Deconstructing Reinforcement Learning Benchmarks: Revealing The Objectives

Sofyan Ajridi[a], **Willem Röpke**[a], **Ann Nowé**[a] **and Roxana Rădulescu**[a;*]
[a]Artificial Intelligence Lab, Vrije Universiteit Brussel, Belgium

Abstract. Real-world problems often involve multiple decision-makers as well as multiple, potentially conflicting, objectives. It is crucial to ensure that the methods developed for these settings capture and are able to handle these aspects. To this end, we first provide a novel benchmark for multi-objective reinforcement learning, in both single and multi-agent settings, starting from the DeepDrive Zero self-driving car simulator. Secondly, we construct and assess the performance of multi-objective versions of actor-critic reinforcement learning approaches in MO-DeepDrive Zero, under both linear and non-linear utility functions, against their single-objective counterparts. Additionally, we benchmark the performance of a multi-policy approach, Pareto Conditioned Networks, in the unknown utility scenario. We emphasise and demonstrate the importance of taking a multi-objective perspective for the quality of the learnt behaviour, when the setting warrants it.

1 Introduction

Upon examining real-world decision problems, we recognise that they often represent multi-objective problems, where multiple, often conflicting, criteria need to be considered. For example, due to cloud computing, big data and an increase in internet users across the globe, more data storage and processing capabilities are needed. Data centres meet this increase in demand by investing in faster hardware, but at the same time aim to reduce the energy required due to the increase in energy prices [10]. This necessitates a trade-off between two conflicting objectives: performance and energy consumption [26].

Nonetheless, when looking at the reinforcement learning literature, we often see such real-world problems modelled using a scalar reward signal constructed using an ad-hoc linear combination of the objectives and engineered to output desirable behaviour. As Hayes et al. [7] argue, this approach comes with severe limitations, including the potential for suboptimal trade-offs as well as impacting the explainability and trustworthiness of the system, since we can no longer relate the learned policy to the individual impact on each considered objective. In this work, we focus explicitly on the multi-objective nature of decision problems and take a utility-based approach, considering the user's utility as the main optimisation goal [7].

The multi-objective reinforcement learning field, and in particular its extension to multiple agents, significantly lacks benchmarks based on realistic settings [17]. As such, the first contribution of this work is deconstructing an existing single-objective multi- and single-agent self-driving car simulator and transforming it into a novel multi-objective benchmark, MO-DeepDrive Zero. While the original simulator considers a single scalar reward, a careful inspection of the reward function reveals that it is constructed as a linear combination of multiple objectives. The second contribution of this paper is extending a single-objective reinforcement learning algorithm, namely, Advantage-Actor Critic, that can tackle the modified self-driving car problem both for multi- and single-agent settings even when the utility function of the decision maker (i.e. the driver) is non-linear. As most single-objective reinforcement learning algorithms make an assumption of linearity, allowing for such non-linear utility functions presents additional challenges.

We compare these modified algorithms with their original counterpart to showcase the learning difference when considering the potential non-linear nature of utilities. For the multi-agent setting, we make use of independent learners. Our findings show a significant difference in learning performance and stability between both variations in the single-agent multi-objective setting.

2 Background

We formally introduce the relevant multi-objective reinforcement learning concepts and the utility-based approach we follow in this work.

2.1 Multi-Objective Reinforcement Learning

Single-agent decision problems where an agent aims to optimise over different objectives can be modelled as a multi-objective Markov decision process (MOMDP) [7]. Formally, an MOMDP is a tuple $(\mathcal{S}, \mathcal{A}, T, \gamma, \mathcal{R})$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $T : S \times \mathcal{A} \times S \to [0, 1]$ the transition function, $\gamma \in [0, 1)$ the discount factor, and $\mathcal{R} : S \times \mathcal{A} \times S \to \mathbb{R}^d$, the d-dimensional reward function with $d \geq 2$.

At each timestep t, the agent in state $s_t \in \mathcal{S}$ picks an action $a_t \in \mathcal{A}$ and receives the next state $s_{t+1} \in \mathcal{S}$ as well as a vector-valued reward $\mathbf{r}_t \in \mathcal{R}$. As in a single-objective MDP, the agent aims to learn a policy $\pi \in \Pi$ that maximises its cumulative discounted reward.

In a single-objective MDP, state value functions, V^π, i.e. the expected cumulative discounted reward, induce a partial ordering over policies. For a given state, the ordering is complete: $\pi \geq \pi' \Leftrightarrow \forall s, V^\pi(s) \geq V^{\pi'}(s)$. The value function of the optimal policy can subsequently be defined as $V^*(s) = \max_\pi V^\pi(s)$, for all states s

* Corresponding Author. Email: roxana.radulescu@vub.be.

[24]. In MOMDPs, we can define a vectorial value function analogously,

$$\mathbf{V}^{\pi} = \mathbb{E}_{\pi}\left[\sum_{t}^{\infty} \gamma^{t} \mathbf{r}_{t} | \pi, \mu_{0}\right]. \tag{2.1}$$

As the value function is vectorial, it no longer allows for a total ordering over policies, even given a state, and necessitates an agent to learn a set of solutions that are potentially optimal.

2.2 Utility-Based Approach

In this work, we take a utility-based approach [7] which assumes the existence of a (potentially unknown) utility function, that reflects the user's preferences over the considered objectives and can be used to map the vectorial return to a scalar value: $u : \mathbb{R}^d \rightarrow \mathbb{R}$.

2.2.1 Optimisation Criteria

An important complication of the utility-based approach is how to optimise for a given utility function. Specifically, given a utility function u it is possible that the decision maker aims to optimise the expected utility of each individual policy execution, leading to the Expected Scalarised Returns (ESR) criterion, formally defined below.

$$V_{u}^{\pi} = \mathbb{E}\left[u\left(\sum_{i=0}^{\infty} \gamma^{i} \mathbf{r}_{i}\right) | \pi, s_{0}\right] \tag{2.2}$$

On the other hand, it is possible that a decision-maker evaluates the utility of a policy based on its expected returns. This is known as the Scalarised Expected Returns (SER) criterion and is defined below.

$$V_{u}^{\pi} = u\left(\mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^{i} \mathbf{r}_{i} | \pi, s_{0}\right]\right) \tag{2.3}$$

Under linear utility functions, ESR and SER are equivalent due to the linearity of expectation. When allowing non-linear utility functions, however, the distinction becomes important and has been shown to lead to different optimal policies and even appropriate solution concepts. In this work, we mainly focus on the SER criterion. We note however that the novel benchmark described in Section 3 does not enforce either of these criteria, or indeed the utility-based approach and is therefore applicable for a range of different interpretations.

2.2.2 Solutions

The solutions we learn target two distinct settings. First, in the unknown utility function scenario, the utility function of the user is not known during the planning or learning phase. As such, a solution set must be computed which contains all policies that are potentially optimal for the decision maker. When assuming monotonically increasing utility functions, i.e. the decision maker prefers more of each objective given all else equal, a natural and well-studied solution set for MOMDPs is the Pareto front. The Pareto front leverages the Pareto dominance relation to establish a partial ordering over policies. Concretely, we say a policy π Pareto dominates another policy π' when $\forall i : \mathbf{V}_{i}^{\pi} \geq \mathbf{V}_{i}^{\pi'} \wedge \exists i : \mathbf{V}_{i}^{\pi} > \mathbf{V}_{i}^{\pi'}$. The Pareto front then contains all policies for which no other policy exists which Pareto dominates it. We note that other solution sets exist with various theoretical and practical motivations (i.a., [3, 8, 21]).

The second setting we target is the known utility function scenario. Here, we assume that explicit knowledge of the utility function is available and as such we may optimise it directly. When the utility function is linear, the problem can be shown to reduce to the regular reinforcement learning objective on the weighted sum. For non-linear utility functions however this does not hold and one needs additional techniques to show convergence to (local) optima [19].

2.3 Multi-objective multi-agent reinforcement learning

Finally, we introduce multi-objective multi-agent reinforcement learning. To model such settings, we can extend the definition of MOMDPs to allow for multiple agents. Concretely, we consider a tuple $(\mathcal{S}, \mathcal{A}, T, \gamma, \mathcal{R})$ with $n \geq 2$ agents and $d \geq 2$ objectives, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of joint actions (with $\mathcal{A} = A_1 \times ... \times A_n$ with $A_i =$ set of actions for agent i), T is the transition function, γ is the discount factor, and $\mathcal{R} = R_1 \times ... \times R_n$ are the environment reward functions (with $R_i : S \times \mathcal{A} \times S \rightarrow \mathbb{R}^d$ is the d-dimensional reward function of agent i). Such settings are known in game-theoretic terms as multi-objective stochastic games.

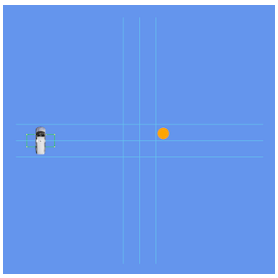
Analogous to single-agent MOMDPs, at each timestep t each agent in state s_t selects an action and obtains the next state s_{t+1} and vectorial reward $\mathbf{r}_t$ from the environment. The vectorial reward associated with each agent may differ and is also influenced by the actions taken by the other agents within the environment. The goal of each agent is to find an optimal policy π_i^*, which maximises its cumulative discounted reward. Depending on whether the environment is cooperative or competitive in nature, agents may choose to coordinate joint actions to ensure higher rewards.

In this work, we focus on the known utility scenario in our multi-agent contributions. In this setting, a popular approach for agents to learn a policy is to enable individual agents to learn using a standard single-agent algorithm and ignore the presence of other agents, referred to as independent learners. While these learning dynamics result in a non-stationary system from the individual perspective of each agent, thereby breaking known convergence and optimality guarantees, it has empirically been demonstrated to work reasonably well [30].

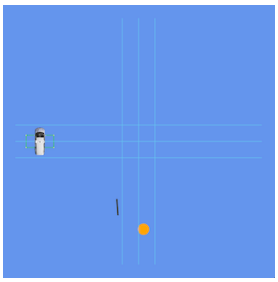
3 Multi-Objective DeepDrive Zero

The domain of MORL is a recent field of study compared to its single-objective counterpart. As such, the accessibility and variety of benchmarks is limited. Modifying existing established single-objective benchmarks to return a vectorial reward is not straightforward as the underlying multi-objective nature needs to be studied and justified concretely. Since the release of MO-Gymnasium [4] at the end of 2022, there has been a standardised API and centralised repository for well-known benchmarks in MORL literature. Environments such as Deep-Sea-Treasure [29] and Minecart [1] have been prevalent due to their longevity in multi-objective research [27, 19] and clear multi-objective nature. As such, these environments have been the go-to benchmarks for evaluating MORL algorithms. However, the drawback of these environments is that they lack direct relevance to a real-life setting.

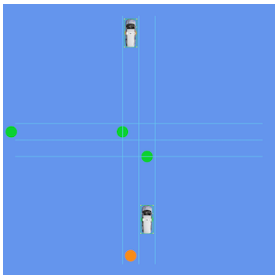
DeepDrive [14] is an open-source self-driving simulator built and rendered in Unreal Engine. It exposes a Python interface with the standard OpenAI Gym API [5] to enable its use for reinforcement learning research. This benchmark can be used as a first step in the



(a) OneWaypoint



(b) StaticObstacle



(c) Intersection

Figure 1: Three environment variations we consider from the deepdrive-zero benchmark. Orange circle represents the end destination. Lines in 1a and 1b should be ignored.

self-driving process by testing the algorithm in the simulator before applying it to a real-world vehicle. The DeepDrive simulator needs substantial resources both computationally and graphically. As such, a lightweight alternative was developed called DeepDrive Zero [15] that can be used to evaluate RL algorithms before using them in the complete DeepDrive benchmark. Unfortunately, to the best of our knowledge, the development for both the DeepDrive and DeepDrive Zero benchmarks has been abandoned. There has been no update since June 2020 for the DeepDrive benchmark and November 2021 for the DeepDrive Zero benchmark[1]. However, the DeepDrive Zero benchmark is still a very interesting simulator for the development and introduction to the field of single-agent, multi-agent, single-objective and multi-objective RL. First, the benchmark has three different environment variations, two single-agent environments and one multi-agent environment, all based on engaging self-driving scenarios. Secondly, depending on the environment variation, the actual problem to solve is very complex due to the large state and action space, For example, the most complicated environment, the intersection environment, has 10^{27} possible actions at every timestep [16]. Lastly, the benchmark has an explicit underlying multi-objective nature that can be explored and used as a benchmark for multi-objective decision problems. Below, we briefly discuss each original environment and subsequently describe the modifications that were made to enable their use in MORL research.

3.1 *OneWayPoint environment*

The OneWayPoint environment is the simplest environment variation in the benchmark and shown in Figure 1a. In this scenario, a single vehicle must drive to the end destination, highlighted by an orange circle. The placement of the orange circle is random and changes every episode thereby ensuring that the vehicle does not learn a fixed route but rather learns to dynamically drive a route using its observations.

[1] See the corresponding GitHub repositories: https://github.com/deepdrive/deepdrive and https://github.com/deepdrive/deepdrive-zero

3.2 *StaticObstacle environment*

The StaticObstacle environment is the second most challenging environment. This scenario includes a static obstacle which is represented as a bike. Naturally, the goal is for the agent to reach its destination while avoiding a collision with the bike. Both the placement of the obstacle and the end destination are random.

3.3 *Intersection environment*

Finally, the Intersection environment is the most difficult environment and introduces an additional agent for more realistic settings. This environment represents the unprotected left turn scenario, one of the most difficult scenarios for both autonomous vehicles and human drivers [16, 6]. In this scenario, two vehicles need to cross an intersection to reach two different destinations without colliding and staying in the correct lane. Vehicle one (top vehicle in Figure 1c) has the easiest trajectory as it only needs to go straight until it reaches the orange circle. Vehicle two's trajectory is more difficult as it needs to go straight, turn to the left and reach the green circle at the left side of the intersection.

3.4 *Action space*

The action space of the environment can be continuous or discrete. However, the discrete action space is an abstraction layer, and the actual discrete action gets converted back to a continuous action. The continuous action space is a three-dimensional vector of variables describing the steer, acceleration and brake. Each variable should be a continuous value between -1 and +1. Steer is the heading angle of the vehicle where -1 is $-360°$, 0 is $0°$ and +1 is $360°$. Acceleration is the acceleration value in m/s^2 and brake is the braking force where -1 is 0g and +1 1g. For the discrete action space, the values lie between 0 and 21, where 0 is 'idle', and 21 is 'large steer right and increase speed'. All other values are a combination of increasing, maintaining or decreasing speed and steering left or right.

3.5 *Observation space*

Depending on the environment variation, the observation space is either a 15-dimensional, 19-dimensional, or 29-dimensional discrete box. The observation includes the previous and current position, steering, acceleration and brake values. It also includes the distance between the vehicle and the lane for the Intersection environment and the distance until the end destination. The observation also includes the positioning of an obstacle or other vehicle in combination with its speed and angle, which is motivated by the fact that in real life, self-driving vehicles make use of their lidar or radar to determine the positioning, speed and angle of other objects and vehicles in their surrounding.

3.6 *Modifications*

Before adapting DeepDrive Zero to the multi-objective setting, we had to make several changes to the existing codebase. First, the original benchmark fails during the initialisation of an environment due to bugs in the initialisation of the observation space. Fixing this issue enables the basic functionality of the environments. Second, the

StaticObstacle environment did not implement an actual collision detection mechanism to verify that the car and bicycle do not collide. Fortunately, the Intersection environment where collisions are also part of the scenario included a collision detection mechanism. Reusing and modifying this mechanism to detect the static obstacle was necessary to make this environment variation work.

The Gym API used by this environment saw a significant version update that introduces breaking changes to the API and is now called Gymnasium. Accordingly, the benchmark was modified to comply with the new API. One significant change concerns the initialisation of the environment. Previously, the environment was first initialised without defining the observation and action space. The observation and action space was later initialised after calling the `configure_env` function as the size depends on the user-configurable parameters. However, the new version of the API requires both spaces to be defined during initiation. As such, the environment configuration is now part of the `env.make` function as a parameter.

The Gymnasium API offers no support for multi-agent environments. As such, the multi-agent Intersection environment is represented as a single-agent environment where every other call to the `step` function alternates the current agent back and forth between vehicle one and vehicle two. Since the release of the PettingZoo API [25] and the RLlib MultiAgentEnv API [11], there have been two well-known APIs that offer multi-agent support. Thereby, we created two environment wrappers that comply with both without changing the underlying environment functionality. Both wrappers are based on the Parallel API, where return values and function calls use dictionaries where the keys are agents.

3.7 Revealing the objectives

Self-driving vehicles need to take a number of different and conflicting objectives into account at the same time. For example, they should bring passengers to their destinations as quick as possible while ensuring a comfortable trip and avoiding dangerous situations. When analysing the DeepDrive Zero benchmark, we see these criteria reflected in a linear scalarisation that is returned as the reward signal. Concretely, the reward is composed of the following objectives:

$$\begin{bmatrix} distance \\ destination \\ g-force \\ jerk \end{bmatrix} \quad \begin{bmatrix} distance \\ destination \\ g-force \\ collision \\ jerk \end{bmatrix} \quad \begin{bmatrix} distance \\ destination \\ g-force \\ collision \\ jerk \\ lane\ violation \end{bmatrix}$$

OneWayPoint StaticObstacle Intersection

Figure 2: Objectives for each environment variation in the deepdrive benchmark

First, the distance objective measures how close the vehicle is to the end destination and therefore serves as an incentive to plan an efficient route. Second, the destination objective either returns 1 if the destination was reached or 0 when this is not the case. Notably, there is a correlation between the distance and destination objective as a higher distance value increases the likelihood of reaching the destination. Next, the g-force and jerk objectives return their respective values and can be used to impose preferences for smoother driving styles. Finally, avoiding collision with other objects or vehicles and lane violations are also presented as distinct objectives to the agent. By exposing these objectives now directly to the learning agent, we hope to inspire research that can find optimal trade-offs in the challenging area of self-driving. Our modified benchmark can be found at https://github.com/sofyanajridi/mo-deepdrive.

4 Methods

We present an adaptation of a well-known single-objective algorithm to function with multi-objectives and a known, potentially non-linear, utility function. Both this algorithm, as well as the multi-policy algorithm we employ in evaluating our novel environment, are optimising for the SER criterion.

4.1 Accrued reward

A central concept in many multi-objective algorithms, including the ones we employ in this work, is the accrued reward. The accrued reward at any point is the cumulative discounted reward that the agent has accrued until that point, i.e. $R_t^- = \sum_{t=0}^{t-1} \gamma^t R_t$. It has been demonstrated that conditioning on an augmented state $\hat{S}_t = (S_t, R_t^-)$ is necessary when optimising non-linear utility functions of the reward as the optimal action depends on the reward already accrued up until that point. Conditioning a policy on the augmented state is analogous to conditioning on the full history, as is common in for example partially observable MDPs. However, while the history grows linearly in the episode length, the accrued reward can be described in the number of objectives d, making it an attractive substitute for learning algorithms. It is interesting to note that accrued reward is not necessary when tackling single-objective MDPs because of the Markov property and stationarity of the environment [13]. We provide an example below to illustrate the computation of the accrued reward and its impact on the decision-making of an agent [19].

Let us assume we are in a MOMDP with two objectives ($d = 2$), $\gamma = 1$ and a non-linear utility function $u(\mathbf{r}) = r_1^2 + r_2^2$. Suppose we are in state S_1 and do not take into account the previous reward $(1, 5)$. The best action to take here is the one that leads us to state S_2. As we have $5^2 + 1^2 = 26$ compared to $3^2 + 4^2 = 25$ when being in state S_3. If we do, however, take into account our accrued rewards, going to state S_3 leads to a higher utility as $u = (1 + 5)^2 + (5 + 1)^2 = 72$ for state S_2 and $u = (1 + 3)^2 + (5 + 4)^2 = 97$ for S_3.

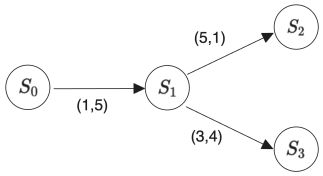


Figure 3: Example of possible state transitions in a MOMDP.

4.2 Multi-Objective Advantage-Actor Critic

We extend the well-known Advantage Actor-Critic (A2C) [12] to handle multiple objectives under the SER criterion. Previous work has explored an extension to the ESR criterion [19]. A2C is a policy gradient algorithm which uses a learnt estimator to approximate the

advantage as a baseline and has been demonstrated to achieve high performance in challenging environments. In addition, it is a classic baseline in single-objective settings, making it an excellent algorithm to evaluate in the multi-objective setting.

To extend A2C to the multi-objective setting with a known utility function, we adapt both the critic and actor. Both models are trained on the augmented state which is the state concatenated with the accrued reward. First, the critic is parametrised by θ and trained to predict the state value function which maximises the given utility function. Concretely,

$$\mathcal{L}(\theta^q) = \mathbb{E}[(y - Q(s, a; \theta^q))^2] \tag{4.1}$$

where $y = r + \gamma \tilde{\max}_{a'} Q(s_{t+1}, a'; \theta)$ and the maximum operator is adapted to take the action which maximises the total utility when combining the accrued reward with the state-action estimate. Second, the actor is updated to follow the policy gradient together with the advantage calculated as the difference in utility between the true action that was taken and the optimal action, defined as

$$U_t = u\left(R^-_{t+1} + \hat{v}\left(S_{t+1}\right)\right) - u\left(R^-_t + \gamma^t \hat{v}\left(S_t\right)\right). \tag{4.2}$$

Therefore, the complete update becomes,

$$\theta^\pi_{t+1} = \theta^\pi_t + \alpha U_t \nabla \ln \pi(A_t|S_t; \theta^\pi_t). \tag{4.3}$$

It is interesting to note that while this loss makes intuitive sense, and empirically can be shown to achieve good performance, a policy gradient update can be derived for non-linear utility functions which takes a different form [19]. We intend to study this in future work.

5 Experiments

We divide our experiments into three main categories:

- single-policy algorithms in a single-agent multi-objective setting
- single-policy algorithms in a multi-agent multi-objective setting
- multi-policy algorithm in a single-agent multi-objective setting

For each category, we perform experiments on the relevant environments. Specifically, for the single-agent multi-objective setting we run experiments on the OneWayPoint and StaticObstacle environment while in the multi-agent multi-objective setting, we run experiments on the Intersection environment.

Finally, in the known utility scenario, we evaluate our A2C adaptation with both a linear and non-linear utility function to compare it with the original A2C. The exact utility functions used and their explanation can be found in Appendix D. For the linear case, we expect similar performance with both algorithms, as the setting can be shown to be equivalent to the scalarised single-objective problem. However, taking a MORL approach helps with the explainability of the agent's decision-making as the effect of different policies on each objective can be measured individually For the non-linear case, however, we hypothesise that a multi-objective approach offers a benefit. In this case, we modify this single-objective variant to apply timestep scalarisation on the received vectorial reward. We run each experiment five times (four times for the multi-agent setting) and average their results.

Additional information on the hyperparameters (Apendix A) and exact utility functions (Appendix D) we use in our experiments can be found in the appendix.

5.1 Results

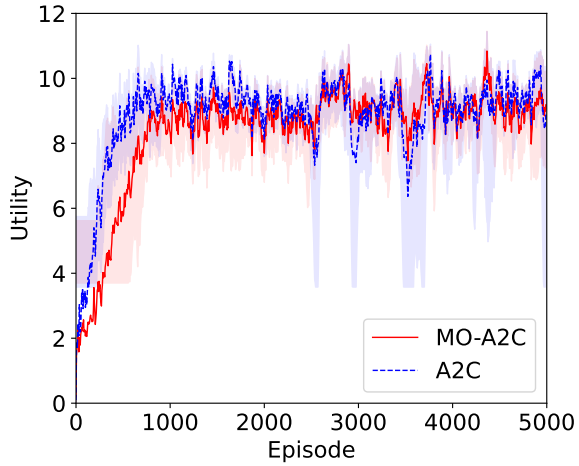
Single-Policy Single-Agent OneWayPoint Looking at Figure 4, we observe for the linear utility function that both MO-A2C and A2C are able to solve the OneWayPoint environment. As expected, the performance is similar for both algorithms which can be attributed to the fact that a multi-objective decision problem can be reduced to a single-objective decision problem if the utility is linear. For the non-linear utility function, we find that MO-A2C reaches a higher and more consistent utility compared to A2C. This happens because we penalise the vehicle more on higher g-force and jerk values to incentivise more comfortable and smoother driving behaviour when the vehicle is making positive progress. This happens whenever the distance objective is higher than 2. Because A2C is not able to handle this non-linearity, it keeps trying to minimise its distance objective to stay under 2, which causes it to perform worse than MO-A2C.

Single-Policy Single-Agent StaticObstacle Figure 5 shows the results of A2C and MO-A2C on the StaticObstacle environment. This is a more challenging environment with an obstacle in the vehicle's path and introduces an extra objective, namely a collision penalty. We can again see the same pattern for the linear utility function. Both MO-A2C and A2C have nearly the same exact utility. The non-linear utility function now not only penalises aggressive driving behaviour when getting close to the destination but also penalises the vehicle when it is staying stationary behind the obstacle. The MO-A2C agent is learning consistently in this setting due to being able to deal with this non-linearity. On the other hand, the A2C agent is not able to converge in this setting, compared to the previous environment. As the non-linear utility function penalises an agent for staying stationary, as well as on g-force and jerk when the vehicle is getting closer to the end destination, the A2C agent learns to always drive in the opposite direction of the destination and obstacle and makes constant negative progress.

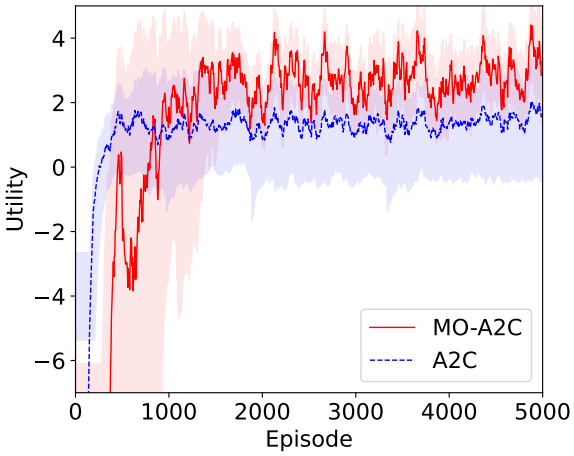
Single-Policy Multi-Agent Intersection We now look at how MO-A2C and A2C perform in the multi-agent Intersection environment. The environment has two agents: vehicle 1 and vehicle 2. As such, we use an independent learner for each of the vehicles. When looking at the results for both the linear and non-linear utility functions in Figure 6 (see Appendix B for complete results), it is apparent that both agents are not learning. The total utility for both mostly stays below zero. For the non-linear utility function, the independent MO-A2C agents are only "failing" better than the independent A2C agents.

We argue that for this problem, independent learners equipped with A2C are not powerful enough. This has been remarked in prior work as well [9]. To the extent of our knowledge, this is the only paper in RL literature that has used the DeepDrive environment as a benchmark for experiments. While their experiments tackle a variation on our setting, it is clear that DeepDrive presents difficult challenges. We note that a more thorough hyperparameter search may help in increasing the stability and performance. However, this implies that the approach is not robust and further highlights the need for additional algorithms in this challenging setting.

Multi-Policy Single-Agent OneWayPoint Finally, we execute the Pareto Conditioned Network (PCN) algorithm [18] on the

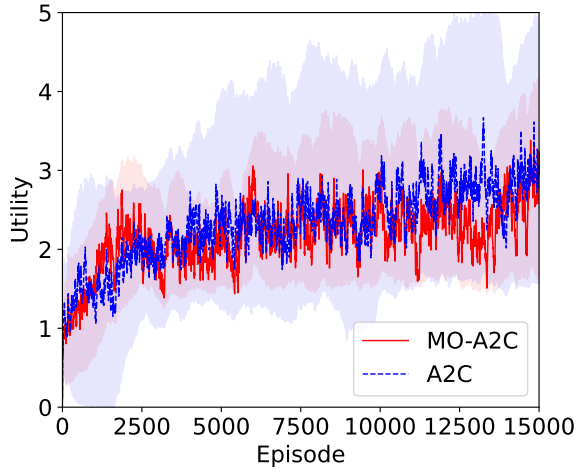


(a) Linear utility function

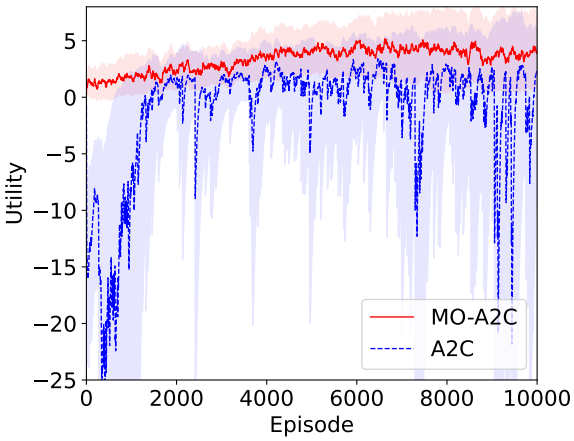


(b) Non-linear utility function

Figure 4: Average utility per episode for MO-A2C and A2C on the OneWayPoint environment

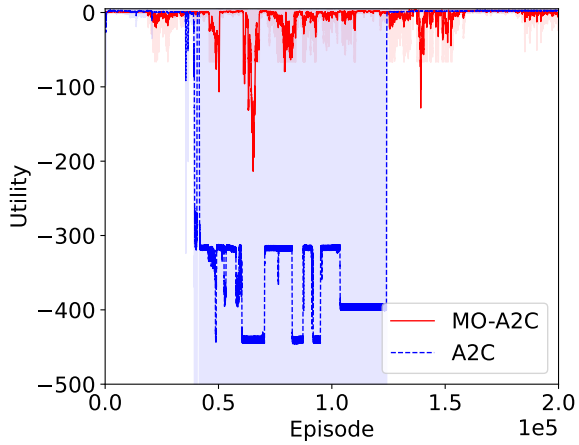


(a) Linear utility function

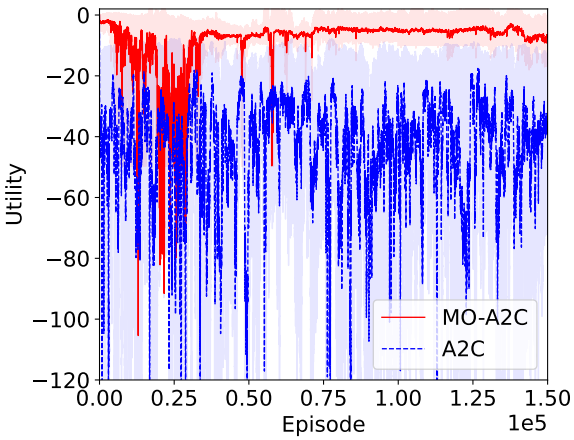


(b) Non-linear utility function

Figure 5: Average utility per episode for MO-A2C and A2C on the StaticObstacle environment

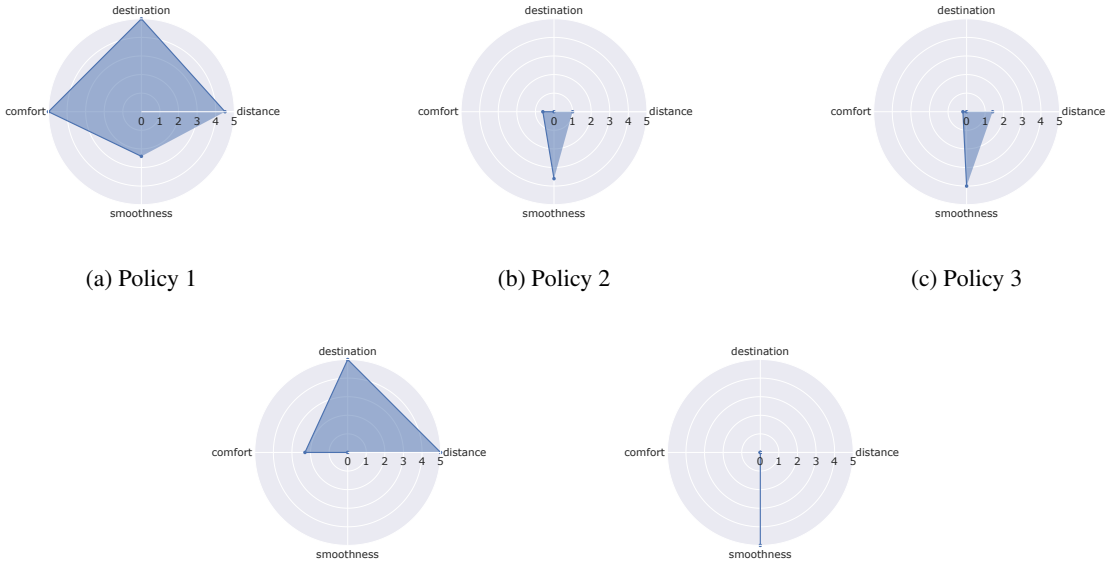


(a) Linear utility function



(b) Non-linear utility function

Figure 6: Average total utility per episode for MO-A2C and A2C on the Intersection environment



(a) Policy 1 (b) Policy 2 (c) Policy 3

(d) Policy 4 (e) Policy 5

Figure 7: The set of policies on the PF obtained using PCN on the OneWayPoint environment.

OneWayPoint environment to determine Pareto optimal policies. Figure 7 showcases the importance of the objectives for every policy on the PF. The smoothness objective represents the smoothness of the acceleration, which is represented using the jerk value. As the smoothness objective increases, the jerk decreases. The comfort objective looks at the g-forces, with low g-forces increasing the comfort of the ride. For example, if one prefers comfort and smoothness over distance, policy one may be more suitable than policy four. However, if a user aims to maximise distance, policy four is preferable. Interestingly, maximising smoothness altogether results in a policy where the vehicle roughly stays in the same place.

6 Related Work

In MORL literature, most proposed algorithms fall either under the known utility function scenario or the unknown utility function scenario. Under the known utility function scenario, algorithms have been proposed for non-linear utility functions with for example the Expected Utility Policy Gradient algorithm which explores the use of accrued reward in policy gradient algorithms [20]. More recently, Multi-Objective Categorical Actor-Critic was proposed for learning an optimal policy under ESR [19]. This algorithm combines an actor-critic approach with the distributional approach. Under the SER criterion, novel policy gradient and actor-critic algorithms have been proposed [2]. Finally, extensions to the popular single-objective algorithms DQN, PPO and A2C have been explored to optimise the non-linear generalised Gini social welfare function [23].

Under the unknown utility function scenario, the literature focuses on multi-policy algorithms that learn a set of policies which are optimal for some utility function in a particular class. One such set is the convex hull, which contains the optimal policies for decision-makers with linear utility functions [3]. Under SER, learning the Pareto front is a long-standing challenge that has been tackled by multiple algorithms [18, 28]. Under the ESR criterion, recent work has explored the use of stochastic dominance in defining novel solution sets [8, 22]. For a thorough overview of multi-objective approaches, we refer to [7].

Finally, research into multi-objective multi-agent settings has been scarce with most attempts focusing on a team-reward team-utility scenario where all agents share both a reward and utility function. For a complete overview, we refer to a recent survey on the topic [17].

7 Conclusion

In this work, we introduced a novel benchmark for MORL, MO-DeepDrive Zero, that simulates a self-driving car and offers three different scenarios including a multi-agent setting. In addition, we propose an extension to the advantage actor-critic algorithm such that it can optimise for a non-linear utility function in MORL settings. We evaluate our algorithm on all three scenarios and find that it outperforms a naive single-objective scalarised approach in single-agent settings. In the multi-agent setting, however, independent learners appear insufficient to learn the challenging task. This motivates further research in the area of multi-agent multi-objective learning. Finally, we evaluate the PCN algorithm on a single-agent scenario to determine the optimal policy under different trade-offs and find that a broad range of behaviour can be learned.

Acknowledgements

Willem Röpke and Roxana Rădulescu are supported by the Research Foundation – Flanders (FWO), grant numbers 1197622N and 1286223N. The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the FWO and the Flemish Government.

References

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

[2] Mridul Agarwal, Vaneet Aggarwal, and Tian Lan, 'Multi-objective reinforcement learning with non-linear scalarization', in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pp. 9–17. International Foundation for Autonomous Agents and Multiagent Systems, (2022).

[3] Lucas N Alegre, Ana LC Bazzan, Diederik M Roijers, Ann Nowé, and Bruno C da Silva, 'Sample-efficient multi-objective learning via generalized policy improvement prioritization', in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 2003–2012, (2023).

[4] Lucas N. Alegre, Florian Felten, El-Ghazali Talbi, Grégoire Danoy, Ann Nowé, Ana L. C. Bazzan, and Bruno C. da Silva, 'MO-Gym: A library of multi-objective reinforcement learning environments', in *Proceedings of the 34th Benelux Conference on Artificial Intelligence BNAIC/Benelearn 2022*, (2022).

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym, 2016.

[6] Jin Chen, Dihua Sun, and Min Zhao, 'Human-Like Control for Automated Vehicles and Avoiding "Vehicle Face-Off" in Unprotected Left Turn Scenarios', *IEEE Transactions on Intelligent Transportation Systems*, **24**(2), 1609–1618, (2023).

[7] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers, 'A practical guide to multi-objective reinforcement learning and planning', *Autonomous Agents and Multi-Agent Systems*, **36**(1), 26, (2022).

[8] Conor F Hayes, Timothy Verstraeten, Diederik M Roijers, Enda Howley, and Patrick Mannion, 'Expected scalarised returns dominance: a new solution concept for multi-objective decision making', *Neural Computing and Applications*, 1–21, (2022).

[9] Eshagh Kargar and Ville Kyrki, 'MACRPO: Multi-Agent Cooperative Recurrent Policy Optimization', *arXiv preprint arXiv:2109.00882*, (2021).

[10] Avita Katal, Susheela Dahiya, and Tanupriya Choudhury, 'Energy efficiency in cloud computing data centers: A survey on software technologies', *Cluster Computing*, **26**(3), 1845–1875, (2023).

[11] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica, 'RLlib: Abstractions for distributed reinforcement learning', in *Proceedings of the 35th International Conference on Machine Learning*, eds., Jennifer Dy and Andreas Krause, volume 80 of *Proceedings of Machine Learning Research*, pp. 3053–3062. PMLR, (2018).

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, 'Asynchronous methods for deep reinforcement learning', in *International conference on machine learning*, pp. 1928–1937. PMLR, (2016).

[13] Martin L Puterman, *Markov decision processes: discrete stochastic dynamic programming*, John Wiley & Sons, 2014.

[14] Craig Quiter. Deepdrive. https://github.com/deepdrive/deepdrive, 2017.

[15] Craig Quiter. Deepdrive Zero. https://doi.org/10.5281/zenodo.3871907, 2020.

[16] Craig Quiter. Smooth Operator. https://crizcraig.medium.com/smooth-operator-92c6c14862fb, 2020. [Online; posted 15-June-2020].

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

[18] Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé, 'Pareto conditioned networks', in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 1110–1118, (2022).

[19] Mathieu Reymond, Conor Hayes, Diederik Roijers, Denis Steckelmacher, and Ann Nowe, 'Actor-Critic Multi-Objective Reinforcement Learning for Non-Linear Utility Functions', *Autonomous Agents and Multi-Agent Systems*, **37**(2), 23, (2023).

[20] Diederik M Roijers, Denis Steckelmacher, and Ann Nowé, 'Multi-objective reinforcement learning for the expected utility of the return', in *Proceedings of the Adaptive and Learning Agents workshop at FAIM*, volume 2018, (2018).

[21] Diederik Marijn Roijers, Shimon Whiteson, and Frans A Oliehoek, 'Computing convex coverage sets for faster multi-objective coordination', *Journal of Artificial Intelligence Research*, **52**, 399–443, (2015).

[22] Willem Röpke, Conor F Hayes, Patrick Mannion, Enda Howley, Ann Nowé, and Diederik M Roijers, 'Distributional multi-objective decision making', *arXiv preprint arXiv:2305.05560*, (2023).

[23] Umer Siddique, Paul Weng, and Matthieu Zimmer, 'Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards', in *International Conference on Machine Learning*, pp. 8905–8915. PMLR, (2020).

[24] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning Series, The MIT Press, second edition edn., 2018.

[25] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al., 'Pettingzoo: Gym for multi-agent reinforcement learning', *Advances in Neural Information Processing Systems*, **34**, 15032–15043, (2021).

[26] Gerald Tesauro, Rajarshi Das, Hoi Chan, Jeffrey Kephart, David Levine, Freeman Rawson, and Charles Lefurgy, 'Managing power consumption and performance of computing systems using reinforcement learning', *Advances in neural information processing systems*, **20**, (2007).

[27] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker, 'Empirical evaluation methods for multiobjective reinforcement learning algorithms', *Machine Learning*, **84**(1), 51–80, (2011).

[28] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik, 'Prediction-guided multi-objective reinforcement learning for continuous robot control', in *Proceedings of the 37th International Conference on Machine Learning*, (2020).

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

[30] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu, 'The surprising effectiveness of PPO in cooperative multi-agent games', *Advances in Neural Information Processing Systems*, **35**, 24611–24624, (2022).

A Hyperparamters

Tables 1 and 2 specify the hyperparameters, as well as the actor and critic neural network architecture, for both A2C and MO-A2C on the different environment variations.

	OneWayPoint StaticObstacle	**Intersection**
Learning rate	1e-4	1e-4
Discount factor	0.99	0.99
Actor network	Linear(state dim, 64), ReLU(), Linear(64, 32), ReLU(), Linear(32, n_actions), Softmax()	Linear(state dim, 128), ReLU(), Linear(128, 64), ReLU(), Linear(64, n_actions), Softmax()
Critic network	Linear(state dim, 64), ReLU(), Linear(64, 32), ReLU(), Linear(32, 1)	Linear(state dim, 128), ReLU(), Linear(128, 64), ReLU(), Linear(64, 1)
Optimizer	Adam	Adam

Table 1: Hyperparameters for the regular A2C algorithm

	OneWayPoint StaticObstacle	**Intersection**
Learning rate	1e-4	1e-4
Discount factor	0.99	0.99
Actor network	Linear(state dim, 64), ReLU(), Linear(64, 32), ReLU(), Linear(32, n_actions), Softmax()	Linear(state dim, 128), ReLU(), Linear(128, 64), ReLU(), Linear(64, n_actions), Softmax()
Critic network	Linear(state dim, 64), ReLU(), Linear(64, 32), ReLU(), Linear(32, n_obj)	Linear(state dim, 128), ReLU(), Linear(128, 64), ReLU(), Linear(64, n_obj)
Optimizer	Adam	Adam

Table 2: Hyperparameters for the MO-A2C algorithm

B Intersection environment individual vehicle utilities

Figures 8 and 9 showcase the individual utility of each vehicle for both A2C and MO-A2C.

C Multi-Policy Single-Agent OneWayPoint

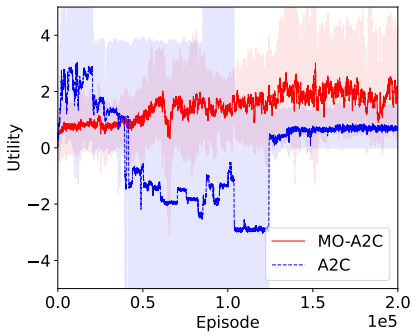
Figure 10 illustrates the wide variety of different policies found by the PCN algorithm on the OneWayPoint environment. Note, however, that the original problem has four objectives. As such, dimensionality reduction was applied to the objectives to visualise them in a two-dimensional plot.

D Utility functions

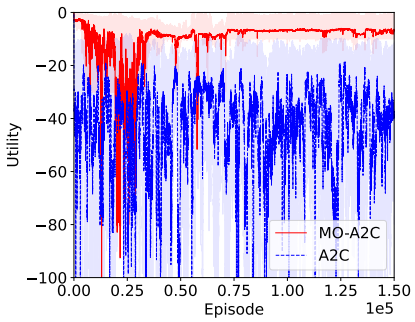
D.1 OneWayPoint Environment

We use the following linear utility function:

$$u = w_0 d + w_1 r - w_2 g - w_3 j$$



(a) Linear utility function



(b) Non-linear utility function

Figure 8: Vehicle 1 utility utility per episode for MO-A2C and A2C on the Intersection environment

where d, r, g and j are distance, destination, g-force and jerk respectively. The following weights are used: $w_0 = 0.50$, $w_1 = 1$, $w_2 = 0.03$ and $w_3 = 3.3 \times 10^{-5}$ As for the non-linear utility function:

$$u = \begin{cases} w_0 d + w_1 dr - w_2 g - w_3 j & \text{if } d > 2, \\ w_0 d + w_1 r - (w_2 g)^2 - (w_3 j)^2 & \text{otherwise.} \end{cases}$$

where $w_0 = 0.50$, $w_1 = 1$, $w_2 = 0.1$ and $w_3 = 0.1$. This non-linear preference models a situation where we penalise the vehicle less on comfortability and smoothness when the vehicle is making negative progress on reaching the end destination and more when the vehicle is following the correct path.

D.2 StaticObstacle Environment

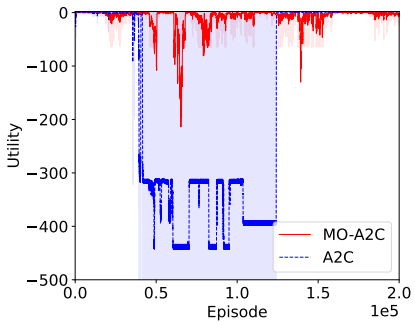
We use the following linear utility function:

$$u = w_0 d + w_1 r - w_2 c - w_3 g - w_4 j$$

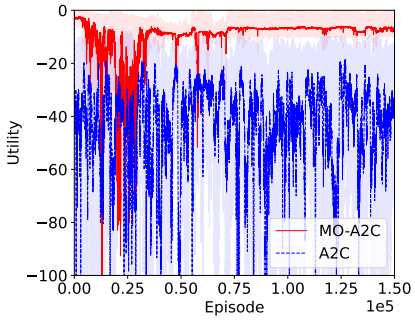
where c is collision. The following weights are used: $w_0 = 0.30$, $w_1 = 1$, $w_2 = 1$, $w_3 = 0.03$ and $w_4 = 3.3 \times 10^{-5}$. As for the non-linear utility function:

$$u = \begin{cases} p & \text{if } d = 0, \\ w_0 d + w_1 r - w_2 c - (w_3 g)^4 - (w_4 j)^4 & \text{if } d < 0, \\ w_0 d + w_1 r - w_2 c - 2 \cdot (w_3 g)^2 & \text{otherwise.} \end{cases}$$

where $w_0 = 0.50$, $w_1 = 1$, $w_2 = 4$, $w_3 = 0.03$, $w_4 = 3.3 \times 10^{-5}$ and $p = -10$. This non-linear utility function models a situation where we again penalise the vehicle less on comfortability and smoothness when the vehicle is making negative progress and more



(a) Linear utility function



(b) Non-linear utility function

Figure 9: Vehicle 2 utility utility per episode for MO-A2C and A2C on the Intersection environment

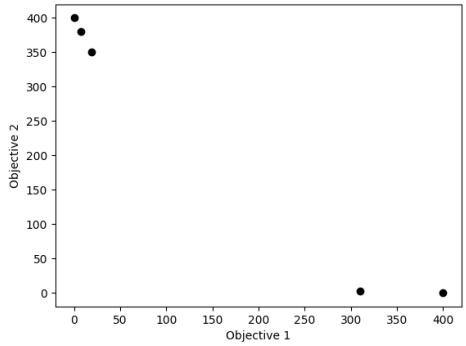


Figure 10: PCN

when it is following the correct path. However, we also penalise the vehicle when it stays still. As the StaticObstacle environment now has a bicycle it needs to avoid, we incentives the vehicle to find a path that avoids the obstacle totally.

D.3 Intersection Environment

We use the following linear utility function:

$$u = w_0 d + w_1 r - w_2 c - w_3 g - w_4 j - w_5 l$$

where l is lane violation. The following weights are used: $w_0 = 0.50$, $w_1 = 10$, $w_2 = 4$, $w_3 = 0.03$, $w_4 = 0.0006$, $w_5 = 0.06$ and $p = -10$. As for the non-linear utility function:

$$u = \begin{cases} p & \text{if } d = 0, \\ w_0 d + w_1 r - w_2 c - (w_3 g)^4 - (w_4 j)^4 - w_5 l & \text{if } d < 0, \\ w_0 d + w_1 r - w_2 c - 2 \cdot (w_3 g)^2 - w_5 l, & \text{otherwise.} \end{cases}$$

where $w_0 = 0.50$, $w_1 = 1$, $w_2 = 4$, $w_3 = 0.03$, $w_4 = 3.3 \times 10^{-5}$, $w_5 = 0.06$ and $p = -10$