Multi-objective Deep Reinforcement Learning for predictive maintenance of road networks

Master Thesis

Konstantinos Krachtopoulos

Supervisors:

Frans Oliehoek

Charalampos Andriotis

Robert Loftin

Faculty of Computer Science, Electric Engineering, Mathematics & Computer Science (EEMCS)

Delft University of Technology

Acknowledgements

Embarking on a Master Thesis project, especially within the esteemed halls of TU Delft, has been a formidable endeavor in itself. The weight of this challenge, coupled with the uncertainties of the future and the daily hurdles, can create an atmosphere that is almost unbearable even for the most motivated individuals.

However, it is during these trying times that the people around you shine the brightest. I am deeply grateful to have such incredible and supportive individuals by my side, guiding me every step of the way.

First and foremost, my heartfelt appreciation goes to my family, whose unwavering support has accompanied me from day one. They have championed every decision I made, stood up for my passions, and provided invaluable mental fortitude. To my family, you are the epitome of what a family should be.

Next, I am immensely fortunate to have had the presence of good friends like Giannis, Thanasis, Christos, Lefteris, and Alex. Their advice and ability to alleviate the pressures of life have been nothing short of lifesaving. Their friendship has instilled in me a sense of confidence and direction, allowing me to forge ahead with determination. I want to especially thank Giannis Kotsakiachidis, who decisively helped me to shape the motivation of this project, with his knowledge in Road maintenance planning operations.

Furthermore, I am blessed to have a life partner in Akrivi, who has stood beside me through both joyful and challenging moments over the past years. We embarked on this entire Master's journey together, and without you, Akrivi, I could not have made it. Thank you for your unwavering support.

Lastly, I extend my gratitude to all the individuals who made this project virtually possible. To my daily supervisors, Charalampos and Robert, and my Thesis advisor, Frans, I express my deepest appreciation for being exceptional mentors throughout this thesis project adventure. Your guidance, responsiveness to my queries, and willingness to brainstorm ideas have been invaluable.

Additionally, I would like to extend my thanks to Wendelin, who graciously agreed to

support my project by becoming a part of the Thesis committee.

Completing this journey fills me with an overwhelming sense of gratitude. Despite being told countless times that "no civil engineer can be admitted to the TU Delft Computer Science master programme", I have reached this milestone, which instills me with unwavering confidence that anything is possible with the right level of determination.

I eagerly anticipate the next adventure that awaits me.

Regards,

Delft University of Technology

Delft, June 2023

Konstantinos Krachtopoulos

Abstract

Operation and maintenance of the built environment have a major effect on socioeconomic stability and sustainability. A significant part of our built world approaches or has well exceeded its designated structural life. As engineers, we need to find efficient ways to extend this life while maintaining acceptable levels of safety and performance. In this direction, smart and adaptive life-cycle inspection and maintenance planning are of paramount importance to reduce costs, increase structural reliability, and minimize resource-intensive interventions. Deep reinforcement learning provides a novel approach to strategize these decisions for systems subject to uncertainties and deterioration.

The goal of this project is to determine optimal life-cycle inspection and maintenance policies for road networks. Road maintenance and inspection planning is a complex task, involving a multitude of different objectives, such as the minimization of lifecycle cost and CO2 emissions (both from the maintenance works and vehicle emissions). However, current literature has underestimated the importance of optimizing for multiple objectives, and doesn't consider the environmental impact during planning. In the current project, we aim to optimize the maintenance and inspection schedule with respect to minimizing the maintenance costs, carbon emissions and vehicle owners costs at the same time. To achieve that, a multi-objective road network environment will be modelled and multiple multi-objective Deep Reinforcement learning approaches will be applied and compared to traditional life-cycle management policies.

Keywords – Deep Reinforcement Learning, Multi-objective optimization, Road maintenance planning

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

CBM Condition Based Maintenance vii, viii, 26, 59, 60, 64–66, 74–77, 91

DOL Deep Optimistic Linear Support Learning v, viii, ix, 47–50, 74, 75, 77, 91

DOL-PR Deep Optimistic Linear Support Learning with partial reuse viii, ix, 48, 50, 51, 74, 75, 77, 78, 80, 92

DRL Deep Reinforcement Learning ix, 2, 5, 12, 13, 20, 26, 27, 49–51, 54, 58, 59

IM Inspection and Maintenance 35, 39, 76

IRI International Roughness Index vi–viii, 31–34, 37, 38, 60, 70–72

MDP Markov Decision Process 9, 12, 15

MORL Multi-objective Reinforcement Learning v, viii, ix, 16, 18, 20, 21, 27, 43, 44, 47, 51, 55, 59, 61, 63, 72–80

OLS Optimistic Linear Support x, 20–25, 47–49

PFA Pareto Following Algorithm vii–ix, 55–59, 73–75, 78, 93

POMDP Partially Observable Markov Decision Process 10

PPO Proximal Policy Optimization vi, 14, 15, 19, 44–47, 50, 51, 54, 59, 64, 66, 67, 74, 77

RA Radial Algorithm v, vii–ix, 53–55, 59, 75, 77, 78, 92

RL Reinforcement Learning vi, 11–14, 19

1 Introduction

1.1 Motivation

Timely and efficient road maintenance is crucial to ensure the safety and functionality of global transportation systems. As a result, road maintenance planning is an essential aspect of the overall transportation management strategy. Weather conditions, natural deterioration over time, and traffic volume are just a few examples of factors that can cause damage and create hazards on roads.

Road maintenance is a complex and multi-faceted issue that requires consideration of multiple actors and objectives (Table 1.1). From governmental decision-makers to private contractors and road users, each stakeholder brings their priorities and concerns to the table either directly or indirectly. At the highest level, government officials must balance competing priorities and make strategic decisions about how to allocate resources to maintain the nation's roads. Most importantly, they need to reach specific sustainability goals. For example, based on the Climate Act of May 28, 2019, the Dutch government has committed to reducing the Netherlands' greenhouse gas emissions by 49% by 2030 compared to the 1990 levels (Ministerie van Economische Zaken, 2020). Meanwhile, public authorities tasked with designing maintenance schedules must prioritize safety above all else, working to ensure that roads are kept in a safe and adequate condition to avoid accidents, while at the same time reduce the maintenance costs as much as possible. Private contractors, on the other hand, are focused on cost efficiency and competitiveness, seeking to maximize their profit margins while delivering high-quality results. Finally, road users themselves have their own set of concerns, including minimizing their traffic delays and their cost for operating and maintaining their vehicle. By considering the needs and objectives of each stakeholder group, a comprehensive road maintenance strategy can be developed that balances the interests of all parties involved.

Actor	Objectives
Government regulators	1. **Reach sustainability goals** 2. Maintain political stability
Public decision-makers	1. Maximize road safety 2. **Minimize maintenance costs**
Private contractors	1. **Maximize profit margin**
Road network users	1. Minimize traffic disruptions 2. **Minimize vehicle costs**

Table 1.1: The different actors involved in the road maintenance problem. The objectives that will be of concern for the present study are displayed in bold.

Hence, trading-off between the different maintenance strategies, such as routine maintenance works and heavy restoration works, is not an easy task. While routine maintenance tasks are cost-effective in the short term, they do not address underlying structural issues. Conversely, heavy restoration tasks can enhance road quality, but at the same time, they come at a high cost and cause significant traffic disruption and carbon emissions.

To achieve efficient and effective road maintenance planning, it is essential to carefully evaluate different objectives, such as the monetary cost of maintenance, the environmental impact, the safety, and the convenience of passengers. Deep Reinforcement Learning (DRL) is a promising tool in the field of asset maintenance scheduling. This revolutionary technology has been effective in finding optimized Maintenance and Inspection (M&I) strategies for various assets, including road networks. For instance, recent studies have shown that DRL approaches can reduce maintenance costs by up to 40% compared to standardized maintenance strategies (Saifullah et al., 2022).

1.1.1 Why we need to plan for multiple situations

The task of road maintenance scheduling is inherently complex, as it involves balancing multiple objectives. However, a significant challenge arises from the uncertainty surrounding the preferences associated with these objectives during the initial planning stages. Various external factors, such as budget allocations and political dynamics, can significantly influence the final maintenance schedule. In this thesis, we aim to delve into the complexities of multi-objective road maintenance scheduling and propose an approach that allows for greater adaptability in response to the dynamic nature of these external

factors.

In order to better understand why this research is important, one needs to be aware of the decision process that takes place within the planning departments of the Municipalities and Provinces, which are responsible for maintaining our roads. This is presented on a high-level in Figure 1.1. A key challenge in road maintenance scheduling is the lack of upfront knowledge regarding the budget to-be-allocated. The planning departments typically have only a rough estimate of the available budget. Consequently, they must plan maintenance activities based on empirical knowledge and assumptions. Once the initial maintenance plan is constructed, a budget proposal is submitted to the respective financial department, which may either approve the proposed budget or deny it. In the case of a denial, asset managers are required to re-plan the road maintenance schedule in accordance with a smaller budget. In this scenario, prioritization becomes crucial, with a focus on the most impactful and urgent maintenance tasks. The iterative process of developing and submitting a maintenance part for approval stops when the proposed budget is approved in its entirety. Based on personal discussions with asset managers of dutch Provinces, it can take more than a year to complete this budget approval process.

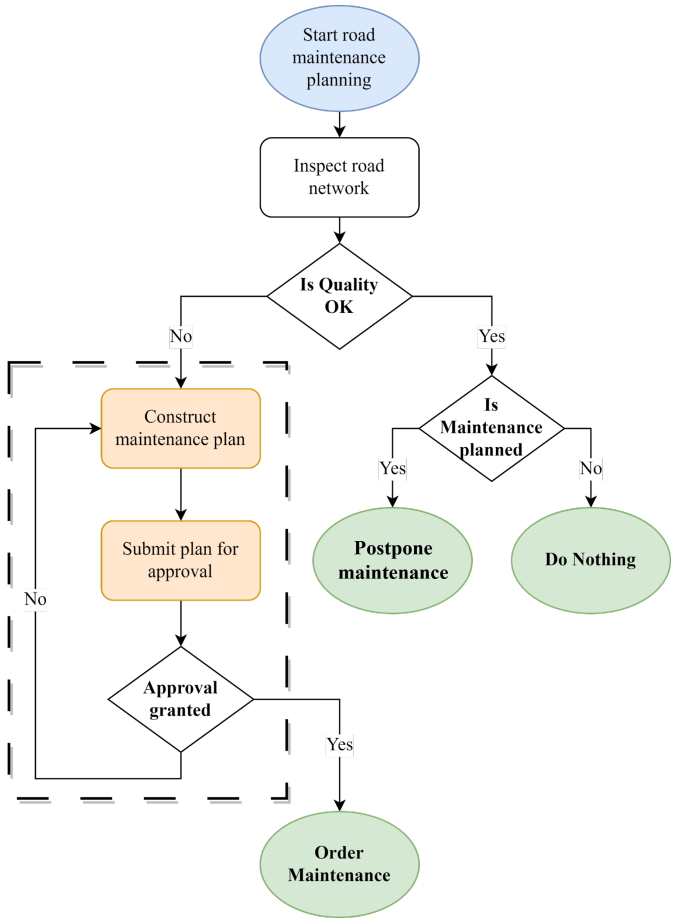


Figure 1.1: Indicative flow chart of how road maintenance planning is performed within a public organization. The black dashed box includes the iterative process of re-constructing the road maintenance plan and requesting budget approval.

In addition to budget uncertainties, preferences in multi-objective road maintenance planning can be heavily influenced by changes in the political system. When a new governing party takes office, it may decide to allocate more resources toward maintaining the road network. Conversely, another party may choose to reduce the road maintenance budget in favor of other sectors. Furthermore, as discussed in the Motivation section, EU regulations now impose stringent climate goals on member countries (Ministerie van Economische Zaken, 2020). These compliance guidelines are expected to shift the maintenance sector's focus towards more eco-friendly solutions in the near future, leading to an increased preference for sustainable road maintenance plans.

Considering the multifaceted factors discussed above, it becomes evident that relying on a fixed maintenance schedule is impractical for effectively addressing the complexities of our problem. Instead, our approach will focus on computing a diverse set of solutions that

cover various combinations of preferences for the studied objectives. By doing so, asset managers will gain the ability to better adapt their work to the dynamic changes in the environment. This flexibility not only ensures that maintenance plans remain relevant and efficient but also enables the effective allocation of resources based on changing priorities.

In conclusion, the multi-objective road maintenance scheduling problem is beset by various challenges due to the unknown preferences associated with objectives and the influence of external factors. Budget uncertainties, political dynamics, and environmental regulations necessitate an adaptable approach that can accommodate evolving preferences and priorities. By computing a diverse set of solutions, our proposed methodology empowers asset managers to navigate the complexity of road maintenance scheduling and make informed decisions that align with the ever-changing circumstances. This flexibility enables the optimization of resource allocation and enhances the ability to adapt to the dynamic nature of the road maintenance environment.

1.1.2 Our proposed approach

As discussed above, road maintenance planning is a complex issue with significant societal impact, and as such, it is crucial that the maintenance and inspection policy is as flexible and adaptable as possible. In a system containing multiple stakeholders with varying priorities and preferences, it is essential to develop a set of policies that can cater to all possible preference combinations. To address this challenge, our study aims to develop a multi-objective Deep Reinforcement Learning (DRL) framework for the road maintenance scheduling problem.

We will begin by creating a realistic multi-objective road network environment and compare the effectiveness of various Multi-Objective DRL algorithms. The algorithms will be evaluated based on their ability to generate a diverse set of policies that can adapt to all possible combinations of preferences for three objectives: the minimization of cost of maintenance, carbon emissions due to traffic disruption, road condition and maintenance actions, and cost for vehicle owners.

By developing a comprehensive set of policies that considers the diverse needs and preferences of all stakeholders, our study will contribute to a better understanding of the potential of Multi-Objective DRL algorithms in the context of road maintenance

planning. This will enable policy-makers to make informed decisions that are in line with the interests of all stakeholders, resulting in more efficient, effective, and sustainable maintenance and inspection policies.

1.2 Research Questions

The research questions for this study can be formulated as follows:

1. *How a Road Network digital twin environment can be enhanced to include, the lifecycle carbon emissions?*
2. *How a Road Network digital twin environment can be enhanced to include the lifecycle cost for vehicle owners?*
3. *How efficient are outer-loop multi-objective Reinforcement Learning algorithms in solving the multi-objective road network maintenance scheduling problem?*

The two first research questions are essentially prerequisites for the third one, since in order to evaluate multi-objective reinforcement-learning algorithms, a multi-objective environment is needed.

2 Background

2.1 Markov Decision Process (MDP)

MDPs provide a mathematical framework for modelling sequential decision-making problems, making them a foundational concept in the field of reinforcement learning. In an MDP, an agent interacts with an environment over time by choosing actions at each time step and receiving rewards based on those actions. The environment changes based on the current state and the action done, and the agent's goal is to maximize its cumulative reward over time. (Sutton and Barto, 2018)

The Markov property, which asserts that the current state contains all relevant information about the past and future, is the fundamental assumption underlying MDPs. In other words, given the current state, the future is conditionally independent of the past. This property enables the agent to make decisions based solely on the current state, without explicitly considering the history of previous states and actions.

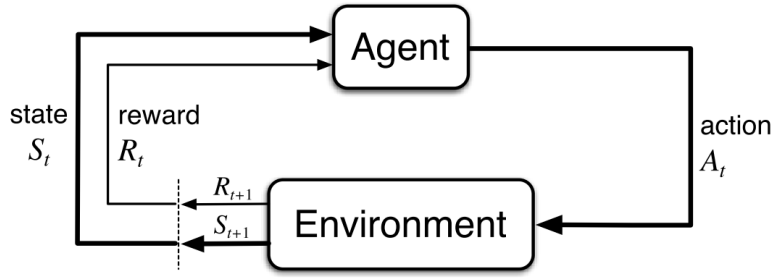


Figure 2.1: The agent–environment interaction in a Markov decision process. (Sutton and Barto, 2018)

Typically, MDPs are represented by the tuple (S, A, P, R, γ), where:

- S is the set of conceivable environmental states.
- A is the set of possible actions the agent can take.
- P is the state transition function, which specifies the probability of transitioning to a new state s' given the agent's current state s and action a : $P(s'|s, a)$.
- R is the reward function, which specifies the agent's instantaneous reward for performing action a in state s : $R(s, a)$.

- γ is the discount factor that determines the relative value of instantaneous rewards versus future rewards. A discount factor of 1 indicates that all prospective rewards are of equal importance, whereas a discount factor of 0 indicates that only immediate rewards are significant.

Once an Markov Decision Process (MDP) has been defined, the agent's objective is to discover the action for each state so that the sum of the discounted rewards it receives over the future is maximized. In particular, it chooses actions a_t to maximize the expected discounted return G_t, over a horizon T:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{T} \gamma^k R_{t+k+1} \tag{2.1}$$

The action-value function $Q_\pi(s, a)$, with $s \in S$ and $a \in A$, is defined as the maximum expected return achievable by following a particular policy $\pi : S \rightarrow A$, after observing some state s and then taking some action a:

$$Q_\pi(s_t, a_t) = \mathbb{E}[G_t \mid s_t = s, a_t = a, \pi] \tag{2.2}$$

Combining Equations 2.1 and 2.2, the following recursive form of the action-value function is retrieved:

$$Q_\pi(s_t, a_t) = R(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}}[Q_\pi(s_{t+1}, a_{t+1}) |\ s_t, a_t] \tag{2.3}$$

Moreover, the value function $V_\pi(s_t)$ from state s_t and for policy π is the expected value of the action-value function over all the possible actions at step t:

$$V_\pi(s_t) = \mathbb{E}_{a_t \sim \pi}[Q_\pi(s_t, a_t)] \tag{2.4}$$

The optimal policy π^* is the one that maximized the value functions V_π^* and Q_π^*, and can be found by using the Bellman Equation:

$$V_\pi^*(s_t) = \max_{a_t \in A} \{R(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} |\ s_t, a_t) V^*(s_{t+1})\} \tag{2.5}$$

Hence, the Bellman Equation is a dynamic formulation of the value function and it can be used to find its optimal V^*, and hence the optimal policy π^*. However, as the state-action space grows, then the number of samples needed, and hence the computational time, grow exponentially (Bellman, 1961).

2.2 Partially Observable MDPs (POMDPs)

POMDPs (Kaelbling et al., 1998) are an extension of MDPs that enable the modeling of decision-making problems in which the agent lacks access to the complete state of the environment. At each time step in a POMDP, the agent only has access to a noisy or fragmentary observation of the environment, as opposed to the full state. This adds uncertainty to the decision-making procedure because the agent must reason about the fundamental state of the environment based on its observations.

Formally, a Partially Observable Markov Decision Process (POMDP) with a finite horizon is represented by the tuple $(S, A, P, \Omega, O, R, \gamma)$, where S, A, P, and R represent the state set, action set, reward function, and transition function, just as in the case of the MDP. Additionally:

- Ω is the observation set.
- O is the probabilistic observation model, where $O(o_0, s_0, a) := P(o_0 |\ s_0, a)$ is the probability that observation o_0 will be perceived if state s_0 was reached after executing action a on the previous time step.

To manage partial observability, the agent employs Bayesian filtering to maintain a estimate on the system state b. The estimate is a probability mass function (PMF) over the system state and is a sufficient statistic for the history of prior actions and observations. The POMDP literature refers to these estimations as belief states. Given the history of prior actions and observations, the belief state at any given time step is defined as the conditional probability distribution over the state. b_0 is the initial belief state before taking any actions or perceiving any observations. Given the current belief state b, the action a, and the resulting observation o', Bayes' rule yields the updated belief state b':

$$b'(s') = \frac{P(o' \mid s', a) \sum_s P(s' \mid a, s) b(s)}{\eta(o' \mid b, a)} \tag{2.6}$$

where the denominator is the prior probability of observing o':

$$\eta(o' \mid b, a) = \sum_{s'} P(o' \mid s', a) \sum_{s} P(s' \mid a, s) b(s) \tag{2.7}$$

2.3 Reinforcement Learning (RL)

In contrast to MDPs and POMDPs, which require specific probabilistic models (state transition function, reward function and observations model) for the environment of the entire system, RL methods directly interact with the environment, eliminating this need (Moerland et al., 2020). A common technique to learn the action-value function, Q, is by implementing temporal difference updates, while collecting samples from the environment (Sutton and Barto, 2018) :

$$Q(s_t, a_t) \leftarrow Q_{s_t, a_t} + \eta(Q(s_t, a_t) - y_t) \tag{2.8}$$

where y_t is the estimate of future return:

$$y_t = R(s_t, a_t) + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a + t + 1) \tag{2.9}$$

and η is the problem learning rate. Temporal difference learning is just one example of a model-free RL algorithm. Model-free RL algorithms directly learn a policy or value function without explicitly learning the dynamics of the environment (Calisir and Kurt Pehlivanoğlu, 2019). The policy or value function is learned through trial and error, by repeatedly choosing actions and observing their consequences. Other model-free RL algorithms include Q-learning (Watkins, 1989) and SARSA (Rummery and Niranjan, 1994). In situations where the environment is complex and building a model of the environment is challenging or impossible, model-free RL is often preferred. Moreover, instead of trying to find the value of every point in the state-action space, the learning problem in RL can be also approached by directly updating the policy. (Sutton and Barto, 2018).

In contrast, model-based RL is a category of RL algorithms that builds an environment model and then utilize it to make decisions. In other terms, the agent constructs an

environment representation by learning the transition dynamics and reward function (Luo et al., 2022). This representation is then used to simulate the environment and plan for the future while optimizing some objective function. Some examples of model-based RL algorithms are Model Predictive Control (MPC) (Camacho et al., 2003) and Dyna (Sutton, 1991). Model-based RL is more sample-efficient and in principle works better in situations where the environment is straightforward enough to be accurately modeled and where optimal decision making requires careful planning (Luo et al., 2022) . However, since this is not usually the case with real-life environments, model-based methods are usually not preferred.

2.3.1 Deep Reinforcement Learning (DRL)

While RL methods have proven very effective in solving relatively simple environments, when the action-state space grows to extreme values, it becomes difficult to find the optimal policy π^*.

For instance, Montezuma's Revenge is a well-known Atari game that requires the player to traverse a maze-like environment while avoiding enemies and traps and gathering rewards. Modelled as an MDP (Brockman et al., 2016), the game features a high-dimensional observation space consisting of a 210x160 pixel screen image, as well as a large action space with 18 possible actions. Due to the complexity of the observation space and the size of the action space, traditional RL methods such as Q-learning or SARSA may struggle to learn an effective policy in this environment.

To tackle the problem with complex environment, DRL was introduced (Mnih et al., 2015). The key idea is to substitute the value and action-value functions with deep neural networks, which are global function approximators such as:

$$F \simeq F'(\cdot|\theta^F) \tag{2.10}$$

where F is one of the value or action-value functions, and F' is a deep network approximation, with parameters $\theta^F \in \Theta$. Therefore, the problem of determining the values for every point of the state-action space, reduces to determining the parameters in Θ, which in principle will be much less than the size of a highly-complex state-action space.

Figure 2.2: The well known Montezuma's Revenge game of Atari console. Due to the very large action-state space, it becomes difficult for traditional RL algorithms to find the optimal policy. (Brockman et al., 2016)

2.3.2 Deep Policy Gradients

Policy gradient methods are a widely used category of DRL algorithms, which rather than trying to find the optimal policy by finding the state-action pairs for each the value function is optimized, directly optimize the policy $\pi_\theta(a_t|s_t)$. They achieve that by initializing a policy π_θ with a parameter vector θ, and then update it by applying stochastic gradient descent on the returns (Equation 2.1), with respect to θ. In Deep Policy Gradients, the policy π_θ is approximated by a neural network with parameters θ. Then the gradient of policy π_θ is given by the Policy Gradient Theorem (Sutton et al., 1999):

$$\nabla_\theta J_\theta = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[G(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{(s_t,a_t)\in\tau} \nabla_\theta \log(\pi_\theta(a_t|s_t))G_t] \tag{2.11}$$

where $\tau \sim \pi_\theta$ corresponds to a trajectory sampled from the distribution of policy π_θ.

However, because the sampled returns G_t can have a high variance, they can be substituted with the advantage function, A_t^π, (Schulman et al., 2018), a bias-free measure which is the difference between the action-value and the value function (Equation 2.12). It expresses

how advantageous an action is, compared to the expected value of the state.

$$A_t^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) \tag{2.12}$$

2.3.3 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a state-of-the-art Actor-Critic RL algorithm that has showcased a great capacity to learn effective policies with high sample efficiency and stability (Schulman et al., 2017). PPO is a policy gradient algorithm that operates on a stochastic policy and generates a probability distribution for the agent's actions.

Instead of trying to optimize for the return like other Policy Gradient methods, PPO is a Trust Region method, optimizing a surrogate objective. The same technique is followed by other Trust Region methods, such as Trust Region Policy Optimization (TRPO) (Schulman et al., 2015). However, the addition of PPO is that it clips the surrogate objective to a region surrounding the current policy, thereby preventing large updates that could lead to instability or divergence (Equation 2.13). This constraint is met by introducing a truncated substitute objective function that limits the gap between the new and old policies.

$$L^{CLIP}(\theta) = \mathbb{E}_t[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{2.13}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)}$ is the probability ratio between the current and old policy.

The clipping trick limits the policy from making very large updates. If the advantage A_t is positive (Figure 2.3, right), meaning that the trajectory had a better than expected return, clipping doesn't allow the new probability of selecting this trajectory $\pi_\theta(a_t, s_t)$ to be a lot higher than the previous probability $\pi_{\theta_{old}}(a_t, s_t)$ (Figure 2.3, left). Conversely, when the advantage is negative, and the trajectory had a worse than expected return, clipping doesn't allow the probability ratio to be less than $1 - \epsilon$ (Figure 2.3, right).

As an Actor-Critic algorithm, it employs two neural networks; an actor network, which calculates the probability distribution over the actions for a state $\pi_\theta(a_t|\ s_t)$, and the Critic network, that calculates the expected value $V(s_t)$ of the current state, which is used in the advantage calculation.

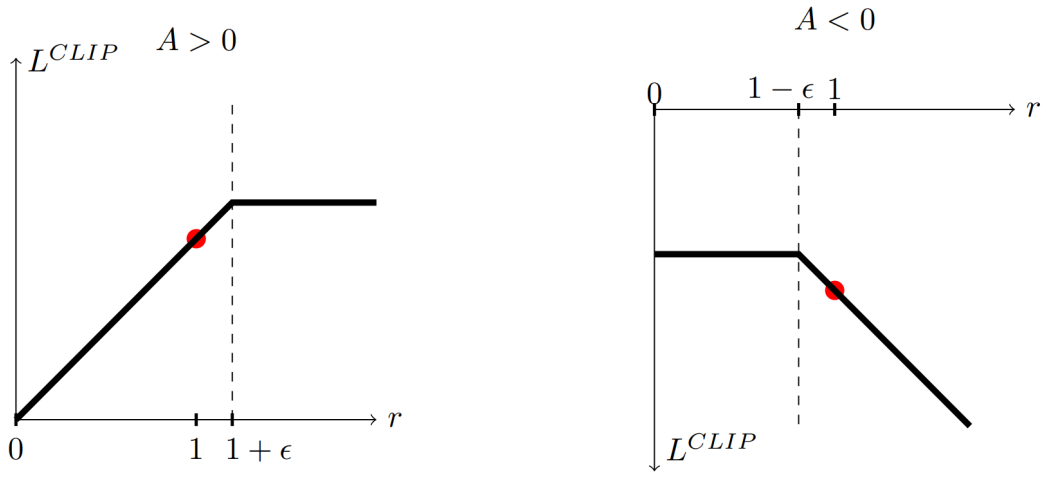


Figure 2.3: Visualization of the clipped surrogate objective of PPO (Schulman et al., 2017).

2.4 Multi-objective Reinforcement Learning (MORL)

2.4.1 Multi-objective MDPs (MOMDPs

A MOMDP is an MDP in which the reward function $R(s_t, a_t) = r_t \in \mathbb{R}^d$ is a vector of size d the number of objectives. The value function has the same form with single-objective MDPs (Equation 2.4) , but is also a vector $\boldsymbol{V_\pi} \in \mathbb{R}^d$. This makes the comparison of the value vectors of different state-action pairs non-trivial.

Having access to a utility function (or scalarization function) that can map the value vector to a single value $\mathbb{R}^d \rightarrow \mathbb{R}$ can make the ordering of value vectors possible, and thus reduce the MOMDP to a single-objective problem.

$$V_u^\pi = u(\boldsymbol{V^\pi}) \tag{2.14}$$

where V_u^π is a scalarized value of the vector value $\boldsymbol{V^\pi}$

However, it is not always the case that a utility function is available. In this case, it is possible to encounter a situation in which $V_i^\pi > V_i^{\pi'}$ for objective i but $V_j^\pi < V_j^{\pi'}$ for objective j. Therefore, only partial ordering of value functions if allowed, and determining the optimal policy is only possible with additional information on the prioritization of the objectives (Hayes et al., 2021).

As a result, an MOMDP decision problem with no additional information for the objectives

can have multiple optimal policies. The solution set that contains all possible policies Π and the corresponding value vectors, for which there is a possible utility function u with a maximal scalarized value is called the **Undominated Set**:

$$U(\Pi) = \{\pi \in \Pi \mid \exists u, \forall \pi' \in \Pi : u(\boldsymbol{V^{\pi}}) \geq u(\boldsymbol{V^{\pi'}})\}) \tag{2.15}$$

The solution set that contains exactly one optimal policy π for every utility function u, is called the **Coverage Set (CS)**, and is a subset of the Undominated Set. In order to keep the MORL solution simple, it is desirable to have a CS that is as small as possible. Of course, the CS set is not always unique for a given problem, as one utility function can have multiple optimal policies.

$$CS(\Pi) \subseteq U(\Pi) \cap \{\forall u, \exists \pi \in CS(\Pi), \forall \pi' \in \Pi : u(\boldsymbol{V^{\pi}}) \geq u(\boldsymbol{V^{\pi'}})\} \tag{2.16}$$

In simple words, the above relation is not an equation, but rather tells us that the coverage is a subset of U, containing one policy π for every utility function u that maximizes this utility function.

A minimal assumption that is made in MORL problems is that the utility function (even if it is unknown) is **monotonically increasing**, which corresponds to "always wanting more value in any of the objectives". More formally, a monotonically increasing utility function u is defined as follows:

$$(\forall i : \ \boldsymbol{V_i^{\pi}} \geq \boldsymbol{V_i^{\pi'}}) \cap (\exists i : \boldsymbol{V_i^{\pi}} > \boldsymbol{V_i^{\pi'}}) \Longrightarrow u(\boldsymbol{V^{\pi}}) \geq u(\boldsymbol{V^{\pi'}}) \tag{2.17}$$

Moreover, several research works have assumed that the utility function has a positively-weighted linear sum shape $u(\boldsymbol{V^{\pi}}) = \boldsymbol{w}^T \boldsymbol{V}^{\pi}$, where the weight vector $\boldsymbol{w} \in \mathbb{R}^d$ specifies the preference for each objective. The solution set that contains all possible policies Π and the corresponding value vectors, for which there exists a $\boldsymbol{w}$ for which the linearly scalarized utility function u with takes its maximal value, is called a **Convex Hull (CH)**. Formally, it is formulated as follows:

$$CH(\Pi) = \{\pi \in \Pi \mid \exists \boldsymbol{w}, \forall \pi' \in \Pi : \boldsymbol{w}^T \boldsymbol{V}^{\pi} \ \geq \boldsymbol{w}^T \boldsymbol{V}^{\pi'}\} \tag{2.18}$$

Finally, in the same fashion as the Undominated Set, the Convex Hull can contain multiple policies that optimize the same utility function. In order to get a set with as few policies as possible, we define the **Convex Coverage Set (CCS)** which is a subset of the CH and contains one optimized policy with respect to every $\boldsymbol{w}$:

$$CCS(\Pi) \subseteq CH(\Pi) \cap \{\forall \boldsymbol{w}, \exists \pi \in CCS(\Pi), \forall \pi' \in \Pi : \boldsymbol{w}^T \boldsymbol{V}^{\pi} \geq \boldsymbol{w}^T \boldsymbol{V}^{\pi'}\} \tag{2.19}$$

2.4.2 Problem Taxonomy

Depending on the problem characteristics a different solution set is needed to describe the optimal solution. Roijers et al. (2014) proposed the problem taxonomy shown in Table 1, that categorizes MOMDPs according to three factors:

- Whether the utility function is known or not.
- Whether the utility function is a linear scalarization of the objective preferences or simply monotonically increasing.
- Whether the produced policies are stochastic or deterministic.

The taxonomy is based on the "Utility-based approach" which implies the use of a scalarization function that derives the scalar utility of the decision problem based on the defined preferences.

	Single policy (known weights)		**Multiple policies (unknown weigths of decision support)**	
	Deterministic	**Stochastic**	**Deterministic**	**Stochastic**
Linear scalarization	One deterministic stationary policy		Convex coverage set of deterministic stationary policies	
Monotonically increasing scalarization	One deterministic non-stationary policy	One mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	Convex coverage set of deterministic stationary policies

Table 2.1: Problem taxonomy of MOMDP's (Roijers et al., 2014)

2.4.2.1 Single vs Multiple policies

The choice of requesting a single policy or multiple policies as the solution to the decision problem lies on whether the utility function and preferences are known at the time of the

execution. For instance, in the example of Wind Farm Control in Hayes et al. (2021), if the user already knows what is power demand and what is the budget for the maintenance of the wind turbines, a single policy should be enough to solve the decision problem of when to power each wind turbine. On the other hand, if the exact power demand is unknown, then learning a coverage set of solutions is required.

2.4.2.2 Linear vs Monotonically increasing

When the utility function is linear, the final utility is calculated as a weighted average of the objective values. Then, depending on whether the utility function (hence the user preferences) are known, the solution to the problem can be one deterministic policy, or a convex coverage set that should cover all the different preference combinations between the objectives. The choice of a linear utility function is the one most usually employed due to its simplicity, as used in Guo et al. (2009) who modelled the problem of Energy Demand Management with respect to price and system stability.

However, in principle, linear utility functions can't be applied effectively when the objectives are related to user's preferences (Vamplew et al., 2008). In such cases, non-linear scalarization functions should be applied, to capture the complex relationship between the objectives. In any case, it is assumed that the scalarization function is monotonically increasing (as per Equation 2.17). This is a rather minimal constraint, as it requires only that, when all other objective values are equal, getting more reward for one objective is always better. Intuitively, it is difficult to think of any function that doesn't have this property (Roijers et al., 2014).

An example of a non-linear utility function is the Threshold Lexicographic Ordering (TLO) (Gábor et al., 1998), where the MORL agent selects actions that prioritize objectives in a lexicographic order, subject to a minimum acceptable threshold for $d-1$ objective, and trying to optimize for the last objective. Due to its nature, TLO requires some knowledge about the problem (Issabekov and Vamplew, 2012).

2.4.2.3 Deterministic vs Stochastic policies

The third factor defining the problem taxonomy of an MOMDP is whether is allows stochastic policies instead of just deterministic ones. Stochastic policies correspond to

policies that are represented as a family of conditional probability distributions $\pi(a|s)$ $s \in S$, where S is the state space. In other words, instead of every state corresponding to one action, the conditioned state only is now mapped to a probability distribution over actions. As a result, an agent following a stochastic policy may not always make the same decisions when in a certain state.

Most of the Policy Gradient RL algorithms, like PPO (discussed in Section 2.3.3), produce stochastic policies.While in most applications there is no reason to limit to only deterministic policies, there are some cases in which stochasticity is unpreferable, or even unethical, like in the work of Lizotte et al. (2012), discussing the clinical decision support.

2.4.3 Categorization of multi-policy algorithms

The multi-policy algorithms family is designed to produce a solution set containing multiple policies as the solution of an RL problem. This is because the utility function is not known a priori, and we want to account for all its different forms. Most multi-policy algorithms are based on model-free RL methods, and are divided into two main categories, the inner loop and outer loop methods.

Outer loop methods execute a single objective problem multiple times, trying to construct an approximation of the CS. In every iteration, different objective preferences are applied, in an effort to capture all the different objective combinations. Multiple such algorithms having proposed in the literature. For instance, Parisi et al. (2014) developed the Radial Algorithm, which for a fixed amount of iterations, tries to optimize approximate the CS by following a weighted sum of the gradients with respect to the problem objectives. In the same work, the proposed Pareto Following algorithm searches for a CS approximation by individually optimizing for each objective in a serial manner. Since performing a series of single-objective executions is an intensive task on each own, multiple works tried to limit the execution time by reusing information between the different iterations. (Parisi et al. (2017) used Importance Sampling to enable sample reuse, while Roijers et al. (2015) developed the Optimistic Linear Support with Alpha Reuse (OLSAR), a method that was based on the Optimistic Linear Support (OLS) algorithm (Roijers, 2016) and could reuse value functions produced when solving single-objective POMDPs in earlier iterations to

more quickly solve scalarized POMDPs in later ones by limiting the solution space. Later, the OLS algorithm was integrated with DRL to produce Deep Optimistic Linear Support (DOL) (Mossalam et al., 2016) that allowed computation of the CCS in high-dimensional problems.

Inner loop methods can produce multiple policies of the CS approximation in one algorithmic execution. Pareto-Q-Learning (PQL) (Moffaert and Nowé, 2014) and MPQ-Learning (Ruiz-Montiel et al., 2017) are methods constructed enhanced versions of Q-learning, where the Q values update of Equation 2.8 takes places for multiple Q-value vectors simultaneously. Afterwards, Mandow and Pérez-de-la Cruz (2018) presented a modification of MPQ-learning algorithm that could approximate the Pareto-optimal set of deterministic policies with an improved number of training steps by pruning dominated values. Finally, the first effort to merge inner loop MORL with DRL was done in Reymond and Nowe (2019), who enhance the PQL algorithm so that the value estimator is a neural network, and update all non-dominated vectors accordingly.

In terms of model-based approaches, Wiering et al. (2014) developed an approach to compute the Pareto front of deterministic policies. After learning the environment dynamics from experience, Pareto-optimal deterministic policies were computed using dynamic programming through CON-MDP (Wiering and de Jong, 2007). Then Yamaguchi et al. (2019) proposed a model-based method for solving stochastic environments using reward occurrence probability (ROP) with unknown weights. With ROP, the average reward of a policy is defined by inner product of the ROP vector and the weight vector. However, there still has been surprisingly little research so far in model-based MORL (Hayes et al., 2021).

Finally, several methods have been proposed for environments with continuous or very large state spaces. In such cases, the use of policy search or actor-critic methods is preferred (Deisenroth et al., 2013). Interestingly, Abdolmaleki et al. (2020) proposed an approach that learns an action distribution for each objective using policy-iteration, and then uses supervised learning to combine the distributions to fit a parametric policy. On the other side, Chen et al. (2018) viewed the MORL as a Meta-learning problem with the task distribution given by a distribution over the preferences. In other words, they tried to find a meta policy that could easily approximate any point of the Pareto front

with few-shot adaptation. However, because different policies can be concentrated in different regions of the CS, Xu et al. (2020) argued that a meta-policy may not be able to approximate all points of the CS. Then, they proposed an evolutionary MORL approach (PGMORL), using a multi-objective extension of PPO, together with an evolutionary algorithm to guide learning towards the most advantageous direction. In each generation of evolution, a prediction model is fitted based on the previous MORL iteration to determine the pairs of policies and preferences that will have the greatest impact on the overall solution. The selected policy-preference pairs are then solved with MORL to generate offspring policies, which are utilized to generate a new generation of policies. By clustering, the final generation is divided into policy families that represent distinct regions of the Pareto Front. At the end, a continuous approximation of the Pareto front is attained by interpolating the policy parameters across the various families. The overview of the full process is presented in Figure 2.4.

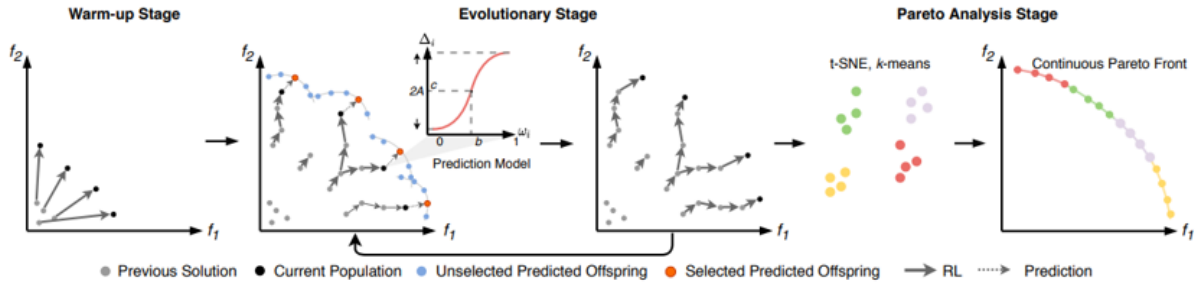


Figure 2.4: Overview of the PGMORL Algorithm for a 2-objective problem. (Xu et al., 2020)

2.4.4 Optimistic Linear Support (OLS)

OLS (Roijers, 2016) is an outer loop MORL approach that incrementally constructs the CCS of a multi-objective decision problem by iteratively solving a single-objective problem, with known preferences over the objectives. The key contribution of OLS is that the preference vector $\boldsymbol{w} \in \mathbb{R}^d$ that suggests to be used by the single-objective optimizer is selected so that the possible improvement in the value of the specified weight is maximized (Equation 2.20).

$$\boldsymbol{w} = argmax_{\boldsymbol{w} \in \mathbb{R}^d}[V^*_{\overline{CCS}}(\boldsymbol{w}) - V^*_{\mathcal{S}}(\boldsymbol{w})] \tag{2.20}$$

where $V^*_{CCS}(\boldsymbol{w})$ is the optimal scalarized value of the true CCS in point $\boldsymbol{w}$, and $V^*_{\mathcal{S}}(\boldsymbol{w})$ is the current value in point $\boldsymbol{w}$ using the currently obtained solution set $\mathcal{S}$. The preference

vectors that are suggested by OLS are called corner weights. OLS execution starts with an empty partial CCS $\mathcal{S}$. After every single-objective iteration, a new value vector $V^*_{\mathcal{S}}(\boldsymbol{w})$ is added to $\mathcal{S}$. During execution, OLS identifies new corner weights, and adds them in a queue, together with their expected improvement. It then selects the weight with the maximum expected improvement, based on Equation 2.20. The OLS process is described into more detail below.

Initialization OLS process begins by initializing the partial CCS $\mathcal{S}$ which will contain the value vectors that will be computed by the single-objective algorithm. It then initializes a set of visited weights $\mathcal{W}$ to keep the preferences that have been visited in previous iterations, and a priority queue $\mathcal{Q}$, to keep the preferences that need to be visited in later equations, together with their priority (i.e. the expected improvement of the value function). For the initialization of the priority queue, the extrema preferences simplex (the preferences for which the value of one objective is 1 and the values for the rest is 0) are added with an infinity priority, to ensure that these preferences will be explored first.

Corner weights After having evaluated the extrema preferences, the partial CCS $\mathcal{S}$ consists of d value vectors, one for each objective. For the next iterations, OLS needs to determine which preference vectors need to be added to the priority queue $\mathcal{Q}$ and with what priority. They will be the ones for which the scalarized values $V^*_{\mathcal{S}}(\boldsymbol{w})$ surface (Figure 2.5.b) changes slope. $V^*_{\mathcal{S}}(\boldsymbol{w})$ is a piecewise linear and convex (PWLC) function, defined as the maximum scalarized value for each w, for all value vectors in $\mathcal{S}$. As presented in Figure 2.5.b, every grey line corresponds to a point in $\mathcal{S}$. So, for point i with value vector $V_i \in \mathbb{R}^d$, the scalarization function is $V_i(w_i) = V_{i,1} * w_1 + V_{i,2} * w_2$, where $w_1 \in [0, 1]$ and $w_2 = 1 - w_1$.

The corner weights are incrementally recomputed every time a new value vector is added to the partial CCS $\mathcal{S}$. The computation process is seen in Algorithm 1.

First, the `newCornerWeights` algorithm computes the set of relevant value vectors $\mathcal{V}_{rel}$. This corresponds value vectors for which the preferences in W_{del} get their maximum scalarization value $V^*_{\mathcal{S}}(w)$. Preferences W_{del} are the preferences for which their maximum scalarization value is achieved with the new policy, $V^{\pi}_{new}(w)$ (Line 2). The boundaries refer to the extrema preferences. Then, for every $d-1$ subset of the relevant value vectors

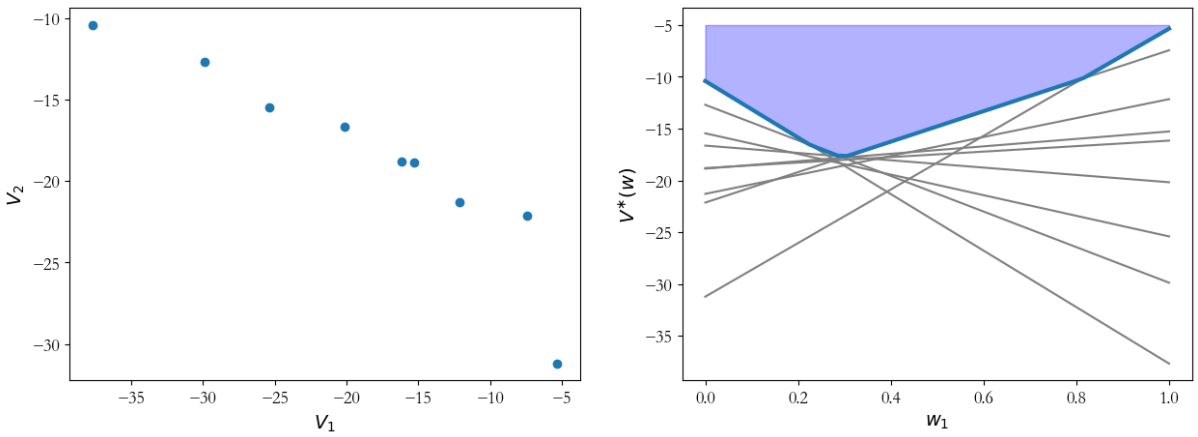


Figure 2.5: (a) The Partial CSS $\mathcal{S}$ of an arbitrary 2-objective problem, containing 9 points. (b) The scalarization function $V^*_{\mathcal{S}}(\boldsymbol{w})$.

Algorithm 1 `newCornerWeights`$(V^{\pi}_{new}, W_{del}, \mathcal{S})$

Require: A new value vector, V^{π}_{new}, a set of obsolete corner weights, w_{del}, and the current partial CCS, $\mathcal{S}$

1: $\mathcal{V}_{rel} \leftarrow \bigcup_{\boldsymbol{w} \in W_{del}} mathcalV_{\mathcal{S}(w)}$
2: $\mathcal{B}_{rel} \leftarrow$ the set of boundaries of the weight simplex involved in any $w \in W_{del}$
3: $W_{new} \leftarrow \emptyset$
4: **for** each subset X of $d-1$ elements from $\mathcal{V}_{rel} \cup \mathcal{B}_{rel}$ **do**
5: $\quad w_c \leftarrow$ compute weight where the vectors/boundaries in X to intersect with $\boldsymbol{V^{\pi}_{new}}$
6: $\quad$ **if** w_c is inside the weights complex **then**
7: $\quad\quad$ **if** $w_c \cdot V^{\pi}_{new} = V^*_{\mathcal{S}}(w_c)$ **then**
8: $\quad\quad\quad W_{new} \leftarrow W_{new} \cup w_c$
9: $\quad\quad$ **end if**
10: $\quad$ **end if**
11: **end for**
12: **return** W_{new}

and boundaries, the preferences of their intersection point are calculated (Line 5). Finally. all the intersection preferences (corner weights) are returned and added to the Priority queue $\mathcal{Q}$.

Prioritization Because for every corner weight added to the priority queue. a new execution of the single-objective problem should be run, it is important for the efficiency of OLS to ensure that the single objective executions run for the preferences with the highest potential impact first. For that reason, OLS prioritizes each corner weight with respect to its maximal possible improvement (Equation 2.20). The value $V^*_{\overline{CCS}}(w)$ is the optimistic value at point w, based on the partial CCS $\mathcal{S}$, which is depicted with dashed line in Figure 2.6.

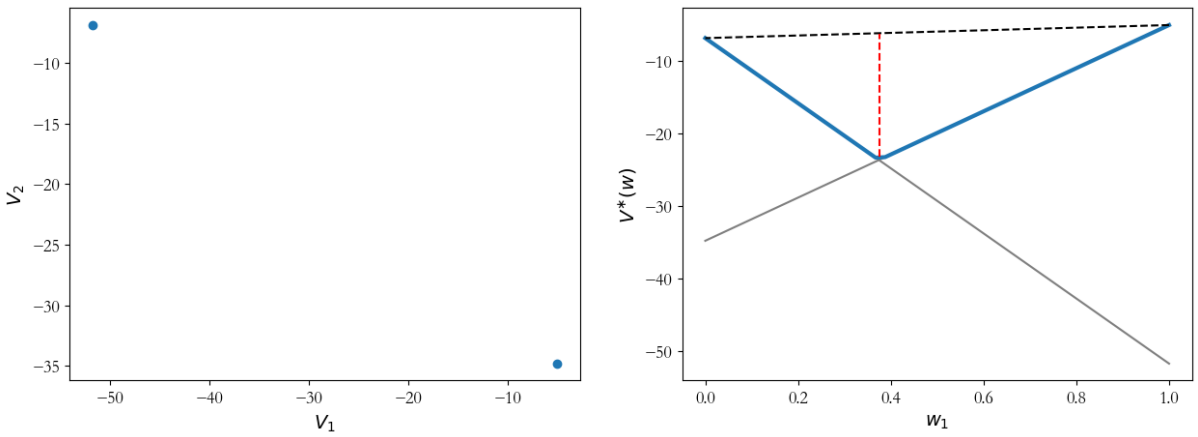


Figure 2.6: (a) The Partial CSS $\mathcal{S}$ of an arbitrary 2-objective problem after solving for the extrema preferences. (b) The scalarization function $V^*_{\mathcal{S}}(\boldsymbol{w})$. The $\overline{CCS}$ is shown with dotted black line. With the red dotted line, the possible improvement of solving corner weight $[0.375, 0.625]$ is shown.

The scalarization value of preference w in the $\overline{CCS}$ is computed by solving the Linear Programming system of Equation 2.21

$$\begin{aligned} &\max && w \cdot v \\ &\text{subject to} && \mathcal{W} \cdot v \leq V^*_{\mathcal{S},\mathcal{W}} \end{aligned} \tag{2.21}$$

where $V^*_{\mathcal{S},\mathcal{W}}$ is a vector containing the maximum scalarization values for all $w' \in \mathcal{W}$.

The final priority is computed as the relative possible improvement, given by Equation 2.22, and ranges from 0 to 1.

$$\Delta(w) = \frac{V^*_{\overline{CCS}}(\boldsymbol{w}) - V^*_{\mathcal{S}}(\boldsymbol{w})}{V^*_{\overline{CCS}}(\boldsymbol{w})} \tag{2.22}$$

It should be noted that for performance reasons, the algorithm accepts a threshold parameter ϵ, which specifies the minimum relative possible improvement for which a corner weight will be stored in the $\mathcal{Q}$ set, and therefore get solved with the single-objective optimizer. In that way, we avoid running single-objective executions for corner weights with a very small expected improvements.

Time & space complexity Based on Theorem 4 from Roijers (2016), the runtime of OLS is

$$O((|\epsilon - CCS| + |\mathcal{W}_{\epsilon-CCS}|)(R_{SO} + R_{PE} + R_{nw} + R_{heur})) \tag{2.23}$$

where:

- $|\epsilon - CCS|$ is the size of the CCS approximation with priority threshold ϵ.
- $|\mathcal{W}_{\epsilon-CCS}|$ is the number of corner weights of the scalarized value of the $\epsilon - CCS$.
- R_{SO} is the runtime of the single-objective optimizer.
- R_{PE} is the runtime of the policy evaluation, after computing the value vector for a specific preference vector.
- R_{nw} is the runtime of the `newCornerWeights` function (Algorithm 1).
- R_{heur} is the runtime of the prioritization value calculation (Equation 2.21).

The total number of iterations of the single-objective optimizer is $|\epsilon - CCS| + |\mathcal{W}_{\epsilon-CCS}|$, coming from the fact that after each single-objective iteration, the resulted value vector will either enter the partial CCS or it will not enter it because it is dominated by another preexisting value vector. The first case corresponds to the number of point in the approximate CCS, while the latter is at most the number of the corner weights of the CCS approximation.

Accordingly, the space complexity is given by Theorem 6 of Roijers (2016):

$$O(d|\epsilon - CCS| + d|\mathcal{W}_{\epsilon-CCS}| + M_{SO} + M_{PE}) \tag{2.24}$$

where:

- d is the number of objectives.
- M_{SO} is the memory requirement of the single-objective optimizer.
- M_{PE} is the memory requirement of the policy evaluation.

As OLS only keeps track of the CCS points and the visited corner weights, it is almost as memory efficient as the single-objective optimizer itself, giving it a huge advantage compared to inner-loop algorithms which need to keep multiple sets of value-vectors and partial policies (Roijers, 2016).

2.5 Predictive Maintenance Planning in road networks

2.5.1 Predictive Maintenance of road networks

Predictive maintenance refers to forecasting when and where it is best to perform each Maintenance or Inspection action, in order to optimize specific objectives. The current industry standard in predictive maintenance in road networks is condition-based maintenance (CBM) (Ayalew et al., 2022). CBM is a simple, yet effective approach to optimize road maintenance operations. It firstly assigns a condition threshold to each of maintenance actions. This threshold specifies the condition after which the specific maintenance action should be applied to the road segment. The assessment of the condition of the road segments is achieved by periodic inspections. As a result, effective maintenance scheduling of the road network can be achieved by defining the rights condition thresholds for maintenance actions, and time thresholds for inspection actions.

2.5.2 Multi-objective approaches

Predictive maintenance planning is a complex problem, involving multiple stakeholders, and a large potential negative impact in case of a disaster. Therefore, the fact that multiple efforts have been made to model it as a multi-objective problem, doesn't come as a surprise. Namely, most studies have been conducted in the field of Genetic Algorithms (GA), trying to approximate the Pareto front (Hamdi et al., 2017), (Yang et al., 2015) (Guan et al., 2022).

On the other side, there have been many attempts to tackle the Predictive Maintenance planning problem as a (PO)MDP with the help of DRL. However, all approaches so far have resulted in a single optimal policy, either because there was only one objective in the study, or because the utility function was linear and known before the execution. Notably, Andriotis and Papakonstantinou (2021) developed a framework for applying objectives in terms of constraints instead of a utility function, which was also applied in Saifullah et al. (2022). This was achieved through state augmentation and Lagrangian relaxation. An overview of the DRL studies, together with the multi-objective GA studies that were traced in the literature is presented in Table 2.2

Author(s)	Method	Objectives	Solution(s)
Hamdi et al. (2017)	GA	Maintenance cost, Road serviceability	Multiple
Yang et al. (2015)	GA	Maintenance cost, Road serviceability	Multiple
Guan et al. (2022)	GA	Maintenance cost, Carbon Emissions, Users cost	Multiple
Ramachandran et al. (2017)	Mixed Integer Linear Programming (MILP)	Road roughness, road deflection	Multiple
Chen and Wang (2023)	DRL	Maintenance cost	Single
Rabbanian et al. (2021)	DRL	Maintenance cost	Single
Andriotis and Papakonstantinou (2021)	DRL	Maintenance cost, Structural risk	Single(scalarized)
Yao et al. (2020)	DRL	Maintenance cost	Single
Wei et al. (2020)	DRL	Maintenance cost, Structural risk	Single(scalarized)
Han et al. (2021)	DRL	Maintenance cost, Structural risk	Single(scalarized)
Latifi et al. (2021)	DRL	Maintenance cost, Carbon emissions	Single (scalarized)
Saifullah et al. (2022)	DRL	Maintenance cost, Structural risk	Single (scalarized)

Table 2.2: Overview of works involving either multi-objective optimization or DRL in road maintenance planning.

2.6 Research Gap

As per Table 2.2, it becomes clear that all efforts utilizing DRL for Road maintenance planning result to single-policy solutions, meaning that they implicitly or explicitly infer the preferences over the different modelled objectives. As a result, there has been no study to model the Road Maintenance scheduling problem as a multi-objective problem with a multi-policy solution. To the best of our knowledge, the present study will be the first effort to model a road network with multiple objectives and apply MORL algorithm to approximate the CS instead of just one scalarized solution.

Moreover, most DRL studies have focused on finding an optimal planning with respect to the maintenance cost and the structural risk of road failure. The environmental aspect of road maintenance has largely been understudied, with only one study Latifi et al. (2021) including it. We believe that accounting for the environmental impact of road maintenance is a really important for sustainable operations, and hence we decided to include it in our study.

Finally, inspired by the study of Guan et al. (2022), we decided to include the "Users' Cost" objective to our multi-objective study. This is especially essential from a societal

standpoint, as roads are a public good should be accessible by anyone, regardless of their economical status. In addition, higher user costs due to bad road condition can result in higher fuel costs and in turn higher carbon emissions (Zaabar and Chatti, 2010).

As this is the first effort to model a road environment with Multi-objective DRL, we will use a linear utility function. As a result, based on the problem taxonomy of Table 2.2, the solution set that we will be looking for will be the Convex Coverage Set (CCS) of deterministic stationary policies.

3 Environment Description

The realistic modelling of the environment is a crucial component of any Reinforcement Learning analysis. In the context of road inspection and maintenance (I&M) optimization it refers to the physical and structural characteristics of the road network, including the topology of the network, the condition of the pavement, the volume of traffic. To develop an effective maintenance schedule for a road network, an in-depth understanding of the road environment and its effect on road deterioration is essential.

This chapter offers a comprehensive description of the road environment utilized in the present study. The chapter begins with a description of the physical road network characteristics, including its topology, the number of road segments, the traffic volume information, and the condition of the pavement. Next, the formulation of the environment as a Markov Decision Process is discussed. The chapter concludes with some preliminary evaluation of the environment.

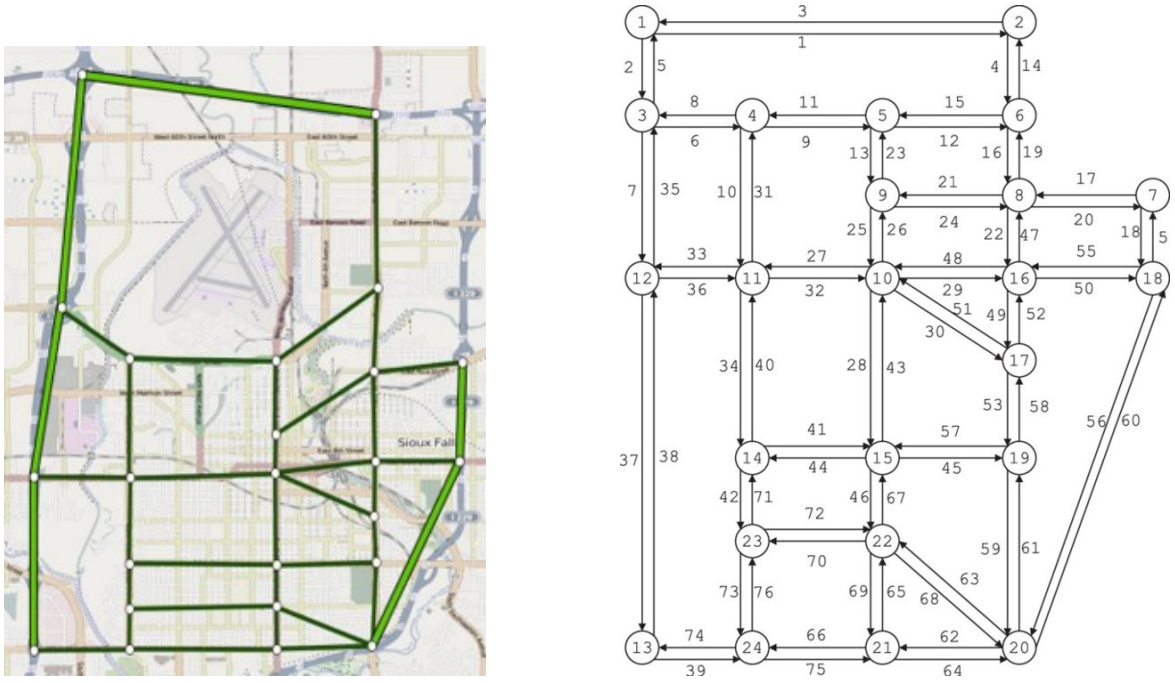
3.1 Overview

3.2 The physical problem

The environment for this work is based on a subset of the Sioux Falls road network (LeBlanc et al., 1975). It is one of the most widely studied networks for traffic assignment problems, as it has been used as a case study in multiple works (Chakirov and Fourie, 2014), (Wang et al., 2013).

The Sioux Falls network has a total length of almost 505 km, and it consists of 24 nodes and 76 links. The original network is presented in Figure 3.1.

Because our research involves multi-objective experiments, it was decided that only a subset of the Sioux falls network will be utilized in an effort to limit the required execution time. The selected subset consists of 10 road segments, and corresponds to the nodes $3, 4, 11, 12$ of the original network. The final subset can be seen in Figure 3.2



(a) The Sioux Falls network on the map. (Chakirov and Fourie, 2014)

(b) The Sioux Falls network as a bidirectional graph. (Long et al., 2015)

Figure 3.1: The Sioux Falls network.

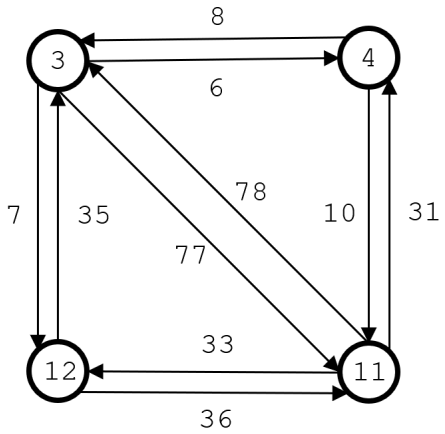


Figure 3.2: The Sioux Falls network subset that will be used for the multi-objective experiments.

It is worth noting that two extra road segments (77 and 78) have been added to the original Sioux Falls network. This addition was to encourage vehicle rerouting when a road segment is under maintenance, and therefore has limited capacity.

For every node, the traffic demand to each of the other nodes (average daily trips) is given in Table 3.1. For every link, its length, start and end node and its traffic capacity are given in Table 3.2.

Network node	3	4	11	12
3	0	2	3	2
4	2	0	15	6
11	3	15	0	14
12	2	6	14	0

Table 3.1: Matrix of trips between each node pair(thousands of vehicles/day). The matrix is symmetric.

Segment	Length (km)	Start node	End node	Nr of lanes	Capacity (vehicles/day)
6	6.44	3	4	2	855.53
7	6.44	3	12	2	585.09
8	6.44	4	3	2	427.76
10	9.66	4	11	2	122.72
31	9.66	11	4	2	245.44
33	9.66	11	12	2	245.44
35	6.44	12	3	2	1170.17
36	9.66	12	11	2	122.72
77	9.66	3	11	2	1295.01
78	9.66	11	3	2	1295.01

Table 3.2: Attributes for every link of the network

We assume that each of the 10 road segments deteriorates independently. Asphalt deterioration is measured using the International Roughness Index (IRI) (Gillespie et al., 1986) in m/km. The IRI is used as a benchmark by highway experts worldwide to assess road surface roughness. A continuous profile is measured and evaluated throughout the route to characterize the attributes of pavement surface irregularities that affect vehicle suspension movement. The IRI, measured in inches per mile or meters per kilometer, represents how much total vertical movement a conventional passenger vehicle's body would experience if driven at 50 mph over a 1-mile portion of the subject pavement. IRI is important for determining the overall ride quality of a pavement; a higher IRI value implies a rougher road surface.

In order to keep the road network in a good condition, the asset managers can decide upon performing a maintenance or/and an inspection actions in any of the road segments once a year. The project horizon is 20 years.

3.3 Belief Space

The belief space models the agent's uncertainty over the state of the environment. It represents all the possible probability distributions over states. For the present environment, we consider a 10-dimensional discrete belief space, where each dimension represents the belief over one of the road segments. The value ranges of the belief space are based on the International Roughness Index (IRI) and were retrieved from Saifullah et al. (2022). More specifically, five discrete conditions were identified, ranging from intact state to damaged state. The value ranges for each of the conditions are shown in Figure 3.3. The condition of each segment is independent of the rest.

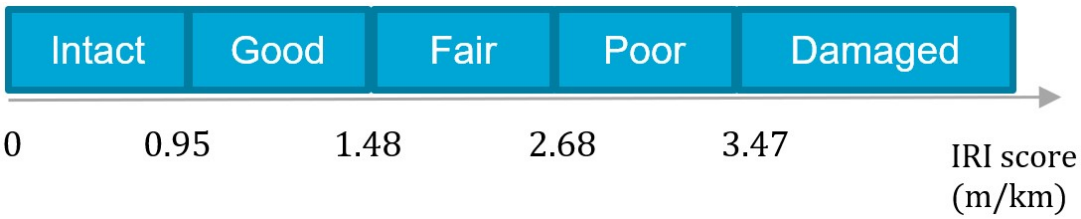


Figure 3.3: The discretization of IRI to 5 discrete states.

It is assumed that at the beginning of the maintenance life cycle, all road segments are in intact state. However, due to different traffic loads, and stochasticity, they may deteriorate in a different way. Hence, at every time step, every segment can have a different belief. The total size of the belief space is 5^{10}.

3.4 Action Space

The ultimate goal of efficient road maintenance is to select the appropriate maintenance and inspection actions such that the road segments remain in a adequate condition. Maintenance actions improve the condition of a road segment, but may come at a great cost. On the other hand, inspection actions are relatively cheap, and give an estimation of the condition of the road segment.

In this environment we assume that the project managers can select between 5 actions for each road segment:

- **Do Nothing**: No action is performed - Asphalt continues deteriorating
- **Repair**: Asphalt is partially restored, through crack filling and moderate patching.

This action improves the IRI score of the road segment, even though the uncertainty upon the true state is preserved

- **Inspect**: An inspection crew goes to the road to inspect its true state. This action reduces the uncertainty of the belief over the states.
- **Repair & Inspect**: A combination of the two actions above. IRI score improves, and the uncertainty of the belief reduces.
- **Replace**: The asphalt of the road segment is fully replaced. After applying this action, the road segment is in intact state, with a full certainty.

Since the road segments deteriorate individually, different actions can be selected each of them. As a result, the action space has a size of 5^{10}.

Every action comes with its corresponding negative rewards. These are the monetary cost and the carbon emissions of the repair, and are relative to the length of the segment. Moreover, if a maintenance action (repair or replace) is performed in a road segment, this road segment will have its capacity reduced by 50% while the action is performed. We assume that maintenance actions start at the beginning of the year an have a certain duration, while inspection actions are performed at the end of the year, with zero traffic disruption.

Finally, if a road segment is found to be in damaged condition, then an urgent replacement action must be performed. This costs 50% more than the original replacement action, due to its urgency.

Table 3.3 shows an overview of the monetary cost, carbon emissions and traffic disruption of every action.

Actions	Duration (days)	Monetary Cost ($/m2)	Carbon Emissions (g/m2)	Traffic capacity reduction (%)
Do Nothing	0	0	0	0%
Repair	168	52	6000	50%
Inspect	0	0.2	0	0%
Repair + Inspect	168	52.2	6000	50%
Replace	365	250	22500	50%

Table 3.3: Overview of the negative rewards after every action, together with its effect to traffic.

3.5 Transition Probabilities

Transition probabilities describe the likelihood of transitioning from one state to another in the environment, given an action taken by the agent. In the context of this thesis project, transition probabilities will model the probability of moving from one IRI state to another. We will consider the stationary model for IRI from (Saifullah et al., 2022). The transition probabilities matrix has a size of (A, S, S), where A is the number of actions, while S is the number of beliefs. The full transition probabilities and are presented in Appendix A1.

3.6 Observation Probabilities

To decrease the uncertainty of the belief, the asset managers can decide to perform a visual inspection, which will help them infer the true condition of a road segments. However, this is not a perfect observation, as it is possible that a wrong state will be infered. Observation probabilities are a core concept in Belief MDPs as they describe the likelihood of observing a particular outcome given a state of the environment and an action taken by the agent. The observation probabilities will be in the form of Equation 3.1, where i is the true state of the road segment and j is the observed one. We assume that inspectors will infer the correct state of the road segment with a probability of $p = 0.8$. The final observation matrix has a size of (S, S), where S is the number of beliefs.

$$O(o_{t+1} = j | s_{t+1} = i, a_t) = \begin{bmatrix} p & 1-p & & & \\ \frac{1-p}{2} & p & \frac{1-p}{2} & & \\ & \frac{1-p}{2} & p & \frac{1-p}{2} & \\ & & \frac{1-p}{2} & p & \frac{1-p}{2} \\ & & & 1-p & p \end{bmatrix} \tag{3.1}$$

3.7 Reward function

The reward function for the environment will be a 3-dimensional vector, where its dimension represents a different objective.

3.7.1 Maintenance cost objective

The first objective of the road network environment is related to the maintenance cost minimization. It includes all the monetary costs related to the maintenance and inspection of the road network. These are:

- Cost of maintenance actions: Cost related to performing a maintenance action to a road segment, as described in Table 3.3
- Cost of inspection actions: Cost related to performing an inspection action to a road segment
- Cost of mobilization of the equipment: Fixed cost to be applied if an action has been picked at least once in the network for the current time step. It relates to the mobilization of the maintenance equipment and crew, and it serves as a motive for grouping same actions in the same time step.
- Cost of urgent actions: When a road segment has reached a damaged condition, then an urgent replacement of the asphalt is mandatory. This costs 50% more than the original replacement action

The reward function for the maintenance cost is seen in 3.2, where δ_i is 0 if the road segment is not in a damaged condition, and is 1 otherwise. $c_i(a_t)$ is the cost of performing action a to component i. Finally $c_i(replacement)$ is the cost of replacing the asphalt in road segment i, and c_{mob} is the mobilization cost for all actions performed in time step t.

$$r_{t,1}(a_t) = -\sum_{i}^{10} [(1-\delta_i)c_i(a_t) + \delta_i 1.50 c_i(replacement)] - c_{mob} \tag{3.2}$$

3.7.2 Carbon emissions reward

The second objective of the road network environment is the minimization of the carbon emissions. This includes:

- The emissions from the maintenance actions, as described in Table 3.3
- The added vehicle emissions due to traffic disruption from Inspection and Maintenance (IM) actions. When a maintenance work is carried out in a road

segment, the capacity of this segment is reduced. This may force vehicles to select other routes to their destination which will longer than the initial one. The increased travel distances will result in more carbon emissions per travel.

- The extra vehicle emissions due to high road roughness. It has been measured that high road roughness levels lead to increased vehicle fuel consumption, and carbon emissions.

The reward function for the carbon emissions is shown in Equation 3.3, where l_i and $flow_i(a_t)$ are the length and traffic flow in road segment i, $extraConsumption_i(s_t)$ is the (%) of extra carbon emissions due to the road segment condition, and avg_{co2e} is the weighted average carbon emissions per km of road, calculated from Table 3.4. $e_i(a_t)$ is the carbon emissions in road segment i from performing action a_t. Finally, E_{init} corresponds to the total network carbon emissions when no maintenance or inspection action is performed.

$$r_{t,2}(a_t, s_t) = -(\sum_{i}^{10} l_i * flow_i(a_t) * (1 + extraConsumption_i(s_t)) * avg_{co2e} + e_i(a_t) - E_{init}) \tag{3.3}$$

Vehicle Type	**% in traffic**	**Average $CO_2e(g/km)$**	**Weighted $CO_2e(g/km)$**
Medium Car	78	143.16	111.66
Van	13	246.2	32.01
Truck	5	649.73	32.49
Motorcycle	3	116.80	3.50
Bus	1	1294.13	12.94
Total	100		avg_{co2e}=**192.60**

Table 3.4: The average carbon emissions per km of road is calculated as a weighted sum of the carbon emissions from the different vehicle types in the road network. The traffic composition in the current table have been retrieved from the Transport and Mobility Report (Statistics Netherlands, 2016), who suggest an average traffic composition for the dutch road. Moreover, the information about the average carbon emissions per vehicle type were retrieved from environmental reports from the UK Government (Department of Transport, 2021), (Department of Business Energy and Industrial Strategy, 2020).

At the same time, the asphalt condition plays an important role in the carbon emissions of the travelling vehicles. According to Zaabar and Chatti (2010), a medium car travelling in a damaged road segment (IRI $> 3.47m/km$) consumes almost 2.5% more fuel, and hence emits proportionally more carbon in the environment. The model for the five main

vehicle types can be seen in Figure 3.4

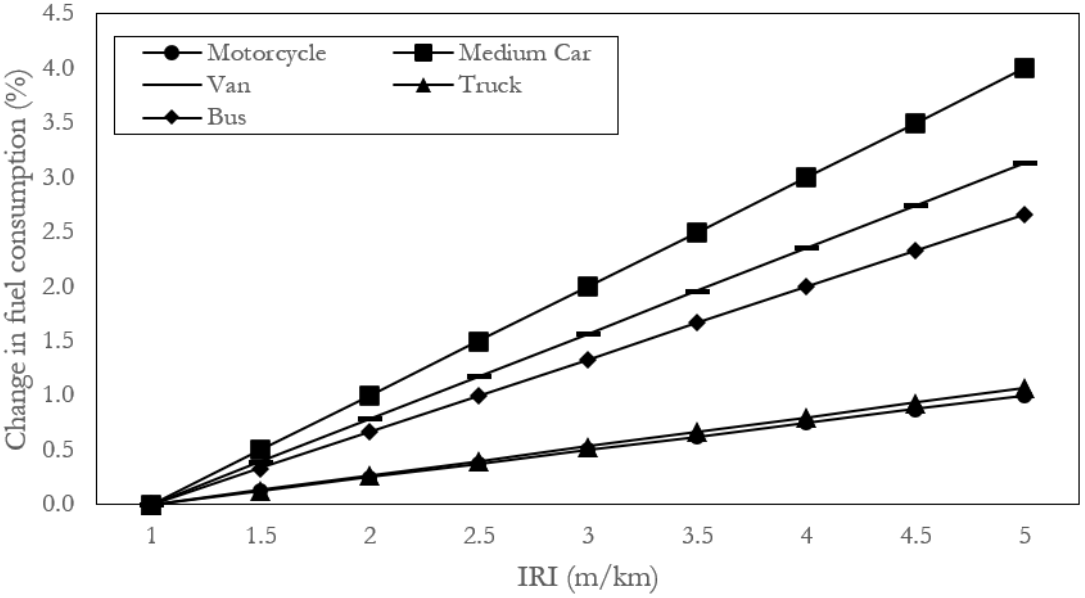


Figure 3.4: Percentage change in fuel consumption with IRI progression. (Zaabar and Chatti, 2010)

Finally, the values for the carbon emissions from maintenance actions were retrieved from Jiang et al. (2020) who performed a life cycle assessment for the use, maintenance and rehabilitation phases of a road network, focusing on a real-life network in Australia. For our analyses, we assume that the "Repair" action corresponds to the Slurry maintenance program, while the replacement action corresponds to the ASOG program. The respective descriptions of each program are presented in Table 3.5.

Action	Maintenance Program	Description
Repair	Slurry	Cold mixed surface treatments, including application of 3–20 mm in-situ mixture of aggregate, cement/lime, polymer modified bitumen emulsion, adhesive, and water
Replacement	ASOG	Asphalt replacement (Asphalt mixing plant, paver and compactor) (30 mm)

Table 3.5: Mapping of the case study actions with maintenance programs, and their descriptions. (Wu and Wang, 2016)

3.7.3 Users cost objective

The third objective is the minimization of the users' cost. This corresponds to all the costs incurred during the operation and maintenance of the road vehicles, for which the vehicle owners are responsible. For instance it includes the fuel cost, lubricants and maintenance material and vehicle degradation. Vehicle owners spend more on the maintenance and operation of their vehicles when the road roughness is high. In the The Highway Design

and Maintenance Standard Model (HDM4) (World Bank Group, 2018), the relation between the user costs and IRI is fitted with a polynomial model. The resulting curves are presented in Figure 3.5

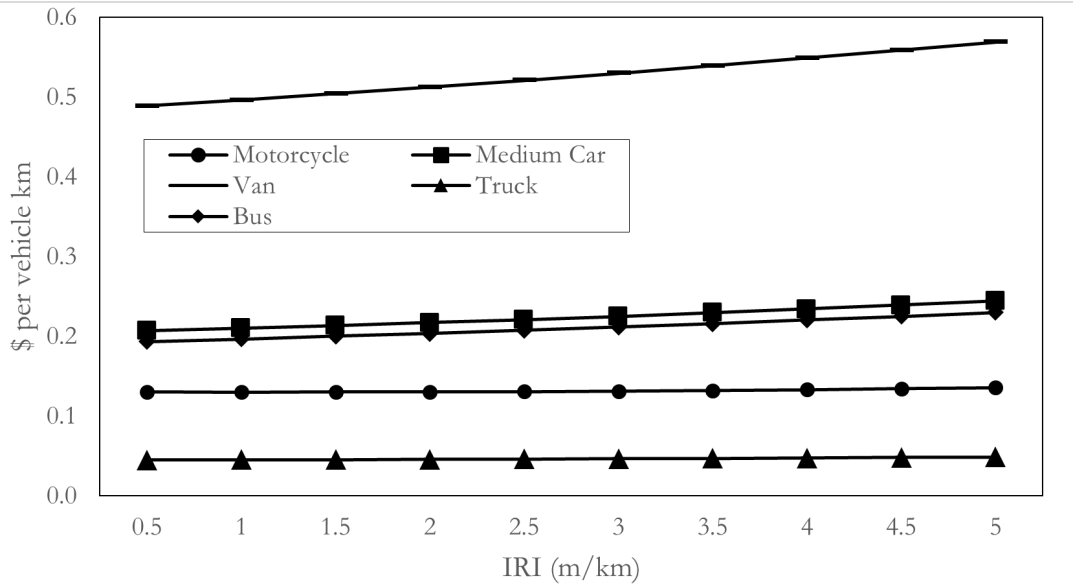


Figure 3.5: User costs per km of road with IRI progression. (Zaabar and Chatti, 2010)

The reward function for the user cost is shown in Equation 3.4, where l_i and $flow_i(a_t)$ are the length and traffic flow in road segment i, while $userCost_i(s_t)$ is the user cost per km of road for segment i.

$$r_{t,3}(s_t, a_t) = -\sum_{i}^{10} l_i * flow_i(a_t) * userCost_i(s_t) \tag{3.4}$$

3.8 The traffic assignment problem

The Traffic Assignment Problem (TAP) is a transportation engineering mathematical optimization problem that tries to arrange cars throughout the network in order to reduce overall trip time or cost for all users while respecting network restrictions. It entails determining the best distribution of demand across available routes in a transportation network while accounting for characteristics such as road capacity, congestion, and trip time. TAP is a valuable resource for transportation planners and engineers in the design and management of efficient and sustainable transportation networks.

In our study, we use the traffic assignment calculation to estimate the traffic disruption

that will be caused by maintenance actions. Since our goal is to study the traffic disruption in yearly time intervals, we can rightfully assume that our traffic assignment model is static, meaning that the traffic flow from a network node to another is fixed.

When a road segment is closed for maintenance, its traffic capacity decreases by 50%. This will result in extended travel times and different traffic rearrangement. We want to know which are the updated flows (number of vehicles travelling across each road segment) and total travel time in the network, in order to assess the efficiency of a certain IM plan. If the plan includes many simultaneous road maintenance actions, it will cause major traffic congestion, negatively impacting the "Carbon Emissions" objective, due to rerouting. Conversely, if no maintenance actions are planned, many segments will reach damaged condition, and this will negatively impact the "Users Cost" objective.

In order to solve the static traffic assignment problem, we will use the User Equilibrium model (Wardrop and Whitehead, 1952), which is based on on Wardrop's first principle, stating that "*no driver can unilaterally reduce his/her travel costs by shifting to another route*". This means that the decision of a driver to change their route will influence the decisions of other drivers with shared routes.

The mathematical formulation of the static traffic assignment problem with user equilibrium is as follows:

$$\begin{aligned} MinimizeZ &= \sum_{\alpha} \int_{0}^{x_\alpha} t_\alpha(x_\alpha) dx \\ s.t. \sum_{k} f_k^{rs} &= q_{rs} \forall r, s \\ x_\alpha &= \sum_{r} \sum_{s} \sum_{k} \delta_{\alpha,k}^{rs} f_k^{rs} : \forall \alpha \\ f_k^{rs} &\geq 0 : \forall r, s, k \\ x_\alpha &\geq 0 : \forall \alpha \in A \end{aligned} \tag{3.5}$$

where:

- k is a path connecting nodes $r - s$
- x_α are the equilibrium flows in road segment α
- t_α is the travel time in the road segment α given by the selected cost function

- f_k^{rs} is the flow on path k connecting nodes $r - s$
- $\delta_{\alpha,k}^{rs} = \begin{matrix} 1 & if\ road\ network\ belows\ to\ path\ k \\ 0 & otherwise \end{matrix}$ is a definitional constraint

The selected cost function to minimize is the BPR (Bureau of Public Roads) function (U.S. Department of Commerce, 1964). It is a mathematical model used in transportation planning and engineering to predict travel time and delay on a road network. It is frequently used to predict the impacts of traffic congestion on travel time and expenses. The BPR cost function considers free-flow travel time, road capacity, and traffic volume, and forecasts travel time and delay as a function of these factors. The BPR function is given in Equation 3.6, where fft_α is the free-flow travel time, $capacity_\alpha$ is the road capacity and x_α is the actual traffic flow in road segment α. a and b are model coefficients with values of 4 and 0.15 respectively.

$$t_{x_\alpha} = fft_\alpha * [1 + a(\frac{x_\alpha}{capacity_\alpha})^b] \tag{3.6}$$

The traffic determination through the traffic assignment problem will take the following inputs:

- Road network graph (nodes, road segments)
- Information about road segments (capacity, length, free flow speed)
- Origin-Destination (OD_{ij}) matrices, denoting the traffic demand from node i to node j

The outputs of the optimization will be:

- The total travel time in the network
- The traffic flow in each road segment

An overview of the road network with the calculated flows-to-capacity ratios per road segment is presented in Figure 3.6.

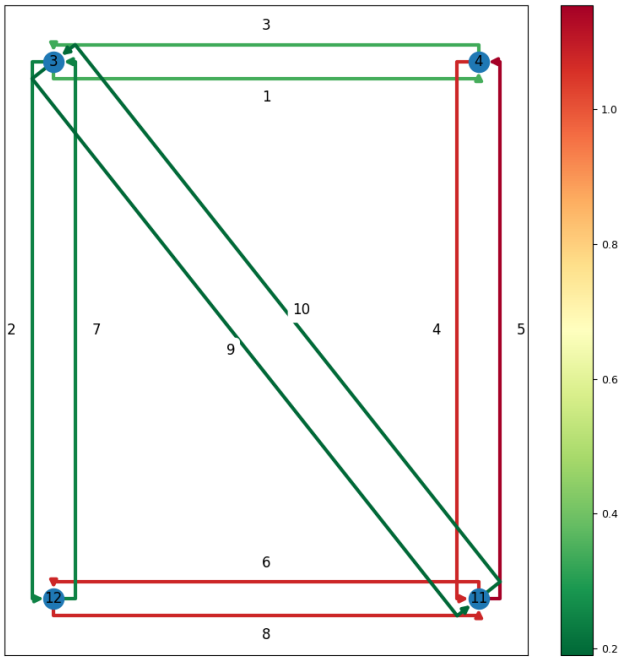


Figure 3.6: Overview of traffic flow to capacity ratio after the traffic assignment calculation, when no maintenance action is performed.

3.9 Timing performance

Time performance of the environment is an important concern in Deep Reinforcement learning, since a large number of environment executions is usually needed in order to converge to optimality. In Figure 3.7 we can see that the time performance of one episode takes on average 0.022 seconds for random policies, when executed on a computer with an Intel Core i-7 processor running at 2.30 GHz using 16 GB of RAM. It is deemed that the time performance is adequate in order to move to the multi-objective executions.

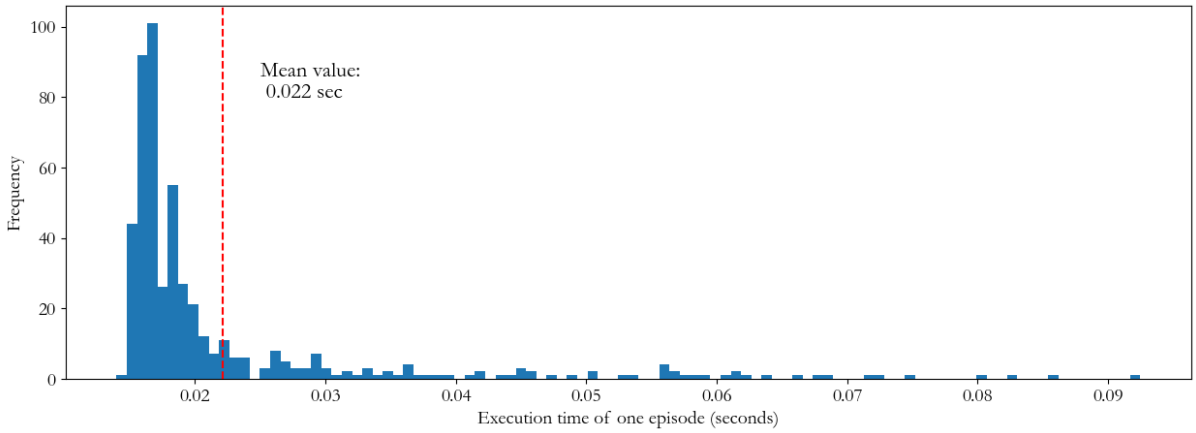


Figure 3.7: Histogram with the time performance of 500 episodes using a random policy.

4 Methodology

In the realm of MORL, various techniques have emerged, as described in the Background section. However, when it comes to addressing the Road maintenance scheduling problem, certain limitations impede the exploration of multiple approaches. The huge size of the state/action space naturally narrows our focus to policy-gradient-based MORL methods. Moreover, as our ultimate aim is to create a practical tool for future researchers in the predictive maintenance of roads, we have chosen to concentrate on outer-loop approaches, which are the most intuitive and straightforward techniques proposed in the field of MORL.

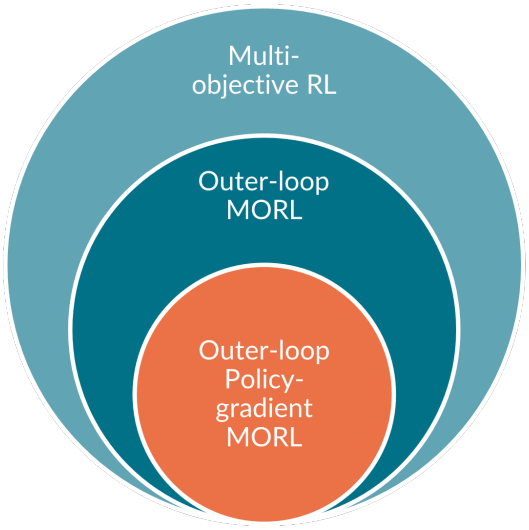


Figure 4.1: A stacked Venn diagram showcasing the category of MORL approaches, on which the present research is focusing.

This section presents the algorithms investigated in our research, as well as the overall framework and experimental setup employed to conduct our study. By delving into these details, we lay the foundation for our work and establish the groundwork for future advancements in road maintenance.

4.1 General Framework

The general framework of the present study corresponds to the execution of an outer-loop MORL algorithm, and is comprised of two main components, presented in Figure 4.2. The first is the selection of preferences for the objectives, while the second one is is the single-objective execution and optimization of the problem with the previously specified

preferences. Iterating over these two components gradually constructs the CS of the problem.

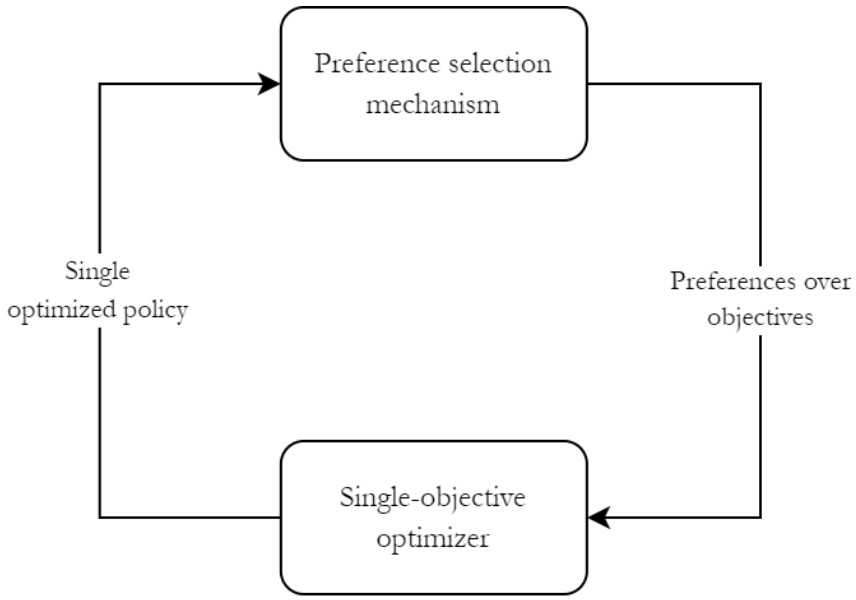


Figure 4.2: The general framework of the study consists of an iterative scheme between a preferences selection mechanism and the single objective optimizer.

In the following sections the specific algorithms that will be used are described. It is worth noticing that two of the approaches that will be discussed (Radial algorithm and Pareto Following algorithm) don't directly proposing a preferences vector for every single-objective execution. Instead, they modify the inner policy gradient of every training update, using a convex combination of the policy gradients over the different objectives. This is essentially an indirect way of proposing a preferences trade-off, and that is why they are still considered as outer-loop MORL approaches.

4.2 Single-objective optimizer

The basis for the MORL experiments is a modified version of PPO. The modifications in the algorithm compared to the vanilla PPO implementation (Schulman et al., 2017) are described below.

The first modification regarding the Actor network output is related to the nature of the road network environment. The second update, regarding the critic network output was performed to expand PPO functionality to handle multiple objectives.

4.2.1 Actor network output

In the traditional PPO implementation, the output of the Actor network is a probability distribution over all potential actions, for the given belief vector b_t. In the modified implementation, we have assumed conditional independence in the deterioration of every road segment, and hence the the policy of each road segment is considered independent as well. In other words:

$$\pi(b_t|a_t) = \prod_{i=1}^{n} \pi_i(a_t^{(i)}|b_t) \tag{4.1}$$

where n is the total number of road segments. In our problem $n = 10$.

Because of the conditional independence assumption, we can compute the policy for each road segment independently. This greatly reduces the size of the output layer of the Actor network to $n \times A << n^A$, which is the permutation of all action combinations. Then, a Softmax activation function is applied for each pair of A outputs, to compute the policy in each road segment as a probability distribution over the A actions.

The above notion of independent policies between the different components of an assets network was firstly introduced in the context of the Deep Centralized Multi-agent Actor Critic (DCMAC) (Andriotis and Papakonstantinou, 2019), an architecture that utilized the independence assumption to reduce the number of action combinations, and produce separate policies for each road segment.

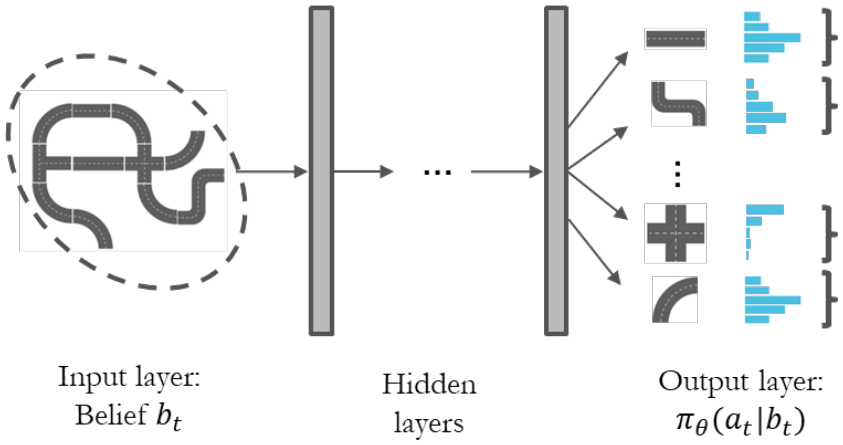


Figure 4.3: Overview of actor architecture for a multi-segment system. The output of the network is a set of n probability distributions, each one describing the policy for a road segment.

4.2.2 Critic network output

In the PPO standard implementation (Schulman et al., 2017), the critic network output is a single value, indicating the expected value of the specified belief (i.e. what is the best possible return that the agent can get from the current state belief). In the modified PPO version of this study, the critic network output is a vector of size d, where d is the number of objectives. This update is required in order to keep all values of the different objectives individually, and use the utility function and current preferences to compute the global value. This implementation was retrieved from Felten and Alegre (2022).

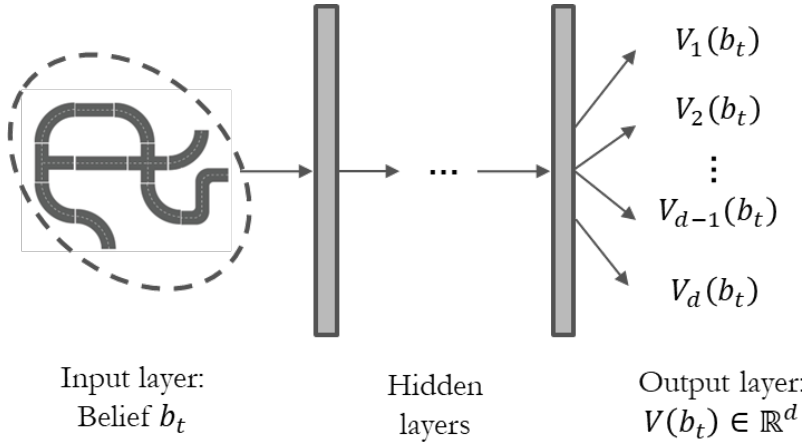


Figure 4.4: Overview of critic architecture for a multi-objective system . The output of the network is a vector of d values, each one corresponding to a different objective.

4.2.3 Modified-PPO training process

The execution process of the Modified-PPO algorithm that will be used is presented in Figure 4.5

For each single-objective execution (an execution of modified PPO with specifiec preferences w), a predefined number of episodes will be executed. Every episode starts with spawning 8 sub-processes, each executing the road environment once. After all subprocesses are complete, we merge the 8 trajectories that have been collected to a batch trajectory, and train the actor and critic network using this. Training is performed for $train\_iters$ training iterations per episode. It is worth noting that there is early stopping mechanism for the training, when the policy before training and the one after training are very different, based on their KL divergence. The full modified PPO parameters with their values are presented in Appendix A2.

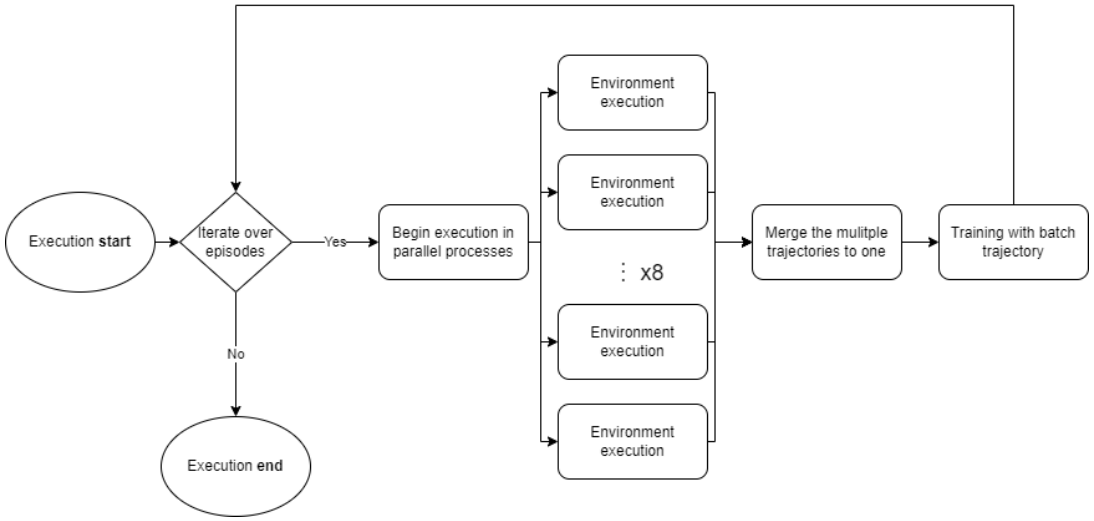


Figure 4.5: Overview of the modified-PPO execution.

4.3 Multi-objective algorithms

In this section, the outer-loop MORL algorithms that have been examined in the context of this work are presented.

4.3.1 Deep Optimistic Linear Support (DOL)

4.3.1.1 Overview

Deep Optimistic Linear Support Learning (DOL) (Mossalam et al., 2016) is a MODRL algorithm that can solve high-dimensional multi-objective decision problems, when the objective preferences are unknown at the time of execution. DOL is a general multi-objective decision framework consisting of two components, analogous to the one in Figure 4.2; the Preference selection mechanism, which in this case is OLS (Roijers, 2016) (illustrated in the Background section), and a single-objective learning algorithm that is OLS-compliant. The only update that is needed to make a learning algorithm OLS-compliant is for it to output vector-valued Q-value $Q(s_t, a_t)$ at the size of the number of objectives. This is a condition that our adjusted PPO algorithm described in section 4.2 already satisfied, hence it can be used in the DOL framework.

4.3.1.2 Reuse of model parameters

After experimenting with OLS, it becomes clear that the fact that information from the different single-objective executions can be exploited to increase the result and/or runtime performance of future executions. For instance, it is reasonable to think that the policy for a 2-objective problem with preferences [0.72, 0.28] is not very different to the policy of preferences [0.68, 0.32]. For this reason, Mossalam et al. (2016) experimented on reusing the model parameters from previous single-objective runs to enhance the performance of the next ones.

Firstly, they proposed DOL with full reuse (DOL-FR), a DOL variation that initializes the single-objective model parameters for a scalarization preference w using the parameters of a previously solved model with scalarization preference w', where w' is the visited preference that is closest to the current preference w.

Moreover, they proposed DOL with partial reuse (DOL-PR). DOL-PR initializes the single-objective model parameters in the same way as DOL-FR, but reinitializes the last layer of the model randomly, in order to escape from local optima.

As a result, Mossalam et al. (2016) proposed three different OLS-based approaches; DOL, with no parameters reuse, DOL-FR with full parameters reuse, and DOR-PR with partial parameters reuse. However, based on experimental evaluation, the authors concluded that partial reuse is more effective than full reuse, as it prevents the model from getting stuck in a policy that was previously optimal. Therefore, for our evaluation we decided to explore DOL and DOL-PR methods.

4.3.1.3 The DOL Algorithm

The pseudocode with DOL algorithm is presented in Algorithm 2.

Algorithm 2 Deep OLS Learning (with different types of reuse)

Require: The number of objectives m, the OLS improvement threshold ϵ, and OLS-compliant DRL architecture, *template*, the type of reuse, *reuse* taking values "none", "partial" and "full".

1: $\mathcal{S} \leftarrow \emptyset$ # empty partial CCS
2: $W \leftarrow \emptyset$ # empty list of explored corner weights
3: $\mathcal{Q} \leftarrow \emptyset$ # Priority queue initializes with the extrema preferences simplex, having infinite priority
4: $DRL\_models \leftarrow \emptyset$ # Empty set to keep the single-objective models executed.
5: **while** $\mathcal{Q}$ is not empty $\cap$ $iteration \leq max\_it$ **do**
6: $w \leftarrow \mathcal{Q}.pop()$
7: **if** $reuse =' none' \cup DRL\_models = \emptyset$ **then**
8: model $\leftarrow$ a random initialization of the *template* model
9: **else**
10: $model \leftarrow$ `copyNearestModel(w, DRL_models)`
11: **if** $reuse =' partial'$ **then**
12: Reinitialize the last layer of model with random weights
13: **end if**
14: **end if**
15: $V, new\_model =$ `scalarizedDRL(m,w,model)` # Single-objective solver execution
16: $W \leftarrow W \cup w$
17: **if** $(\exists w')$ $w' \cdot V > max_{U \in \mathcal{S}} w' \cdot U$ **then**
18: $W_{del} \leftarrow W_{del} \cup$ corner weights made obsolete by V from $\mathcal{Q}$
19: $W_{del} \leftarrow W_{del} \cup \{w\}$
20: Remove W_{del} from $\mathcal{Q}$
21: Remove vectors from $\mathcal{S}$ that are no longer optimal for any w after adding V
22: $W_V \leftarrow$ `newCornerWeights(`$\mathcal{S}$`, V)` (Algorithm 1)
23: $\mathcal{S} \leftarrow \mathcal{S} \cup \{V\}$
24: $DRL\_models[w] = new\_model$
25: **for** each $w' \in W_V$ **do**
26: **if** `estimateImprovement(w',W,`$\mathcal{S}$`)` $> \epsilon$ **then** (Equation 2.21)
27: $\mathbf{Q}.add(w')$
28: **end if**
29: **end for**
30: **end if**
31: $iteration++$
32: **end while**
33: **return** $\mathcal{S}, DRL\_models$

It can be easily observed that the largest share of the algorithm has to do with OLS execution. Again, the algorithm starts by initializing the partial CCS $\mathcal{S}$ (Line 1), the list of visited corner weights $\mathcal{W}$ (Line 2) and the priority queue $\mathcal{Q}$ containing the extrema preferences (Line 3). Additionally, DOL initializes a list to keep all the executed single-objective DRL models (Line 4).

DOL execution continues with solving the extrema preferences with the selected DRL model. The initialization of the single-objective DRL model depends on the reuse mode (Lines 7-14). The subroutine `copyNearestModel(w, DRL_models)` (Line 10) finds the model that is closest to the one that is about to be executed, by comparing the preference of the previously solved models and the current one. The subroutine `scalarizedDRL(m,w,model)` (Line 15) refers to the execution of a single-objective, OLS-compliant DRL algorithm using the preferences w. After solving the single-objective problem, the iteration continues exactly in the same way as OLS.

The obsolete corner weights are removed from the priority queue (Line 20), while value vectors that are dominated are also removed from the partial CCS (Line 21). Afterwards, the subroutine `newCornerWeights(S, V)` computes the new corner weights in the scalarization function $V^*_{\mathcal{S}}(w)$ and described in Paragraph 2.4.4. Finally, subroutine $estimateImprovement(w', W, \mathcal{S})$ computes the estimated improvement in the scalarization value $V^*_{\mathcal{S}}(w)$ for the current weight w, by solving the Equation 2.21. This subroutine is descibed into more detail in Paragraph 2.4.4. The execution continues by selecting the preference vector with the highest priority from the priority queue. Execution stops when there are no more preference vectors to explore in the priority queue.

4.3.1.4 DOL experiments parameters

In Table 4.1 the parameters that were used for DOL experiments are defined.

Parameter	Description	Value
m	Number of objectives	3
ϵ	Priority threshold	0.05
reuse	Reuse mode, determines the DRL models initialization	"none"
template	Single-objective DRL model	Modified PPO (Section 4.2)

Table 4.1: Parameters used for DOL experiments.

Respectively, Table 4.2 contains the parameters that were used for the DOL-PR experiments.

Parameter	Description	Value
m	Number of objectives	3
ϵ	Priority threshold	0.05
reuse	Reuse mode, determines the DRL models initialization	"partial"
template	Single-objective DRL model	Modified PPO (Section 4.2)

Table 4.2: Parameters used for DOL-PR experiments.

4.3.2 Radial Algorithm (RA)

4.3.2.1 Overview

Radial Algorithm (RA) (Parisi et al., 2014) is an outer-loop MORL algorithm that performs a series of single-objective policy iterations to incrementally construct a uniformly spaced CCS. It starts by defining p the number of points that are required to approximate CCS, and then performs single-objective optimization for each point, following a different ascent direction within the ascent simplex. The ascent simplex is defined by the convex combination of the single-objective gradients.

4.3.2.2 Ascent Directions

RA extends the concept of policy gradients defined in 2.11, so that the policy gradient is a vector $\nabla_\theta J_\theta \in \mathbb{R}^d$ of gradients, where d is the number of objectives.

The ascent simplex is the convex combination of the gradient vectors:

$$S(w,\theta) = \sum_{i=1}^{d} w_i \nabla_\theta J_i(\theta) \quad subject\ to\ \sum_{i=1}^{d} w_i = 1 \forall i, w_i \geq 0 \tag{4.2}$$

Intuitively, we know that any direction (i.e. combination of values λ_i) will allow us to approach the Pareto Frontier. However, only a subset of them allows us to improve all objectives at the same time.

In order to better understand the concept of objectives improvement, we will first focus on a single-objective problem with two parameters. When we try to optimize for our toy problem, a line that is perpendicular to the gradient divides the parameter space into two sub-spaces, the dominated area, which is opposite to the direction of the gradient, and the non-dominated area, which is at the direction of the gradient. In principle, any

point within the non-dominated sub-space can improve the objective. In the same way, in multi-objective problems, the parameter space is divided into, at most 2^d mutually exclusive directional cones. One of the cones simultaneously increases all the objectives, and is called the Ascent cone. Another cone simultaneously decreases all the objectives and is called the descent cone, while all the rest decrease at least one of the objectives. Ascent directions $l = [l_1, l_2, ..., l_d]^T \in \mathbb{R}^d$ are the directions within the Ascent cone, and increase all objectives at the same time, i.e. have a positive gradient for all objectives (Equation 4.3).

$$l \cdot \nabla_\theta J_i(\theta) \geq 0 \quad \forall i = 1, ..., d \tag{4.3}$$

When the solution of the multi-objective problem is far from the CCS, then the Ascent simplex lies within the Ascent cone (Point θ_A in Figure 4.6). However, when the solution approaches the CCS, gradients of the different objectives become increasingly conflicting, and the width of the Ascent cone decreases (Point θ_B in Figure 4.6). In this case, the Ascent simplex contains directions that are outside the Ascent cone.

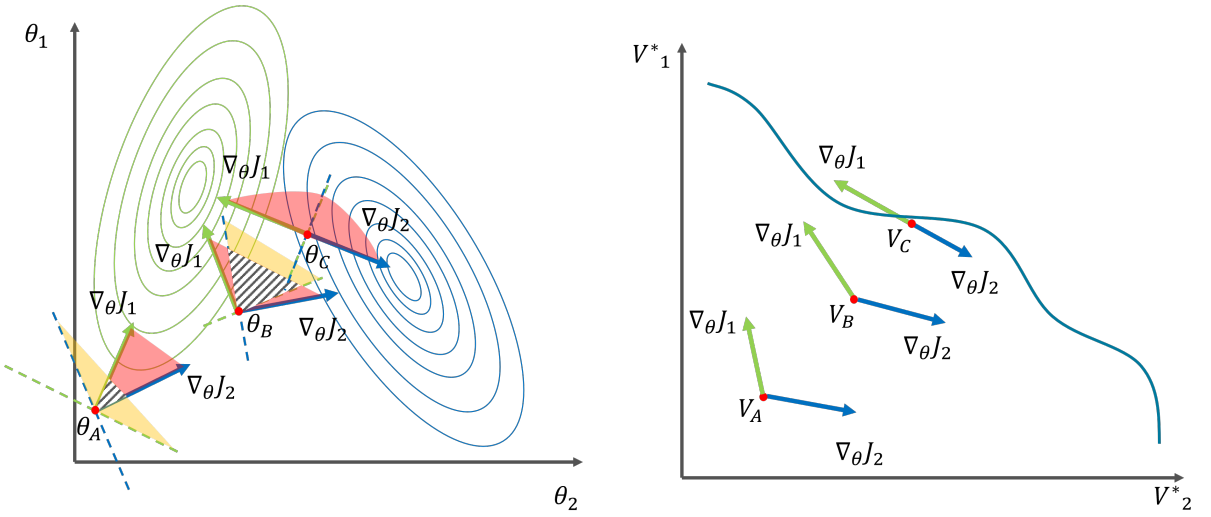


Figure 4.6: (a): Illustration of the ascent simplex (in red), ascent cone (in yellow) and Pareto ascent cone (in gradient fill), for a two-parameter, two-objective problem, for three different parameter vectors θ_A, θ_B and θ_C. The green contour lines correspond to the values of the first objective J_1, while the blue lines correspond to objective J_2. The dashed lines for each objective are perpendicular to the objective gradient, and divide the parameter space to a dominated and non-dominated subspace. (b): The respective value vectors of the models with parameters θ_A, θ_B and θ_C together with their relative placement with respect to the Pareto front. The closest a point is to the Pareto front, the more conflicting its gradients will be.

4.3.2.3 Stopping condition

For every single-objective problem, RA follows a convex combination of the objective gradients. However, as in the case of point C in Figure 4.6, when the directions of the different objective gradients become too conflicting, a point in the CCS has been reached, and therefore, execution can stop. In order to understand when this condition has been reached, the optimization problem of Equation 4.4 can be solved:

$$\begin{aligned} & min_{\beta,l} && \beta + \tfrac{1}{2}||l||_2^2 \\ & \text{subject to} && (\boldsymbol{G} \cdot l)_i \geq \beta \ \forall i = 1, ..., q \end{aligned} \tag{4.4}$$

where $\boldsymbol{G}_{i,j} = \frac{\partial J_i}{\partial \theta_j}(\theta)$ is the Jacobian matric containing the gradient for every model parameter, for every objective, and q is the number of model parameters. In other words, we try to find the Ascent direction $l \in \mathbb{R}^d$ such that the minimum improvement among all the individual objectives is maximized. The L2-norm in the objective function is added as a regularization term. If $\beta = 0$, then the current model parametrization produces a Pareto optimal point, and therefore the equation can stop. In $\beta > 0$, then solution l of the quadratic problem corresponds to the best Pareto directions.

4.3.2.4 The RA Algorithm

The pseudocode for RA algorithm is presented in Algorithm 3.

At first, the number of points p to approximate the CCS are defined. Afterwards, an empty set $\mathcal{S}$ to keep the value (Line 1), and an empty set DRL_models to keep the single-objective models (Line 2) are initialized. The preference vectors $w_i,\ i \in 1, ..., p$ are defined, so that they uniformly cover the whole objective space (Line 3). Then, for each point individually, a single-objective optimizer initialized (Line 5), and iteratively trained using the Ascent simplex with respect to the preferences w_i of point i. (Line 9). The single-objective optimizer runs for every point i until either Pareto optimality is reached through Equation 4.4, or after the maximum number of episodes gets exceeded (Line 8). In practice, Pareto optimality is reached when there is no gradient direction that can improve all objectives at the same time. A parameter vector $\theta_i^{(t)}$ is considered Pareto optimal when the norm of its Ascent direction $l \in \mathbb{R}^d$ is smaller than ϵ. After the end of a single-objective execution, the value vector is computed with policy evaluation (line

Algorithm 3 `RadialAlgorithm`$(d, p, \epsilon, template)$

Require: the number of objectives, d, the number of points to explore, p, pareto optimality threshold ϵ, single-objective model template, *template*.

1: $\mathcal{S} \leftarrow \emptyset$ # an empty set to keep the value vectors for every point.
2: $DRL\_models \leftarrow \emptyset$ # Empty set to keep the single-objective models executed.
3: $\{w_i\}_{i=1}^{p} \leftarrow$ uniform sampling of $\mathbb{R}^d$
4: **for** $i = 1, ..., p$ **do**
5: $\quad\theta_i^{(0)} \leftarrow$ random initialized parameters for model *template*
6: $\quad d_i^{(0)} \leftarrow S(w_i, \theta_i^{(0)})$
7: $\quad t \leftarrow 1$
8: $\quad$ **while** $(not$ `isParetoOptimal`$(\theta_i^{t-1}, \epsilon)) \cup$ maximum iterations reached **do**
9: $\quad\quad\theta_i^{(t)} \leftarrow \theta_i^{(t-1)} + ad_i^{(t-1)}$ # Train the policy using the convex combination of objective gradients
10: $\quad\quad d_i^{(t)} \leftarrow S(w_i, \theta^{(t)})$
11: $\quad\quad t \leftarrow t + 1$
12: $\quad$ **end while**
13: $\quad V, new\_model =$ `policyEvaluation`(θ_i^t)
14: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{V\}$
15: $\quad DRL\_models[w] = new\_model$
16: **end for**
17: **return** $\mathcal{S}, DRL\_models$

15). Finally, the CCS-approximation points, and the single-objectives models what were executed are returned (Line 17).

4.3.2.5 RA experiments parameters

In Table 4.3 the parameters that were used for RA experiments are defined.

Parameter	Description	Value
p	Number of point to explore	66 (step=0.1)
m	Number of objectives	3
ϵ	Optimality threshold for ascent directions	0.1
template	Single-objective DRL model	Modified PPO (Section 4.2)

Table 4.3: Parameters used for RA experiments.

By setting p to 66, we ensure that the tested preferences will form a grid with $step = 0.1$. In other words, the generated preferences for our 3-objective problem will be in the form of $[1.0, 0.0, 0.0], [0.9, 0.1, 0.0], [0.9, 0.0, 0.1], [0.8, 0.2, 0.0], [0.8, 0.1, 0.1], ..., [0.0, 0.0, 0.1]$. A visual overview of the preferences grid is shown in Figure 4.7. Naturally, the surface defined by the preferences is the described by the function $x + y + z = 1$.

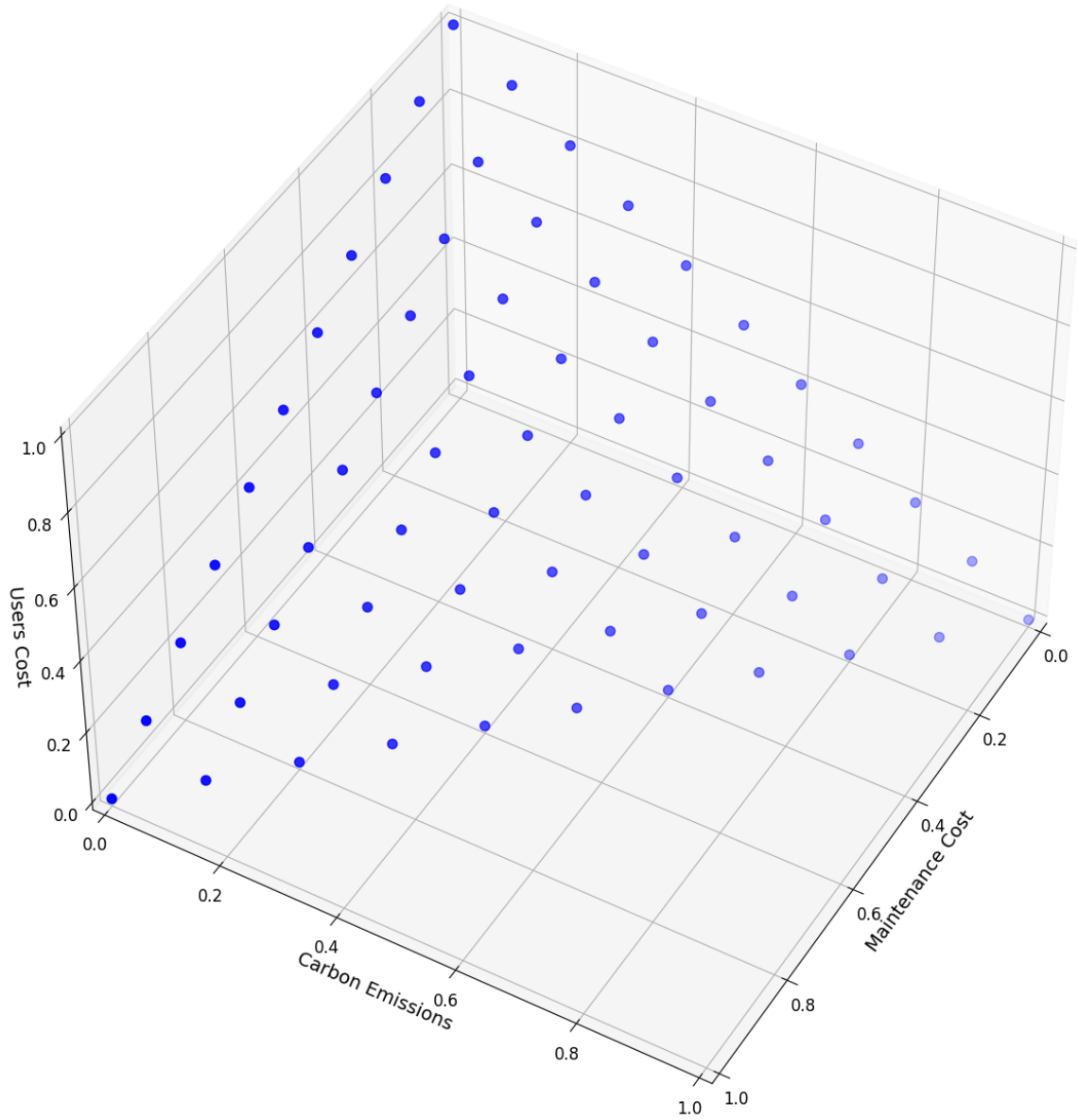


Figure 4.7: The grid of preferences used in RA experiments.

4.3.3 Pareto Following Algorithm (PFA)

4.3.3.1 Overview

Pareto Following Algorithm (PFA) (Parisi et al., 2014) is an outer-loop MORL algorithm that uses the notion of directed optimization on the Pareto front. In other words, PFA begins from a point in the Pareto front, and gradually moves to other points by improving some objectives, and decreasing some others.

The main concept revolves around optimizing one objective at a time. Therefore, in

the 2-objective case, where the Pareto front is a line, PFA starts by optimizing with respect to one of the objectives, and then gradually moves towards points in the Pareto front that have an increased preference over the other objective. However, starting from a point in the Pareto front and optimizing towards one objective potential produces dominated solutions. To emerge to the Pareto front again, a Pareto ascent direction can be followed (as described in Paragraph 4.3.2.2). A visual explanation of how PFA works for a 2-objective problem is presented in Figure 4.8

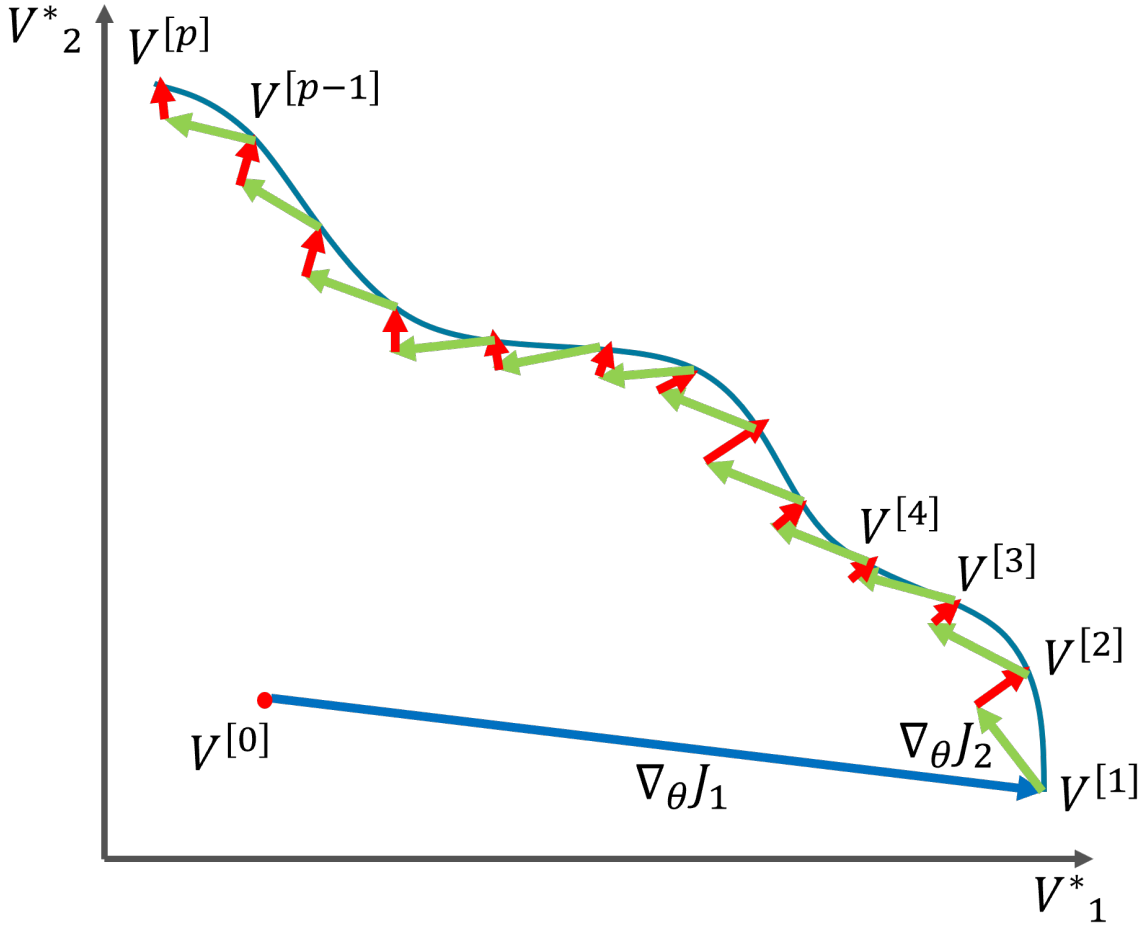


Figure 4.8: The behavior of PFA in a 2-objective problem. First, the single-objective problem is solved with respect to the 1st objective (blue line). Afterwards, the problem is incrementally solved with respect to the second gradient (green lines). In order to ensure that intermediate points in the CCS will be found, optimization iterations (red lines) are executed in specific intervals.

As a result, movement from one Pareto point to another involves the serial execution of an optimization step (solve the single-objective problem with respect to one objective), and a correction step (solve the single-objective problem using the best Ascent direction) using the Quadratic Programming Equation 4.4.

The same notion is followed when running PFA for 3-objective problems. Again, the single-objective problem is firstly solved for the 1st objective. Afterwards, incremental iterations towards the 2nd objective are executed. During this phase, all intermediate points on the Pareto front are kept. After the 2nd objective has been completely optimized, and after having collected a set of Pareto front points, we begin single-objective executions starting from all collected points. In that way, the whole Pareto surface is covered in a grid-like manner. A visual explanation of PFA in 3-d problems is presented in Figure 4.9.

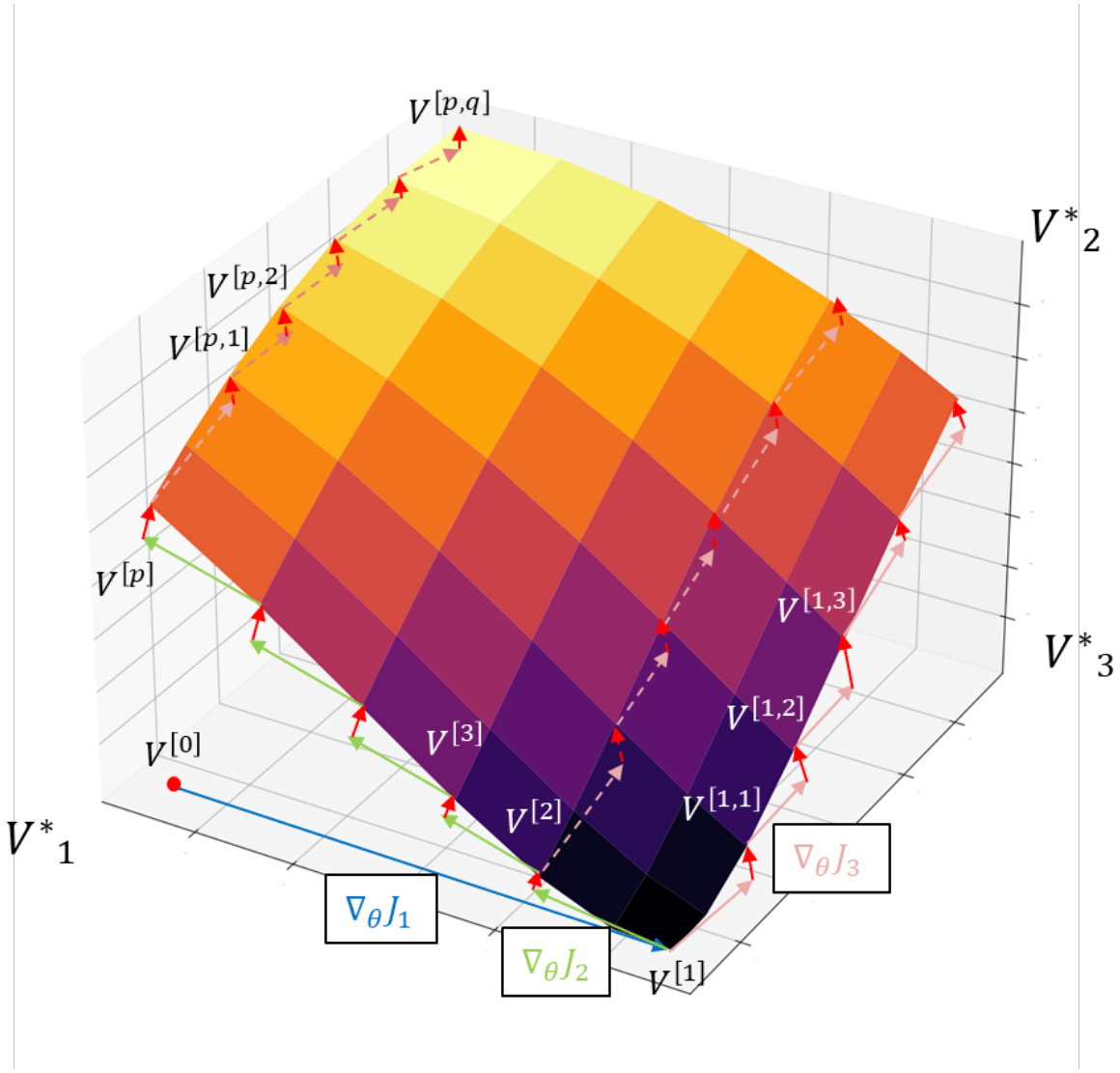


Figure 4.9: The behavior of PFA in a 3-objective problem. First, the single-objective problem is solved with respect to the 1st objective (blue line). Afterwards, the problem is incrementally solved with respect to the second gradient (green lines), in combination with optimization steps using the Pareto directions. After p steps, the single-objective problem is solved with respect to the 3rd objective, starting from all previously-computed Pareto points, for q steps.

4.3.3.2 PFA algorithm

The pseudocode for PFA algorithm is presented in Algorithm 5.

Algorithm 4 `ParetoFollowingAlgorithm`$(d, p, n_{corr}, n_{opt}, \epsilon, template)$

Require: the number of objectives, d, the number of points to explore per direction, p, the number of correction episodes, n_{corr}, the number of optimization episodes, n_{opt}, pareto optimality threshold ϵ, the single-objective model template, *template*.

1: $\mathcal{S} \leftarrow \emptyset$ # an empty set to keep the value vectors for every point.
2: $DRL\_models \leftarrow \emptyset$ # Empty set to keep the single-objective models executed.
3: $c \leftarrow 1$ # The number of objective to optimize. Start with the 1st one.
4: $model \leftarrow$ random initialized parameters for model *template*
5: $new\_model =$ `scalarizedDRL(c,model)` # Single-objective solver execution w.r.t to objective c
6: $V =$ `policyEvaluation`(new_model)
7: $\mathcal{S} \leftarrow \mathcal{S} \cup \{V\}$ #Add value vector to CCS
8: $DRL\_models \leftarrow DRL\_models \cup new\_model$ #Add model to DRL models
9: **while** c < d **do**
10: $c \leftarrow c + 1$ # Go to the next objective
11: $DRL\_models\_prev \leftarrow DRL\_models$ # Get the models solved up until the previous objective
12: **for** each model in DRL_models_prev **do**
13: **for** i *in* $1..p$ **do**
14: $V, new\_model =$ `scalarizedDRL`$(c, n_{opt}, model)$ # Single-objective solver execution w.r.t to objective c for n_{opt} episodes
15: $new\_model =$ `correctionDRL`$(n_{corr}, new\_model, \epsilon)$ # Perform correction by following the Pareto directions for n_{corr} episodes
16: $V =$ `policyEvaluation`(new_model)
17: $\mathcal{S} \leftarrow \mathcal{S} \cup \{V\}$ #Add value vector to CCS
18: $DRL\_models \leftarrow DRL\_models \cup new\_model$ #Add model to DRL models
19: **end for**
20: **end for**
21: **end while**
22: **return** $\mathcal{S}, DRL\_models$

The algorithm begins by initializing the empty set $\mathcal{S}$ to keep the value vectors (Line 1), and an empty set DRL_models to keep the single-objective models (Line 2). Afterwards, the multi-objective problem is solved with respect to only the first objective (Lines 3-6). After having calculated the Pareto point with respect to the first objective, we move to the next objective (Line 10). There, we perform p single-objective executions with respect to the new objective. More specifically, starting from the previous Pareto point we first execute a series of optimization episodes with respect to the new objective (Line 14). Then, correction episodes are performed (function `correctionDRL`), so that we again emerge to

a Pareto point (Line 15). During these episodes, the optimization problem of equation 4.4 is solved, and the optimized gradients' combination is used. The execution stops either after n_{cor} episodes have passed, or when a Pareto optimal parameter vector has been reached (based on 4.4 and optimality threshold ϵ). Finally, the value vector for the specific preferences is calculated from the `PolicyEcaluation` function, which evaluates a given policy. After completing all single-objective executions for the 2nd objective, we move to the next one. There, we iterate over all previously collected Pareto points (Line 12). Starting from them, we perform additional optimization-correction pair executions with respect to the 3rd objective. After completing all the steps for the third objective, PFA execution terminates. The value vectors of the CCS $\mathcal{S}$ are returned, together with the solved single-objective models DRL_models.

4.3.3.3 PFA experiments parameters

In Table 4.4 the parameters that were used for PFA experiments are defined.

Parameter	Description	Value
d	Number of objectives	3
p	Number of point to explore per direction	10
n_{cor}	Number of correction episodes	5000
n_{opt}	Number of optimization episodes	100
ϵ	Optimality threshold for ascent directions	0.1
template	Single-objective DRL model	Modified PPO (Section 4.2)

Table 4.4: Parameters used for PFA experiments.

4.4 Benchmarking

Since the multi-objective road network maintenance scheduling problem doesn't have a true Pareto Front as a ground truth, it is useful to compare the solutions generated from the MORL algorithms with a theoretical industry-standard. For that reason, we have added Multi-objective CBM in the evaluation. Multi-objective CBM is a naive implementation of a MORL algorithm using the CBM approach, discussed in Section 2.5.1. The multi-objective benchmark works as follows. At first, we define a set of preferences for which we want to execute CBM. We use the same set of preferences as the one used by RA for comparability (Figure 4.7). Afterwards, we treat CBM as an outer-loop approach,

and iteratively solve the single-objective problem for every set of preferences. In this way we gradually construct the CS for our benchmark. The pseudocode of the multi-objective CBM is presented in Algorithm 0.

Algorithm 5 `MultiObjectiveCBM`(d, p, T)

Require: the number of objectives, d, the number of points to explore, p, the time horizon, T.

1: $\mathcal{S} \leftarrow \emptyset$ # an empty set to keep the value vectors for every point.
2: $\{w_i\}_{i=1}^{p} \leftarrow$ uniform sampling of $\mathbb{R}^d$
3: **for** $w_i,\ i = 1, ..., p$ **do** # Iterate over the preference vectors
4: $repair\_thres_{opt} \leftarrow -1$ # Initialize the optimal IRI condition for repair action
5: $replace\_thres_{opt} \leftarrow -1$ # Initialize the optimal IRI condition for replace action
6: $insp\_intv_{opt} \leftarrow -1$ # Initialize the best time interval to perform inspection
7: $V_{opt}, V_{scal,opt} \leftarrow -\infty, -\infty$ # Initialize the value of the best return so far
8: **for** $inspect\_intv\ in\ T$ **do** # Try all different inspection intervals within time horizon to find for the optimal one
9: **for** $repair\_cond\ in\ IRI\_conditions$ **do** # Try all different IRI values for the optimal repair condition
10: **for** $replace\_cond\ in\ IRI\_conditions$ **do** # Try all different IRI values for the optimal replace condition
11: $V =$ `execute_environment`$(inspect\_intv, repair\_cond, replace\_cond)$
12: $V_{scal} = w_i^T \cdot V$ # Compute the scalarized value
13: **if** $V_{scal} > V_{opt}$ **then**
14: $V_{scal,opt} = V_{scal}$
15: $V_{opt} = V$
16: $repair\_thres_{opt}, replace\_thres_{opt}, insp\_intv_{opt} \leftarrow repair\_cond, replace\_cond, inspect\_intv$
17: **end if**
18: **end for**
19: **end for**
20: **end for**
21: $\mathcal{S} \leftarrow \mathcal{S} \cup V_{opt}$
22: **end for**
23: **return** $\mathcal{S}$

Given the number of objectives, number of points to explore and the time horizon, multi-objective CBM finds the best IRI condition to perform repair and replacement, and the best timestep to perform inspection for a several different preferences. Namely, function `execute_environment`$(inspect\_intv, repair\_cond, replace\_cond)$ calculates the value vector of a policy where an inspection takes place every $inspect\_intv$ timesteps, a repair is performed when a road segment reaches $repair\_cond$ condition, and a replacement is performed whenever a segment reaches $repair\_cond$ condition (Line 11). Afterwards, the scalarized value is computed based on the given preferences (Line 12). The optimal value

vector for every preference is the one for which the scalarized value is maximized (Lines 13-16). Finally, the algorithm returns the optimal value vectors for all the explored points p (Line 23).

4.5 Evaluation metrics

At this point, it is important to remind the reader that the multi-objective road maintenance scheduling problem that we are trying to tackle doesn't have a known Pareto front. For that reason, error-based evaluation metrics like the *ϵ-metric* (Zitzler et al., 2008), the Coverage ratio (Yang et al., 2019) and the Maximum Utility Loss (MUL) (Zintgraf et al., 2015) can't be used in our study.

Moreover, an interesting evaluation metric, that doesn't require a ground-truth to the Pareto front, is the Expected Utility Metric (EUM), again developed by Zintgraf et al. (2015). EUM computes the expected average utility of a solution set $\mathcal{S}$ for a family of scalarisation functions $V_{\pi}^{*}(w) = f(V_{\pi}, w)$ and given a probability distribution $P(w)$ over the preferences w. It is a utility-based metric, meaning that instead of evaluating the abstract Pareto front shape, it reasons based on the true user utility. However, to do that, it requires some prior knowledge about the user preferences in order to determine the probability distribution $P(w)$. Since we have no prior knowledge of user preferences in our problem, we avoid using EUM as well.

The only metrics found in the literature that do not require a ground-truth for the Pareto front, neither a prior knowledge about user preferences are the Hypervolume metric and the Sparsity metric.

The Hypervolume metric (HV) (Zitzler and Thiele, 1999) is the most widely used metric in the MORL literature. It measures the hypervolume of the value vectors of an approximate coverage set with respect to a given reference point (Figure 4.10). It can be used to compare the sets of solutions produced by different multi-policy algorithms, as the larger hypervolume corresponds to a better quality of the CCS.

The HV formula is given in Equation 4.5

$$HV(CS, V_{ref}) = \bigcup_{\pi \in CS} Volume(V_{ref}, V^{\pi}) \tag{4.5}$$

where V_{ref} is the reference value vector, and $Volume(V_{ref}, V^{\pi})$ is the volume of the hypercube with one corner in vector V_{ref} and another one in vector V_{π}. For our analyses, we chose point $(-100, -100, -100)$ as the reference for the Hypervolume calculation.

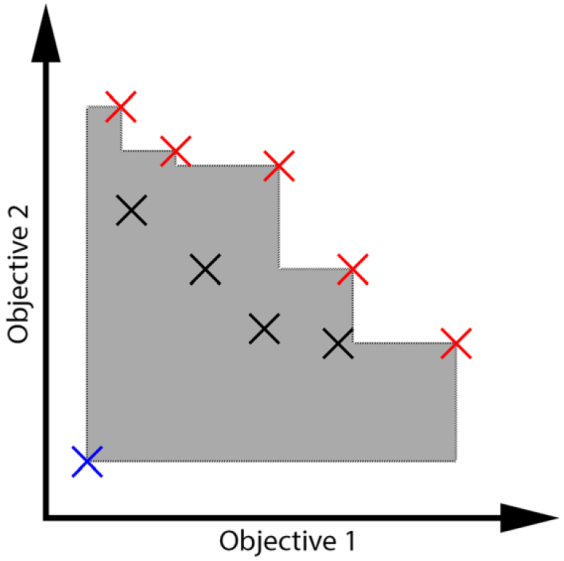


Figure 4.10: Hypervolume visualization for a 2-objective problem. The red marks are the non-dominated value-value vectors that construct the CS, while the blue mark is the reference point . The hypervolume corresponds to the grey area (Hayes et al., 2021).

If we have two CS sets with (approximately) equal size, then intuitively we should select the one, the value vectors of which are more spread over the value space. For this reason Xu et al. (2020) proposed a Sparsity metric that calculates the average distance between the sorted value vectors of a CS set $\mathcal{S}$:

$$Sp(\mathcal{S}) = \frac{1}{|\mathcal{S}| - 1} \sum_{j=1}^{m} \sum_{i=1}^{|S|-1} (\tilde{\mathcal{S}}_j(i) - \tilde{\mathcal{S}}_j(i+1))^2 \tag{4.6}$$

where $\tilde{\mathcal{S}}_j(i)$ is the i-th value in the sorted list of the j-th objective value vectors and m is the number of objectives.

It is however not trivial to evaluate the sparsity metric. One could argue that between two CS with the same number of value vectors, the one with the lowest sparsity has a better spread of the value vectors over the value space. On the other side, when comparing CS

with different sizes, its intuitive to think that the one with the most points has a lower sparsity. However, as discussed in the Background section, we try to keep the CS as small as possible. Because of this ambiguity, it is deemed that sparsity comparisons are only meaningful when the CS size between two solution sets is comparable.

4.6 Experimental setting

The experiments with the previously described MORL algorithms were executed in DelftBlue compute cluster (Delft High Performance Computing Centre , DHPC), using 9 CPU cores of 20GB each, in every experiment. Every MORL algorithm was executed 3 times with different seeds, to ensure reproducibility. The results of the experiment are reported in the respective Results section.

4.7 Initial reflection on the MORL approaches

Before performing any experiments, we can make some educated assumptions on how each algorithm is expected to work. Firstly, concerning the DOL and DOL-PR approaches, we expect that DOL-PR will perform better due to the model parameters reuse. On the other hand, DOL may perform better than DOL-PR in the first single-objective iterations because the initial preference vectors will not be close enough for DOL-PR to reuse the parameters effectively.

Moreover, RA is expected to achieve reasonable results since the algorithm will uniformly pick preference vectors over the value space. This will allow for a thorough exploration of the value space.

Finally, PFA optimizes with respect to each objective individually for a certain number of weights. This raises a concern about its effectiveness since we don't know the exact shape of the Pareto front a priori and it is difficult to finely tune the parameter n_{opt}.

5 Results

The present chapter contains the results from the evaluation of the developed multi-objective environment, as well and the results from the multi-objective experiments.

5.1 Environment evaluation

To gain a deeper understanding of the multi-objective road maintenance environment, we conducted a series of experiments to evaluate its performance. Our first experiment involved optimizing for each of the three objectives individually using the modified PPO algorithm described in Section 4.2, followed by visualizing the resulting policies. This allowed us to gauge the level of conflict between the objectives and gain insight into their respective importance.

In the second experiment, we examined the individual components that contribute to each objective, providing valuable information on the impact of each component on the final policy.

Finally, we compared the action sequences generated by the optimized policies, enabling us to evaluate their effectiveness and make informed decisions on the best maintenance strategies to implement. Through these experiments, we gained a more comprehensive understanding of the multi-objective road network environment.

5.1.1 Single-objective policies comparison

After optimizing with respect to a single objective each time, we can see that modified PPO manages to surpass the CBM benchmark within the the first 5,000 episodes for all benchmarks, as shown in Figure 5.1.

It is interesting to understand why the user cost benchmark is very easily surpassed by the algorithm. A main disadvantage of the CBM algorithm, is that it applies the same policy to all road segments. On the other side, the policy generated by modified PPO is in principle different per road segment, giving more combinations of actions to explore. In the case of the "User cost", we will see that the optimal single-objective PPO policy results in a very diverse maintenance actions sequence for the different road segment,

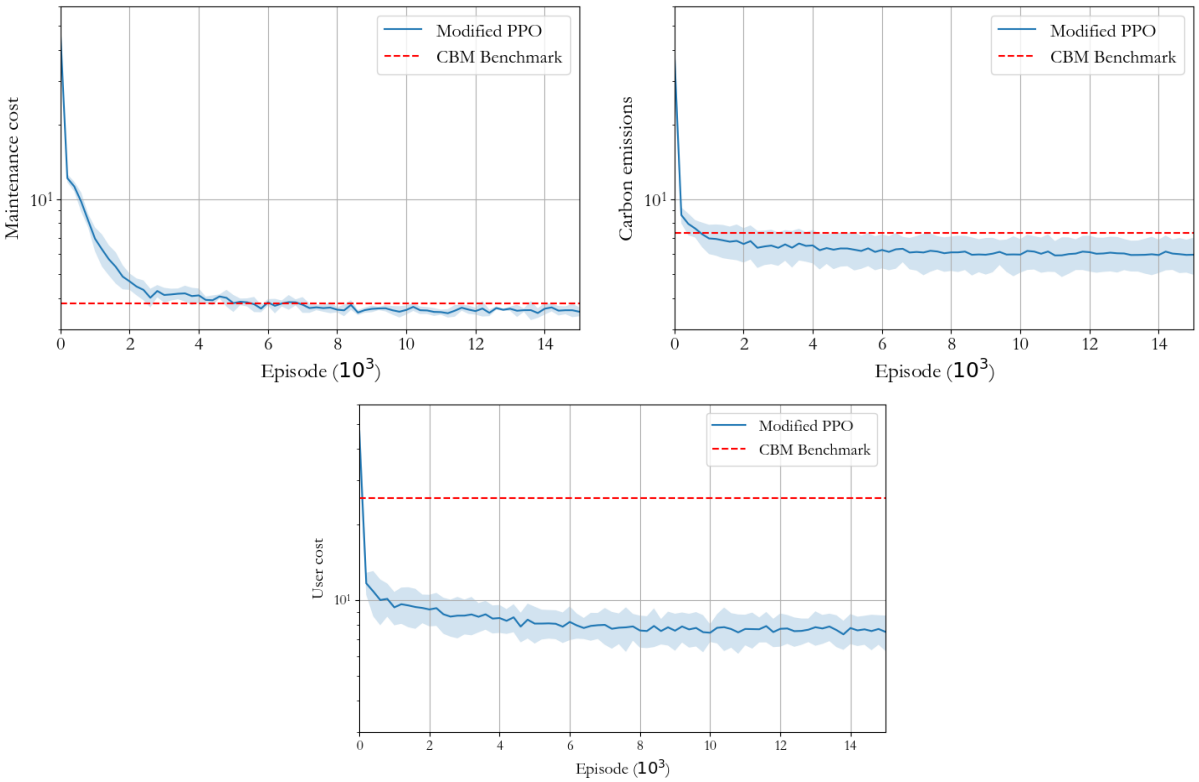


Figure 5.1: Comparison of the single-objective executions with the CBM benchmark, for the maintenance cost (top left), carbon emissions (top right) and user cost objectives (bottom). The training curves have resulted as the average of 5 executions, while the confidence interval with confidence 95% is visualized.

which is something that CBM can't achieve, and thus results in sub-optimal returns.

Moreover, by comparing the value vectors returned from each policy, we can get an idea of how conflicting the three problem objectives are. The three value vectors generated from the three single objectives policies are visualized in the value space in Figure 5.2. We realize that the value vectors of the maintenance cost-optimized and the carbon emissions-optimized policies appear to be in the same region, where the scalarized values for both these objectives are relatively small. This gives us a first indication that the two objectives have some correlation.

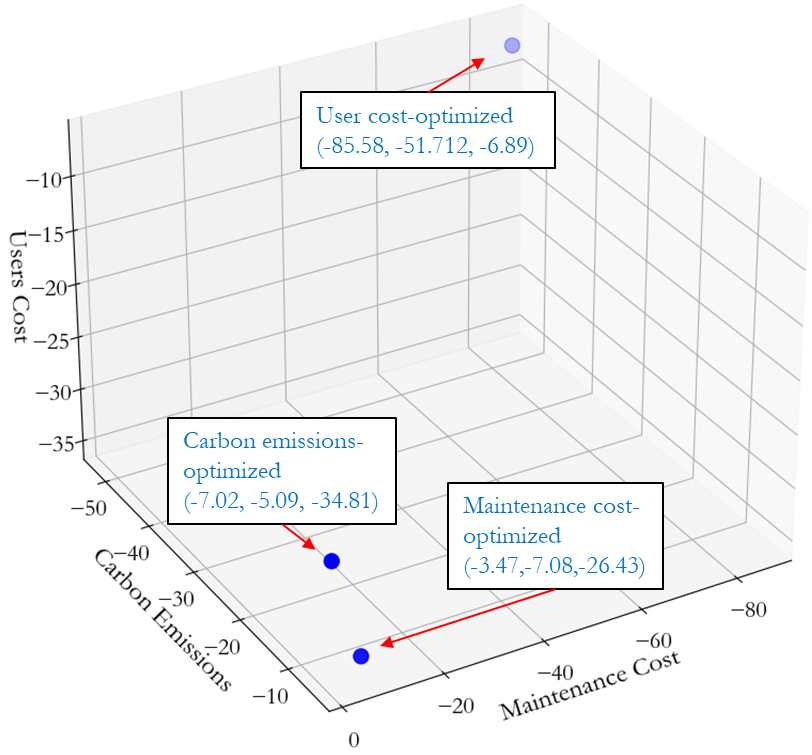


Figure 5.2: Comparison of the normalized value vectors for three extreme preferences policies.

To get a better idea of the degree of correlation between the different objectives, and how this is expressed in real-life metrics, Table 5.1 displays the difference in the maintenance costs between the three policies and the benchmark in relative and absolute metrics. The modified PPO execution that was optimized w.r.t the maintenance cost results in 7.13% less maintenance costs compared to he CBM benchmark. Moreover, we observe that the PPO policies optimized for the other objectives (and especially the user cost) result to

substantially higher maintenance costs.

Maintenance Cost (Million USD)			
Policy	**Return**	**Difference with CBM (%)**	**Difference with CBM (absolute)**
PPO w.r.t. Maintenance cost	83.4	-7.13%	-6.4
PPO w.r.t. Carbon emissions	331.8	+269.60%	+242.1
PPO w.r.t User cost	1955.1	+2077.17%	+1865.3
CBM	89.8	0.00%	0.00

Table 5.1: Comparison of single-objective policies in terms of maintenance cost objective.

Next, the respective information for the Carbon emissions objective are displayed in Table 5.2. Here we observe that the policy that is optimized with respect to the carbon emissions objective emits 0.13% less greenhouse gases than the benchmark. It is interesting to understand that for a road network of roughly 85km, even such a difference equals to $2,530$ tons of CO2e emissions, which corresponds to powering almost 300 houses for one year (Environmental Protection Agency, 2023)!

Total Carbon Emissions (Tons CO2e)			
Policy	**Return**	**Difference with CBM (%)**	**Difference with CBM (absolute)**
PPO w.r.t. Maintenance cost	1.88E+07	-0.03%	-6.29E+03
PPO w.r.t. Carbon emissions	1.87E+07	-0.13%	-2.53E+04
PPO w.r.t User cost	2.03E+07	8.26%	1.55E+06
CBM	1.88E+07	0.00%	0.00

Table 5.2: Comparison of single-objective policies in terms of carbon emissions objective.

Finally, the comparisons for the user cost objective are presented in table 5.3. We observe that the modified PPO policies that are not optimized with respect to the user cost objectives lead to substantially higher costs for the vehicle owners.

User costs (Million USD)			
Policy	**Return**	**Difference with CBM (%)**	**Difference with CBM (absolute)**
PPO w.r.t. Maintenance cost	1.13E+04	0.11%	1.26E+01
PPO w.r.t. Carbon emissions	1.15E+04	1.89%	2.13E+02
PPO w.r.t User cost	1.10E+04	-2.80%	-3.05E+02
CBM	1.13E+04	0.00%	0.00

Table 5.3: Comparison of single-objective policies in terms of user cost objective.

5.1.2 Reward components analysis

As previously stated, the "Maintenance Cost" and "Carbon Emissions" reward functions contain multiple components.

Namely, the "Maintenance Cost" reward function contains the following negative rewards:

- Cost of maintenance actions
- Cost of inspection actions
- Cost of mobilization of the equipment
- Cost of urgent actions

On the other hand, the "Carbon Emissions" function includes:

- The emissions from the maintenance actions
- The added vehicle emissions from rerouting due to maintenance actions.
- The extra vehicle emissions due to high road roughness.

In order to estimate the impact of each component in the objective functions, we visualize the realization of the policy that is optimized with respect to the maintenance cost and the one with respect to the carbon emissions, and view the negative rewards breakdown.

The cumulative return of the maintenance cost-optimized policy is presented in Figure 5.3. One can observe that the dominant component is the cost of the maintenance actions. Moreover, the cost of urgent actions is zero, indicating that in this policy, no component

ever reaches the damaged condition.

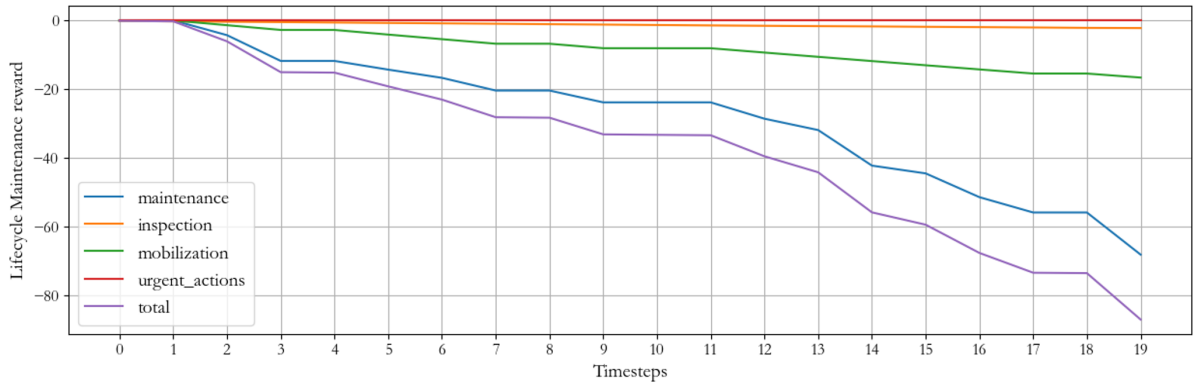


Figure 5.3: Cumulative negative rewards of the policy that was optimized with respect to the "Maintenance Cost" objective.

Moreover, the optimized policy with respect to the "Carbon Emissions" objective is presented in Figure 5.4. It is interesting to notice that the "rerouting" cumulative reward value is positive for the first 8 years, meaning that due to rerouting, less carbon is emitted from the vehicles. This happens because the traffic assignment optimizer chooses the optimal traffic assignment with respect to the travel time, while the carbon emissions are relative to the travel distance. To what is more, the most dominant components are the ones related to the condition of the asphalt and the rerouting.

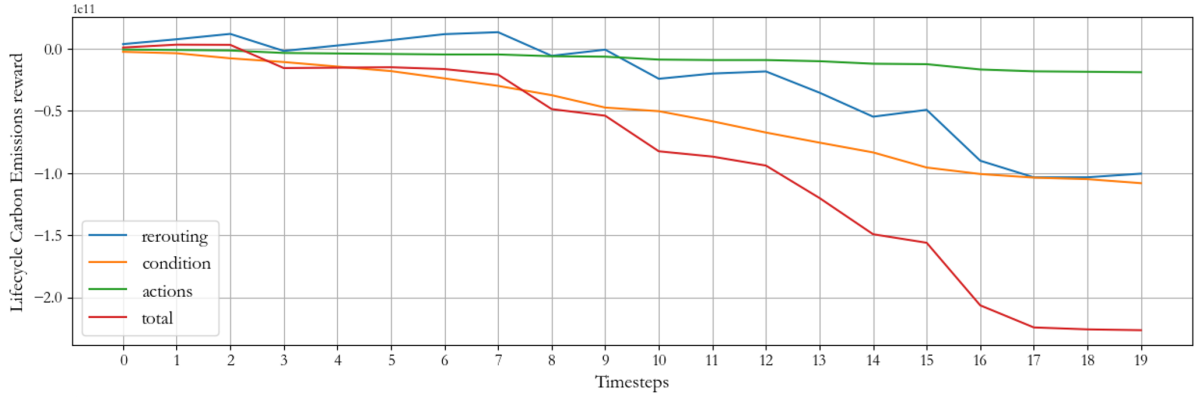


Figure 5.4: Cumulative negative rewards of the policy that was optimized with respect to the "Carbon Emissions" objective.

5.1.3 Policy realizations

Next, we visualize the realizations for each of the three single-objective policies. The realization of the policy that is optimized with respect to the maintenance cost is presented in Figure 5.5.

Figure 5.5: Policy realization for all road segments when following a policy that is optimized with respect to maintenance cost minimization. On the x-axes is the timestep, while on the y-axes is the IRI rating. The action for every timestep is depicted with a yellow mark. No mark means "Do nothing" action was picked.

We can see that no segment reaches the "damaged state" ($IRI = 4$), hence avoiding the extra urgent action cost. Also, we see that no replacement action is performed. This is logical, as the cost of a replacement is very high, and the algorithm cleverly opts to substituting it with several repair actions. Finally, inspection actions are executed almost in every timestep. This is done to minimize the uncertainty over the belief, and hence avoid the risk not maintaining segment that approaches the damaged condition.

The realization of the policy that is optimized with respect to the carbon emissions is presented in Figure 5.6.

In this case, we observe that there exist multiple segments $(1, 2, 4, 7, 10)$ that at some point reach the damaged state and get replaced afterwards. Moreover, we observe that road segment 5 and 8, connecting nodes $12 \rightarrow 11 \rightarrow 4$ are preserved in an almost intact

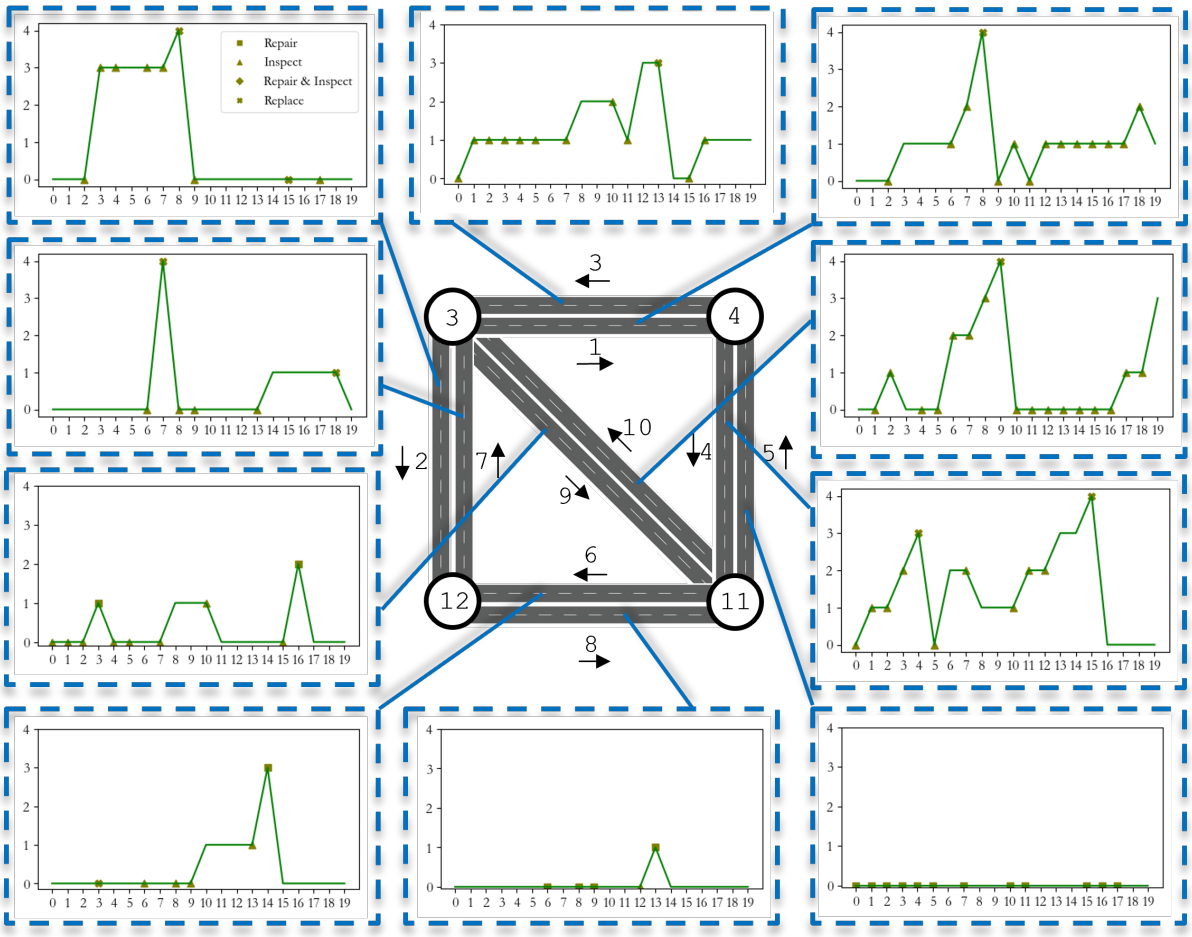


Figure 5.6: Policy realization for all road segments when following a policy that is optimized with respect to carbon emissions minimization. On the x-axes is the timestep, while on the y-axes is the IRI rating. The action for every timestep is depicted with a yellow mark. No mark means "Do nothing" action was picked.

condition throughout their lifecycle. This is chosen because by performing multiple repair actions on segments 5 and 8, their traffic capacity decreases, and therefore traffic from node 12 to node 4 needs to reroute. The alternative route $12 \rightarrow 3 \rightarrow 4$ may be slower, but has a lower total length, and hence it results in fewer CO2e emissions.

The realization of the policy that is optimized with respect to the user cost is presented in Figure 5.7.

Here, it is first observed that the agent has picked a yearly replacement schedule for all segments from-and-to node 3. A possible explanation for this behavior lies on the fact that for the user cost objective, the agent needs to search for a trade-off solution between having a nearly intact road condition, and avoiding to travel extra km, as the cost for vehicle owners increases with the distance. Therefore, because the traffic demand

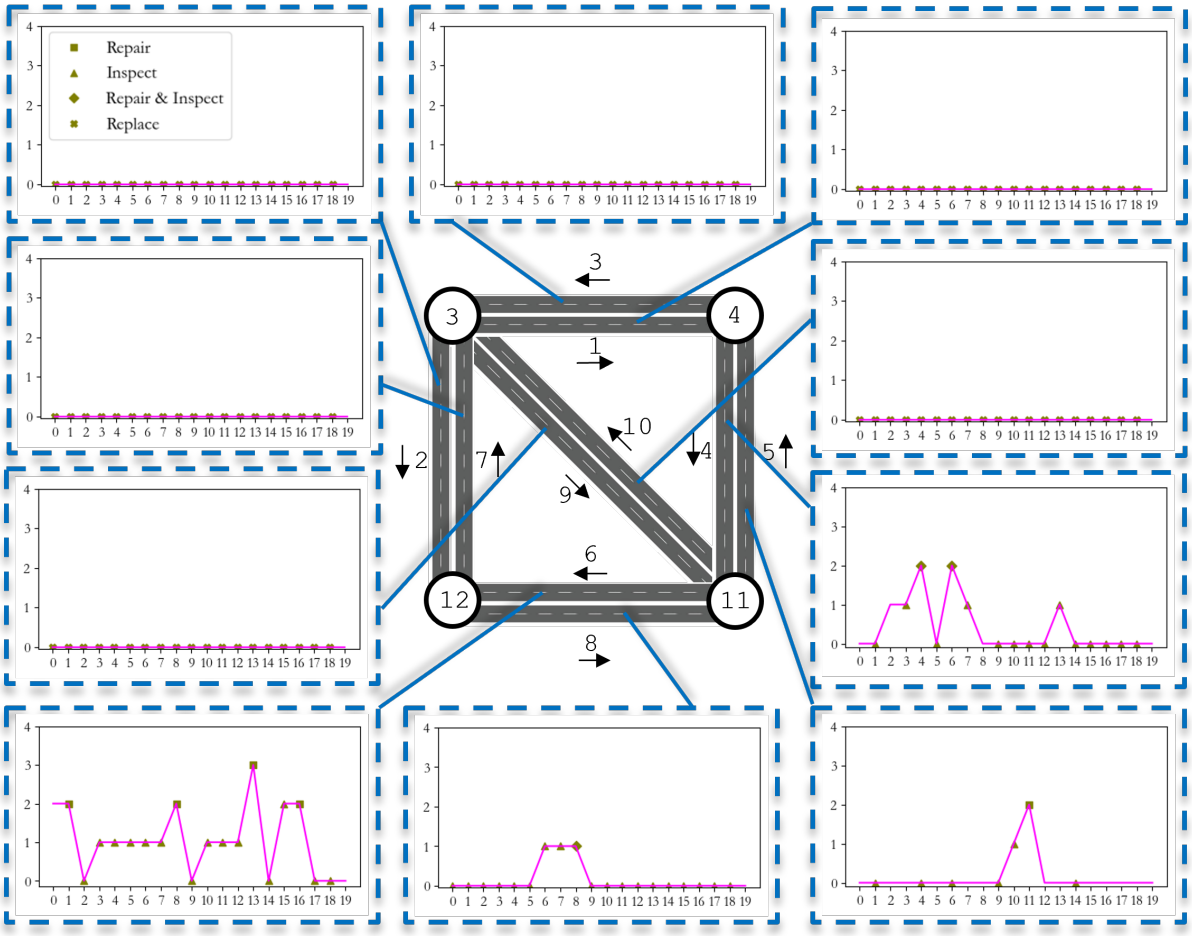


Figure 5.7: Policy realization for all road segments when following a policy that is optimized with respect to user cost minimization. On the x-axes is the timestep, while on the y-axes is the IRI rating. The action for every timestep is depicted with a yellow mark. No mark means "Do nothing" action was picked.

from-and-to node 3 is small (as illustrated in Figure 3.6), the limited capacity due to road replacement is enough to accommodate the traffic load. For the rest of the segments, a conservative maintenance and inspection schedule is selected as the best combination between traffic disruption and condition improvement.

5.2 Multi-objective experiments

In this section, we evaluate the results for the experiments with the MORL algorithms. Firstly, we visually compare the Coverage sets generated, while afterwards we compare the MORL approaches in terms of the hypervolume and sparsity metrics, and sample efficiency.

5.2.1 Coverage sets visualization

Since the problem we attempt to solve is a 3-objective one, it is not trivial to visually compare the generated Coverage sets on paper. Therefore, it was deemed that visual comparison should be performed in pairs of objectives. The 3-dimensional visualizations of the Coverage set for the best experiment of each MORL algorithm are presented in Appendix A3. The Coverage sets comparison in paired objectives are presented in Figure 5.8

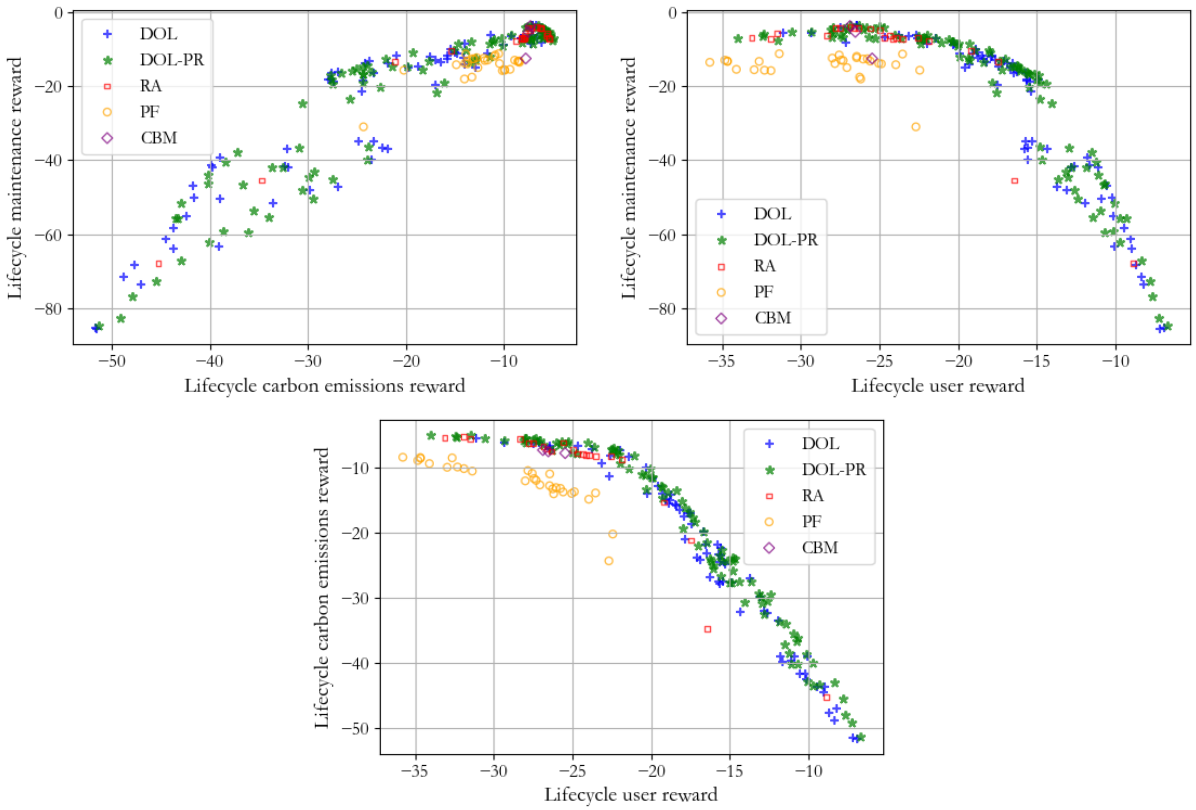


Figure 5.8: Comparison of multi-objective experiment for the MORL algorithms in paired objectives. The results presented are the best output out of 3 experiments per MORL algorithm.

The first thing that we can notice from Figure 5.8.a is that the "Maintenance cost" and "Carbon emissions" objectives are positively correlated for all MORL algorithms . However, the "User cost" objective is conflicting with the former two, as the 2-dimensional coverage sets containing the value vectors between the "User cost" objective and one of the other two have a clear convex shape.

Moreover, we observe that PFA results in a poorer approximation compared to the other

algorithms. There are regions where no PFA value vectors are present, while there are other regions where the PFA vectors are dominated from other solution vectors.

Finally, the CBM benchmark seems to have produced very few points. On the other side of things, it looks like the DOL approaches produced many value vectors that are well distributed across the values space.

5.2.2 Comparison of MORL algorithms

In this section we perform numerical evaluation of the examined algorithms with respect to the metrics defined in Section 4.5. The numerical results for all MORL algorithms and the benchmark are presented in Table 5.4.

MORL Algorithm	**# CS Points**	**# Iterations**	**Hypervolume**	**Sparsity**	**Runtime/iteration (minutes)**
DOL	70	96	784450.14	7.96	46.73 ± 0.96
DOL-PR	97	119	792041.05	4.02	67.58 ± 3.37
RA	22	66	757946.92	95.15	144.02 ± 98.05
PFA	29	121	626173.16	8.5	48.74 ± 64.38
CBM	3	66	664089.01	27.72	10.02 ± 0.11

Table 5.4: Numerical comparison between the different MORL algorithms. The margins in the runtime correspond to one standard deviation, using a sample of 3 experiments per MORL algorithm. Reminding the reader that sparsity comparisons can only be conducted between Coverage sets with (roughly) the same size. In this case, the lower sparsity equals a betters spread across the value space.

We observe that three out of four MORL algorithms perform better than the multi-objective CBM benchmark in terms of hypervolume. Comparing the sparsity is not applicable because of the large difference in the number of CS points between multi-objective CBM and MORL algorithms. DOL-PR achieves the best hypervolume, followed by the DOL with no reuse. DOL-PR also results in the most CS points, even though the number of iterations (i.e. the number of different preferences explored) is roughly the same as DOL. Moreover, in terms of runtime, one DOL-PR iteration takes approximately 50% more than a DOL iteration. This has to do with the KL divergence stopping mechanism of the modified PPO training. KL divergence is used as a mechanism to stop training for an episode where the actor parameters change more than a threshold. Therefore, during the first episodes of modified PPO in DOL experiments, training is terminated early due to high KL divergence. On the other hand, during the DOL-PR experiments, the policy is already near-optimal from the first episodes, and training does not terminate early,

hence adding more computational time.

To what is more, we observe that PFA shows the worst performance in terms of hypervolume, and sample efficiency, as it generated only 29 CS points from 121 single-objective iterations.

RA has a better sample efficiency and resulted to a higher hypervolume compared to PFA. However, its really large sparsity suggests that the points in the CS are not evenly distributed. This can be further validated from Figure 5.8, where the RA value vectors are mostly concentrated in the low-maintenance cost, low-carbon emissions, high-user costs region.

Finally, another thing to notice is the high variability in the runtime for RA and PFA. This is attributed to the stopping mechanism that has been applied to both algorithms (described in Paragraph 4.3.2.3), that terminates a single-objective execution when the norm of the Ascent directions is close to 0.

Visual comparison of hypervolume and sparsity metrics over iterations is presented in Figure 5.9. It's important to note that for RA and for the CBM benchmark, by design, the order of iterations has no effect in the resulting hypervolume.

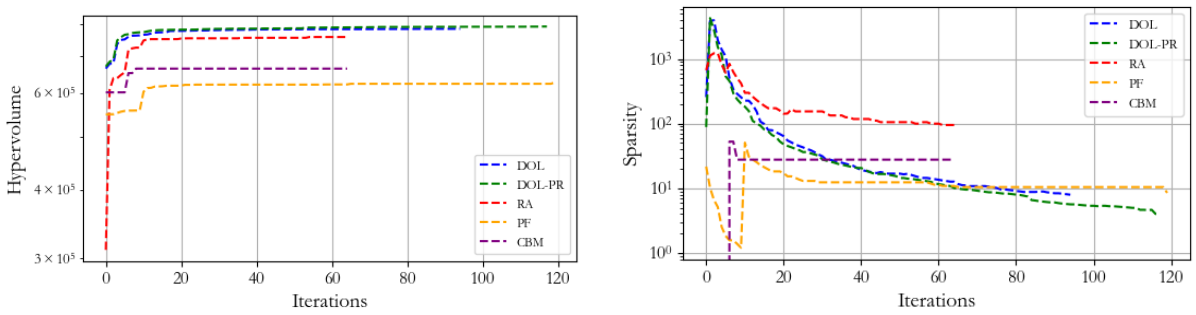


Figure 5.9: Comparison of the MORL algorithm results with respect to the hypervolume and sparsity metrics over inner-loop iterations. Comparison is performed for the best experiment of each algorithm in terms of hypervolume.

One can observe that from the first single-objective iterations, DOL-PR and DOL result in higher hypervolumes, indicating that they consistently produce value vectors that are closer to the (unknown) true Pareto front, compared to the other MORL approaches.

6 Discussion - Conclusion

6.1 Discussion of results

In this study, we modelled a multi-objective road network environment and tried to find the optimal inspection and maintenance policy with respect to maintenance cost minimization, carbon emissions reduction and users cost minimization. In this section are discussing the results of the project. We will first analyze the results retrieved by the environment evaluation, and then review our key finding concerning the MORL experiments.

6.1.1 Environment evaluation

We developed a realistic model of a road network environment, containing 10 road segments, with a total length of roughly 84km.

By performing single-objective optimization for each one of the three objectives, we realized that there is correlation between the Maintenance cost and Carbon emissions objectives, while the User cost objective is conflicting with the rest. Moreover, we were able to surpass the CBM benchmark in all single-objective executions.

To what is more, by reviewing the different reward components 5.1.2, we understood that the monetary cost of maintenance actions themselves accounts for most of the cost incurred within the IM operations of the road (78%). At the same time, we observed that because of the heavy traffic in the road network, the large majority (98%) of the carbon emissions comes from the vehicle emissions, while the smallest part comes from the emissions due to the IM actions.

Additionally, based on Tables 5.1, 5.2 & 5.3, optimization with respect to the maintenance cost objective can have the highest relative impact, as we were able to save 76.5$/km of road (-7.21%) compared to the benchmark. On the other hand, the relative impact by optimizing with respect to carbon emissions and user cost objectives is lower (0.14% and 2.78% respectively). This is due to the fact that carbon emissions and user costs are indirectly related to maintenance and inspection of the road network. Even so, our optimization still results in large absolute savings.

Finally, from the policy realizations in Figures 5.5, 5.6 & 5.7, we observed that the agent was able to find policies that intuitively are reasonable, and in the case of the User cost-optimized execution, apply different policies per road segment to radically improve the resulted return.

All above observations show that developed the road network environment accurately models the three objectives under examination, producing reasonable policies.

6.1.2 Multi-objective experiments

For the MORL experiments, we observed that DOL-PR achieves the best results in terms of hypervolume in all experiments. Employing partial reuse of the model parameters, the agent can transfer information from a single-objective execution to another, and hence improve in terms of sample-efficiency and performance. The above result comes in alignment with Mossalam et al. (2016), who also observed superior behavior of DOL-PR over DOL without reuse.

The multi-objective CBM benchmark was able to produce a CS of only three points, and was surpassed by 3 out of the 4 examined MORL approaches in terms of Hypervolume. This tells us that CBM isn't adequate to provide a solution for varying preferences. Moreover, as CBM assigns the same maintenance and inspection thresholds for all the road segments, it results in very rigid and limited policies, that are unable to treat road segments individually. This weakness was especially observed in the case of the single-objective optimization with respect to the "User cost" objective, where PPO was able to find a policy where some of the road segments were repaired very often, while others were not, resulting in much lower user costs. The results of this experiment are presented in Table 5.3.

Moreover RA was proven to be sample inefficient, having run 66 inner-loop iterations and resulting to just 29 CS points. The stopping mechanism that computes the Ascent direction based on Equation 4.4 produces sub-optimal solutions for many of the preferences that were tested. Moreover the spread of the CS points over the value space is far from uniform (due to the very high sparsity), with most of the CS points residing in the region with high values for the "Maintenance cost" and "Carbon emissions" objectives. Nevertheless, RA-generated CS still beats the benchmark. Finally, as discussed in the

Results chapter, the high variability in the runtime per iteration is explained by the stopping mechanism.

Finally, the CS generated from PFA is the one producing the most sub-optimal hypervolume, staying below the benchmark. Based on these results, we conclude that PFA isn't able to approximate the solution for the multi-objective problem of the road network. This comes in alignment with Xu et al. (2020), who reported that PFA is hard to implement in environments with more than 2 objectives. This happens because PFA blindly shifts from a preference vector to another, making it difficult to properly tune the n_{opt} and n_{cor} parameters, such that the true Pareto front is approximated. Parisi (2020) also reports PFA can't properly scale to multiple objectives, and hence achieve worse hypervolume than RA in the 3-objective Linear Quadratic Regulator (LQR) problem.

Overall, DOL-PR is the most effective MORL approach to tackle to multi-objective road maintenance scheduling problem, between the algorithms tested, achieving the best results in terms of CS quality.

In practice, the application of a MORL approach, such as the DOL-PR algorithm, holds great promise for solving the complex maintenance scheduling problem in road networks. By utilizing MORL, it becomes possible to construct a comprehensive Coverage set in the value space, providing asset managers with a range of potential policies to consider.

Building upon the discussion in the Motivation section, where the need for multiple re-evaluations of the maintenance plan due to budget updates was highlighted, MORL offers a valuable solution. Asset managers can fine-tune the preferences of the MORL model to align with the specific requirements of each situation. This adaptability allows for efficient and effective adjustments to the maintenance schedule, significantly reducing the duration of planning iterations. The utilization of an automated tool like MORL in generating the maintenance schedule brings numerous benefits. First and foremost, it provides asset managers with the ability to experiment with different preferences over objectives, enabling them to explore various trade-offs and make informed decisions. The MORL approach produces a diverse set of maintenance policies, covering a wide range of possible solutions, thereby expanding the decision-making options available to asset managers. Moreover, the incorporation of MORL into the maintenance scheduling process leads to a substantial reduction in planning iteration duration. By automating

the generation of maintenance schedules, MORL eliminates the need for time-consuming manual adjustments and iterations. The ability to quickly generate and evaluate multiple maintenance schedules significantly streamlines the planning process, allowing asset managers to allocate their time and resources more efficiently.

6.2 Limitations

When interpreting the results of this study, it is important to consider its limitations. Firstly, the learning model used assumes that each segment of the road network deteriorates independently, whereas there may be some correlation between the deterioration of different segments. Additionally, the study assumes that carbon emissions from vehicle fuel consumption are proportional to travel time when in reality this relationship is more complex. Furthermore, the study assumes that traffic patterns, traffic composition, vehicle emissions, and maintenance action emissions are fixed within the time horizon. Lastly, the International Roughness Index (IRI) is just one of the metrics used to capture the condition of asphalt effectively.

Secondly, the study's purview is limited as it only explores outer-loop MORL algorithms. A more comprehensive investigation should include inner-loop approaches as well. Additionally, the study only considers three objectives: maintenance cost minimization, carbon emission reduction, and user cost minimization. Other significant objectives, such as safety maximization and travel time minimization, are not considered.

Thirdly, computational limitations restrict the number of experiments that can be conducted due to the high time complexity of the calculations. As a result, for each MORL algorithm, only three multi-objective experiments were conducted.

Lastly, because time complexity of multi-objective approaches scales exponentially with the number of objectives, we limited our work to three objectives, even though a real-life road network scheduling problem includes much more (f.i. travel safety, traffic disruptions). Outer-loop MORL algorithms typically require many more inner-loop iterations to reach a Coverage set. Therefore, scalability is an important constraint in a complex problem with many stakeholders and, that should be noted.

To conclude, this study provides valuable insights into the application of MORL algorithms

to road network maintenance planning. However, it is important to acknowledge the aforementioned limitations when interpreting the results.

6.3 Future research

The present project can become the basis of multiple interesting future research ideas.

First, related to the road network environment, it can be enhanced to include other objectives that are important for the road maintenance stakeholders, such as travel safety and travel time. Also, to better understand the environment dynamics, it would be interesting to perform sensitivity analyses of several environment parameters (f.i. traffic data, road segments capacity) and assess their impact in the resulting policies. Finally, it would be interesting to explore the use of a surrogate model as the traffic assignment optimizer, as the one in the work of Yuan et al. (2022).

Secondly, related to the MORL approaches, extending the scope of the study to include other state-of-the-art outer-loop MORL approaches like Chen et al. (2018) and Xu et al. (2020), or inner-loop approaches would possibly result in better approximations of the problem's true Pareto front. Finally, as previously stated, the partial model parameter reuse employed by DOL-PR results in faster learning for the single-objective runs. However, with a fixed number of episodes per single-objective execution, the execution time can be even greater compared to the no-reuse experiments. Implementing an early stopping mechanism for DOL-PR, could improve the sample efficiency and at the same time preserve its superior performance. This update is also suggested as a future research from the authors of DOL-PR (Mossalam et al., 2016).

References

1. A. Abdolmaleki, S. H. Huang, L. Hasenclever, M. Neunert, H. F. Song, M. Zambelli, M. F. Martins, N. Heess, R. Hadsell, and M. A. Riedmiller. A distributional view on multi-objective policy optimization. *CoRR*, abs/2005.07513, 2020. URL https://arxiv.org/abs/2005.07513.

2. C. Andriotis and K. Papakonstantinou. Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering System Safety*, 191:106483, 2019. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2019.04.036. URL https://www.sciencedirect.com/science/article/pii/S0951832018313309.

3. C. Andriotis and K. Papakonstantinou. Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints. *Reliability Engineering System Safety*, 212:107551, 2021. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2021.107551. URL https://www.sciencedirect.com/science/article/pii/S095183202100106X.

4. G. G. Ayalew, M. G. Meharie, and B. Worku. A road maintenance management strategy evaluation and selection model by integrating fuzzy ahp and fuzzy topsis methods: The case of ethiopian roads authority. *Cogent Engineering*, 9(1):2146628, 2022. doi: 10.1080/23311916.2022.2146628. URL https://doi.org/10.1080/23311916.2022.2146628.

5. R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, 1961. ISBN 9781400874668. doi: doi:10.1515/9781400874668. URL https://doi.org/10.1515/9781400874668.

6. G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

7. S. Calisir and M. Kurt Pehlivanoğlu. Model-free reinforcement learning algorithms: A survey. pages 1–4, 04 2019. doi: 10.1109/SIU.2019.8806389.

8. E. F. Camacho, C. Bordons, and J. E. Normey-Rico. Model predictive control springer, berlin, 1999, isbn 3540762418, 280 pages. *International Journal of Robust and Nonlinear Control*, 13(11):1091–1093, 2003. doi: https://doi.org/10.1002/rnc.752. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.752.

9. A. Chakirov and P. J. Fourie. Enriched sioux falls scenario with dynamic and disaggregate demand. 2014.

10. J. Chen and Y. Wang. A deep reinforcement learning approach for maintenance planning of multi-component systems with complex structure. *Neural Computing and Applications*, 2023. doi: 10.1007/s00521-023-08542-9. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85152803004&doi=10.1007%2fs00521-023-08542-9&partnerID=40&md5=4aeed264c0e882dab0e322184c314dcf. Cited by: 0.

11. X. Chen, A. Ghadirzadeh, M. Björkman, and P. Jensfelt. Meta-learning for multi-objective reinforcement learning. *CoRR*, abs/1811.03376, 2018. URL http://arxiv.org/abs/1811.03376.

12. M. P. Deisenroth, G. Neumann, and J. Peters. 2013.

13. Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 1). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1, 2022.

14. U. Department of Business Energy and Industrial Strategy. Greenhouse gas reporting: Conversion factors 2020, Jul 2020. URL https://www.gov.uk/government/publications/greenhouse-gas-reporting-conversion-factors-2020.

15. U. Department of Transport. Transport and environment statistics: Autumn 2021, May 2021. URL https://www.gov.uk/government/statistics/transport-and-environment-statistics-autumn-2021/transport-and-environment-statistics-autumn-2021.

16. U. Environmental Protection Agency. Greenhouse gas equivalencies calculator, 2023. URL https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator.

17. F. Felten and L. N. Alegre. Morl-baselines: Multi-objective reinforcement learning algorithms implementations. https://github.com/LucasAlegre/morl-baselines, 2022.

18. Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 197–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

19. T. Gillespie, W. Paterson, and M. Sayers. Guidelines for conducting and calibrating road roughness measurements. *World Bank technical paper*, 1986. URL http://documents.worldbank.org/curated/en/851131468160775725/Guidelines-for-conducting-and-calibrating-road-roughness-measurements.

20. J. Guan, X. Yang, L. You, L. Ding, and X. Cheng. Multi-objective optimization for sustainable road network maintenance under traffic equilibrium: Incorporating costs and environmental impacts. *Journal of Cleaner Production*, 334:130103, 2022. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2021.130103. URL https://www.sciencedirect.com/science/article/pii/S0959652621042694.

21. Y. Guo, A. Zeman, and R. Li. A reinforcement learning approach to setting multi-objective goals for energy demand management. *Int. J. Agent Technol. Syst.*, 1(2):55–70, apr 2009. ISSN 1943-0744. doi: 10.4018/jats.2009040104. URL https://doi.org/10.4018/jats.2009040104.

22. Hamdi, S. P. Hadiwardoyo, A. G. Correia, and P. Pereira. Pavement maintenance optimization strategies for national road network in indonesia applying genetic algorithm. *Procedia Engineering*, 210:253–260, 2017. ISSN 1877-7058. doi: https://doi.org/10.1016/j.proeng.2017.11.074. URL https://www.sciencedirect.com/science/article/pii/S1877705817360691. Performance of materials and structures under extreme conditions.

23. C. Han, T. Ma, and S. Chen. Asphalt pavement maintenance plans intelligent decision model based on reinforcement learning algorithm. *Construction and Building Materials*, 299:124278, 2021. ISSN 0950-0618. doi: https://doi.org/10.1016/j.conbuildmat.2021.124278. URL https://www.sciencedirect.com/science/article/pii/S0950061821020377.

24. C. F. Hayes, R. Radulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, E. Howley, A. A. Irissappane,

P. Mannion, A. Nowé, G. de Oliveira Ramos, M. Restelli, P. Vamplew, and D. M. Roijers. A practical guide to multi-objective reinforcement learning and planning. *CoRR*, abs/2103.09568, 2021. URL https://arxiv.org/abs/2103.09568.

25. R. Issabekov and P. Vamplew. An empirical comparison of two common multiobjective reinforcement learning algorithms. In M. Thielscher and D. Zhang, editors, *AI 2012: Advances in Artificial Intelligence*, pages 626–636, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-35101-3.

26. R. Jiang, C. Wu, Y. Song, and P. Wu. Estimating carbon emissions from road use, maintenance and rehabilitation through a hybrid life cycle assessment approach – a case study. *Journal of Cleaner Production*, 277:123276, 2020. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2020.123276. URL https://www.sciencedirect.com/science/article/pii/S0959652620333217.

27. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

28. M. Latifi, F. G. Darvishvand, and O. Khandel. A deep reinforcement learning model for predictive maintenance planning of road assets: Integrating LCA and LCCA. *CoRR*, abs/2112.12589, 2021. URL https://arxiv.org/abs/2112.12589.

29. L. J. LeBlanc, E. K. Morlok, and W. P. Pierskalla. An efficient approach to solving the road network equilibrium traffic assignment problem. *Transportation Research*, 9(5):309–318, 1975. ISSN 0041-1647. doi: https://doi.org/10.1016/0041-1647(75)90030-1. URL https://www.sciencedirect.com/science/article/pii/0041164775900301.

30. D. J. Lizotte, M. Bowling, and S. A. Murphy. Linear fitted-q iteration with multiple reward functions. *Journal of Machine Learning Research*, 13(105):3253–3295, 2012. URL http://jmlr.org/papers/v13/lizotte12a.html.

31. J. Long, W. Szeto, Q. Shi, Z. Gao, and H.-J. Huang. A nonlinear equation system approach to the dynamic stochastic user equilibrium simultaneous route and departure time choice problem. *Transportmetrica A Transport Science*, 11(5):388–419, 2015. ISSN 2324-9935. doi: https://doi.org/10.1080/23249935.2014.1003112. URL https://www.sciencedirect.com/science/article/pii/S2324993522000173.

32. F.-M. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, and Y. Yu. A survey on model-based reinforcement learning, 2022.

33. L. Mandow and J.-L. Pérez-de-la Cruz. Pruning dominated policies in multiobjective pareto q-learning. In F. Herrera, S. Damas, R. Montes, S. Alonso, Ó. Cordón, A. González, and A. Troncoso, editors, *Advances in Artificial Intelligence*, pages 240–250, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00374-6.

34. L. e. I. Ministerie van Economische Zaken. Climate policy, Jan 2020. URL https://www.government.nl/topics/climate-change/climate-policy.

35. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236.

36. T. M. Moerland, J. Broekens, and C. M. Jonker. Model-based reinforcement learning: A survey. *CoRR*, abs/2006.16712, 2020. URL https://arxiv.org/abs/2006.16712.

37. K. V. Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *J. Mach. Learn. Res.*, 15:3483–3512, 2014.

38. H. Mossalam, Y. M. Assael, D. M. Roijers, and S. Whiteson. Multi-objective deep reinforcement learning. *CoRR*, abs/1610.02707, 2016. URL http://arxiv.org/abs/1610.02707.

39. S. Parisi. *Reinforcement Learning with Sparse and Multiple Rewards*. PhD thesis, Technische Universität, Darmstadt, January 2020. URL http://tuprints.ulb.tu-darmstadt.de/11372/.

40. S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta, and M. Restelli. Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2323–2330, 2014. doi: 10.1109/IJCNN.2014.6889738.

41. S. Parisi, M. Pirotta, and J. Peters. Manifold-based multi-objective policy search with sample reuse. *Neurocomputing*, 263:3–14, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.11.094. URL https://www.sciencedirect.com/science/article/pii/S0925231217310986. Multiobjective Reinforcement Learning: Theory and Applications.

42. S. S. Rabbanian, M. Nemati, and G. M. Knapp. A deep reinforcement learning approach for maintenance planning. page 932 – 937, 2021. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85120951185&partnerID=40&md5=6c294f9feeb35ed9757f87bf55d30521. Cited by: 1.

43. S. Ramachandran, C. Rajendran, A. Veeraragavan, and R. Ramya. A framework for maintenance management of pavement networks under performance-based multi-objective optimization. volume 2017-August, page 209 – 221, 2017. doi: 10.1061/9780784480922.019. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85048843142&doi=10.1061%2f9780784480922.019&partnerID=40&md5=5b29edb89d59fd5cce7c39910b7a3bdc. Cited by: 3.

44. M. Reymond and A. Nowe. 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*, May 2019. URL https://ala2019.vub.ac.be. null ; Conference date: 13-05-2019 Through 14-05-2019.

45. D. M. Roijers. *Multi-objective decision-theoretic planning*. PhD thesis, 2016.

46. D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *CoRR*, abs/1402.0590, 2014. URL http://arxiv.org/abs/1402.0590.

47. D. M. Roijers, S. Whiteson, and F. A. Oliehoek. Point-based planning for multi-objective pomdps. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1666–1672. AAAI Press, 2015. ISBN 9781577357384.

48. M. Ruiz-Montiel, L. Mandow, and J.-L. P. de-la Cruz. A temporal difference method for multi-objective reinforcement learning. *Neurocomputing*, 263:15–25, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.10.100. URL https://www.sciencedirect.com/

science/article/pii/S0925231217310998. Multiobjective Reinforcement Learning: Theory and Applications.

49. G. Rummery and M. Niranjan. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*, 11 1994.

50. M. Saifullah, C. Andriotis, K. Papakonstantinou, and S. Stoffels. Deep reinforcement learning-based life-cycle management of deteriorating transportation systems. *11th International Conference on Bridge Maintenance, Safety and Management (IABMAS)*, 2022. URL https://par.nsf.gov/biblio/10350545.

51. J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

52. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

53. J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

54. T. m. R. Statistics Netherlands. Transport and mobility 2016, Oct 2016. URL https://www.cbs.nl/en-gb/publication/2016/25/transport-and-mobility-2016.

55. R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, jul 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

56. R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press Ltd, 2018.

57. R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

58. B. o. P. R. U.S. Department of Commerce. Traffic assignment manual : for application with a large high speed computer, 1964.

59. P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry. On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In W. Wobcke and M. Zhang, editors, *AI 2008: Advances in Artificial Intelligence*, pages 372–378, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-89378-3.

60. S. Wang, Q. Meng, and H. Yang. Global optimization methods for the discrete network design problem. *Transportation Research Part B: Methodological*, 50:42–60, 2013.

61. J. G. Wardrop and J. I. Whitehead. Correspondence. some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(5):767–768, 1952. doi: 10.1680/ipeds.1952.11362. URL https://doi.org/10.1680/ipeds.1952.11362.

62. C. Watkins. Learning from delayed rewards. 01 1989.

63. S. Wei, Y. Bao, and H. Li. Optimal policy for structure maintenance: A deep reinforcement learning framework. *Structural Safety*, 83:101906, 2020. ISSN 0167-4730. doi: https://doi.org/10.1016/j.strusafe.2019.101906. URL https://www.sciencedirect.com/science/article/pii/S0167473019303704.

64. M. A. Wiering and E. de Jong. Computing optimal stationary policies for multi-objective markov decision processes. *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 158–165, 2007.

65. M. A. Wiering, M. Withagen, and M. M. Drugan. Model-based multi-objective reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 1–6, 2014. doi: 10.1109/ADPRL.2014.7010622.

66. s. World Bank Group. Hdm-4 road use costs model documentation, version 1.10 (english), 2018. URL http://documents.worldbank.org/curated/en/620891468337297899/HDM-4-road-use-costs-model-documentation-version-1-10.

67. P. Wu and X. Wang. Sustainable asset management–selecting optimal maintenance strategies based on multi-criteria decision making–a research agenda. 2016.

68. J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

69. T. Yamaguchi, S. Nagahama, Y. Ichikawa, and K. Takadama. Model-based multi-objective reinforcement learning with unknown weights. In S. Yamamoto and H. Mori, editors, *Human Interface and the Management of Information. Information in Intelligent Systems*, pages 311–321, Cham, 2019. Springer International Publishing. ISBN 978-3-030-22649-7.

70. C. Yang, R. Remenyte-Prescott, and J. Andrews. Pavement maintenance scheduling using genetic algorithms. *International Journal of Performability Engineering*, 11(2):135 – 152, 2015. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84930248381&partnerID=40&md5=4273dd757cefe52ddc1e150984b066da. Cited by: 27.

71. R. Yang, X. Sun, and K. Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *CoRR*, abs/1908.08342, 2019. URL http://arxiv.org/abs/1908.08342.

72. L. Yao, Q. Dong, J. Jiang, and F. Ni. Deep reinforcement learning for long-term pavement maintenance planning. *Computer-Aided Civil and Infrastructure Engineering*, 35(11):1230 – 1245, 2020. doi: 10.1111/mice.12558. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85084995854&doi=10.1111%2fmice.12558&partnerID=40&md5=1ce36615b5d8e2647eec45d042e899d8. Cited by: 56.

73. Q. Yuan, Y. Ye, Y. Tang, Y. Liu, and G. Strbac. A novel deep-learning based surrogate modeling of stochastic electric vehicle traffic user equilibrium in low-carbon electricity–transportation nexus. *Applied Energy*, 315, 2022. doi: 10.1016/j.apenergy.2022.118961. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85127530140&doi=10.1016%2fj.apenergy.2022.118961&partnerID=40&md5=dfd07394bab6b413087e6a043c79aae4. Cited by: 2.

74. I. Zaabar and K. Chatti. Calibration of hdm-4 models for estimating the effect of pavement

roughness on fuel consumption for u.s. conditions. *Transportation Research Record*, 2155 (1):105–116, 2010. doi: 10.3141/2155-12. URL https://doi.org/10.3141/2155-12.

75. L. M. Zintgraf, T. V. Kanters, D. M. Roijers, F. A. Oliehoek, and P. Beau. Quality assessment of morl algorithms: A utility-based approach. 2015.

76. E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3 (4):257–271, 1999. doi: 10.1109/4235.797969.

77. E. Zitzler, J. Knowles, and L. Thiele. *Quality Assessment of Pareto Set Approximations*, pages 373–404. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-88908-3. doi: 10.1007/978-3-540-88908-3_14. URL https://doi.org/10.1007/978-3-540-88908-3_14.

Appendix

A1 Road network environment dynamics

A1.1 Transition probabilities

Below, the transition probabilities for the actions retrieved by Saifullah et al. (2022) are presented.

$$P(s'|s, \{"Do\ nothing", "Inspect"\}) = \begin{bmatrix} 0.839 & 0.121 & 0.039 & 0. & 0. \\ 0. & 0.787 & 0.142 & 0.07 & 0. \\ 0. & 0. & 0.708 & 0.192 & 0.099 \\ 0. & 0. & 0. & 0.578 & 0.421 \\ 0. & 0. & 0. & 0. & 1. \end{bmatrix} \tag{.1}$$

$$P(s'|s, \{"Repair", "Repair + Inspect"\}) = \begin{bmatrix} 1. & 0. & 0. & 0. & 0. \\ 0.95 & 0.05 & 0. & 0. & 0. \\ 0.8 & 0.2 & 0. & 0. & 0. \\ 0.7 & 0.25 & 0.05 & 0. & 0. \\ 0.45 & 0.35 & 0.2 & 0. & 0. \end{bmatrix} \tag{.2}$$

$$P(s'|s, \{"Replace"\}) = \begin{bmatrix} 1. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. & 0. \end{bmatrix} \tag{.3}$$

A1.2 Observation probabilities

Below, the observation probabilities to update the belief when an inspection action is selected are presented. Transition probabilies were retrieved from Saifullah et al. (2022).

$$O(o|s', \{\text{"}Do\ nothing\text{"}, \text{"}Repair\text{"}, \text{"}Replace\text{"}\}) = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \tag{.4}$$

$$O(o|s', \{\text{"}Inspect\text{"}, \text{"}Repair + Inspect\text{"}\}) = \begin{bmatrix} 0.9 & 0.1 & 0. & 0. & 0. \\ 0.05 & 0.9 & 0.05 & 0. & 0. \\ 0. & 0.05 & 0.9 & 0.05 & 0. \\ 0. & 0. & 0.05 & 0.9 & 0.05 \\ 0. & 0. & 0. & 0.1 & 0.9 \end{bmatrix} \tag{.5}$$

A2 Modified-PPO experiments parameters

In Table A2.1 the parameters that were used for PPO single-objective executions are defined.

Parameter	Description	Value
processes	Number of parallel environment runners	8
n_epochs	Number of episodes per single-objective execution	15000
actor_architecture	Architecture of actor network	MLP
actor_hidden_layers	Hidden layers of actor network	[80, 80]
actor_activation_fn	Activation function after every hidden layer of the actor network	relu
lr_act	Initial actor learning rate	1.00E-03
lr_act_min	Minimum actor learning rate	1.00E-06
optimizer_act	Actor optimizer	Adam
critic_architecture	Architecture of critic network	MLP
critic_hidden_layers	Hidden layers of critic network	[80, 80]
critic_activation_fn	Activation function after every hidden layer of the critic network	relu
lr_crit	Initial critic learning rate	1.00E-02
lr_crit_min	Minimum critic learning rate	1.00E-05
optimizer_crit	Critic optimizer	Adam
lr_decay_step	Decay step for the learning rate of actor and critic networks	200
lr_decay_episode_perc	Percentage of episodes in which the learning rate will reach its min value	0.8
batch_size	Number of samples used in every training iteration	40
train_iters	Number of training iterations per episode	50
clip_range	PPO clip	0.2
normalize_advantage	Standard normalization of advantages	TRUE
ent_coef	Entropy coefficient for actor loss	0.03
vf_coef	Value coefficient for actor loss	0.01
target_kl	Target KL coefficient for training pruning	0.025
lam	GAE λ	0.95

Table A2.1: Modified PPO parameters used for the experiments.

A3 MORL Coverage sets

The CS from all MORL algorithms are presented in the following figures. The best experiment is shown for each algorithm

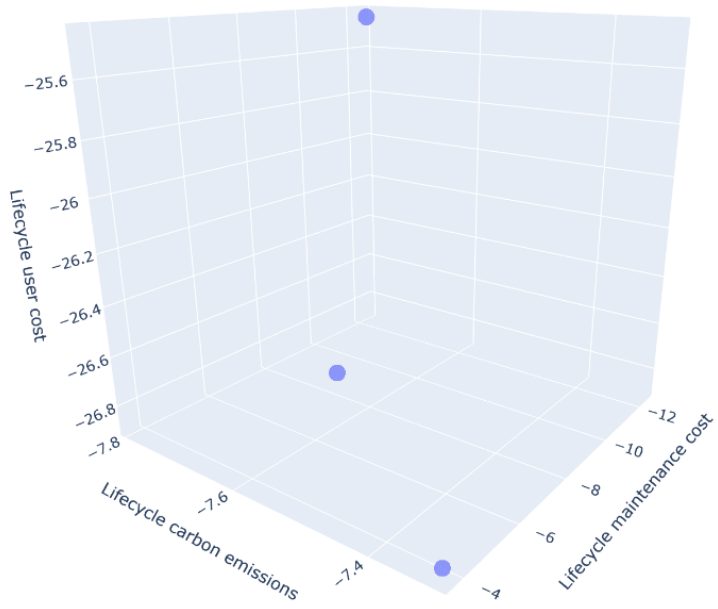


Figure A3.1: CS generated using the multi-objective CBM.

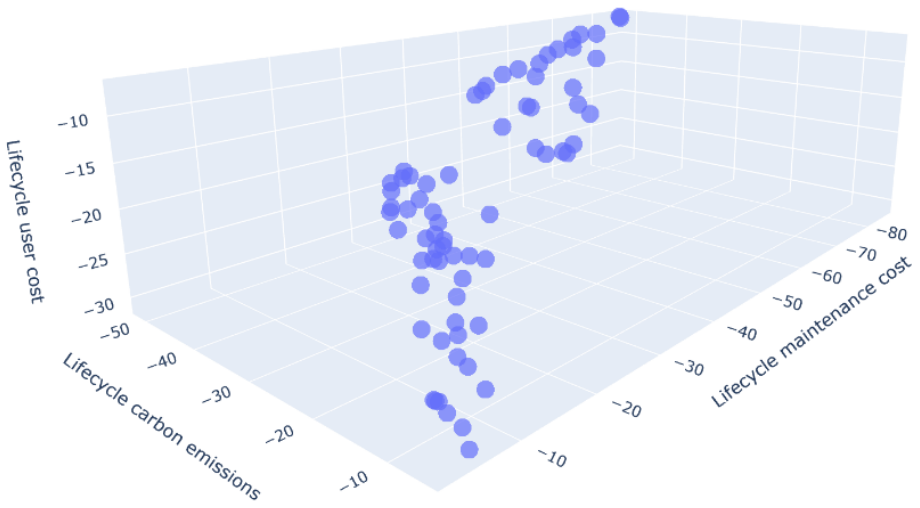


Figure A3.2: CS generated using DOL.

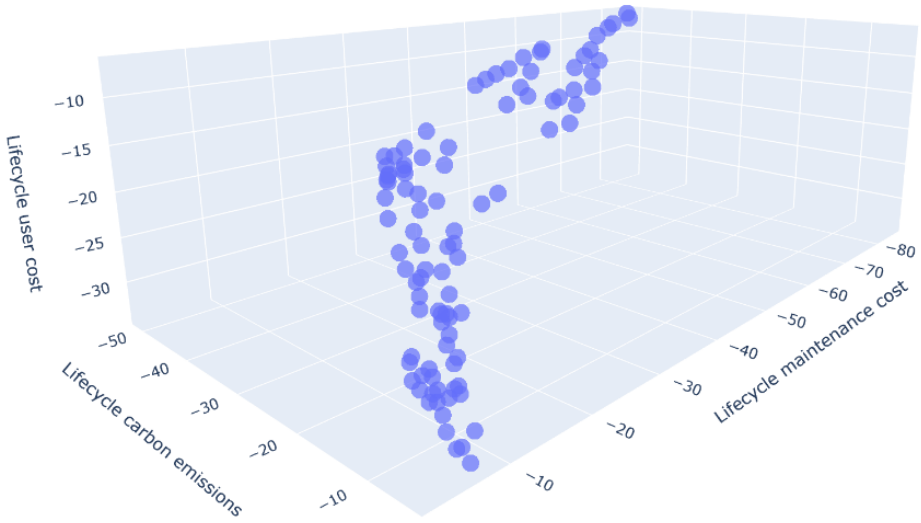


Figure A3.3: CS generated using DOL-PR.

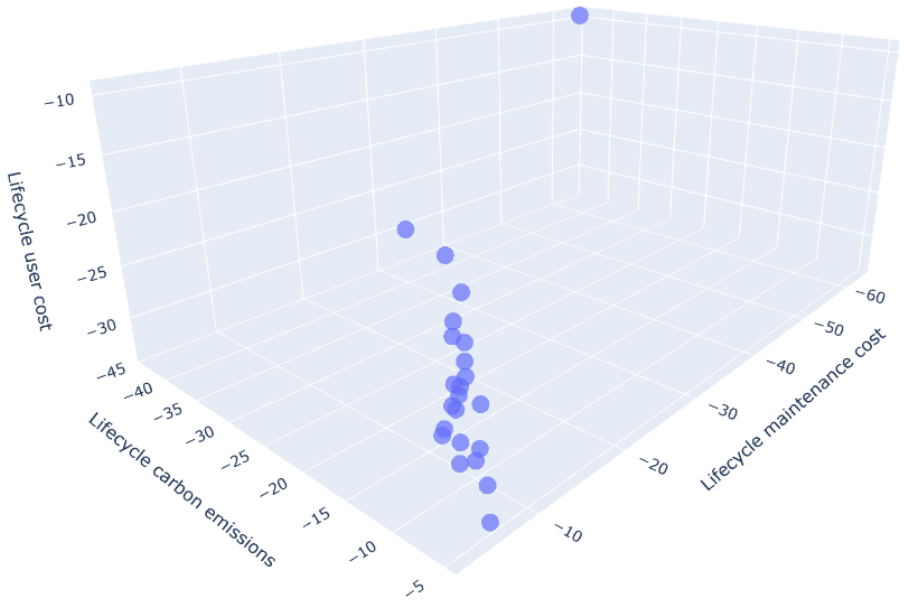


Figure A3.4: CS generated using RA.

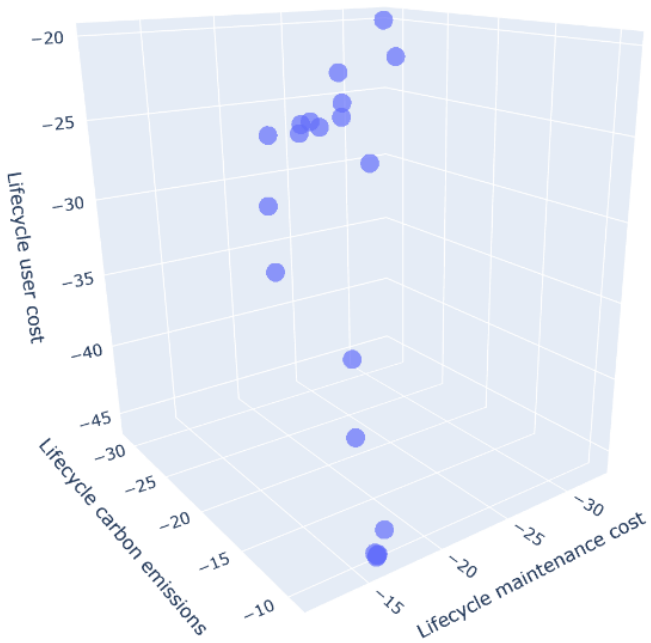


Figure A3.5: CS generated using PFA.