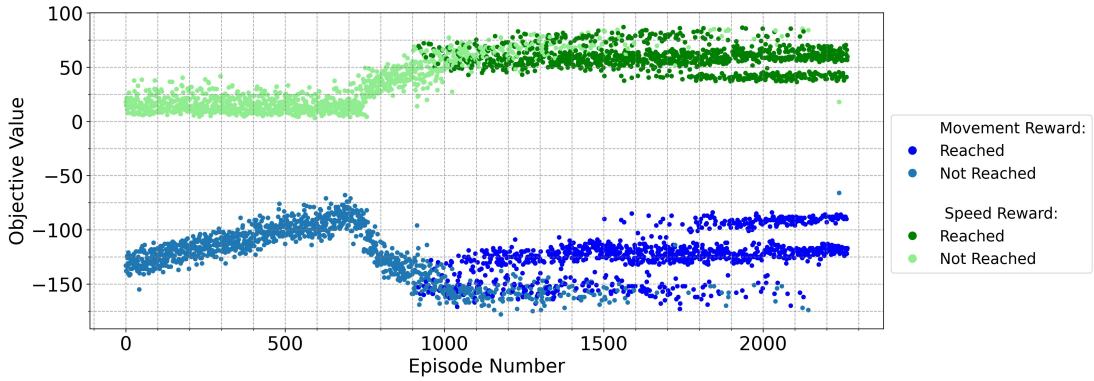



Figure 4.10: Reward accumulated by each objective while training on the (0.05, 0.95) scalarization of the Movement-Speed reward function. The agent was trained for 400,000 time steps. The dark blue and the dark green points indicate the episodes in which the agent can reach the goal state, while the light blue and the light green indicate the episodes in which the agent cannot reach the goal state. Each point for the speed reward has a corresponding point for the time reward.


Figure 4.11: Objective Reward Correlation for every training episode of the (0.05, 0.95) scalarization. The agent was trained for 400,000 time steps. Green points indicate episodes in which the agent reached the goal state, and the red points indicate the ones in which the agent did not reach the goal state.

episodes being found by the agent, having a range of 100 for the movement objective reward. Among the distributed episodes there are a few successful ones, signalling that the agent has found the goal state. The reason for the distribution of episodes could be the exploration factor (epsilon) and the agent's lack of understanding of how its steps affect the final outcome. The relative consistency and convergence for (0.07, 0.93) appear around the $1500^{th}$ episode. Though, unlike the training process for (0.05, 0.07) shown in Figure 4.11, where there were two clear sets of solutions, the training process for (0.07, 0.93) shown in Figure 4.13, only finds one set. The set found in (0.07, 0.93) is similar to set B for the (0.05, 0.95) training process, defined in Section 4.2.5. The reason for finding only one set of solutions could be delayed convergence since the solutions in set A were found towards the end of the (0.05, 0.95) training process. The differences in the training process help us conclude that providing higher priority to the

movement objective may delay the algorithm's convergence. Additionally, it leads to a very different training process with a significant deviation from the movement objective optimization. Lastly, the agent could only find the goal state in scalarizations, which provided the movement objective weights lower than or equal to 0.1.

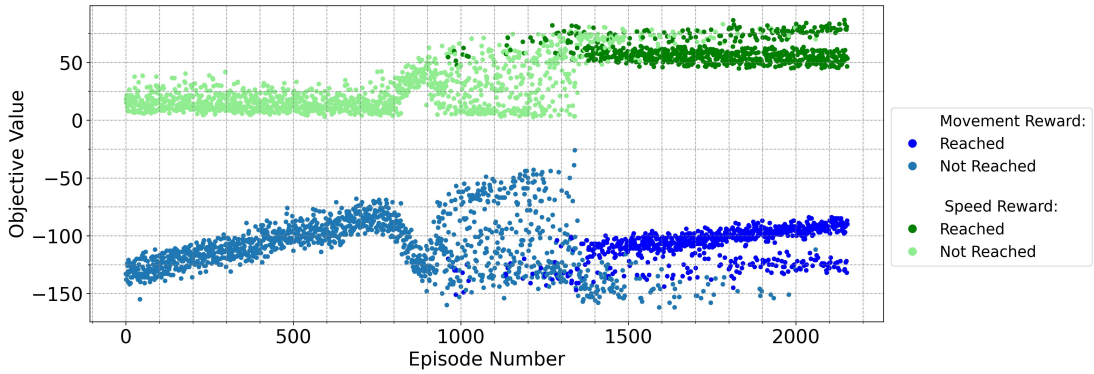



Figure 4.12: Reward accumulated by each objective while training on the (0.07, 0.93) scalarization of the Movement-Speed reward function. The agent was trained for 400,000 time steps. The dark blue and the dark green points indicate the episodes in which the agent can reach the goal state, while the light blue and the light green indicate the episodes in which the agent cannot reach the goal state. Each point for the speed reward has a corresponding point for the time reward.

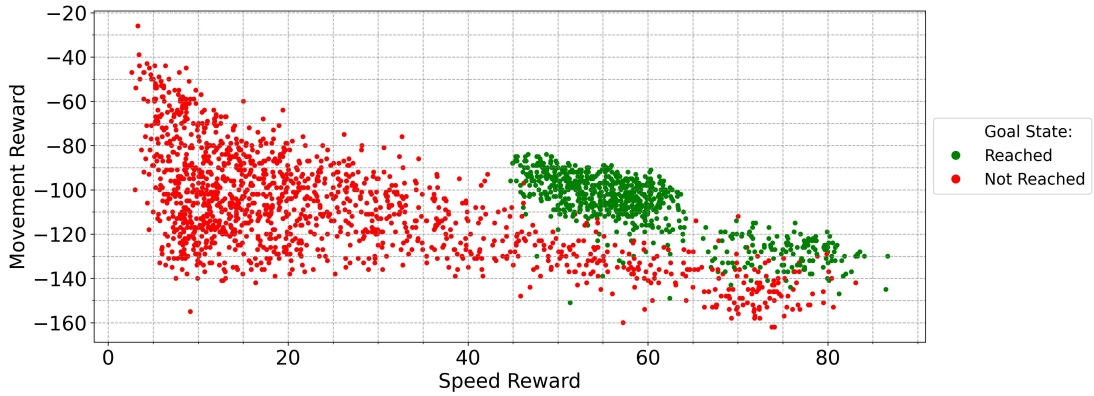



Figure 4.13: Objective Reward Correlation for every training episode of the (0.07, 0.93) scalarization. The agent was trained for 400,000 time steps. Green points indicate episodes in which the agent reached the goal state, and the red points indicate the ones in which the agent did not reach the goal state.

### 4.3.5 Trajectory Based Analysis

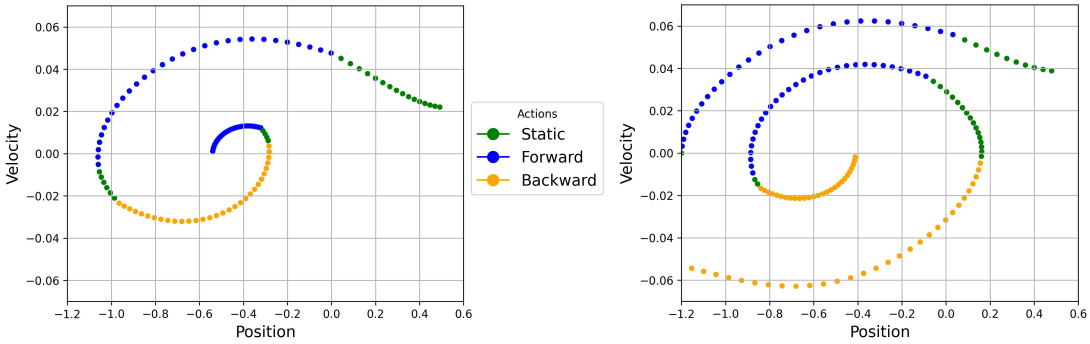
In Section 4.3.3, we highlighted the emergence of two different sets of solutions for (0.05, 0.95). After uniformly sampling trajectories from the two sets, we found that the stochasticity of the starting position of the Mountain Car was the reason for distinct sets of solutions. As outlined in Section 3.1.1, the starting position is uniformly sampled from [-0.4,-0.6]. Consequently, certain starting points prompt the agent to initially move

forward, while in other cases, the agent moves backwards first, changing the trajectory it adopts.

Trajectory starting with the agent moving forwards is illustrated in Figure 4.14a. We can observe that the agent takes one swing before it can reach the goal state. The agent utilizes the velocity gained to get a position higher on the hill by taking the zero-throttle, indicated by the green points. It learns that gravity acting on it will slow it down to almost negligible velocity, therefore there is no requirement for it to exert acceleration when it wishes to move in the other direction. The additional distance gained expands its range of movement when moving in the opposite direction, as it can gather additional momentum.

In comparison, the trajectory in which the agent starts moving backwards initially is illustrated in Figure 4.14b. We observe that the agent changes its direction thrice because moving backwards first does not provide sufficient energy for it to swing to the goal state. The agent hits the virtual backboard since it tries to go beyond the lower bound of -1.2, mentioned in Section 3.1.1. The environment loses all its speed and its speed state variable is re-initialized to 0.

The trajectory graphs for the Movement-Speed reward function illustrated in Figure 4.14 show that the zero-throttle step is taken quite frequently. This is unlike the Time-Speed trajectories, shown in Figure 4.9, where it is not observed at all. The trajectories which commence with the agent moving backwards take longer than the ones in which the agent moves forwards initially. In the backward first trajectories, the agent reaches a higher magnitude of velocity (speed) and goes far more backward than in the trajectory where the agent starts forwards. Therefore, it is clear that more swings lead to more time steps taken to reach the goal state and the agent attains positions and speed values which are far higher than needed to scale the hill, hence allowing us to conclude that starting forwards is more efficient.




(a) Trajectory starting with forward steps, takes about 119 steps

(b) Trajectory starting with backward steps, takes about 142 steps

Figure 4.14: Trajectories sampled from sets A and B, defined in Section 4.3.3. The sets are found in the training process of (0.05, 0.95).

### 4.3.6 Learnings and Summary

From the experiments conducted on the proposed Movement-Speed reward function, we can conclude that:

- The Movement-Speed reward function addresses the local optima problem.
- The agent adopts zero-throttle moves before changing directions, this allows it to gain additional distance as velocity approaches 0. Verifying the claim by Vamplew et al. (2010) that the zero-throttle step is very valuable in the multi-objective case.
- In the trajectories where the agent moves backwards initially, the agent hits the virtual backboard.
- A threshold is apparent for the speed objective rewards corresponding to each movement objective reward, signifying that the agent must exceed this threshold for the associated movement reward to have a chance to reach the goal state.
- Providing higher priority to the movement objective may lead to the agent taking longer to converge to a dependable and optimal policy.
- The threshold for the Movement-Speed reward function was found to be (0.1,0.9). Any higher priority assigned to the movement objective results in the agent being unable to solve the problem.
- Due to the inability of the agent to solve the problem with only the movement objective, the Pareto front cannot be generated for this environment configuration.

### 4.3.7 Potential Limitations

While the Movement-Speed reward function has the required effect of overcoming the local optima problem and shows that the reward shaping introduced through the speed at time step $t$ has been effective, there are a few limitations of the suggested reward function.

- Due to the forward and backward movement getting the same negative reward, it might be hard for the agent to differentiate between the actions.
- One of the reasons why the movement objective was added to the reward function was to prevent the agent from hitting the backboard in its trajectories. However, Figure 4.9b shows that the reward function was unable to achieve this aim and therefore, it could have severe repercussions in a real-world situation.
- The lack of a Pareto Graph is due to the agent's inability to solve the environment using a high priority for movement. Therefore, we do not have insight into which scalarizations are optimal.
- The solution space is small because when priorities higher than 0.1 are provided to the movement objective, the agent is led to the local optima solution.

# Chapter 5

# Results and Discussion

We compare the merits of the two proposed reward functions, highlighting the differences and similarities in the agent's behaviour when trained on each. Additionally, we offer a reward function which is a direct replacement for the original which was introduced in Section 3.1.1. Lastly, we check if the conclusions drawn confer to literature and highlight a few limitations of our work.

## 5.1 Comparison between the Reward Functions

The Time-Speed reward function's Pareto Graph can be estimated, though the same cannot be done for the Time-Movement reward function. The movement objective is harder for the agent to balance than the time objective. Therefore, the agent has to train for twice as long for movement-based reward functions. Agents trained on the Time-Speed reward function do not use the zero-throttle action, while the ones trained on Movement-Speed use it to slow down and gain additional distance without accelerating. The trajectories vary significantly depending on whether the agent starts by moving backwards or forwards. When moving backwards first, the trajectories for Movement-Speed take fewer time steps on average than Time-Speed. This is despite Movement-Speed having a considerable amount of zero-throttle steps. While, when moving forwards first, the trajectories for Movement-Speed take higher time steps on average than Time-Speed. However, the number of acceleration steps (backwards and forwards) is fewer on average than in Movement-Speed. Therefore, the Movement-Speed trajectories are more energy efficient on average than the Time-Speed trajectories. The backward trajectories in both reward functions lead to the agent hitting the backboard. The speed with which the backboard is hit when the agent is trained on Time-Speed is higher than when trained on Movement-Speed.

## 5.2 Three Objective Case

We combine the reward functions on Time-Speed and Movement-Speed to create a Time-Movement-Speed configuration, with the reward shaping implemented through the speed objective. Vamplew et al. (2010) had suggested the multi-objective Mountain

Car problem to provide an environment configuration with 3 objectives to test the generality of MORL systems. The original three objective reward function, introduced in Section 3.1.1, suffers from the local optima problem, as shown in Section 3.2. Therefore, a replacement for it was found. The negative reinforcement provided by movement and time objectives is offset by the speed objective. The double negative objectives further reinforce the case of sparse rewards. Therefore, the weight assigned to the speed objective should ideally be higher than the combined weights assigned to the other objectives to prevent the local optima situation from arising. The upper bound we found for the movement objective still applies; the agent is unable to find the goal state in training when weights higher than of 0.1 are assigned to the movement objective.

To showcase the admissibility of the three objective reward function and show the correlation to the training processes for the Time-Speed and Movement-Speed reward function, the scalarization weight combination of (0.3, 0.05, 0.65) has been illustrated in the Figure 5.1. The combination was chosen because it represents a middle ground between the lower and higher extremes of scalarization. The figure shows glimpses of the training curves for the Time-Speed and Movement-Speed reward functions, with the movement objective converging to 0 until the speed objective is optimized and the reward received for the time objective being mostly -200 till the agent starts to find the goal state consistently. Similar to the training process of (0.07, 0.93) of the Movement-Speed reward function, the agent finds the goal state around the $1300^{th}$ episode, though it starts consistently reaching it after the $1600^{th}$ episode. Upon consistently reaching the goal state, the time and movement objectives progressively approach each other. Though, the movement objective rewards are less negative since the time objective penalizes any action taken in the action space, unlike the movement objective which only penalizes the backward or forward movements. Sets of solutions are observed towards the end of training for all three objectives. Therefore, we can conclude that the Time-Movement-Speed is a viable three-objective setting for the Mountain Car problem. Additionally, the problem can be expanded to a four-objective setting, by converting the movement objective into backwards and forwards objectives, as were part of the reward function in Section 3.1.1.

## 5.3 General Trends

We extract conclusions applicable to the wider MORL community and compare them to previous literature to showcase their generalizability and highlight the significance of our research endeavours.

Van Moffaert et al. (2013) mentioned that scalarization can be used to aggregate the agent's performance and lead to changes in the agent's performance. This is visible in the reward functions suggested in this work. Different scalarization combinations lead the agent to behave differently depending on the weight assigned to the different objectives.

Kusari and How (2020) found that there is a relation between the weights provided to the reward function and the optimal value of the MORL problem for the GridWorld,

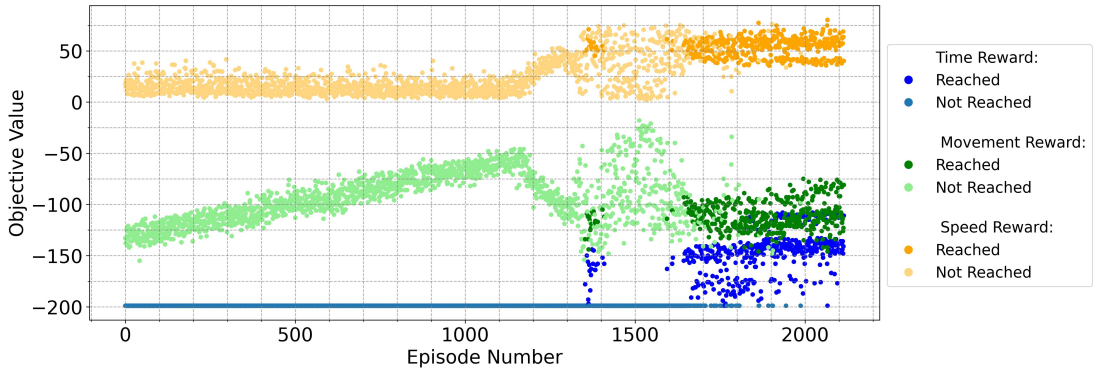



Figure 5.1: Reward accumulated by each objective while training on the (0.3, 0.05, 0.65) scalarization of the Time-Movement-Speed reward function. The agent was trained for 400,000 time steps. The dark blue, dark green and dark orange points indicate the episodes in which the agent can reach the goal state, while the light blue, light green and light orange indicate the episodes in which the agent cannot reach the goal state. Each point represents the rewards received by the objective in a given episode. Every point for one objective has a corresponding point in each of the other objectives for that particular episode.

ObjectWorld and Pendulum problems. In the Pareto plot, illustrated in Figure 4.3, there are a greater number of points on the Pareto front which have a higher priority provided to the speed objective than to the time objective.

Gupta et al. (2022) mentioned that reward shaping can provide task-related direction to the agent's behaviour and Devidze et al. (2022) concluded that it is a viable technique for environments having sparse rewards. This is indeed the case in our experiments where reward shaping takes care of the local optima problem and lead the agent to finding the goal state.

Brys et al. (2014) mentioned that a single objective problem can be multi-objectivized by adding objectives which are correlated to the single objective problem. Our experiments indicate that our suggested reward functions of the Multi-Objective Mountain Car problem have been effective with the introduction of the speed objective, which is directly correlated to the time and movement objectives.

Brys et al. (2014) mentioned that composite reward functions, made from adding reward shaping to existing reward functions, enable the agent to find good actions quicker due to being more informative. Therefore the agent can solve the problem faster and better. We observe that optimal and non-optimal scalarizations of the Time-Speed reward function, find better solutions and learn faster than the single objective case, which was shown in figure 3.3.

## 5.4 Critical Analysis and Limitations

We have estimated only the convex portion of the Pareto front for the Time-Speed reward function to demonstrate its intended effect. However, it is worth noting that

there might exist a non-convex portion of the Pareto plot, which remains unexplored, and will be mentioned in future work.

The experiments conducted on the Time-Speed reward function were run for fewer time steps than the experiments conducted on the Movement-Speed reward function. This discrepancy in duration came from the agent's inability to find successful episodes within the shorter timeframe for Movement-Speed. A limitation of running the training for longer is that the DQN algorithm employs epsilon greedy, as explained in Section 3.4, which means that epsilon, which controls the degree of exploration starts at 1 and reaches 0 at the end of training. Therefore, if we run the algorithm for longer in training, the epsilon decay will be slower. Consequently, the training instances may not be directly comparable.

The scalarizations utilized for comparison from each reward function are distinct. While the Time-Speed scalarization chosen was an optimal point, part of the Pareto Front, though, it remains unclear whether the scalarization for the Movement-Speed reward function is an optimal point.

The conclusions have been drawn based on a small sample size and a single benchmarking environment, therefore, they may not generalize well to the broader domain. Thus, the trends are compared to the ones seen in literature and are indeed verifiable.

# Chapter 6

# Conclusion

The goal of this project was to draw conclusions benefitting the field of multi-objective reinforcement learning by running experiments on a particular use case. The general goals chosen after exploring the field were understanding the effect of reward shaping in the multi-objective problem. In particular, we considered problems which originally had a single objective but were later multi-objectivized. Additionally, the project aimed to investigate any changes in the agent's behaviour upon the introduction of weights (i.e., priority) to the different objectives.

The use case chosen was the Multi-Objective Mountain Car problem, an important environment in the benchmarking suite. It suffers from a local optimal problem, which prevents the agent from learning the intended behaviour. Therefore, the aim was to provide a viable reward function by introducing reward shaping. A speed-based objective was added to implement reward shaping, and two different reward functions were suggested, Time-Speed and Movement-Speed. Analysis was conducted on the training and evaluation level due to many researchers mentioning a lack of evaluation being conducted on suggested benchmarking environments and reward functions and to provide researchers with an idea of how the suggested reward functions affect the agent's behaviour. The Pareto Graph for the Time-Speed reward function was estimated. Analysis was conducted on how objectives are balanced and their correlation with each other during training for different scalarization combinations. Additionally, the agent's behaviour was examined after the training process.

The experiments conducted helped us conclude, in line with the literature, that reward shaping is a viable option to multi-objectivize environments and can be utilized to propagate positive rewards into the system. Additionally, it makes the multi-objective configuration converge faster and perform better than the single-objective configuration. For scalarization, we found that different combinations alter the agent's training process and performance considerably. There is an inversely proportional relation between the priority provided to the positive rewards and the time taken by the agent to begin optimizing them. This may even affect the convergence of the agent and lead to the agent being unable to find certain sets of solutions.

We were able to draw several conclusions about the Mountain Car problem and its

suggested reward functions. Most importantly, the reward functions suggested, Time-Speed and Movement-Speed, address the local optimal problem. The inclusion of the zero-throttle step is beneficial at certain positions since it helps the agent conserve energy while ensuring it continues its journey to the goal. Notably, the agent takes different trajectories depending on whether it begins its journey by moving forwards or backwards. Training the Pareto points for the Time-Speed reward function for longer can lead to the agent's performance improving, though the same cannot be said about the non-Pareto points. In the Movement-Speed reward function, for most movement objective reward values, there is a threshold speed objective reward value and the agent must get higher than that to ensure it can reach the goal, without succumbing to the local optima. Lastly, the reward functions were condensed into a three objective case, Time-Movement-Speed, to provide the community with a benchmark to test the generality of MORL systems.

## 6.1 Future Work

Possible developments to this problem which could further our understanding of the field of MORL and similar benchmarking problems are given below.

- Finding non-convex Pareto Fronts of the Time-Speed reward function using non-scalarization-based techniques or the Chebyshev scalarization (Van Moffaert et al., 2013);
- Testing the suggested reward functions across multi-objective benchmarking algorithms to understand how the performance changes (Wang et al., 2020)
- Adding stochasticity to the Mountain Car state transitions to improve generalizability (Vamplew et al., 2010)

# Bibliography


Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., & Bellemare, M. G. (2021). Deep reinforcement learning at the edge of the statistical precipice. *CoRR*. https://doi.org/10.48550/arXiv.2108.13264

Akbari, M., Asadi, P., Besharati Givi, M., & Khodabandehlouie, G. (2014). Artificial neural network and optimization. *Advances in Friction-Stir Welding and Processing*, 543–599. https://doi.org//10.1533/9780857094551.543

Alhammadi, H. Y., & Romagnoli, J. A. (2004). Process design and operation. *The Integration of Process Design and Control*, 264–305. https://doi.org//10.1016/s1570-7946(04)80063-4

Almasan, P., Suárez-Varela, J., Rusek, K., Barlet-Ros, P., & Cabellos-Aparicio, A. (2022). Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Computer Communications*. https://doi.org//10.1016/j.comcom.2022.09.029

Alvarez-Vázquez, L. J., García-Chan, N., Martínez, A., & Vázquez-Méndez, M. E. (2008). Multi-objective pareto-optimal control: An application to wastewater management. *Computational Optimization and Applications*, *46*(1), 135–157. https://doi.org//10.1007/s10589-008-9190-9

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., & Mané, D. (2016). Concrete problems in ai safety. *ArXiv*, *abs/1606.06565*. Retrieved March 16, 2024, from https://api.semanticscholar.org/CorpusID:10242377

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., & Hochreiter, S. (2019). Rudder: Return decomposition for delayed rewards. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., & Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. *CoRR*, *abs/2003.13350*. Retrieved March 16, 2024, from https://arxiv.org/abs/2003.13350

Barrett, L., & Narayanan, S. (2008). Learning all optimal policies with multiple criteria. *CiteSeer X (The Pennsylvania State University)*. https://doi.org//10.1145/1390156.1390162

Barto, A. G. (1997). Chapter 2 - reinforcement learning. In O. Omidvar & D. L. Elliott (Eds.), *Neural systems for control* (pp. 7–30). Academic Press. https://doi.org/10.1016/b978-012526430-3/50003-9

Bechtle, S., Rai, A., Lin, Y., Righetti, L., & Meier, F. (2019). Curious ilqr: Resolving uncertainty in model-based rl. *ArXiv*, *abs/1904.06786*. Retrieved March 16, 2024, from https://api.semanticscholar.org/CorpusID:119287289

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *CoRR*, *abs/1606.01540*. Retrieved March 16, 2024, from http://arxiv.org/abs/1606.01540

Brys, T., Harutyunyan, A., Vrancx, P., Taylor, M. E., Kudenko, D., & Nowe, A. (2014). Multi-objectivization of reinforcement learning problems by reward shaping. *2014 International Joint Conference on Neural Networks (IJCNN)*, 2315–2322. https://doi.org/10.1109/IJCNN.2014.6889732

Castelletti, A., Pianosi, F., & Soncini-Sessa, R. (2008). Water reservoir control under economic, social and environmental constraints. *Automatica*, *44*(6), 1595–1607.

Cheng, X. (2022). Learning classifier systems for multi-objective reinforcement learning problems. https://doi.org/10.26686/wgtn.21568983

Devidze, R., Kamalaruban, P., & Singla, A. (2022). Exploration-guided reward shaping for reinforcement learning under sparse rewards. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems* (pp. 5829–5842, Vol. 35). Curran Associates, Inc. Retrieved March 16, 2024, from https://proceedings.neurips.cc/paper_files/paper/2022/file/266c0f191b04cbbbe529016d0edc847e-Paper-Conference.pdf

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016, June). Benchmarking deep reinforcement learning for continuous control. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning* (pp. 1329–1338, Vol. 48). PMLR. Retrieved March 16, 2024, from https://proceedings.mlr.press/v48/duan16.html

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., & Hester, T. (2021). Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*. https://doi.org//10.1007/s10994-021-05961-4

Eschmann, J. (2021). Reward function design in reinforcement learning. *Studies in Computational Intelligence*, 25–33. https://doi.org/10.1007/978-3-030-41188-6_3

Felten, F., Alegre, L. N., Nowe, A., Bazzan, A. L. C., Talbi, E. G., Danoy, G., & da Silva, B. C. (2023). A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. Retrieved March 16, 2024, from https://openreview.net/forum?id=jfwRLudQyj

Gábor, Z., Kalmár, Z., & Szepesvári, C. (1998). Multi-criteria reinforcement learning. *ICML*, *98*, 197–205.

Gobbi, M., Levi, F., Mastinu, G., & Previati, G. (2014). On the analytical derivation of the pareto-optimal set with applications to structural design. *Structural and Multidisciplinary Optimization*, *51*(3), 645–657. https://doi.org/10.1007/s00158-014-1152-5

Gunantara, N. (2018). A review of multi-objective optimization: Methods and its applications (Q. Ai, Ed.). *Cogent Engineering*, *5*(1). https://doi.org/10.1080/23311916.2018.1502242

Gupta, A., Pacchiano, A., Zhai, Y., Kakade, S. M., & Levine, S. (2022). Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2210.09579

Hare, J. (2019). Dealing with sparse rewards in reinforcement learning. *CoRR*, *abs/1910.09281*. https://doi.org/10.48550/arXiv.1910.09281

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., Howley, E., Irissappane, A. A., Mannion, P., Nowé, A., Ramos, G., Restelli, M., Vamplew, P., & Roijers, D. M. (2022). A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, *36*(1). https://doi.org/10.1007/s10458-022-09552-y

Knox, W. B., Allievi, A., Banzhaf, H., Schmitt, F., & Stone, P. (2023). Reward (mis)design for autonomous driving. *Artificial Intelligence*, *316*, 103829. https://doi.org/10.1016/j.artint.2022.103829

Kusari, A., & How, J. P. (2020). Predicting optimal value functions by interpolating reward functions in scalarized multi-objective reinforcement learning. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 7484–7490. https://doi.org/10.48550/arxiv.1909.05004

Lee, R., Sun, B., Hajiesmaili, M., & Lui, J. C. (2024). Online search with predictions: Pareto-optimal algorithm and its applications in energy markets. *The 15th ACM International Conference on Future and Sustainable Energy Systems*, 50–71.

Minsky, M. (1995). Steps toward artificial intelligence. *Proceedings of the IRE*, *49*(1), 8–30. https://doi.org/10.1109/JRPROC.1961.287775

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *CoRR*. https://doi.org/10.48550/arXiv.1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org//10.1038/nature14236

Moore, A. W. (1990). *Efficient memory-based learning for robot control* [Doctoral dissertation, University of Cambridge].

Natura, B., Neuwohner, M., & Weltge, S. (2022). The pareto cover problem. In S. Chechik, G. Navarro, E. Rotenberg, & G. Herman (Eds.), *30th annual european symposium on algorithms (esa 2022)* (80:1–80:12, Vol. 244). Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Ng, A., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *International Conference on Machine Learning*. Retrieved March 16, 2024, from https://api.semanticscholar.org/CorpusID:5730166

Ngatchou, P., Zarei, A., & El-Sharkawi, A. (2005). Pareto multi objective optimization. *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, 84–91. https://doi.org/10.1109/ISAP.2005.1599245

Nguyen, M. T., & Cao, T. (2019). A multi-method approach to evaluate land combat vehicle system. *International Journal of Applied Decision Sciences*, *12*(4), 337–337. https://doi.org//10.1504/ijads.2019.102639

Nguyen, T. T., Nguyen, N. D., Vamplew, P., Nahavandi, S., Dazeley, R., & Lim, C. P. (2020). A multi-objective deep reinforcement learning framework. *Engineering Applications of Artificial Intelligence*, *96*, 103915. https://doi.org/10.1016/j.engappai.2020.103915

Pérez-Gil, Ó., Barea, R., López-Guillén, E., Bergasa, L. M., Gómez-Huélamo, C., Gutiérrez, R., & Díaz-Díaz, A. (2022). Deep reinforcement learning based control for autonomous vehicles in carla. *Multimedia Tools and Applications*, *81*(3), 3553–3576. https://doi.org//10.1007/s11042-021-11437-3

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, *22*(1), 1–8. Retrieved March 16, 2024, from http://jmlr.org/papers/v22/20-1364.html

Roijers, D. M., & Whiteson, S. (2017, January). *Multi-objective decision making*. Morgan & Claypool Publishers. https://doi.org//10.1007/978-3-031-01576-2

Röpke, W., Reymond, M., Mannion, P., Roijers, D. M., Nowé, A., & Rădulescu, R. (2024). Divide and conquer: Provably unveiling the pareto front with multi-objective reinforcement learning. *arXiv (Cornell University)*. https://doi.org//10.48550/arxiv.2402.07182

Singh, V., Chen, S.-S., Singhania, M., Nanavati, B., kar, A. k., & Gupta, A. (2022). How are reinforcement learning and deep learning algorithms used for big data based decision making in financial industries–a review and research agenda. *International Journal of Information Management Data Insights*, *2*(2), 100094. https://doi.org//10.1016/j.jjimei.2022.100094

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Second). The MIT Press. Retrieved March 16, 2024, from http://incompleteideas.net/book/the-book-2nd.html

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., & Dekker, E. (2010). Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, *84*(1-2), 51–80. https://doi.org/10.1007/s10994-010-5232-5

Van Moffaert, K., Drugan, M. M., & Nowé, A. (2013). Scalarized multi-objective reinforcement learning: Novel design techniques. *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 191–199. https://doi.org/10.1109/ADPRL.2013.6615007

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., & Ba, J. (2020). Benchmarking model-based reinforcement learning. Retrieved March 16, 2024, from https://openreview.net/forum?id=H1lefTEKDS

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* [Doctoral dissertation, University of Cambridge].

Zhu, B., Dang, M., & Grover, A. (2023). Scaling pareto-efficient decision making via offline multi-objective RL. *The Eleventh International Conference on Learning Representations*. Retrieved March 16, 2024, from https://openreview.net/forum?id=Ki4ocDm364