UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

GIOVANI DA SILVA

Analysis and improvements of Multi Objective Reinforcement Learning Algorithms based on Pareto Dominating Policies

Work presented in partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering

Advisor: Prof. Anderson Rocha Tavares

Porto Alegre
September 2023

“I care too much, wanna share too much, in my head too much
I shut down too, I ain’t there too much
I’m a complex soul, they layered me up
Then broke me down, and morality’s dust, I lack in trust
This time around, I trust myself
Please everybody else but myself
All else fails, I was myself
Outdone fear, outdone myself”

— KENDRICK LAMAR

ACKNOWLEDGMENTS

I would like to start by expressing my deep gratitude to my parents, Lani and Nelson. Thank you for being incredibly patient with me over these 5 years, for being understanding and supportive, even when I expressed my love or desires in an unconventional way. Know that a great part of who I am today and the constantly evolving person I am becoming is a result of the unconditional love and support I received from you. Even with my flaws, you have always stood by my side, and my greatest wish has always been and continues to be to make you proud. Without a doubt, you are the best parents in the world.

I am immensely grateful to my girlfriend, Jaqueline, for facing so many challenges by my side, for her understanding and for tolerating my quirks and my sometimes "in another world" way of seeing things. We make quite a team, and these last 5 years together, even when sometimes distant, have been marked by mutual care and affection. Your presence in my life is an invaluable gift.

I cannot fail to thank my second mothers as well, Aunt Tata and Grandma Evonir. The love you've consistently shown at every encounter is essential to the person I've become, and I'm immensely grateful to have you in my life. The way you keep our family united and care for everyone's well-being is a constant inspiration to me. I love you very much.

My sister, Gabi, deserves a special mention. Her presence and support in difficult times were crucial in keeping us united as a family. Moreover, sharing experiences that only siblings who pretend to hate each other understand has always been and continues to be funny.

To my brother-in-law, Bruno, thank you for the years of friendship and for being my eternal FIFA opponent. To my uncles, Aunt Léia, Uncle Lauro, Uncle Luiz, and Uncle Doda, you are inexhaustible sources of inspiration and affection. Nothing of what I am would be possible without the example and love I received from each of you.

My cousin Mateus, you have always been an inspiration and one of my best friends, regardless of the physical distance. Know that your presence is essential for me to be here, and I hope to see you again soon.

To my cousins Fê and Bruno, it's always a pleasure to share happy moments with you and strengthen our family bonds.

To the family friends, Márcia, Aunt Tereza, and Aunt Romilda, you played important roles in my upbringing, and I wish the best for each of you.

To school friends, Victor, I thank you for the runs at "Feijão" and for our almost philosophical discussions ranging from soccer to veganism. Leo and Cassiano, thank you for understanding my ways and always being present to talk about anything and laugh along with me. Nadine, for the countless outings and random conversations, you are a special friend.

To my futsal friends over the years, Caue, Pedro, Natan, João, and Pepe, those were some great years playing and winning with you. It's always good to see you.

To my friends Bruno and Marcelo, thank you very much for the hours of conversation, talking nonsense. It's always a pleasure.

To my friend Ricardo, I'm grateful for the years of friendship. Even though we're different in so many ways, our mutual understanding is remarkable. The fact that both of us are "small-town" guys who came to Porto Alegre at 17, full of fear and uncertainties, might have brought us even closer.

To my college friends, you were essential in making the field of computing more welcoming and less lonely. I thank you for the help, the laughter, and the explanations on the subjects.

To the friends from Dunk Park (Anderson, Antônio, Careca, Dani, Dudu, Edu, Fe, Gian, Mi) and RB Cãopiladores, practicing the sports I love was fundamental to my physical and mental well-being over these years. Thank you for sharing these moments with me; I'm always happy to see you.

To the professionals who helped me, like the psychologist, psychiatrist, and nutritionist, you were crucial for my health and well-being over these 5 years.

To the professors at the Institute of Informatics at UFRGS, over these 5 years, you made me fall in love with computing. I entered as a student who didn't know what an "if" was, and today I leave graduation, passing into the master's in the field of artificial intelligence. If you had told me this 5 years ago, I would do it all over again. The young Giovani who watched "I, Robot," "Ex Machina," and other works of science fiction would be proud. Special thanks to professors Renata Galante, Ribas, Mariana, and Bampi. Each, in your own way, made me like topics completely diverse in the field of computing. I'd also like to thank Professor Gabriel Ramos for being my co-advisor during the period of scientific initiation. His patience, teaching, and helpfulness were crucial. He and Professor Anderson made me love the area of AI and reinforcement learning even more. So, I am very grateful. Also, I would like to thank FAPESP (Fundação de Amparo a

Pesquisa do Estado de São Paulo) for the resources and the support in the COMA project during my scientific initiation.

To my advisor, Professor Anderson, thank you for the incredible patience, time, teachings, and discussions for the sake of science that we had in the last year and a half. I couldn't have chosen a better advisor. Your teachings have certainly made me a better scientist and a lover of AI.

Finally, I want to thank myself. In these years, I went through moments of uncertainty and found myself again with calmness and wisdom. The ability to make difficult decisions and remain calm is a personal achievement that I value and will continue to improve.

In summary, I am grateful to all who are part of my life, as each of you has contributed to shaping who I am today. Your presence and support have been invaluable, and I feel blessed to have each of you on my journey. These 5 years of graduation were special, challenging, but incredible.

ABSTRACT

In today's complex optimization landscape, challenges often transcend the pursuit of a solitary objective, instead requiring the simultaneous consideration of multiple, sometimes conflicting, goals. This complexity has given rise to the field of Multi-Objective Optimization. In recent years, researchers have begun to integrate these multi-objective approaches into Reinforcement Learning, leading to the emergence of Multi-Objective Reinforcement Learning (MORL). The field is gaining traction, especially due to the capabilities of model-free Reinforcement Learning algorithms. Essentially, these model-free MORL algorithms strive to balance multiple, often conflicting, objectives without necessitating prior knowledge of the environment. This thesis provides an in-depth analysis of model-free MORL algorithms anchored in Pareto Dominating Policies (PDP), specifically focusing on two key algorithms: Pareto Q-Learning (PQL) and Pareto Deep Q-Networks (PDQN), these algorithms were selected for their model-free characteristics and their resemblance to well-known reinforcement learning algorithms like Q-Learning and Deep Q-Networks.

This study features implementations from scratch of both the PQL and PDQN algorithms. It evaluates the performance of PQL in the Deep Sea Treasure environment and assesses PDQN in both the Deep Sea Treasure and a simulated urban traffic setting. This research identifies common challenges, such as the generation of non-optimal policies and the difficulties associated with managing large state spaces.

Our findings reveal that the application of the PDQN algorithm in real-world scenarios, such as Gym City Flow (ZHANG et al., 2019), has led to no improvements, thereby demonstrating their inefficacy. To address these challenges, this work proposes enhancements to the PDQN algorithm and introduces a new MORL technique based on Pareto-Dominating Actions. Preliminary tests indicate that this innovative approach shows promise in enhancing the effectiveness of MORL algorithms.

The primary contributions of this work lie in its examination of the current state of MORL algorithms based on Pareto Dominating Policies: discussing their architecture, their challenges and their possible improvements, while also testing their effectiveness in MO scenarios. By doing that, we are trying to shed light on their inherent limitations and challenges. In light of these limitations, we propose enhancements to the PDQN algorithm through an innovative approach that holds the potential to establish an effective approach to MORL algorithms in the future. This work serves as both a critical review of existing

methodologies and a forward-looking exploration of the future landscape of MORL centered on Pareto Optimality.
Keywords: Reinforcement Learning, Multi-objective, Artificial Intelligence, Multi-Objective Reinforcement Learning, Pareto Dominating Policies, Pareto Deep Q-networks.

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
Env	Environment
ML	Machine Learning
MO	Multi-Objective
MORL	Multi-Objective Reinforcement Learning
NN	Neural-network
PDQN	Pareto Deep Q-Networks
PQL	Pareto Q-Learning
RL	Reinforcement Learning
PDP	Pareto Dominating Policies

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

In many real-world scenarios, optimization extends beyond refining a single criterion to balancing multiple, often conflicting, objectives. For instance, consider the design of a car where engineers must balance fuel efficiency, safety, speed, and cost. Such complexity gives rise to multi-objective optimization, where the challenge lies in identifying solutions that offer the best rewards among various goals. Such solutions are termed Pareto optimal or non-dominated solutions, and their collective representation is known as the Pareto Frontier—a visual depiction of the most favorable trade-offs. While a deeper exploration of these concepts will follow, they set the stage for the subsequent discussion on Multi-Objective Reinforcement Learning, where agents navigate a multifaceted objective landscape.

Model-free Multi-Objective Reinforcement Learning (MORL) algorithms have gained significant attention in recent years due to their independence from a model of the environment, making them highly relevant for solving real-world problems. These problems span diverse domains such as traffic environments (OMAR; GOMAA, 2014), energy management (QIN et al., 2020), healthcare (JALALIMANESH et al., 2017) and even virtual-world applications, like video games (SHEN et al., 2020). Among the various approaches to MORL, algorithms that utilize Pareto Dominating Policies (PDP) have become increasingly popular due to their effectiveness in optimizing multiple objectives. These algorithms aim to create a policy or decision that improves one objective's outcomes without negatively impacting the outcomes of any other objectives.

This work presents a comprehensive analysis of MORL algorithms based on PDP. We assess the strengths and limitations of two existing algorithms: Pareto Q-Learning(PQL) (Section 2.3.1) and Pareto Deep Q-Networks (PDQN) (Section 2.3.2). Additionally, we evaluate both algorithms in the Deep Sea Treasure Environment (Section 3.1 and Section 3.2), also specifically test PDQN in a simulated urban traffic environment of Traffic Light Control (Section 3.3). We identify common issues such as suboptimal optimization and difficulty in handling large states space. Furthermore, building upon this analysis, we propose enhancements to PDQN (Chapter 4) and also introduce a novel MORL technique (Section 4.2). This innovative approach is rooted in Pareto-Dominating Actions, enumerated through an on-policy reinforcement learning algorithm, as opposed to PDP. Notably, our preliminary tests showcase promising results with this technique, highlighting its potential to outperform existing methods.

The primary aim of this study is to make a meaningful contribution to the field by offering valuable insights into the present landscape of Multi-Objective Reinforcement Learning (MORL) algorithms rooted in Pareto Dominating Policies (PDP). By thoroughly examining the current state of these algorithms(their architecture and effectiveness), we shed light on their inherent challenges and limitations.

Beyond the analysis, this work also presents a forward-looking perspective by proposing an innovative approach to address these challenges. We strive to offer novel solutions to the issues encountered in this context, using the concept of non-dominated actions. By exploring this alternative strategy, we seek to further enrich the MORL domain, potentially paving the way for more effective and robust solutions.

The central question of this research revolves around assessing the performance of MORL algorithms based on PDP in varying environments and identifying avenues for their optimization. Our hypothesis led us to implement and rigorously test these algorithms, revealing key areas for improvement such as architectural tweaks and the inclusion of non-dominated actions. The study's contributions are an exhaustive analysis of existing algorithms, the introduction of a new approach to MORL based on non-dominated actions, and a critical analysis of the PDQN in a real-world setting. By addressing these facets, this research not only fills existing gaps but also charts a course for future developments in the MORL based on the Pareto Optimality landscape.

This work is structured as follows:

- **Chapter 2: Background**
  This chapter establishes the foundational concepts of RL. It progresses into Multi-Objective Optimization, elucidating its importance and inherent challenges. The discourse then shifts to MORL, emphasizing its intricacies and distinctions. The chapter concludes with a comprehensive overview of the algorithms central to this monograph.
- **Chapter 3: Results and Algorithm Evaluation**
  This chapter delves into the practical application of the introduced algorithms. It commences with the deployment of the algorithm in the Deep Sea Treasure Environment, underscoring its effectiveness and versatility. Subsequently, the narrative transitions to a real-world context, illustrating the potential applicability of the PDQN algorithm.
- **Chapter 4: Improvements to the PDQN Algorithm**
  This concluding chapter contemplates enhancements and refinements to the existing

methodology. It provides insights into the current approach's constraints and proposes directions for future exploration. A fresh perspective on tackling the problem is introduced, suggesting innovative avenues in the domain of MORL.

2 BACKGROUND

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a Machine Learning technique that enables agents to learn by interacting with their environment through trial and error. This environment is typically represented as a Markov Decision Process (MDP), a structured framework for decision-making where outcomes are partly random and partly under the control of a decision-maker. The MDP is defined by $M = (S, A, T, R, \gamma)$. Here, S denotes the set of environment states, representing the various situations or positions the agent can encounter. A signifies the set of actions available to the agent in any given state. The transition function, $T : S \times A \times S \to [0, 1]$, determines the probability of the agent transitioning from one state to another after performing a specific action. The immediate reward function, $R : S \to \mathbb{R}$, assigns a reward value to each state-action pair. Lastly, the discount factor γ quantifies the importance of future rewards compared to immediate ones, guiding the agent in evaluating the long-term consequences of its actions.

The objective of an RL agent is to find a policy that maps the state to an action that yields the best total reward in the long run. To accomplish this, the agent explores the environment, takes actions based on the current state, receives feedback (reward), and utilizes it to adjust its policy on-the-fly to optimize it and maximize the total reward.

Two key factors of RL algorithms are exploration and exploitation. Exploration involves trying out actions to learn about the environment and discover ways to achieve potentially higher rewards, ultimately aiding in finding the optimal policy. Exploitation, on the other hand, involves taking actions that the agent knows will yield the highest value.

There are several techniques available to address the trade-off between exploration and exploitation in RL algorithms, the most popular one being the ϵ-greedy technique. This technique explores by selecting a random action with probability (ϵ) and exploits the action with the highest estimated value with a probability of 1-ϵ.

2.1.1 Q-learning

Q-Learning (QL) is a widely used RL algorithm, primarily due to its model-free nature, which means it doesn't require a model of the environment to learn. Another distinctive feature is its off-policy approach, allowing it to discover the optimal policy even when following any exploratory strategy. In QL, the agent aims to learn the optimal action-selection policy for an MDP. This learning process focuses on maximizing cumulative rewards over time, and it achieves this without relying on a predefined environment model In QL, the agent iteratively refines its knowledge. At each timestep, it observes the current state s, chooses an action a, and receives the resulting reward r and next state s'. The agent then uses this experience tuple $< s, a, r, s' >$ to update its Q-value estimates, as illustrated in Eq.2.1.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a) \right] \ . \tag{2.1}$$

In this equation, $Q(s,a)$ represents the estimated Q-value of taking an action in the current state. The term $\max_{a'} Q\left(s', a'\right)$ denotes the highest Q-value estimate for all potential actions a' in the next state. The parameters α and γ are the learning rate and discount factor, respectively. While α determines the influence of new information in the Q-value update, γ balances the significance of immediate versus future rewards.

The Q-values are updated iteratively until they converge to their optimal values (if certain conditions are obeyed). Once the Q-values have converged, the agent can follow the optimal policy by selecting the action with the highest Q-value for each state.

To store and update these Q-values, a table is typically used. It is a table that stores the estimated Q-values for each state-action pair.

The table approach is sufficient for small and discrete environments. However, as the number of state-action pairs increases, the table size grows exponentially, making it impractical for large and complex environments. In order to address this issue, approximate methods such as neural networks can be used. This is discussed in the next section.

The description of QL can be seen in Algorithm 1.

Algorithm 1: Tabular Q-learning

```
Input: Initial state $s$, learning rate $\alpha$, discount factor $\gamma$, exploration
  parameter $\epsilon$, number of episodes $E$;
Output: Learned Q-value function $Q(s, a)$;
Initialize $Q(s, a)$ arbitrarily for all state-action pairs;
for $t = 1$ to $E$ do
    Initialize state $s$;
    while $s$ is non-terminal do
        Choose action $a$ from state $s$ using an $\epsilon$-greedy policy derived from $Q$;
        Take action $a$ and observe next state $s'$ and reward $r$;
        Update $Q(s, a)$ using Eq. 2.1;
        Update $s \leftarrow s'$;
    end
end
```

2.1.2 Deep Q-Network

Deep Q-Network (DQN) (MNIH et al., 2015) is an algorithm used in Reinforcement Learning that combines Q-Learning with deep neural networks. DQN is particularly effective in handling high-dimensional environments, which can be challenging for tabular Q-Learning algorithms.

The key idea behind DQN is to use deep neural networks to approximate the Q function by leveraging a combination of supervised learning and RL.

The set of experiences that the DQN algorithm uses is called replay memory. This replay memory stores experience tuples $< s, a, r, s' >$, which consist of the agent's current state s, the action taken a, the immediate reward received r, and the resulting next state s'. During training, the algorithm samples experiences randomly from this buffer, which allows the agent to learn from a diverse set of experiences and avoid overfitting to recent experiences.

The training of the neural network consists of minimizing the squared error between its predictions and the target Q-values. This is achieved through backpropagation, with the loss function calculated as the squared difference between the target Q-value and the predicted Q-value, as shown in Eq.2.2.

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')} \sim U(D)[(y_i - Q(s,a;\theta_i))^2] \tag{2.2}$$

Where $y_i = r + \gamma \max_{a'} Q(s', a'; \theta_i^-)$ is the target Q-value, θ_i are the parameters of the Q-network at iteration i, $U(D)$ is the uniform random sampling from the replay memory D.

In addition to the replay memory, DQN uses a target network to stabilize the training process. This target network, with parameters θ^-, is used to stabilize the training process. This is done because when updating the Q-network, the Q values become highly correlated, leading to the network getting stuck in a local minimum and failing to converge to the optimal solution. To address this issue, the target network is periodically updated with the Q-network values, with the frequency determined by a hyper-parameter C. This approach allows the Q-network to learn from a more stable set of Q values since the target network is updated less frequently than the Q-network. The target network plays a critical role in helping the network avoid correlation problems and gradually adjust to changes in the Q values over time. Overall, the use of a target network is an important feature of DQN that helps the algorithm converge.

The complete algorithm of DQN with replay memory is shown in Algorithm 2.

Algorithm 2: Deep Q Network with replay memory

```
Data: Replay memory $D$ with capacity $B$, action-value function $Q$ with
      weights $\theta$, and target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for episode = $1$ to $E$ do
    Get initial state $s_t$;
    for $t = 1$ to $T$ do
        With probability $\epsilon$ select a random action $a_t$, otherwise select
          $a_t = \operatorname{argmax}_a Q(s_t, a; \theta)$;
        Execute action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$;
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $D$;
        Sample random minibatch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $D$;
        Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$;
        Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect
          to the network parameters $\theta$;
        if copy to Target then
            $\hat{Q} \leftarrow Q$;
        end
    end
end
```

2.2 Multi-objective optimization

Multi-objective problems refer to optimization problems with more than one objective function that need to be optimized simultaneously. In such problems, there is usually a trade-off between the objectives, making it difficult to find a single optimal solution that satisfies all the objectives simultaneously. To cope with multiple objectives, this work uses the concept of Pareto efficiency, also known as Pareto optimality. Pareto optimality arises in situations where no allocation of resources can make one individual better off without causing harm to another (LUC, 2008, p.481). This principle can be applied to multi-objective optimization, as we elaborate next.

In multi-objective problems, the ideal solution would be one that outperforms all others in every objective, but this is often not feasible. Instead, some solutions stand out as clearly superior to others, as they excel in one objective without performing much worse in any other. These are known as non-dominated solutions, which form the Pareto Frontier. Illustrated in Fig. 2.1.

However, among non-dominated solutions, a dilemma arises when switching solutions: improving the value of one objective often leads to a decrease in another objective. Despite this challenge, the objective of a multi-objective optimization algorithm is to accurately identify and enumerate all the points of the Pareto Frontier.

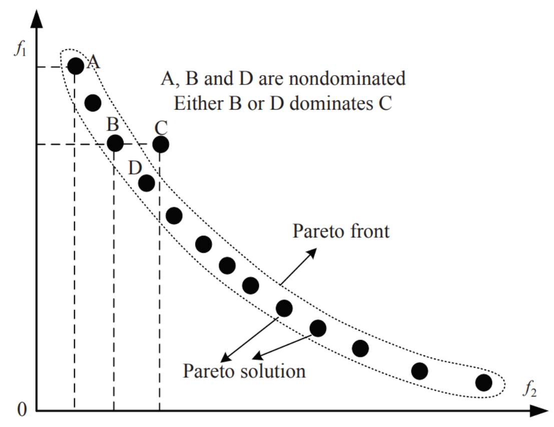


Figure 2.1 – Example of Pareto Frontier for a 2-objective minimization problem. Here, f_1 and f_2 are the objective functions. Points A, B, and D are better in one objective without compromising too much of the second one compared to C. Therefore, A, B, and D are part of the Pareto Frontier and dominate C. As one moves along the Pareto Frontier, improvement in one objective results in a trade-off, worsening the other objective. (CAI; LIJIA; GONG, 2014)

2.3 Multi objective Reinforcement Learning

RL traditionally focuses on maximizing a scalar reward. In scenarios where multiple objectives need to be optimized simultaneously, the standard RL approach might not be sufficient. This is where Multi-objective Reinforcement Learning (MORL) comes into play.

MORL is a type of RL where the agent receives a vector-valued reward function instead of the scalar reward, as is the case in single-objective RL. Specifically, the reward function is defined as $R : S \rightarrow \mathbf{R}^d$, where d represents the number of objectives. The primary challenge in MORL is not just to obtain the maximum reward for a given state-action pair but to find an optimal policy that can simultaneously maximize rewards across multiple objectives. This becomes particularly challenging when these objectives conflict or are antagonistic to each other. For instance, in a driving scenario, one might want to reach a destination quickly (objective 1) but also want to save fuel (objective 2). These objectives can sometimes be opposite of each other, making it complex to find a balanced solution.

A fundamental aspect of MORL is its reliance on the concept of the Pareto Frontier, which we discussed in Section 2.2. In the context of MORL, the Pareto Frontier helps identify solutions that balance multiple objectives effectively. There are numerous methods available for determining the Pareto Frontier in optimization problems. Primarily, what sets MORL apart in many control problems is its intrinsic ability to learn without depending on prior knowledge of the environment's dynamics (due to its many model-free implementations).

In the realm of MORL, there are predominantly two approaches:

- **Inner-loop Methods:** These algorithms adapt traditional single-objective RL techniques to handle vector operations. Their main objective is to simultaneously find solutions that optimize all objectives. Essentially, they transform standard RL algorithms to accommodate multiple objectives directly within the learning process (HAYES et al., 2022).
- **Outer-loop Methods:** These algorithms adopt a different strategy. They repeatedly employ a single-objective RL algorithm, but each iteration applies it to a different scalarization of the reward function. The goal is to unearth optimal policies by exploring various scalarized versions of the objectives (HAYES et al., 2022).

Several MORL algorithms use different techniques to identify non-dominated policies. Algorithms like Pareto Q-learning (MOFFAERT; NOWé, 2014) and Multi-objective Actor-Critic (MOAC) (REYMOND et al., 2021) employ Pareto dominance to identify solutions. On the other hand, algorithms like Multi-objective Deep Deterministic Policy Gradient (MODDPG) (YU et al., 2021) utilize scalarization methods to convert multiple objectives into a single composite objective. This work delves deeper into algorithms that emphasize Pareto Dominating Policies, specifically Pareto Q-Learning (Section 2.3.1) and Pareto Deep Q-Networks (Section 2.3.2). Distinctively, these algorithms do not need a pre-defined scalarization function to balance their objectives. Instead, they directly optimize the vector-valued policy. This approach not only obviates the need for manual tuning but also ensures a thorough exploration of the solution space, fostering better generalization across diverse environments.

2.3.1 Pareto Q-Learning

Pareto Q-Learning (PQL)(MOFFAERT; NOWé, 2014), is an extension of the traditional single-objective Q-learning algorithm, tailored to manage multiple objectives. Characterized as an inner-loop approach, PQL maintains a set of non-dominated solutions for each state-action pair combination. Initially, this set is empty but gets updated using the non-dominated set derived from the next state across all potential actions. The expected returns for a specific state-action combination are articulated as the vector addition of the immediate reward and the non-dominated set of prospective returns, this is represented as the $Q_{set}(s,a)$. This can be seen in Eq. 2.3.

$$Q_{set}(s,a) \leftarrow \bar{R}(s,a) \oplus \gamma ND_t(s,a) \tag{2.3}$$

Where $\bar{R}$ is the average immediate reward, ND_t is the set of non-dominated future returns, γ is the discount factor, and $\oplus$ is an operator that adds a vector to a set of vectors.

To better understand the operation of $\oplus$, let's consider a numerical example. Suppose $\overline{R}(s,a)$ (the return vector of rewards for a given state-action pair) is $[16,-28]$. This vector will be added to each of the non-dominated vectors already found by ND$_t$, for example, $[1,-1]$, $[3,-5]$, and $[124,-21]$, each multiplied by the value of γ. For simplicity, let's assume $\gamma = 1$ in this example. After this addition, we get the vectors $[17,-29]$, $[19,-33]$, and $[140,-49]$. A simple illustration of the $\oplus$ operator is shown in Fig. 2.2.

After that, we check with the Frontier Points operator to see if it could be a new non-dominated point in the Frontier.

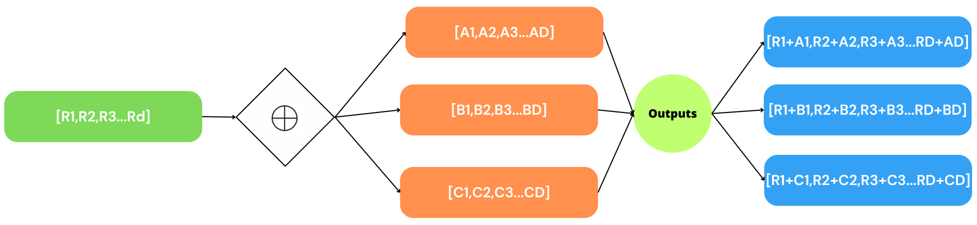

[R1,R2,R3...Rd]
⊕
[A1,A2,A3...AD]
[B1,B2,B3...BD]
[C1,C2,C3...CD]
Outputs
[R1+A1,R2+A2,R3+A3...RD+AD]
[R1+B1,R2+B2,R3+B3...RD+BD]
[R1+C1,R2+C2,R3+C3...RD+CD]

Figure 2.2 – Illustration of the ⊕ operator. Source: The Author.

When an agent receives a new reward r from the environment for taking action a in state s, the immediate reward estimate $\bar{R}(s,a)$ is updated as follows:

$$\bar{R}(s,a) \leftarrow \bar{R}(s,a) + \frac{r - \bar{R}(s,a)}{n(s,a)} \tag{2.4}$$

Where $n(s,a)$ is the number of times the state-action pair (s,a) has been visited. The Equation 2.4 is an incremental calculation of the simple arithmetic mean and updates the estimate of the average reward for the state-action pair (s,a).

ND_t represents the set of non-dominated future returns, and it is updated using the non-dominated set of all the $Q_{set}(s,a)$'s of the next state (MOFFAERT; NOWé, 2014), This is done as follows:

$$ND_t(s,a) \leftarrow FP\left(\cup_{a' \in A} Q_{set}\left(s',a'\right)\right) \tag{2.5}$$

Or it can be calculated as follows, where FP (Frontier Points)[1] is a function that keeps only the non-dominated solutions from the set:

$$ND_t(s,a) \leftarrow FP\left(\cup_{a' \in A} \bar{R}\left(s',a'\right) \oplus \gamma ND_t\left(s',a'\right)\right) \tag{2.6}$$

[1]In the reference papers (MOFFAERT; NOWé, 2014) and (REYMOND; NOWE, 2019), this operator is named ND, but for the sake of reader clarity, we chose to rename it due to the presence of NDt (Non-dominated estimator) in this Section and in Section 2.3.2.

To add clarity to the FP operator, let's consider a numerical example. Suppose we have a set of points with values $[1, -1]$, $[1, -10]$, $[2, -5]$, $[3, -7]$, $[32, -15]$, $[2, -3]$, and $[3, -32]$. Let's consider the first objective as a treasure reward we are trying to maximize and the second as a time penalty we are trying also to maximize(get it closer to zero). The FP operator will evaluate this set of vectors and retain only those that are non-dominated. For instance, the vector $[1, -1]$ is non-dominated as it dominates all other vectors in terms of the second objective (it has the lowest time penalty). The vector $[32, -15]$ is also non-dominated as it dominates all other vectors in terms of the first objective, having the highest reward value. Similarly, the vector $[2, -3]$ is non-dominated because no other vector dominates it in both objectives, as it dominates $[1, -1]$ in terms of reward but loses in time, and it dominates $[32, -15]$ in terms of time but loses in reward. The same applies to the vector $[3, -7]$, which is also non-dominated. Therefore, the Pareto Frontier operator will return these non-dominated vectors $[1, -1]$, $[2, -3]$, $[3, -7]$, and $[32, -15]$, and discard the dominated ones, such as $[1, -10]$, $[2, -5]$, and $[3, -32]$. This example is illustrated in Figure 2.3.

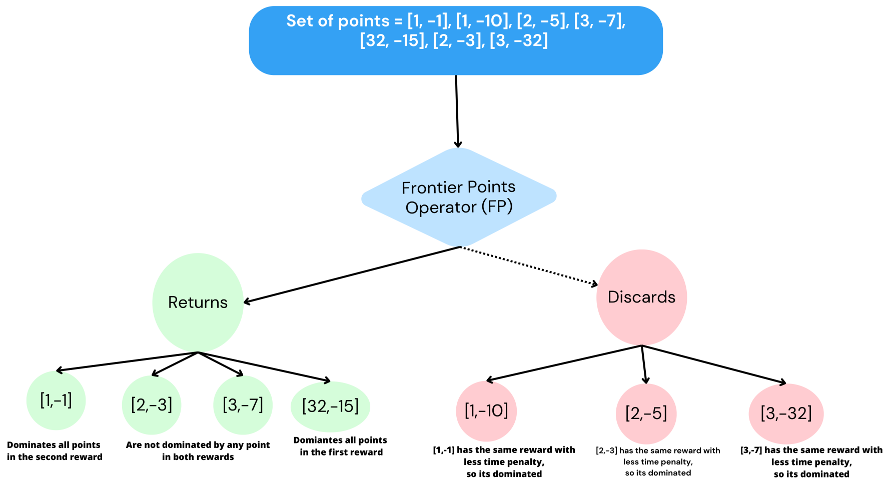

Set of points = [1, -1], [1, -10], [2, -5], [3, -7], [32, -15], [2, -3], [3, -32]
Frontier Points Operator (FP)
Returns
Discards
[1,-1]
[2,-3]
[3,-7]
[32,-15]
[1,-10]
[2,-5]
[3,-32]
Dominates all points in the second reward
Are not dominated by any point in both rewards
Domiantes all points in the first reward
[1,-1] has the same reward with less time penalty, so its dominated
[2,-3] has the same reward with less time penalty, so its dominated
[3,-7] has the same reward with less time penalty, so its dominated

Figure 2.3 – Illustration of the FP operator process. Source: The Author.

Continuing with PQL, as observed, the traditional Q-learning (Eq.2.1) update rule has been adapted in PQL to accommodate vector rewards (Eq. 2.3). Also, the method for selecting the optimal action evaluates its quality using the hypervolume indicator, illustrated in Figure 2.4. This indicator computes the total volume under all the Q-values ($\mathcal{Q}$) relative to a specified reference point, a hyper-parameter that sets the minimum possible return. The action selection is determined by the ϵ-greedy strategy based on the hypervolumes of each action.

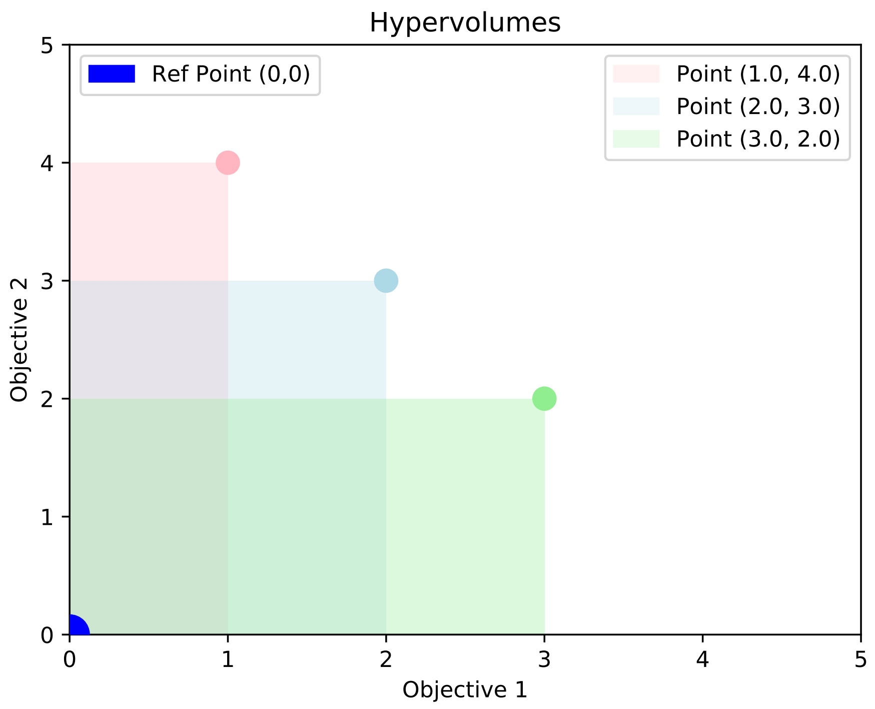


Figure 2.4 – Illustration of the hypervolume evaluator. The Q-values (1,4),(2,3),(3,2) are shown with respect to a reference point (0,0). The hypervolume in this 2D scenario, the hypervolume would be the area between each point and the reference point: 4 for the first point, 6 for the second and 6 for the third point. Source: The Author.

A detailed exposition of PQL is provided in Algorithm 3.

For this work, the PQL algorithm was implemented, tested, and evaluated in the Deep Sea Treasure environment. Some of the results can be seen in Section 3.1. Additionally, the implementation is publicly available online at <https://github.com/GiovaniCenta/MOparetoQlearning>.

2.3.2 Pareto Deep Q-Networks

The Pareto Deep Q-Networks (PDQN) is an innovative approach that fuses the principles of the classic Deep Q-Learning algorithm (Section 2) with the multi-objective strategies and mechanisms of Pareto Q-Learning (Section 2.3.1).

The PQL algorithm, much like its single-objective QL counterpart, faces difficulties in handling large environments. This challenge is inherent in its architecture, which keeps a set of Q-values (or non-dominated points) for each state-action pairing. Such a design can lead to increased computational demands and instability (SUTTON; BARTO, 2018, Chapter 11).

Algorithm 3: Pareto Q-Learning Algorithm (MOFFAERT; NOWé, 2014)

```
Input: Empty sets for each state-action pair: $Q_{set}(s,a) \leftarrow \varnothing$, number of
       steps, $\alpha \in (0,1]$, discount factor $\gamma \in (0,1)$
for each episode $t$ do
    Initialize state $s$
    repeat
        Choose action $a$ from $s$ using a policy derived from $Qset$
        Take action $a$ and observe the next state $s'$ and reward vector $r$
        Update the ND policies of $s'$ in $s$:
            $Q_{set}(s',a') \leftarrow FP(\{Q(s',a') \cup Q_{set}(s',a')) \mid a' \in A$
        Update average immediate rewards:
            $R(s,a) \leftarrow R(s,a) + \frac{1}{n(s,a)}(r - R(s,a))$
        Proceed to next state: $s \leftarrow s'$
    until terminal state is reached;
end
```

To enhance its efficiency, PDQN utilizes neural networks to approximate both the Pareto Frontier and the average immediate reward. By adopting this neural network-based methodology, PDQN can possibly handle large environments. The Q-set within PDQN is derived from Eq. 2.3, with both the immediate reward and the non-dominated points being estimated using neural networks. Subsequent sections will delve into the specifics of how the neural network estimates the immediate reward and the intricacies of the Non-dominated Estimator.

2.3.2.1 Estimating the immediate reward

In RL algorithms, it's essential to draw a clear distinction between two concepts:

- The **immediate reward** that an agent gains when it performs a specific action in a given state.
- The **Q-value** of a state-action pair, which represents its long-term value.

In the context of PDQN, a neural network is employed to estimate the immediate reward. This approach allows the algorithm to approximate a complex, non-linear function that correlates the state-action pair with the reward. The network takes the state-action pair as its input and yields a reward vector, denoted as $r \in \mathbf{R^d}$, with d being the number of objectives.

The primary goal is to train the neural network to reduce the quadratic error between the observed and estimated rewards, similar to a regression task. This approach aims to improve the network's proficiency in formulating optimal policies.

2.3.2.2 Estimating the non-dominated set

In the PDQN approach, alongside approximating the reward R, there's a need to approximate NDt, which represents the non-dominated returns. A significant challenge arises due to the variable size of this set. Specifically, the number of non-dominated points returned varies depending on the state. This variability makes it impractical to use only state and actions as inputs to output a fixed number of points.

One might consider limiting the set to a predefined number of top non-dominated points, say j. However, this introduces another challenge: the sequence in which these j points appear becomes crucial. For consistent network updates, each output neuron must be compared with a consistent target value, given the same input. Any change in the sequence of points would alter the target, potentially disrupting the training process. Furthermore, maintaining this sequence is not feasible. Whenever a new non-dominated point emerges (which is state-dependent), an existing point would need to be removed, leading to a sequence alteration.

To address these challenges, the network is designed to accept not only a state s and action a as input but also $d - 1$ additional values. These values correspond to the rewards associated with each of the first $d - 1$ objectives for the given state-action pair. Specifically, the network takes as input a tuple $(s, a, o_1, \ldots, o_{d-1})$, where s is the current state, a is the current action, and $(o_1, \ldots, o_{d-1})$ are the rewards for the first $d - 1$ objectives for that state-action pair. The network then outputs the value o_d, which is the estimated reward for the last objective. By combining this output o_d with the input values, a single point in $\mathbf{R}^d$ for (s, a) is derived. Adding this to the estimated reward value $\bar{R}$ (as per Eq. 2.5), a single point on the Pareto Frontier is obtained. This procedure is iterated p times, where p is a hyper-parameter denoting the desired number of points on the Pareto Frontier, resulting in p distinct points on the Frontier. This process is illustrated in the diagram of Fig. 2.5.

This method of approximating the Pareto Frontier has some limitations, specifically, the Pareto Frontier for some states may only exist within a subdomain of $\mathbb{R}^{d-1}$, meaning that the predictor may sometimes produce objective values that are not actually possible within that Frontier.

To handle such scenarios, the predictor is specifically trained to be conservative in its predictions. If it encounters points that are likely not part of the Pareto Frontier, it assigns them values that are worse than the smallest possible reward. This approach ensures that even if the predictor makes big mistakes, it leans towards caution, preventing

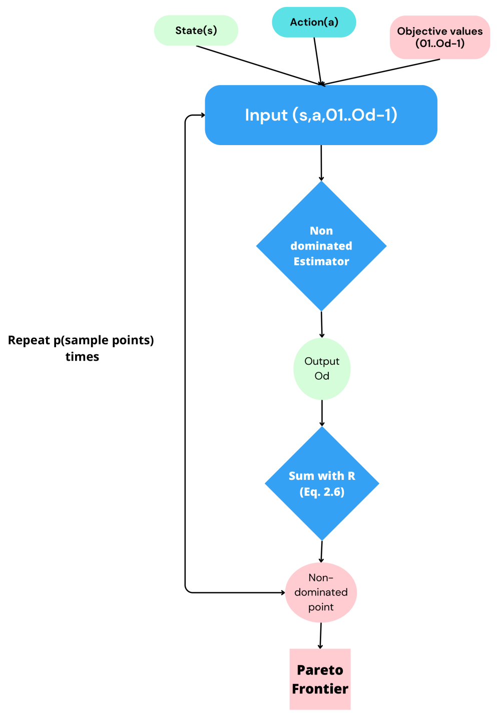


Figure 2.5 – Illustration of the Non-dominated Estimator process, Source: The Author.

overestimation.

For example, let's analyze a Pareto Frontier of the Deep Sea Treasure (Fig 3.1) problem with the points: `points = [(1, -1), (3, -5), (16, -9), (24, -14), (50, -15), (124, -19)]`. Suppose we attempt to add a point with the value of $(74, -2)$. This point is impossible to obtain - the submarine cannot reach a reward of 74 in so few steps. If we were to add this point to the Pareto Frontier, the entire Frontier would be compromised. Points such as $(3, -5)$, $(16, -9)$, $(24, -14)$, and $(50, -15)$ would all be dominated, leading to incorrect results, as we can see in Figure 2.6. To address this issue, we can simply compare this point to the maximum value that the time penalty reward can achieve, which is -17. By doing so, we can adopt a conservative approach and exclude this point from the Frontier, assigning low reward values to the second objective, so it can be dominated by all existing points and does not interfere with the existing Frontier. This is illustrated in Figure 2.7.

By using this approach, the Pareto Frontier is not biased towards any particular region of the domain, and the architecture can identify all the non-dominated points in the given state-action space. Overall, this should allow for more accurate and effective approximations of the Pareto Frontier in complex problems (REYMOND; NOWE, 2019).

Another challenge encountered in the PDQN method is the potential instability

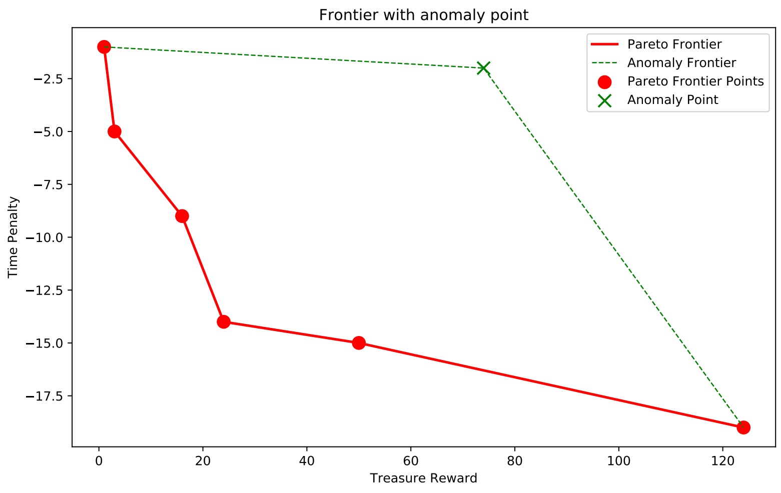


Figure 2.6 – Illustration of adding an anomaly point to a correct Pareto Frontier and the anomaly Frontier generated. Source: The Author.

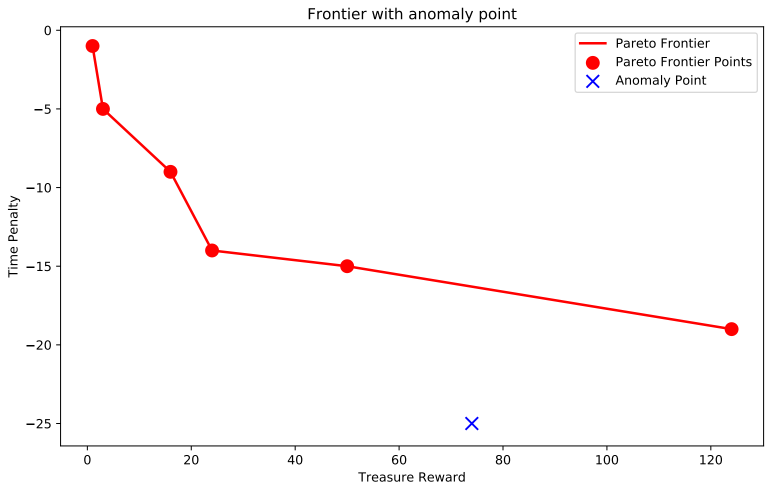


Figure 2.7 – Illustration of the correction of the anomaly point. Source: The Author.

introduced by newly discovered non-dominated points. When these new points are integrated, they can inadvertently alter the predictions for existing points on the Pareto Frontier, leading to inconsistencies in the approximation. To mitigate this, the PDQN method adopts a comprehensive update strategy. Instead of solely updating with the new points, the entire Pareto Frontier is refreshed. This involves using the current predictions of all points as their own targets, ensuring a consistent and stable representation of the Pareto Frontier. Stability in the Pareto Frontier is vital for hypervolume calculation. If the Pareto Frontier experiences frequent shifts or replacements, the hypervolume can exhibit significant fluctuations. Such inconsistencies can misguide the optimization process and lead to miscalculations.

The algorithm sometimes encounters situations where it must artificially introduce points to ensure comprehensive coverage of the $\mathbb{R}^{d-1}$ domain. This is crucial to maintain a consistent representation of the Pareto Frontier across the entire domain. Two primary scenarios necessitate this artificial introduction of points:

1. **Terminal States:** In certain situations, when the algorithm reaches terminal states, the Pareto Frontier is solely defined by the terminal reward. This means that the Pareto Frontier is not derived from a combination of multiple objectives but is instead represented by a singular value. This can lead to gaps and inconsistencies in the representation of the Pareto Frontier across the domain. This is very similar to what we described in Figure 2.6.
2. **Shift in ND_t Range:** After specific calculations, particularly as per Eq. 2.5, there's a shift in the range that ND_t covers. This shift can create regions in the $\mathbb{R}^{d-1}$ domain that lack representation on the Pareto Frontier.

To add clarity to the shift in the ND_t Range problem, let's exemplify it: When the FP (Frontier Point) operator is used in PDQN, as mentioned earlier, the entire Frontier is updated simultaneously. This can lead to situations where the shift in the Frontier results in points that are impossible to obtain. For instance, let's say we currently have a Frontier with the points `[(1, -2), (2, -4), (3, -8), (8, -11), (16, -13)]`. After adding a new non-dominated point and updating the Frontier, we obtain a new set of points `[(1, 0), (2, -2), (3, -7), (8, -9), (16, -11)]`. Based on the true Frontier, we know that it is impossible to achieve treasure values of 1 with a penalty of 0 and treasure reward 2 with -2 as a penalty, this problem is described in Fig.2.8. Therefore, a correction is needed. To address this, we discard the points that cannot exist while retaining the rest of the updated Frontier, in addition to sample points from the region, this process can be seen in Fig.2.9.

To address these gaps and ensure that the entire $\mathbb{R}^{d-1}$ domain is covered, the algorithm samples points from the uncovered regions. Each of these sampled points is then evaluated based on its dominance relationship with existing points:

- **Non-Dominated Points:** If a sampled point is not dominated by any existing point on the Pareto Frontier, it's assigned a conservative value, similar to how points outside the Frontier are treated, like in Fig. 2.7.
- **Dominated Points:** On the other hand, if a sampled point is dominated by an existing point, it adopts the value of that dominating point.

In essence, this strategy of artificially introducing points and assigning them val-

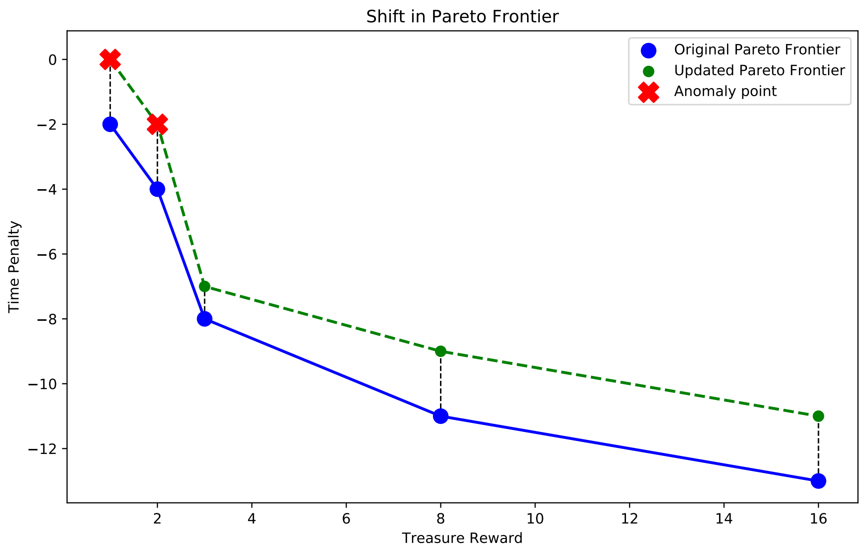


Figure 2.8 – Illustration of the shift anomaly in the domain region of the Pareto Frontier. Source: The Author.

ues based on their dominance relationship ensures that the Pareto Frontier remains consistent, accurate, and comprehensive across the entire $\mathbb{R}^{d-1}$ domain.

An overview of the PQDN is provided in Algorithm 4.

For this work, the PDQN algorithm was implemented, tested, and evaluated in the Deep Sea Treasure environment. Some of the results can be seen in Section 3.2. This implementation is available online at <https://github.com/GiovaniCenta/Pareto-Deep-Q-Networks>. Also, the PDQN was implemented and tested in the Gym City Flow urban traffic environment of Traffic Light Control (Section 3.3), the implementation is available online at <https://github.com/GiovaniCenta/Pareto-Deep-Q-networks-in-Traffic-Environment-Gym-City-Flow>.

2.4 Related works

The theoretical underpinnings of the Reinforcement Learning (RL) components in this research are primarily informed by the book "Reinforcement Learning: An Introduction" by Richard Sutton and Andrew Barto (SUTTON; BARTO, 2018). This seminal text not only provides a comprehensive overview of RL but also introduces key algorithms and methodologies that have been adapted in this study. The book's discussion on model-free learning methods and policy optimization techniques has been particularly influential in shaping the RL algorithms we employed.

In the realm of Multi-Objective Reinforcement Learning (MORL), the paper "Multi-

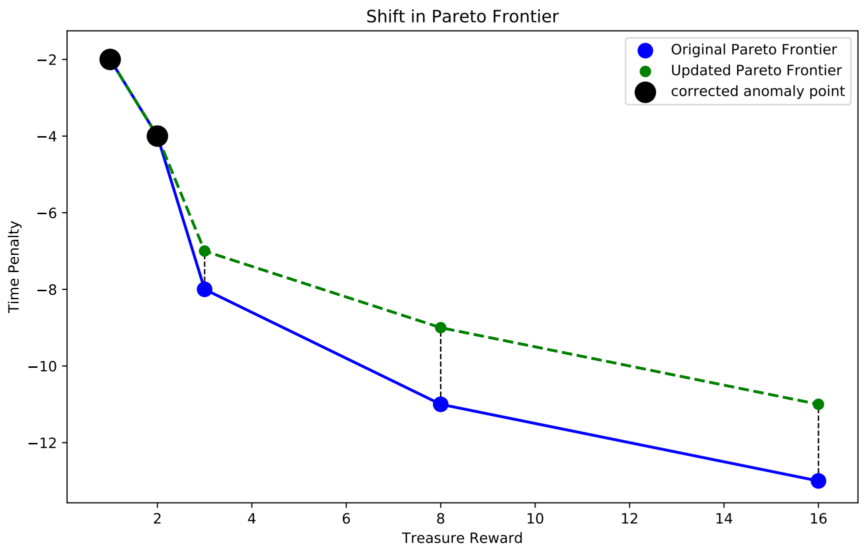


Figure 2.9 – Illustration of the correction made in the shift of the Pareto Frontier domain. Source: The Author.

Objective Reinforcement Learning using Sets of Pareto Dominating Policies" by Kristof Van Moffaert and Ann Nowé (MOFFAERT; NOWé, 2014) serves as a cornerstone. This paper is pivotal for its introduction of Pareto Dominating Policies (PDP) as a formalized approach to MORL. The authors methodology for identifying and utilizing Pareto-optimal solutions has been directly incorporated into our research, influencing both the algorithms we chose to study and the metrics we used for evaluation.

The algorithmic architecture of Pareto Deep Q-Networks (PDQN) in our study is heavily influenced by the paper "Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems" by Mathieu Reymond and Ann Nowé (REYMOND; NOWE, 2019). This paper not only provides a detailed architecture of PDQN but also discusses its applicability in complex decision-making scenarios. The authors insights into the limitations and potential improvements of PDQN have guided the enhancements we proposed in our own work.

For the application to real-world Traffic Light Control scenarios, we turned to the paper "On the Explainability and Expressiveness of Function Approximation Methods in RL-Based Traffic Signal Control" by Lincoln V. Schreiber, Lucas N. Alegre, Ana L.C. Bazzan, and Gabriel de O. Ramos (SCHREIBER et al., 2022b). This research offers a comprehensive framework for applying RL algorithms in traffic control settings, including the challenges and considerations for real-world deployment. The paper's empirical evaluations and methodological insights have been instrumental in shaping the design and metrics of our own experiments in a similar context.

Algorithm 4: PDQN algorithm (REYMOND; NOWE, 2019)

```
Input: Replay memory $D$, Reward Estimator $\bar{R}$, Non-dominated Estimator
       $ND_t$, target Non-dominated Estimator $\hat{ND_t}$, hyper-parameters $\gamma$, $\epsilon$,
       number of episodes: $M$, copy target step: $C$, number of sample
       points: p
for episode = 1 to $M$ do
    $terminal$ ← False;
    while $\neg terminal$ do
        Sample points $p$ from $\mathbb{R}^{d-1}$;
        $Qset(s, ., p) \leftarrow R(s, .) \oplus \gamma ND_t(s, ., p)$;
        $hv$ ← hypervolume($Q_{set}(s, ., p)$);
        $a$ ← $\epsilon$-greedy($hv$);
        Execute $a$ in environment, observe state $s'$, reward $r$, terminal $t$;
        Add transition $(s, a, r, s', t)$ to $D$;
        Sample minibatch $(s_i, a_i, r_i, s'_i, t_i)$ from $D$;
        Sample points $p_i$ from $\mathbb{R}^{d-1}$;
        $y_i \leftarrow \begin{cases} ND(\bigcup_{a' \in A} \hat{Q}_{set}(s'_i, a'_i, p_i)) & \textbf{if not } t_i \\ r_i & \textbf{otherwise} \end{cases}$;
        Update $ND_t$ by performing gradient descent step on
          $(y_i - Q_{set}(s_i, a_i, p_i))^2$;
        Update $R$ by performing gradient descent step on $(r_i - R(s_i, a_i))^2$;
        Every $C$ steps copy $ND_t$ to $\hat{ND_t}$
        $terminal \leftarrow t$;
    end
end
```

3 EVALUATING PQL AND PDQN

To gain a deeper understanding of the algorithms and their limitations, we implemented and evaluated them in specific environments. For PQL, we used the Deep Sea Treasure environment, and for PDQN, we utilized both the Deep Sea Treasure and Gym City Flow environments. We compared our findings with previous results and identified limitations, which are discussed in the following chapter.

3.1 PQL in Deep Sea Treasure Environment

The Deep Sea Treasure is a deterministic environment that serves as a benchmark in the domain of MORL. In this environment, an agent controls a submarine with the primary goal of collecting treasures as quickly as possible. The state space is defined by a grid that represents the submarine's coordinates. The action space comprises the four allowed directional movements: Up, Down, Left, and Right. Additionally, when the submarine attempts to cross a boundary, it remains in the same place and incurs a time penalty. The agent aims to optimize the balance between maximizing treasure rewards and minimizing time consumption. It's worth noting that rewards escalate with increasing water depth, as depicted in the Deep Sea Treasure map (Figure 3.1). However, each movement of the submarine incurs a time penalty of -1. This MO environment was provided by (ALEGRE et al., 2022).

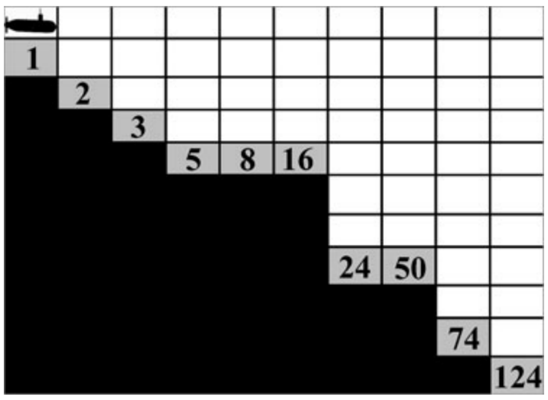


Figure 3.1 – Illustration of the Deep Sea Treasure environment (VAMPLEW et al., 2011)

Given its relatively simple state and action spaces, the Deep Sea Treasure environment is particularly suited for evaluating the Pareto Q-Learning algorithm. Consequently, we implemented and tested the algorithm within this setting. The experiment utilized the following hyper-parameters: $\gamma = 0.96$, number of episodes $= 35000$, hypervolume reference point = $[0, -25]$, $\epsilon = 1$, and ϵ decrease = 0.999.

To understand the effectiveness of our implementation, it's crucial to compare it against a benchmark. The true Pareto Front of the Deep Sea Treasure problem serves this purpose. In this problem, determining the Pareto Front analytically is straightforward: one simply needs to consider the shortest path to each reward. Each pair of time and reward constitutes a point on the front. This front represents the set of non-dominated solutions where no objective can be improved without sacrificing another. To gauge the performance of the PQL algorithm, their respective Pareto Fronts were compared to this true Pareto Front.

For a more refined and precise representation, the algorithms were run using the best policies with a greedy action selection (i.e., $\epsilon = 0$).

The results of the Pareto Frontier in our implementation of the PQL algorithm are illustrated in Figure 3.2. The plot demonstrates the algorithm's capability to establish a boundary that balances both objectives: minimizing the time penalty while maximizing treasure rewards. While the current results are promising, particularly for the initial treasure rewards, there is room for further optimization with respect to the latter rewards. This could be achieved through hyperparameter tuning or by enhancing the algorithm's ability to learn optimal policies, which is the primary focus of the PDQN algorithm, which will be seen in the following Section 3.2.

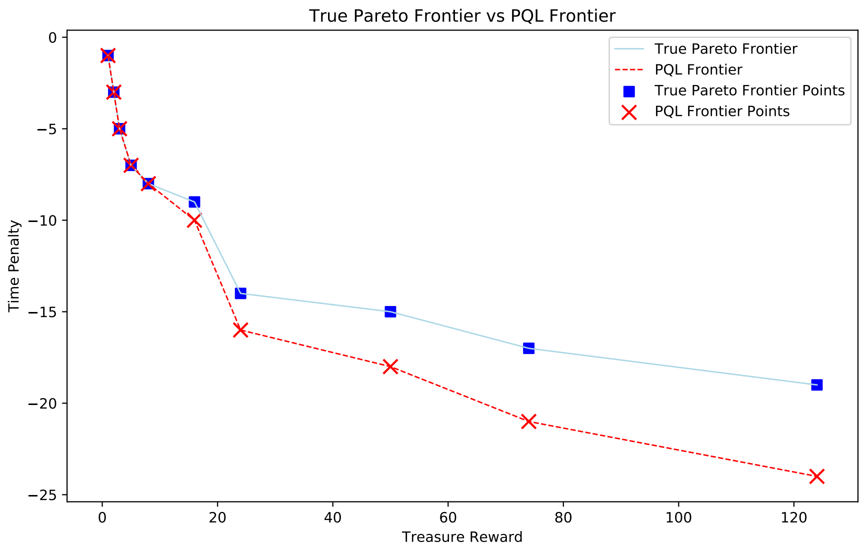


Figure 3.2 – Plot of the comparison between the True Pareto Frontier and the PQL Pareto Frontier. Source: The Author.

Figures 3.3 and 3.4 depict the algorithm's performance in terms of rewards over episodes. These figures show the algorithm's ability to obtain good rewards while managing time. Although there is room for improvement, the results validate the PQL algorithm's capability in navigating the complexities of the Deep Sea Treasure environment. The algorithm did not follow an optimal policy for most episodes but managed to follow a satisfactory one, with rewards close to 16 and a time penalty close to -9. This accounts for the non-optimal results for the latter treasure rewards in Fig.3.2. Once the algorithm settled on a preference for the 16 treasure reward, it did not explore enough to find optimal policies for subsequent rewards like 24, 50, 74, and especially 124.

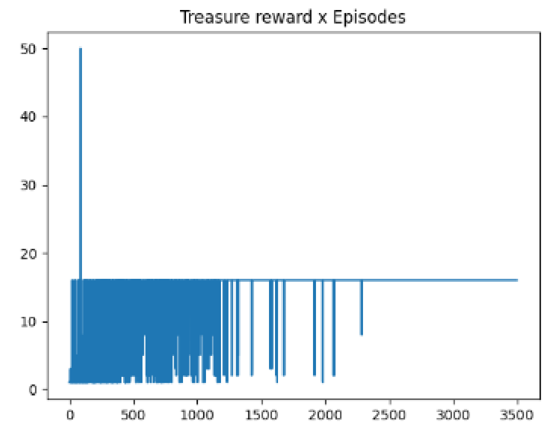


Figure 3.3 – Plot of the treasure reward x number of episodes, Source: The Author.

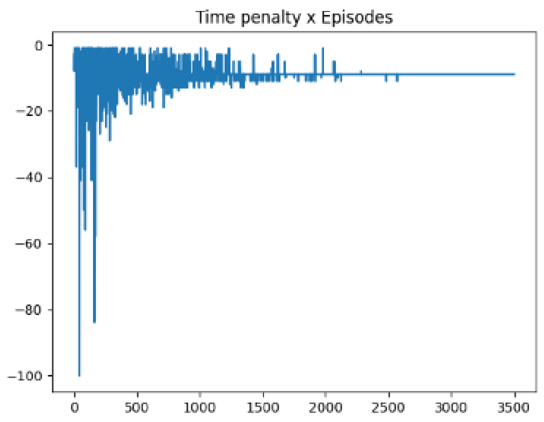


Figure 3.4 – Plot of the time penalty x number of episodes, Source: The Author.

3.2 Results of PDQN in the Deep Sea Treasure Environment

After our exploration with the PQL algorithm, we applied the PDQN algorithm to the Deep Sea Treasure Environment. The methodology for determining the Pareto Frontier with PDQN was equal to our approach for PQL. For this experiment, the hyperparameters were set as follows: Discount factor (γ) was 0.99, number of episodes was

35000, hypervolume reference point was $[0, -25]$, the initial exploration rate (ϵ) was 1, rate of decrease for ϵ was 0.9999, number of points p was 16, learning rate for Replay Estimator was 0.001, and learning rate for Non-dominated Estimator was 0.0001.

The outcomes of the PDQN algorithm Pareto Frontier are illustrated in Figure 3.5.

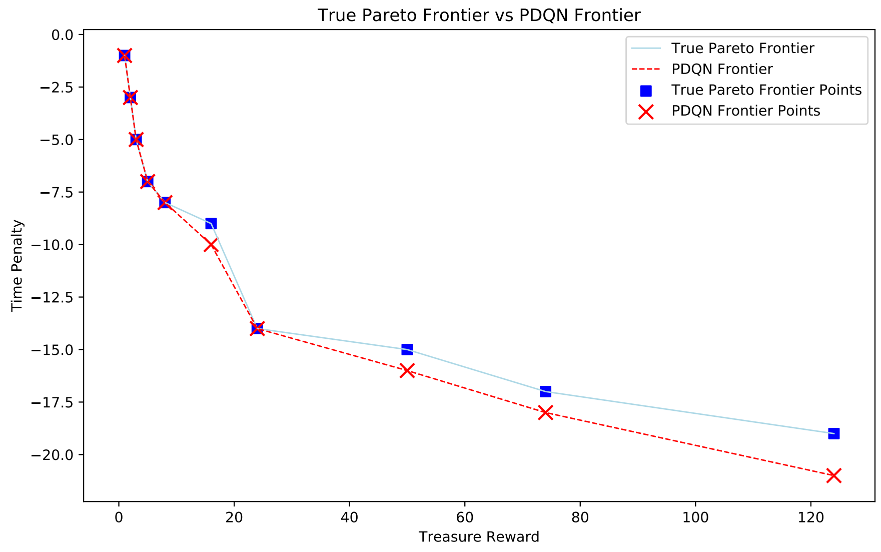


Figure 3.5 – Comparison between the true Pareto Frontier and the PDQN Pareto Frontier algorithm implemented by The Author. Source: The Author.

The analysis reveals interesting results. Notably, the initial data points on the PDQN's Pareto Frontier align closely with those of the true Pareto Frontier. However, discrepancies become apparent in the later points, indicating areas for improvement. Additionally, when compared to the PQL Frontier results (Fig. 3.6), the PDQN outperforms PQL in the latter treasure rewards, (50,74, and 124 rewards).

3.3 Results of PDQN in a simulated Traffic Light Control environment

With the continuous expansion of urban areas and the increasing number of vehicles on the roads, urban traffic congestion has become a significant challenge in today's society (ZHANG; LEVINSON, 2004). This issue affects not just daily commuting and transportation efficiency but also has broader implications on social well-being (ELIASSON, 2009), public health (BRAUER; HYSTAD, 2019), and environmental sustainability (HO; MULLEY, 2013). As cities search for effective solutions, traffic light control has emerged as a promising strategy (ABDOOS; MOZAYANI; BAZZAN, 2011), leveraging existing infrastructure in many urban centers.

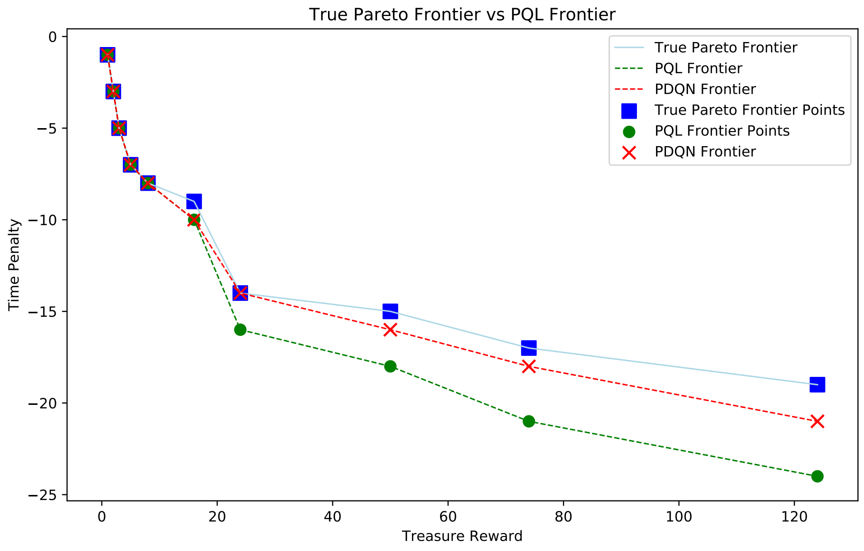


Figure 3.6 – Comparison between the PQL Pareto Frontier, the PDQN Pareto Frontier and the PQL Pareto Frontier from the algorithms implemented by The Author. Source: The Author.

Over the years, many studies have utilized RL algorithms to optimize traffic light control systems. These algorithms, often combined with neural networks and other advanced techniques, have shown potential in improving traffic flow and reducing congestion (SHAH; ZHOU, 2019). However, there remains a gap in the application of Multi-objective algorithms specifically for this context.

For the testing scenarios, we utilized realistic traffic datasets sourced from cameras in Hangzhou, China, as provided by (ZHENG et al., 2019). Which included a single four-lane intersection layout. Every intersection maintained a speed limit of 11.11 meters per second across all four lanes. With each approach measuring 300 meters, Conforming to standard traffic regulations, we followed the sequence of a green light transitioning to a 3-second yellow light and finally to a 2-second red light across all signals. These scenarios were recreated using the CityFlow simulator (ZHANG et al., 2019). This traffic scenario is illustrated in Figure 3.7.

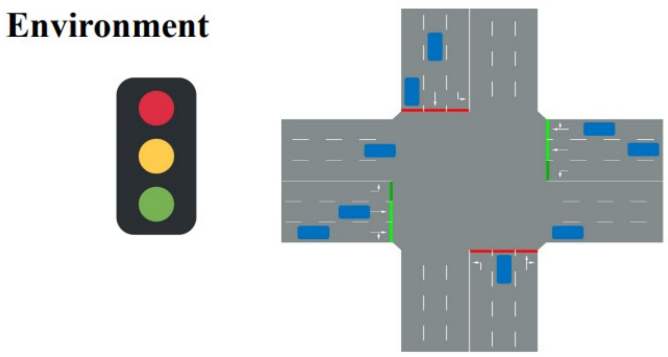


Figure 3.7 – Illustration of the traffic scenario experiment (ZHENG et al., 2019).

To address this, we applied our version of the PDQN algorithm to this scenario. We aimed to optimize the Traffic Lights control environment and assess the adaptability of the PDQN in this setting.

In this scenario, each traffic light at the intersection serves as an individual agent.Let's define the state space, action space, and reward functions of this complex environment:

- **Action Space:** The set of actions available to the agent comprises eight distinct phases, each corresponding to a specific movement defined in Table 3.1. These movements represent different traffic flow configurations at the intersection. Once the agent selects a particular phase, it remains active for a fixed duration of 20 seconds, during which the agent determines the order of these phases. It is important to note that the timing of each phase is predetermined and remains constant; thus, the agent's decisions focus on the sequencing of the phases rather than their individual durations. While the agent can opt for the same phase multiple times within the 20-second interval, such choices are made only once every 20 seconds.

Table 3.1 – Allowed Phases

Phases	**Allowed Movements**	**Allowed Directions**
0	W → E, E→W	→, ←
1	S→N, N →S	↓, ↑
2	W →N, E→S	⬏, ⬎
3	S→W, N →E	⬐, ⬑
4	W →E, W →N	→, ⬏
5	E→W, E→S	←, ⬎
6	S→N, S→W	↑, ⬐
7	N →E, N →S	⬑, ↓

- **State Space:** The state space is modeled as a continuous real-valued vector, denoted by S^d, where each of the d components corresponds to the queue length of a specific movement at the intersection. At a given time step t, the state is represented as S_t = $[F_0, F_1, F_2, F_3, F_4, F_5, F_6, F_7]$, where F_0 through F_7 represents the queue lengths of movements 1 through 8, respectively. The possible movements can be seen in table 3.2. These queue lengths provide critical information about the number of vehicles waiting in each traffic lane. By considering the state of each movement, the agent gains insights into the current traffic conditions, allowing it to make informed decisions for optimizing traffic flow.
- **Reward Functions:** In our MO scenario, we have identified two pivotal reward

Table 3.2 – Possible movements

Movement	**Direction**	**Symbol**
0	W → E	→
1	E→W	←
2	S→N	↑
3	N →S	↓
4	W →N	⬏
5	E→S	⬎
6	S→W	⬐
7	N →E	⬑

functions to address the complexities of urban traffic control. The first reward is centered on the intersection pressure, which is quantified as the difference between the density of vehicles entering the intersection and those departing. By minimizing this pressure, we aim to substantially enhance the overall traffic flow and mitigate congestion, ensuring smoother transit for commuters. For the second reward, we emphasize the lane waiting count. This metric represents the number of vehicles queued (waiting to be able to move) in each lane. We've chosen this particular reward because it provides a direct measure of localized congestion and can be instrumental in identifying specific bottlenecks or areas that would require immediate attention.

This definition of action and state spaces provides a structured framework for the agent to make intelligent decisions and effectively optimize traffic signal control. The real-valued state representation allows the agent to perceive the traffic situation in a continuous manner, providing a more nuanced understanding of the traffic dynamics at the intersection. Similarly, the discrete action space enables the agent to exert control over the traffic signals in a practical and manageable way. Furthermore, the reward functions have been thoughtfully designed to facilitate simultaneous optimization, enabling the agent to achieve favorable outcomes in multiple objectives.

For the sake of consistency and to allow for direct comparison of results, we conducted our experiment using the same scenarios as Lincoln V. Schereiber in his paper "On the Explainability and Expressiveness of Function Approximation Methods in RL-Based Traffic Signal Control" (SCHREIBER et al., 2022b). This facilitated our focus on realistic traffic datasets referenced in previous studies.

The PDQN algorithm was designed to address the challenge of managing a large number of state-action pairs, a common issue with traditional Pareto Q-Learning approaches. Using a neural network to approximate both the Pareto Frontier and the average immediate reward, the PDQN proved somehow effective at managing state-action pairs in smaller environments as we have seen before 3.2.

To enable a comparison with the current literature, we employed the same performance metrics as (SCHREIBER et al., 2022b), namely:

- **Travel Time:** This measures the duration a vehicle takes to traverse an intersection. It is calculated by noting the difference (in seconds) between the vehicle's entry and exit times at the intersection.
- **Speed Score:** This index measures how swiftly a vehicle can cross the intersection, while still adhering to the speed limit. It is calculated by dividing the actual speed of the vehicle by the speed limit of the road.
- **Throughput:** This is a count of the number of vehicles that have completed their trips. It's determined by tallying the number of completed steps in each vehicle's route.

In addition to these metrics, we also compared our results against established traffic management algorithms such as:

- **Fixed Time (LI et al., 2016):** This model keeps the traffic light phases on a pre-set, unchanging cycle, with each phase lasting 58 seconds, a duration found to be most efficient in Schreiber's tests (SCHREIBER et al., 2022b).
- **Self-organizing traffic lights (SOTL) (COOLS; MOONS; WETS, 2013):** The Self-Organizing Traffic Lights Control strategy changes the traffic phases when either a certain number of waiting vehicles is exceeded or the minimum green light duration is reached.
- **Max-Pressure (HAO; YANG, 2019):** Regarded as the current best approach in traffic signal control, Max-Pressure gives priority to the traffic phase with the most halted or incoming vehicles, balanced by the number of departing vehicles. The decision to switch or maintain phases is made every 20 seconds.

This approach allows for an in-depth analysis and comparison of our results against standard methods and the findings of Schereiber's paper (SCHREIBER et al., 2022b).

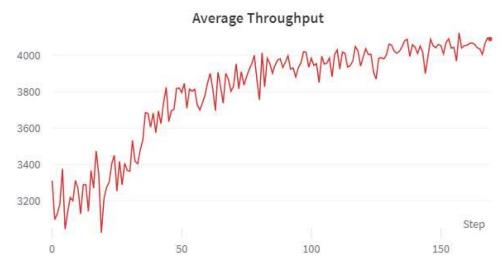
Upon implementing the PDQN algorithm within the CityFlow gym environment, we quickly discerned the algorithm's difficulties in managing the multifarious aspects of urban traffic dynamics. The initial outcomes not only fell short of expectations but also deviated significantly from benchmarks established by contemporary studies. Vital performance indicators, including average travel time and vehicle throughput, lagged behind ideal standards, especially when contrasted with results achieved using other RL algorithms. A table of these comparison results can be seen in table 3.3. These observations underscored the limitations in the adaptability of the PDQN algorithm within such intricate contexts.

Table 3.3 – Results in (SCHREIBER et al., 2022a) in comparison with PDQN.

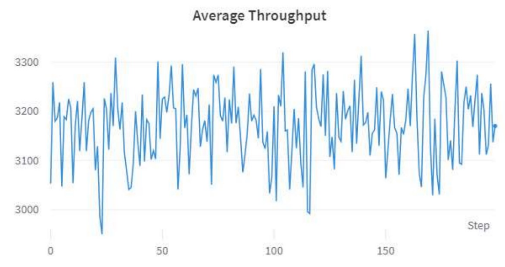
Results in Hangzhou environment

Algorithm	**TravelTime**	**SpeedScore**	**Throughput**
PDQN	594.9216	0.197	3170
FixedTime	558.837	0.218	3313
SOTL	304.32	0.276	3899
MaxPressure	131.91	0.425	4327
DQL-XGB	124.96 ± 4.80	0.468 ± 0.015	4323
DQL-FB	131.70 ± 9.76	0.428 ± 0.022	4320
DQL-NN	132.47 ± 4.88	0.431 ± 0.018	4280

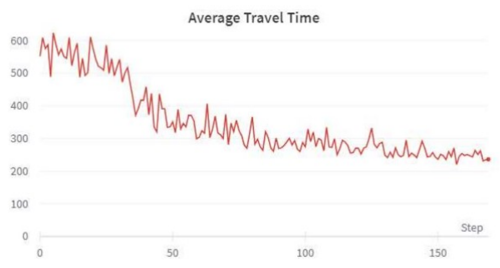
When analyzing the performance over the last 150 episodes for both the standard DQN and PDQN approaches, a distinct disparity emerged. The DQN algorithm successfully optimized both throughput and travel time—increasing the throughput while reducing the travel time. In contrast, the PDQN did not demonstrate a similar level of optimization. For a visual comparison of their throughput performances, refer to Figures 3.8a and 3.8b. For travel time performances, see Figures 3.9a and 3.9b.



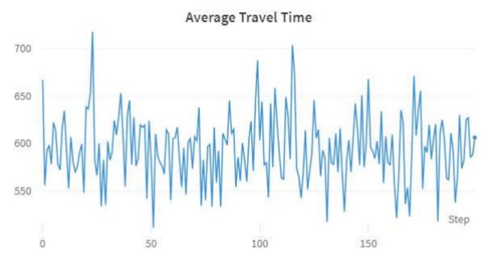
(a) Average throughput of the DQN algorithm over the last 150 episodes.



(b) Average throughput of the PDQN algorithm over the last 150 episodes.



(a) Average travel time of the DQN algorithm over the last 150 episodes.



(b) Average travel time of the PDQN algorithm over the last 150 episodes.

Upon analyzing the performance of the PDQN algorithm in this context, several questions emerged. Although the primary objective of the PDQN algorithm was to excel in larger environments, the results presented in this section indicate its shortcomings in achieving that goal. Consequently, in the subsequent chapter, we aim to enhance the algorithm's learning capabilities by modifying its architecture (Section 4.1) and introducing a novel approach based on non-dominated actions (Section 4.2).

4 ENHANCEMENTS IN THE PDQN ALGORITHM

4.1 Incorporation of the Target Reward Estimator

To try enhance the stability of the PDQN algorithm, we introduced a Target Reward Estimator. This component is analogous to the Target Network found in the Deep Q-Learning framework, as referenced in Alg. 2. The primary reason for introducing this stability mechanism is to ensure that the neural network controlling the agent is not updated at every timestep, which would otherwise result in more erratic control behavior.

Our main objective was to improve the stability of the Reward Estimator. While the Non-dominated Estimator already incorporates a target for a similar purpose, our aim was to further stabilize the Target Reward approximations. It should be noted that this is more about enhancing the stability in the agent's control rather than in the estimators themselves.

In our proposed approach, the Target Reward Estimator's role is to produce Target Reward values for the PDQN algorithm. These values are then used to adjust the priorities of experiences in the replay buffer and to determine approximation errors.

To implement this, the Target Reward Estimator is updated at regular intervals, copying the weights from the Reward Estimator every C episodes. During the PDQN training, the values from the Target Reward Estimator would be used similarly to the Target Non-dominated values. This is expected to aid in calculating approximation errors and updating the neural network parameters.

By integrating the Target Reward Estimator into the PDQN algorithm, we hoped to reduce fluctuations in reward predictions, potentially leading to smoother learning and better algorithm performance.

However, our tests showed that adding the target Estimator didn't bring significant benefits. The Pareto Frontier benchmark was equal to the results reported in Section 3.2. Also, it slightly increased the algorithm's computational demands, as shown in Table 4.1. One reason could be the added complexity from the Target Reward Estimator, possibly introducing more noise and affecting reward prediction accuracy. The existing robustness of the algorithm might mean the added Estimator doesn't bring enough value to justify the computational cost.

Also, In RL, Q-values denote the cumulative rewards for executing a specific action in a particular state. As the agent deepens its understanding of the environment, these

Table 4.1 – Average of computational metrics with and without the Target Reward Estimator after 10.000 Episodes in 10 runs

	With Target Reward Estimator	Without Target Reward Estimator
Memory usage	56.7%	54.9%
CPU Usage	99.9%	99.85%
Running time(seconds)	612	597

Q-values evolve, classifying them as non-stationary targets. On the other hand, immediate rewards remain consistent and unaltered, categorizing them as stationary targets. For instance, in the Deep Sea Treasure environment, every time the agent arrives at the state with coordinates $(x, y) = (0, 1)$ (refer to Figure 3.1), it receives an immediate reward of 1. However, the corresponding Q-value might vary based on the trajectory taken to reach that state, as dictated by its update equation (see Eq. 2.1).

The discussion on Q-values and immediate rewards highlights the concept of bootstrapping and stability in RL algorithms. In the case of the Q-values in the PDQN algorithm, the estimates are updated using another estimate of Q, specifically the Q-value of the future state. This means we are updating an estimate with another imprecise estimate made by the agent itself. This bootstrapping process introduces non-stationarity in the Q-value updates, making stabilization mechanisms like backpropagation crucial. Backpropagation averages updates over multiple iterations, ensuring that the network's predictions are not swayed by transient fluctuations in the data and are grounded in a consistent understanding of the environment.

On the other hand, immediate rewards have a stationary target that is not influenced by the agent's imprecise estimates. These rewards are updated exclusively based on samples from the environment, negating the need for stabilization mechanisms. This inherent stability in immediate rewards stands in contrast to the non-stationary nature of Q-value updates, highlighting the importance of stabilization mechanisms for the latter while demonstrating their lack of necessity for the former.

This distinction holds significance: the unwavering nature of immediate rewards negates the requirement for an auxiliary target network. While such networks are conventionally employed to stabilize learning by furnishing consistent Q-value targets, the presence of stationary rewards allows for direct updates to the primary network based on the immediate rewards obtained.

4.2 Another approach on action selection

In our investigation of PDQN, our primary objective was to identify mechanisms to eliminate the reliance on sample points within the algorithm. This exploration led us to the promising proposition of incorporating a hypervolume Estimator. This innovative approach seemed poised to render artificial points redundant and to facilitate the direct computation of the maximum Q-value for that state and action pair, replacing the need for p sample points and the need for the equation $Qset(s,.,p) \leftarrow R(s,.) \oplus \gamma ND_t(s,.,p)$ in the PDQN algorithm(Alg. 4).

This hypervolume Estimator would be designed to amalgamate the two core approximations intrinsic to the PDQN algorithm – the Reward Estimator and the Non-dominated Estimator – yielding a unified framework. The central idea here was to make the algorithm more efficient by utilizing the existing state and actions to estimate the hypervolume with the locally non-dominated actions, thereby bypassing the p points sampling.

Yet, this approach seemed to clash with a pivotal premise posited by van Moffaert and Anne Nowe in their foundational work (MOFFAERT; NOWé, 2014, p.3679), which claimed, "Selecting actions that are locally non-dominated within the current state does not guarantee that the entire policy is globally Pareto optimal." This assertion posed a challenging dichotomy prompting deeper exploration.

In MORL, actions are deemed non-dominated when they can't be surpassed by any other action across all objectives. This definition is grounded in the principle of Pareto optimality, where any action on the Pareto Frontier is, by definition, non-dominated.

To dissect this enigma further, we embarked on empirical studies using a MO tabular approach. For this purpose, we utilized two foundational algorithms: Q-Learning and SARSA. Each algorithm utilizes d tables, where d represents the number of objectives. Each table is designed to estimate the expected return of a state-action pair for a specific objective. From these tables, locally non-dominated values are extrapolated for each action in every state. Subsequently, a non-dominated action is randomly selected, serving as the basis for the decision-making policy. The MO Q-Learning algorithm is described in Alg. 5 and the MO SARSA algorithm is described in Alg. 6.

Our experimental setup involved implementing both algorithms in a simplified version of the Deep Sea Treasure environment, illustrated in Fig. 4.1. Both algorithms were configured with identical hyperparameters: α = 0.4, γ = 0.98, ϵ = 1, epsilon decrease =

Algorithm 5: Multi-Table Q-Learning

```
Input: Initial state $s$, learning rate $\alpha$, discount factor $\gamma$, exploration
  parameter $\epsilon$, number of episodes $E$, number of objectives $d$;
Output: Learned Q-value functions $Q_1(s,a), Q_2(s,a), \ldots, Q_d(s,a)$;
Initialize $Q_i(s,a)$ arbitrarily for all state-action pairs and for $i = 1, \ldots, d$;
for $e = 1$ to $E$ do
    Initialize state $s$; Initialize done = False;
    while not done do
        a ← $\epsilon$-greedy in FP([$Q_1$,$Q_2$ ..., $Q_d$]);
        Take action $a$ and observe reward,$s'$,done;
        $r_1, r_2, \ldots, r_d$ ← reward;
        if not done then
            for $i = 1$ to $d$ do
                $Q_i(s,\mathrm{a}) \leftarrow Q_i(s,\mathrm{a}) + \alpha\,[r_i + \gamma \max_{a'} Q_i(s',a') - Q_i(s,\mathrm{a})]$;
            end
        end
        else
            for $i = 1$ to $d$ do
                $Q_i(s,\mathrm{a}) \leftarrow r_i$;
            end
        end
        $s$ ← s';
    end
end
```

0.996, and number of episodes = 3600. The non-dominated actions for every state were vividly depicted in Figures 4.2 and 4.3. An examination of the Q-Learning results, revealed a discernible flaw: it incorrectly identified the upward action as non-dominated on the first state. This choice is inconsistent with the principle of non-dominance, as this action would merely increase further time penalties without getting any rewards. Conversely, SARSA's implementation did not display these anomalies. The correct non-dominated actions – moving downward (securing a low-value treasure with minimal time penalty) and shifting rightward (navigating towards treasures with higher values but incurring larger time penalties) – were correctly identified. This observation indicates that when Q-Learning randomly selects one of the non-dominated actions, choosing the 'Up' action could result in global non-dominance issues, such as the generation of non-optimal policies. This is a pitfall that the SARSA approach avoids.

By closely examining the Q-values (Q_1 for the treasure and Q_2 for the time penalty) associated with each action in the initial state, we uncover intriguing differences between Q-Learning and SARSA in this multi-objective setting.

In table 4.2, the action to move right is considered non-dominated because it op-

Algorithm 6: Multi-Table SARSA

```
Input: Initial state $s$, learning rate $\alpha$, discount factor $\gamma$, exploration
  parameter $\epsilon$, number of episodes $E$, number of objectives $d$;
Output: Learned Q-value functions $Q_1(s,a), Q_2(s,a), \dots, Q_d(s,a)$;
Initialize $Q_i(s,a)$ arbitrarily for all state-action pairs and for $i = 1, \dots, d$;
for $e = 1$ to $E$ do
    Initialize state $s$; Initialize done = False;
    a ← $\epsilon$-greedy in FP([Q$_1$,Q$_2$ …, Q$_d$]);
    while not done do
        Take action $a$ and observe reward,$s'$,done;
        $r_1, r_2, \dots, r_d \leftarrow$ reward;
        a' ← $\epsilon$-greedy in FP([Q$_1$,Q$_2$ …, Q$_d$]);
        if not done then
            for $i = 1$ to $d$ do
                $Q_i(s,\text{a}) \leftarrow Q_i(s,\text{a}) + \alpha\,[r_i + \gamma Q_i(\text{s'},\text{a'}) - Q_i(s,\text{a})]$;
            end
        end
        else
            for $i = 1$ to $d$ do
                $Q_i(s,\text{a}) \leftarrow r_i$;
            end
        end
        $a \leftarrow a'$;
        $s \leftarrow$ s';
    end
end
```

timizes the first reward but has a worse time penalty. The action to move down tends to generate lower time penalties while achieving a minimum reward, so it is also considered non-dominated. However, both the Up and Left actions are considered non-dominated as they provide intermediate values of reward and penalty. They dominate the action to move right in terms of time penalty and dominate the action to move down in terms of reward. This is not ideal, as both actions should not be considered non-dominated in this environment.

Table 4.2 – Q-values for actions in the first state in MO Q-Learning

Action	Q_1	Q_2
Up	2.71176239	-1.98
Down	1.0	-1.0
Left	2.71176239	-1.98
Right	2.76710448	-2.9404

The problem with Q-Learning is that it mixes things up: the maximum value for one objective may take the value of one action in state s', but for another objective, it may

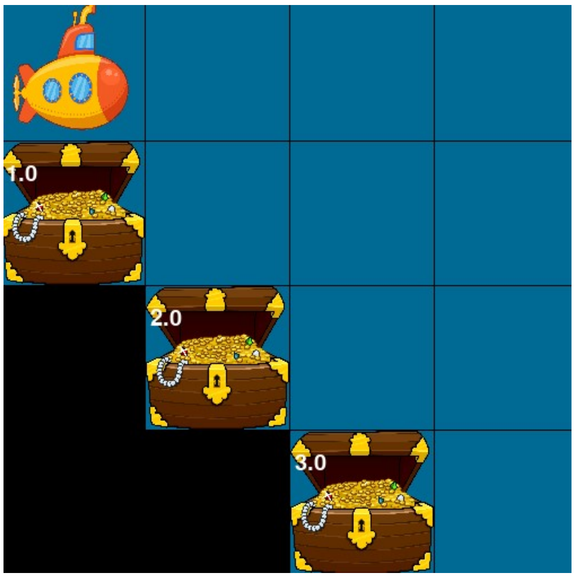


Figure 4.1 – Illustration of the Deep Sea Treasure map utilized in the MO Q-Learning and MO SARSA approaches. Source: The Author.

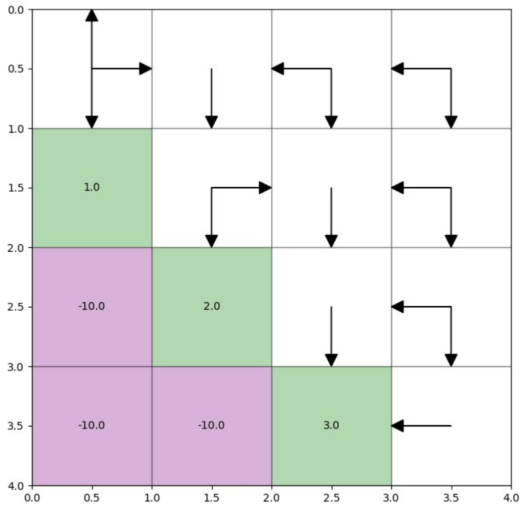


Figure 4.2 – Illustration of the states(rectangles) and Non-dominated actions (arrows) for our MO Q-Learning approach, Source: The Author.

take the value of another action.

On the other hand, SARSA does not face this issue. In table 4.3, the action to move down correctly dominates all other actions in terms of time, while the action to move right dominates all others in terms of reward. Both the actions to move left and up do not dominate either right or down in any objective.

Let's consider an illustrative example to clarify the differences between Q-Learning and SARSA in this multi-objective setting. Imagine a scenario where taking the action of going upwards leads to a second state with Q-values as shown in Table 4.4. In Q-Learning, the independent Q_1 and Q_2 tables would select different actions due to the max operator in Eq.2.1. Specifically, Q_1 would opt for the 'Up' action to maximize its value,

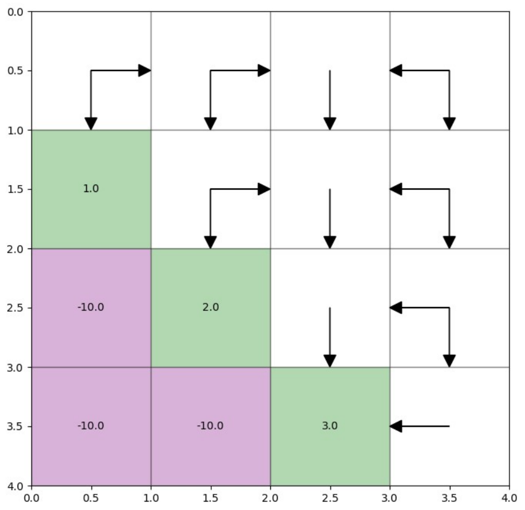


Figure 4.3 – Illustration of the states(rectangles) and Non-dominated actions (arrows) for our MO SARSA approach, Source: The Author.

Table 4.3 – Q-values for actions in the first state in MO SARSA

Action	Q_1	Q_2
Up	1.86478052	-5.93641483
Down	1.0	-1.0
Left	1.99666967	-6.10895009
Right	2.76710448	-4.80396016

while Q_2 would choose 'Down'.

In contrast, SARSA's on-policy approach would select the same action based on its current policy. This difference stems from the update equations for SARSA (Eq. 4.1) and Q-Learning (Eq. 2.1). While Q-Learning updates its Q-values based on the maximum Q-value for each objective, SARSA chooses the update for both objectives based on the action chosen by its policy.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{4.1}$$

Table 4.4 – Q-values for actions in a secondary imaginary state

Action	Q_1	Q_2
Up	9.0	0.0
Down	0.0	9.0
Left	1.0	2.0
Right	2.0	1.0

Using Eq. 2.1, let's consider an imaginary first state Q-table where the 'Up' action initially has a Q-value of 0 for both Q_1 and Q_2. With hyperparameters set to $\gamma = 0.9$ and $\alpha = 0.5$, and assuming a reward of 0 for transitioning to the next state, the updated Q-values for both Q_1 and Q_2 would be 4.05, as calculated using Eq. 4.2. As shown in Table 4.5, both Q-values end up being equal, since they both select Q-values of 9 in the next state. In the context of non-dominated actions, 'Down' and 'Right' would be considered non-dominated. Also, in this case, the 'Up' action would also be categorized as non-dominated.

$$Q_{1,2}(0,\uparrow) \leftarrow 0 + 0.5\left[0 + 0.9 * 9 - 0\right] \;= 4.05. \tag{4.2}$$

Table 4.5 – Q-values for actions in a first imaginary state

Action	Q_1	Q_2
Up	4.05	4.05
Down	15	1
Left	2	3
Right	1	15

Applying the SARSA equation in the same context, let's assume that the next action selected by the policy is 'Down'. The corresponding Q-values for this action in the subsequent state are $(0,9)$ for Q_1 and Q_2, respectively. Using these values in Eq. 4.1, we obtain updated Q-values of 0 for Q_1 and 4.05 for Q_2, as detailed in Eq. 4.3 and Eq. 4.4, respectively. The updated Q-values are presented in Table 4.6. In this particular case, the 'Up' action loses its non-dominated status, as it is outperformed by all other actions with respect to the first Q-value, so only 'Down' and 'Right' are non-dominated actions.

$$Q_1(0,\uparrow) \leftarrow 0 + 0.5[0 + 0.9 * 9 - 0] = 4.05 \tag{4.3}$$

$$Q_2(0,\uparrow) \leftarrow 0 + 0.5[0 + 0.9 * 0 - 0] = 0 \tag{4.4}$$

Table 4.6 – Q-values for actions in a first imaginary state using SARSA

Action	Q_1	Q_2
Up	0	4.05
Down	15	1
Left	2	3
Right	1	15

The key distinction to get these results lies in the look-ahead strategy of Q-Learning and SARSA. While Q-Learning uses the estimated return of the greedy policy, SARSA bases its decision on a single action recommended by the current policy. This avoids the issue in Q-Learning where the best action in a state differs for each objective. Both algorithms look ahead, but they differ in their approach: Q-Learning considers the value of the best action in the next state, whereas SARSA considers the value of the action chosen by the policy in the next state.

When we expanded our research to a bigger Deep Sea Treasure Environment, the results were even more illuminating. Our straightforward Multi-Objective (MO) SARSA algorithm showcased an ability to navigate this environment. This simple algorithm could navigate and find non-dominated actions for various states in the bigger environment, as can be seen in Fig. 4.4. This observation was particularly striking. It suggests that even a relatively basic MO SARSA-based approach, when fine-tuned and applied correctly, has the potential to decipher and adopt a globally optimal policy from locally optimized policies.

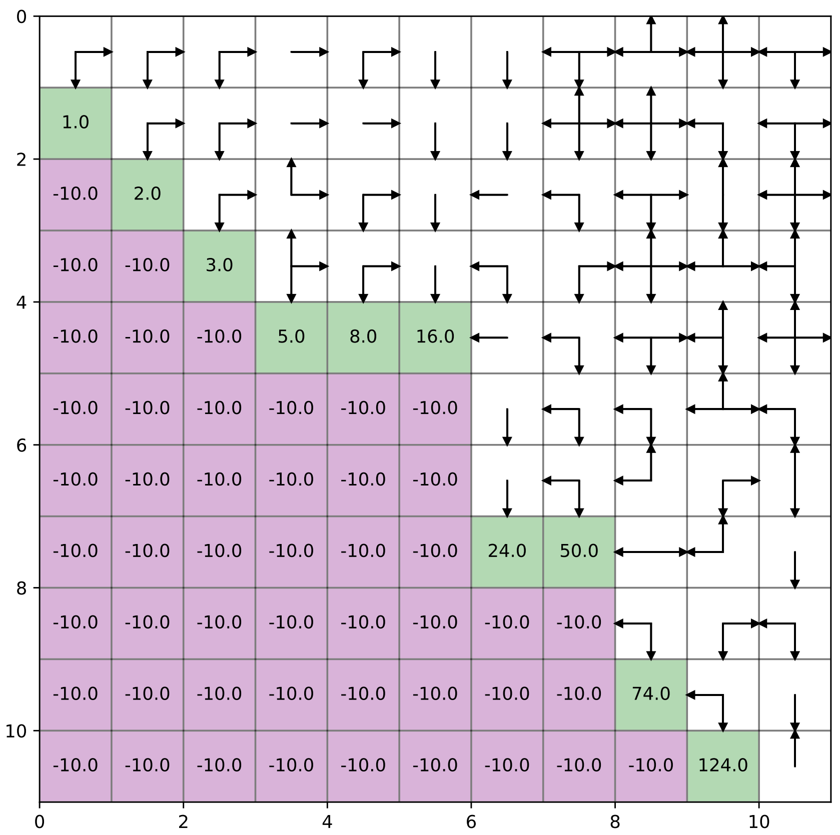


Figure 4.4 – Illustration of the Deep Sea Treasure map environment with the non-dominated actions and states, the algorithm has a good result in finding the non-dominated actions in most states. Source: The Author.

Given these revelations, it's clear that the drawbacks observed in Q-learning stem from its tendency to optimize actions for one objective, often neglecting others in the process. This can inadvertently result in wrong non-dominated actions. SARSA, with its on-policy design, offers a more adaptable and immediate approach. It prioritizes the action chosen by the policy, ensuring a consistent alignment with the current environment and objectives.

While our findings shed a promising light on this approach, it's essential to tread with caution. Our experiments have thus far been constrained to grid-sized environments, characterized by their limited state and action spaces. This specificity raises questions about the scalability and general applicability of our SARSA method in more expansive and intricate domains.

Furthermore, in our quest for a deeper understanding and to potentially validate or challenge our insights, we reached out to the original authors of (MOFFAERT; NOWé, 2014). Our primary goal was to discuss the nuances between locally and globally optimized solutions, seeking clarity about the topic. Sadly, we have yet to receive a response.

In light of this, integrating SARSA's into the PDQN framework seems like a natural progression. This could not only streamline the algorithm but potentially enhance its computational speed and accuracy. By eliminating the necessity for artificial sample points, the algorithm could become more straightforward and more attuned to real-time decision-making.

5 CONCLUSION

As seen in this work, the landscape of MORL is becoming increasingly complex. Balancing multiple objectives is often a challenging task, especially in model-free environments. This study contributes to the field by providing a comprehensive analysis of MORL algorithms based on Pareto Dominating Policies (PDP), with a specific focus on Pareto Q-Learning (PQL) (Section 2.3.1) and Pareto Deep Q-Networks (PDQN) (Section 2.3.2).

Also, our research evaluated the performance of PQL in the Deep Sea Treasure environment (Section 3.1) and assessed PDQN in both the Deep Sea Treasure (Section 3.2) and a simulated urban traffic setting (Section 3.3). We identified prevalent challenges in these algorithms, such as the generation of non-optimal policies for later rewards and the poor results in managing large state spaces.

To address these challenges, we proposed enhancements to the PDQN algorithm and introduced a novel MORL technique rooted in Pareto-Dominating Actions (Chapter 4). Preliminary tests indicate that this innovative approach shows promise in enhancing the effectiveness of action selection in such contexts (Section 4.2). This serves as the primary contribution of this study, offering both a critical analysis of existing methods and a forward-looking perspective on the field of MORL based on Pareto Optimality.

While our research has made strides in understanding the complexities and challenges of MORL algorithms based on PDP, there is still much to be done. Future work could focus on:

1. Extensive testing of the proposed enhancements to the PDQN algorithm.
2. Further refinement of the novel SARSA MORL technique based on Pareto-Dominating Actions.
3. Exploration of other real-world applications where these algorithms can be effectively deployed.
4. Investigating the scalability of the SARSA MO algorithm in larger state spaces.

By thoroughly examining the current state (their architecture and effectiveness in dealing with MO problems) of MORL algorithms based on PDP, this study has shed light on their inherent challenges and limitations. More importantly, it has proposed innovative solutions to these challenges, paving the way for other MORL algorithms based on Pareto Optimality.

In summary, this study not only sheds light on the inherent challenges and limitations of current MORL algorithms based on PDP but also proposes a solution based on non-dominated actions. This innovation can pave the way for more effective and robust MORL algorithms, thereby enriching the domain and inspiring future research endeavors.

REFERENCES

ABDOOS, M.; MOZAYANI, N.; BAZZAN, A. Traffic light control in non-stationary environments based on multi agent q-learning. In: . [S.l.: s.n.], 2011.

ALEGRE, L. N. et al. MO-Gym: A library of multi-objective reinforcement learning environments. In: **Proceedings of the 34th Benelux Conference on Artificial Intelligence BNAIC/Benelearn 2022**. [S.l.: s.n.], 2022.

BRAUER, M.; HYSTAD, P. Traffic-related air pollution and health in canada. **Environmental Pollution**, Elsevier, v. 252, p. 1075–1083, 2019.

CAI, Q.; LIJIA, M.; GONG, M. A survey on network community detection based on evolutionary computation. **International Journal of Bio-Inspired Computation**, v. 8, 01 2014.

COOLS, O.; MOONS, E.; WETS, G. Self-organising traffic lights: A tool for pedestrian management. **Transportation Research Part C: Emerging Technologies**, Elsevier, v. 26, p. 58–74, 2013.

ELIASSON, J. The social costs of traffic congestion: A review of the literature. **Transport Policy**, Elsevier, v. 16, n. 6, p. 362–368, 2009.

HAO, S.; YANG, L. Traffic network modeling and extended max-pressure traffic control strategy based on granular computing theory. **Journal of Advanced Transportation**, Hindawi, v. 2019, p. 1–12, 2019.

HAYES, C. et al. A practical guide to multi-objective reinforcement learning and planning. **Autonomous Agents and Multi-Agent Systems**, v. 36, 04 2022.

HO, C. Q.; MULLEY, C. Environmental impacts of urban traffic congestion: A review. **Transport Policy**, Elsevier, v. 29, p. 217–224, 2013.

JALALIMANESH, A. et al. Multi-objective optimization of radiotherapy: distributed q-learning and agent-based simulation. **Journal of Experimental & Theoretical Artificial Intelligence**, Taylor Francis, v. 29, n. 5, p. 1071–1086, 2017. Disponível em: <https://doi.org/10.1080/0952813X.2017.1292319>.

LI, Z. et al. Traffic signal timing optimization based on intersection delay. **Transportation Research Part C: Emerging Technologies**, Elsevier, v. 71, p. 464–478, 2016.

LUC, D. T. Pareto optimality. In: ____. **Pareto Optimality, Game Theory And Equilibria**. New York, NY: Springer New York, 2008. p. 481–515. ISBN 978-0-387-77247-9. Disponível em: <https://doi.org/10.1007/978-0-387-77247-9_18>.

MNIH, V. et al. Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 518, n. 7540, p. 529–533, fev. 2015. ISSN 00280836. Disponível em: <http://dx.doi.org/10.1038/nature14236>.

MOFFAERT, K. V.; NOWé, A. Multi-objective reinforcement learning using sets of pareto dominating policies. **Journal of Machine Learning Research**, v. 15, n. 107, p. 3663–3692, 2014. Disponível em: <http://jmlr.org/papers/v15/vanmoffaert14a.html>.

OMAR, M. K.; GOMAA, W. Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. **Engineering Applications of Artificial Intelligence**, v. 29, 03 2014.

QIN, Y. et al. An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning. **The Journal of Supercomputing**, v. 76, 01 2020.

REYMOND, M. et al. Actor-critic multi-objective reinforcement learning for non-linear utility functions. In: . [S.l.: s.n.], 2021.

REYMOND, M.; NOWE, A. Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems. In: **Proceedings of the Adaptive and Learning Agents Workshop 2019 (ALA-19) at AAMAS**. [s.n.], 2019. Null ; Conference date: 13-05-2019 Through 14-05-2019. Disponível em: <https://ala2019.vub.ac.be>.

SCHREIBER, L. et al. On the explainability and expressiveness of function approximation methods in rl-based traffic signal control. In: **2022 International Joint Conference on Neural Networks (IJCNN)**. Padova, Italy: IEEE, 2022. Disponível em: <https://doi.org/10.1109/IJCNN55064.2022.9892422>.

SCHREIBER, L. V. et al. On the explainability and expressiveness of function approximation methods in rl-based traffic signal control. In: **2022 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2022. p. 01–08.

SHAH, M.; ZHOU, B. A survey of urban traffic management and the application of artificial intelligence techniques. **Transportation Research Procedia**, Elsevier, v. 37, p. 92–99, 2019.

SHEN, R. et al. Generating behavior-diverse game ais with evolutionary multi-objective deep reinforcement learning. In: . [S.l.: s.n.], 2020. p. 3343–3349.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. Cambridge, MA, USA: A Bradford Book, 2018. ISBN 0262039249.

VAMPLEW, P. et al. Empirical evaluation methods for multiobjective reinforcement learning algorithms. **Machine Learning**, v. 84, p. 51–80, 07 2011.

YU, Y. et al. Multi-objective optimization for uav-assisted wireless powered iot networks based on extended ddpg algorithm. **IEEE Transactions on Communications**, PP, p. 1–1, 06 2021.

ZHANG, H. et al. CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In: **The World Wide Web Conference**. ACM, 2019. Disponível em: <https://doi.org/10.1145%2F3308558.3314139>.

ZHANG, L.; LEVINSON, D. Traffic congestion and reliability: Trends and advanced strategies for congestion mitigation. **Transport Policy**, Elsevier, v. 11, n. 3, p. 229–245, 2004.

ZHENG, G. et al. Learning phase competition for traffic signal control. In: . [S.l.: s.n.], 2019. p. 1963–1972. ISBN 978-1-4503-6976-3.