Quality Assessment of MORL Algorithms: A Utility-Based Approach

Luisa M. Zintgraf[1] Timon V. Kanters[1] Diederik M. Roijers[1] Frans A. Oliehoek[1,2] Philipp Beau[1]

[1]*Informatics Institute, University of Amsterdam, The Netherlands*
[2]*Department of Computer Science, University of Liverpool, United Kingdom*

{luisa.zintgraf,timon.kanters,philipp.beau}@student.uva.nl
{d.m.roijers,f.a.oliehoek}@uva.nl

Abstract

Sequential decision-making problems with multiple objectives occur often in practice. In such settings, the utility of a policy depends on how the user values different trade-offs between the objectives. Such valuations can be expressed by a so-called scalarisation function. However, the exact scalarisation function can be unknown when the agents should learn or plan. Therefore, instead of a single solution, the agents aim to produce a solution set that contains an optimal solution for all possible scalarisations. Because it is often not possible to produce an exact solution set, many algorithms have been proposed that produce approximate solution sets instead. We argue that when comparing these algorithms we should do so on the basis of user utility, and on a wide range of problems. In practice however, comparison of the quality of these algorithms have typically been done with only a few limited benchmarks and metrics that do not directly express the utility for the user. In this paper, we propose two metrics that express either the expected utility, or the maximal utility loss with respect to the optimal solution set. Furthermore, we propose a generalised benchmark in order to compare algorithms more reliably.

1. Introduction

In sequential decision problems, agents aim to maximize their utility. When this utility can be expressed as a single (discounted) sum of scalar rewards, the problem can often be modelled as a *Markov decision process (MDP)*. MDPs have been studied extensively and many solution methods, as well as benchmarks to test these methods[1], have been proposed.

A complicating factor in many real-world decision problems however, is the existence of multiple possibly conflicting objectives, e.g., maximising the economic benefits of timber harvesting while minimising ecological damage in a forest management scenario [2]. In such cases, the rewards are vectors rather than scalars. Sometimes, it might be possible to scalarise these reward vectors using a pre-specified *scalarisation function* that expresses the utility of different trade-offs between the objectives, and reduce the problem to a single-objective problem. However, when the *parameters* of the scalarisation function are not known in advance, this approach does not apply. In such cases we need a model that expresses the multiple objectives explicitly, i.e., a *multi-objective MDP (MOMDP)* [10].

In MOMDPs, the agent's goal is to find a *solution set* that contains an optimal policy for each function in a given family of scalarisation functions. However, when the model is unknown to the agent, or there is not enough time or memory, it is not possible to produce an exact solution set. In such cases, different methods typically produce different approximate solution sets. The quality of such approximations should be assessed w.r.t. the expected utility [10].

In single-objective methods, it is straightforward to compare the quality of the policies found by different algorithms. In particular, the expected (discounted) sum of rewards, i.e., the *value*, of each policy is a scalar, and the policy with the highest value is best. However, it is not so straightforward to evaluate the relative quality of different approximate solution sets because we do not know the parameters of the scalarisation function that ultimately determine user utility. This makes quality assessment and comparison of MOMDP algorithms a challenging task.

Previous research on quality assessment for MOMDP methods follows an *axiomatic approach*. It assumes axiomatically that the Pareto front is the optimal solution set. Following this assumption, metrics (such as the widely used *hypervolume* metric) are proposed to express how well a solution set approximates the Pareto front. The maximal value of such a metric given a MOMDP can only be attained by identifying the Pareto front. An important limitation of these metrics however, is that they do not take into account the loss of utility for the user due to approximation of the

[1]E.g., `http://www.rl-competition.org/` and [18]

solution set.

Another important issue in the quality assessment of MOMDP algorithms is the limited choice in known benchmarks [15]. Often, algorithms are evaluated on a small amount of benchmarks, or specific (real-world inspired) problems which authors are interested in solving. While solving specific problems of course presents its own usability, it obfuscates the comparison between different algorithms. Furthermore, when algorithms are evaluated using the few available benchmarks, there is the risk of *method overfitting* [18], i.e., algorithms might perform really well on the specific available benchmarks, but that does not imply a good performance on different problems with widely different problem properties.

This paper contributes to quality assessment of MOMDP methods in two meaningful ways. Firstly, using the utility-based approach, we propose two methods to assess the quality of an approximate solution. We advocate the ε-metric [22] and show mathematically how this metric can be applied to bound the maximal loss in ultimate utility of the user. We propose the expected utility metric which can be used when a prior over scalarisation functions is known. Secondly, we propose a benchmark that we made publicly available. In order to ensure the quality of this benchmark, we outline criteria for a generalised benchmark for MOMDPs, and discuss the importance of preventing method overfitting through using a generalised benchmark. We illustrate how algorithms can be thoroughly compared using the proposed benchmark by comparing two state-of-the-art algorithms.

2. Background

Before we discuss quality assessment of MORL algorithms in-depth, we first provide the definitions and notations we will use, as well as an overview of previous work in quality assessment.

2.1. MOMDPs

A *Multi-Objective Markov Decision Process* (MOMDP) [10] is a tuple $< S, A, T, R, \mu, \gamma >$ where

- S is a finite set of states,
- A is a finite set of actions,
- $T : S \times A \times S \to [0, 1]$ is a transition function specifying, for each state, action, and next state, the probability of that next state occuring,
- $R : S \times A \times S \to \mathbb{R}^n,\ n \in \mathbb{N}$, is a reward function, specifying, for each state, action, and next state, the expected immediate reward vector of length n (one element for each objective),
- $\mu : S \to [0, 1]$ is a probability distribution over initial states,
- $\gamma \in [0, 1)$ is a discount factor specifying the relative importance of immediate rewards.

In the context of a MOMDP m, we denote the set of all possible (and allowed) policies π with Π^m. The value of a policy π is the expected cumulative discounted sum of reward vectors

$$\mathbf{V}^\pi = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{k+1} | \pi\right],$$

where $\mathbf{r}_t$ is the vector of the rewards received at time t. Unlike in single-objective MDPs, there is typically not a single policy that maximises the value in all objectives, i.e., there is not a single optimal policy. In such cases, what the best policy is depends on the preferences of end user(s) regarding the trade-offs between the different objectives. These preferences can be expressed in terms of a *scalarisation function*, also known as a *utility function*, f, which collapses the multi-objective value of a policy, $\mathbf{V}^\pi$, to a scalar value according to the preferences of the user [10]. This scalarised value is given by

$$V^\pi_{\mathbf{w}} = f(\mathbf{V}^\pi, \mathbf{w})$$

where $\mathbf{w}$ is a weight vector parameterising f. Following [10], we assume that when the end user selects a policy for execution scalarisation always takes places, either explicitly (by applying the scalarisation function) or implicitly, embedded in the thought-process of the end user.

The *utility-based approach* [10] highlights that agents aim to maximise the utility of the user, as expressed by the scalarisation function. Specifically, the agent should maximise $f(\mathbf{V}^\pi, \mathbf{w})$ where $\mathbf{w}$ is unknown during planning. Therefore the agent has to provide a solution set containing an optimal policy for each possible $\mathbf{w}$. Given a solution set, the user can then pick the policy that maximises the scalarisation function he decides upon. Unlike in the axiomatic approach, information about the scalarisation function, along with which policies are allowed, is thus used to *derive* which policies should be in the solution set. Furthermore, as we argue in this paper, the information about the scalarisation function can be used to bound the loss of utility due to approximation of this solution set.

The most common assumptions regarding the scalarisation function are that it is a) monotonically increasing in all objectives, or b) that it is also known to be *linear*, i.e., a weighted sum over the objectives. For MOMDPs where the scalarisation function is known to be monotonically increasing (but not necessarily linear) and only deterministic policies are allowed, the *Pareto front* (PF) is the optimal solution set [10]:

$$PF(\Pi^m) = \{\pi \in \Pi^m \mid \neg\exists\pi' \in \Pi^m : \mathbf{V}^{\pi'} \succ_{\mathcal{P}} \mathbf{V}^\pi\}$$

where $\succ_{\mathcal{P}}$ is the *Pareto dominance relation*,

$$\mathbf{V}^\pi \succ_{\mathcal{P}} \mathbf{V}^{\pi'} \Leftrightarrow (\forall i : V^\pi_i \geq V^{\pi'}_i) \wedge (\exists i : V^\pi_i > V^{\pi'}_i).$$

In words, the Pareto front is the set of all policies that are not Pareto-dominated by any other policy for m. However, when either f is monotonically increasing and stochastic policies are allowed, or f is linear, the optimal solution is the *convex hull* (CH):

$$CH(\Pi^m) = \{\pi \in \Pi^m \mid \exists \mathbf{w}\, \forall \pi' \in \Pi^m : \mathbf{w}\cdot\mathbf{V}^\pi \geq \mathbf{w}\cdot\mathbf{V}^{\pi'}\}.$$

I.e., the convex hull is the set of all policies that maximise the weighted sum over objectives for some weight vector $\mathbf{w}$. Other possible assumptions about f lead to different solution sets, e.g., a sense of fairness may lead to a Lorenz optimal set [9].

The utility-based approach stands in contrast to the axiomatic approach where the main assumption is that the PF is always the optimal solution set; any information about the user and his preferences should be incorporated.

2.2. Quality assessment

Quality metrics based on the axiomatic approach assess the quality of an approximate solution set by other criteria than user utility. Instead, these metrics aim to optimise for certain desirable characteristics. The most popular three features are [21]: *uniformity*, i.e., how uniformly the solutions are distributed (e.g., the Schott spacing metric [13]), how well the solutions are *spread* over the solution space (e.g., the maximum spread metric [21]), and *convergence*, i.e., how closely they approximate the Pareto front (e.g., the ε-indicator [22]). The popular hypervolume metric [22] combines all of these criteria and is able to reflect an improvement in any of the three mentioned characteristics. Therefore, the hypervolume is sometimes considered most suitable for MOMDP quality assessment [15]. However, the value of the hypervolume metric, just as some other quality metrics, strongy depends on the rather arbitrary choice of a reference point [15] and it is unclear what the values of such metrics that are used in the axiomatic approach mean. E.g., are two approximate sets with equal values for a certain metric actually equally good, and how should approximate solution sets be ranked if they have different rankings according to different metrics? This problem arises in the axiomatic approach because the ability of an agent to maximise utility — which we argue is ultimately the goal in solving any decision problem — is not directly assessed. However, this does not mean that contributions to quality assessment that have been done from the axiomatic point of view cannot be used. In fact, we will show that for the ε-indicator, we can bound the loss of utility in a utility-based point of view for certain classes of scalarisation functions.

Another important factor in the quality assessment of MOMDP algorithms is which problems to test the algorithms on. A key contribution in this respect was made by Vamplew et al. [15], who address the lack of overlap between test problems used by different authors and motivate the necessity of a standard benchmark for MOMDP algorithms. They establish a range of benchmark characteristics which are required to fully evaluate the performance of different MORL algorithms (see also Section 4), and propose a set of four specific test problems. However, they not address the risk of *method overfitting*.

Method overfitting [4] occurs when an algorithm overfits on the problem, i.e., a specific (type of) MOMDP. In [18], Whiteson et al. discuss the importance of countering method overfitting when evaluating RL algorithms in single-objective MDPs, and propose using *generalisation*, i.e., using multiple instances of the same class instead of a single MDP. Without generalisation, benchmark scores can be misleading in that algorithms which beat certain benchmark scores may only be capable of solving the given MDP, rather than solving (a large class of) MDPs in general.

3. Evaluation of candidate solution sets

When assessing the quality of an approximate solution set to an MOMDP, we follow the *utility-based approach*, i.e., we aim to assess the ability of an agent to maximise user utility.

We propose two ways to assess this ability directly. First, we propose the *expected utility metric (EUM)*, which calculates the expected utility given a prior over scalarisation functions. Second, when such a prior is not available, we advocate the use of the ε-metric, and prove that for an important class of scalarisation functions, i.e., Lipschitz-continuous monotonically increasing scalarisation functions, we can bound the *maximal utility loss (MUL)*.

3.1. Expected utility

The *expected utility metric (EUM)* indicates how much utility an agent can expect to gain for the user on average, given the solution set it returns and a family of scalarisation functions. This metric is related to the *R Indicator Family* [22] which compares two solution sets directly based on a family of scalarisation functions. Our metric however, evaluates one solution set at a time. The expected utility of a solution set $\mathcal{S}$ for a given family of scalarisation functions $V_w^\pi = f(\mathbf{V}^\pi, \mathbf{w})$ and a probability distribution $P(\mathbf{w})$ over weights $\mathbf{w}$ is given by the expected average utility the optimal policy from the solution set gains,

$$EUM(\mathcal{S}, f, P) = \mathbb{E}[\max_{V^\pi \in \mathcal{S}} f(\mathbf{V}^\pi, \mathbf{w})].$$

Sometimes it is possible to calculate this expectation exactly for a given solution set. When this is not feasible, sampling methods, i.e., drawing a set of K samples $\mathbf{w}$ from $P(\mathbf{w})$, can be used to approximate this expected value,

$$EUM(\mathcal{S}, f, P) \simeq \frac{1}{K} \sum_{k=1}^{K} \max_{V^\pi \in \mathcal{S}} f(\mathbf{V}^\pi, \mathbf{w}_k).$$

This metric is particularly useful when it is important that the agent can perform well across a family of scalarisation functions, for example in a case when several policies from the solution set will be used over time. The standard deviation can give additional information about how this performance fluctuates over scalarisation functions. It is crucial however that the family of scalarisation functions and its probability distribution, or a good prior are known.

3.2. Utility loss of ε-approximate Pareto fronts

When no prior over $\mathbf{w}$ can be defined, we cannot compute EUM. For this case, we propose the *maximal utility loss (MUL)* metric that gives a sense of the utility that is lost due to approximation, given the available information about f.

Definition 1. *For a given solution set $\mathcal{S}$ and a family of scalarisation functions $f(\mathbf{V}^\pi, \mathbf{w})$ we define the maximum utility loss $MUL(\mathcal{S}, f)$ as the maximum scalarised value that is lost due to approximation, i.e.,*

$$\forall \mathbf{V}^\pi \in P \, \exists \mathbf{V}^{\pi'} \in \mathcal{S} : \quad f(\mathbf{V}^\pi, \mathbf{w}) \leq f(\mathbf{V}^{\pi'}, \mathbf{w}) + MUL(\mathcal{S}, f) \, .$$

For an ε-version of convex hulls and corresponding MUL see, e.g., [11]. Here, we show that for an ε-approximate PF, we can bound MUL for the family of Lipschitz-continuous scalarisation functions, i.e., functions f for which there is a Lipschitz-constant $K \geq 0$ s.t.

$$|f(x) - f(y)|_2 \leq K|x - y|_2 \, .$$

Lipschitz-continuous scalarisation functions are widely used in RL [3, 5, 7].

An ε-approximate Pareto front $\mathcal{S}$ [22] is a solution set that approximates the Pareto front $\mathcal{P}$, and there are two versions of the ε-indicator which express how good this approximation is. The **additive ε-indicator** is given by

$$I_{\varepsilon+} = \inf_{\varepsilon \in \mathbb{R}} \{ \forall \, \mathbf{V}^\pi \in \mathcal{P} \, \exists \, \mathbf{V}^{\pi'} \in \mathcal{S} : \quad \forall i = 1, \ldots, n : \; V_i^\pi \leq V_i^{\pi'} + \varepsilon \}$$

and the **multiplicative ε-indicator** is given by

$$I_{\varepsilon+} = \inf_{\varepsilon \in \mathbb{R}} \{ \forall \, \mathbf{V}^\pi \in \mathcal{P} \, \exists \, \mathbf{V}^{\pi'} \in \mathcal{S} : \quad \forall i = 1, \ldots, n : \; V_i^\pi \leq V_i^{\pi'}(1 + \varepsilon) \}$$

where n is the number of objectives in the MOMDP. We now use the ε-indicator to bound MUL for monotonically increasing Lipschitz-continuous scalarisation functions.

Theorem 1. *Let $f(\mathbf{V}^\pi, \mathbf{w})$ be a monotonically increasing scalarisation function that is Lipschitz-continuous for all weights $\mathbf{w}$ with Lipschitz-constant $L_\mathbf{w}$. Let further $L = \max_\mathbf{w} L_\mathbf{w}$. Then*

(a) if a solution set $\mathcal{S}$ is an ε-approximate Pareto front in the sense of the additive *ε-indicator,*

$$MUL(\mathcal{S}, f) = \varepsilon \sqrt{n} L.$$

(b) If a solution set $\mathcal{S}$ is an ε-approximate Pareto front in the sense of the multiplicative *ε-indicator, then*

$$MUL(\mathcal{S}, f) = \frac{\varepsilon}{1+\varepsilon} \sqrt{n} L \max_{\substack{i=1,\ldots,n, \\ V^\pi \in \mathcal{P}}} |V_i^\pi|.$$

Proof. Let $\mathbf{w} \in \mathbb{R}_+^n$ be an arbitrary but fixed weight vector and denote $f(\cdot, \mathbf{w}) = f_\mathbf{w}(\cdot)$. Further let $\mathbf{V}^{\pi_0} \in \mathcal{P}$ be a policy value from the true Pareto front $\mathcal{P}$ for which $f_\mathbf{w}$ is maximal across all policies (exists in $\mathcal{P}$ because $f_\mathbf{w}$ is monotonically increasing),

$$\mathbf{V}^{\pi_0} \in \{ \mathbf{V}^\pi \in \mathcal{P} | \forall \pi' \in \Pi^m : f_\mathbf{w}(\mathbf{V}^\pi) \geq f_\mathbf{w}(\mathbf{V}^{\pi'}) \} \quad (1)$$

and let $\mathbf{V}^{\pi_\varepsilon} \in \mathcal{S}$ be a policy value from the solution set $\mathcal{S}$ for which $f_\mathbf{w}$ is maximal across $\mathcal{S}$,

$$\mathbf{V}^{\pi_\varepsilon} \in \{ \mathbf{V}^\pi \in \mathcal{S} | \forall \mathbf{V}^{\pi'} \in \mathcal{S} : f_\mathbf{w}(\mathbf{V}^\pi) \geq f_\mathbf{w}(\mathbf{V}^{\pi'}) \}. \quad (2)$$

(a) Let $\mathbf{V}^{\pi_{0\varepsilon}} \in \mathcal{S}$ be a policy value that additively ε-dominates the above defined $\mathbf{V}^{\pi_0} \in \mathcal{P}$,

$$\mathbf{V}^{\pi_{0\varepsilon}} \in \{ \mathbf{V}^\pi \in \mathcal{S} | \forall i = 1, \ldots, n : V^{\pi_0} \leq V_i^\pi + \varepsilon \} \, . \quad (3)$$

Then from (3) and with $\boldsymbol{\varepsilon} := (\varepsilon, \ldots, \varepsilon) \in \mathbb{R}^n$ follows

$$\begin{aligned} V_i^{\pi_0} \leq V_i^{\pi_{0\varepsilon}} + \varepsilon \Leftrightarrow V_i^{\pi_0} - \varepsilon \leq V_i^{\pi_{0\varepsilon}} & \\ \overset{(*)}{\Rightarrow} f_\mathbf{w}(\mathbf{V}^{\pi_0} - \boldsymbol{\varepsilon}) \leq f_\mathbf{w}(\mathbf{V}^{\pi_{0\varepsilon}}) & \\ \overset{(2)}{\Rightarrow} f_\mathbf{w}(\mathbf{V}^{\pi_0} - \boldsymbol{\varepsilon}) \leq f_\mathbf{w}(\mathbf{V}^{\pi_\varepsilon}) & \quad (4) \end{aligned}$$

and further,

$$\begin{aligned} f_\mathbf{w}(\mathbf{V}^{\pi_0}) - f_\mathbf{w}(\mathbf{V}^{\pi_\varepsilon}) & \\ & \overset{(4)}{\leq} f_\mathbf{w}(\mathbf{V}^{\pi_0}) - f_\mathbf{w}(\mathbf{V}^{\pi_0} - \boldsymbol{\varepsilon}) \\ & \overset{(*)}{=} |f(\mathbf{V}^{\pi_0}) - f_\mathbf{w}(\mathbf{V}^{\pi_0} - \boldsymbol{\varepsilon})|_2 \\ & \overset{(**)}{\leq} L_\mathbf{w} |\mathbf{V}^{\pi_0} - (\mathbf{V}^{\pi_0} - \boldsymbol{\varepsilon})|_2 \\ & = L_\mathbf{w} |\boldsymbol{\varepsilon}|_2 \\ & \leq \varepsilon \sqrt{n} L \end{aligned}$$

(b) Let $\mathbf{V}^{\pi_{0\varepsilon}} \in \mathcal{S}$ be a policy value that multiplicatively ε-dominates $\mathbf{V}^{\pi_0}$,

$$\mathbf{V}^{\pi_{0\varepsilon}} \in \{ V^\pi \in \mathcal{S} | \forall i = 1, \ldots, n : V_i^{\pi_0} \leq (1+\varepsilon) V_i^\pi \}. \quad (5)$$

Then

$$
\begin{aligned}
&f_{\mathbf{w}}(\mathbf{V}^{\pi_0}) - f_{\mathbf{w}}(\mathbf{V}^{\pi_\varepsilon}) \\
&\quad\overset{(2)}{\leq} f_{\mathbf{w}}(\mathbf{V}^{\pi_0}) - f_{\mathbf{w}}(\mathbf{V}^{\pi_{0\varepsilon}}) \\
&\quad\overset{(5,*)}{\leq} f_{\mathbf{w}}(\mathbf{V}^{\pi_0}) - f_{\mathbf{w}}\left(\frac{1}{(1+\varepsilon)}\mathbf{V}^{\pi_0}\right) \\
&\quad\overset{(*)}{=} |f_{\mathbf{w}}(\mathbf{V}^{\pi_0}) - f_{\mathbf{w}}\left(\frac{1}{(1+\varepsilon)}\mathbf{V}^{\pi_0}\right)|_2 \\
&\quad\overset{(**)}{\leq} L_{\mathbf{w}}|\mathbf{V}^{\pi_0} - \frac{1}{(1+\varepsilon)}\mathbf{V}^{\pi_0}|_2 \\
&\quad= L_{\mathbf{w}}|\frac{\varepsilon}{1+\varepsilon}\mathbf{V}^{\pi_0}|_2 \\
&\quad= L_{\mathbf{w}}\frac{\varepsilon}{1+\varepsilon}|\mathbf{V}^{\pi_0}|_2 \\
&\quad\leq \frac{\varepsilon}{1+\varepsilon}\sqrt{n}L \max_{\substack{i=1,\ldots,n,\\ V^\pi\in\mathcal{P}}} |V_i^\pi|
\end{aligned}
$$

(*) because f is monotonically increasing
(**) because f is Lipschitz-continuous □

We showed how the MUL metric can be used to estimate an upper bound on the possible loss of utility for any Lipschitz-continuous f. We see from the factor $\sqrt{n}$ that this loss depends on the number of objectives of the MOMDP. The higher the dimensions, the higher the MUL for the same ε. For larger number of dimensions, it will become increasingly harder to keep the maximal possible loss of utility small, however, this will be the case across all planning and learning algorithms.

The MUL metric overcomes the drawback of the expected reward metric, as it does not require the knowledge of the distribution over weights.

In practice, the true Pareto front is often infeasible to compute, which makes calculating ε exactly infeasible too. In this case, we can work with a good reference set instead of the PF, and compare algorithms on this set. When comparing two algorithms, the union of the final solution set of both algorithms can be used as a reference set (however, when comparing a new algorithm these reference sets change precluding their reuse).

Some multi-objective algorithms exist that guarantee producing an ε-approximate Pareto front without ever calculating the true Pareto front [8]. In this case, the MUL is known without needing a reference set if we make use of the fact that $\max_{\substack{i=1,\ldots,n,\\ V^\pi\in\mathcal{P}}} |V_i^\pi| \leq \max_{\substack{i=1,\ldots,n,\\ V^\pi\in\mathcal{S}}} |(1+\varepsilon)V_i^\pi|$.

4. Criteria for a good benchmark

Having defined metrics with which to compare different approximate solution sets, we can now compare the output of different MOMDP algorithms. However, in order to compare the performance of these algorithms, we require test problems that provide a deeper insight into how the algorithms behave as a function of different problem features. Vamplew et al. [15] propose the following list of features that a benchmark should contain:

a. two or more objectives
b. stochasticity in transition dynamics and/or rewards
c. continuous state or action spaces
d. state dimensionality high enough to require the use of function approximation
e. partially-observable states
f. a mixture of episodic and continuing tasks
g. different Pareto front features such as concavities and discontinuities

In essence, these features demand a benchmark (suite) to be able to test performance on a large *variety* of problems, suitable for many different types of algorithms.

In addition to the above-mentioned features, we stress the importance of generalisation. Generalisation prevents algorithms from overfitting to the problem, leading to a more reliable evaluation of the performance [18]. To prevent method overfitting, a benchmark's environment should be able to keep the difficulty parameters (Section 5.2) of the problem the same, while shuffling the exact way these properties are implemented. E.g., in the benchmark we propose in the next section, these are different resources. While the number of resources influences the difficulty of the problem, the exact position of the resources can be determined randomly, without changing the difficulty. Shuffling these positions randomly causes the agent to be presented with different MOMDPs. While testing algorithms on problems with given difficulty parameters, one should always test several instances of such a *generalised benchmark*. Furthermore, because a generalised benchmark is parameterised, it is possible to test how well different algorithms perform *as a function* of these parameters, leading to a deeper insight into the relative performance of the algorithms.

5. The Collecting Traveller benchmark

We propose a new benchmark inspired by resource gathering problems, such as the one described in [15]. Our benchmark has many adjustable settings that shape the problem and, as such, is able to satisfy all criteria.[2]

5.1. Shape of the environment

In our proposed problem, The Collecting Traveller, the agent starts in the bottom-left corner of a field and has to move to the goal in the top-right corner. Within the field are several resources of different types. Each type corresponds to an objective. Collecting a resource, i.e., moving to its location, results in a positive reward in the objective corresponding to the resource type. Furthermore, a reward of -1

[2]The satisfaction of the last feature (g) is somewhat limited, as it is possible that not all Pareto front features can be created; we were unable to prove whether it is possible to create discontinuities in the Pareto front. I.e., gaps in an otherwise continuous front.

is given in a separate objective for each action performed. An example of shapes that the benchmark can take is seen in Figure 1.

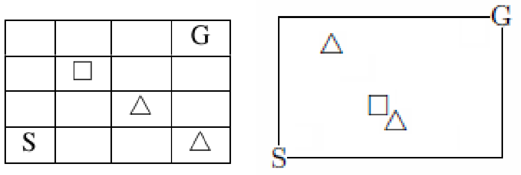

Figure 1. An example field with the start position S, the goal position G, two resource types △ and □, one resource of □ and two resources of △, either discrete (left) or continuous (right).

Default definition

- A state $s = (Pos_x, Pos_y, Res_1, \dots, Res_{N_r}) \in S$, where (Pos_x, Pos_y) is the position of the agent and Res_i are booleans indicating whether or not resource i has been picked up.[3] The initial position is $(0,0)$ (lower left). The goal (top right) is a terminal state.
- A set of actions: $A = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}$ corresponding to one step in that direction.
- Deterministic transitions; actions always succeed. If an agent walks into the border, his position remains the same.
- Deterministic rewards. An immediate reward vector has the form $R = (-1, B_1, \dots, B_{N_t})$, where B_i is a reward for collecting a resource of type i. For one problem instance, the positions of resources are the same for each episode.
- A planning horizon of 1000 steps (finite horizon). The rewards are not discounted.

5.2. Adjustable parameters

In order to support generalisation and adjust difficulty, the following parameters can be set:

Field size – The field's width and height can be set.

Resources – Collecting a resource will give the agent a reward in the objective corresponding to the resource's type. More resources can be placed at the same location. The number of resources can be set by the user.

The number N_t of types of resources can be set, but cannot be more than the amount of resources. The number of objectives is $N_t + 1$.

The locations of the resources are randomised in order to generalise. Each resource can have a minimum and maximum reward assigned where a uniformly random chosen reward from that range is awarded when the resource is collected. However, exact locations and rewards of the resources can also be specified if wanted.

Pick up when collecting – By default, resources are picked up when collected. Alternatively, they can be left, allowing them to be collected multiple times. To facilitate this, the action *wait* is added, allowing the agent to consecutively collect the same resource.

Limited pick-ups – The amount of resources an agent can collect can be limited in order to require an extra consideration during the decision making process. After the limit has been reached, attempting to collect a resource will not give a reward or have the resource picked up in the state. This helps create an additional challenge, as well as reducing the amount of states.

Action space – There are three possible discrete action spaces that can be selected: a *tiny*, $A = \{\uparrow, \rightarrow\}$, a *small* $A = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}$, and a *full* action space $A = \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow, \swarrow, \leftarrow, \nwarrow\}$, in which each arrow corresponds to one step in that direction.

When collected resources are not picked up and can be collected multiple times, an additional *wait* action is available that leaves the agent in the same position. This leaves the infinite horizon setting with action spaces of 3, 5 or 9 actions.

Transition stochasticity – It is possible to provide an action failure probability. Upon action failure, a randomly selected action from the action space is performed in addition to the chosen action. If the failure causes the agent to skip the position of a resource, it will not be collected.

Horizon – The problem can have either a fixed amount of time steps as horizon, or an infinite horizon.

Discount factor – For infinite horizon settings, the discount factor $\gamma \in [0, 1)$ can be set.

5.3. Extensions

We offer two extensions to the standard MOMDPs:

Partial observability – Only the locations of resources within a Manhattan distance of 1 of the agent will be visible, i.e., at each time step, the agents receives a vector of length N_r in which each resource is denoted as picked up, not picked up or not observed. The probability of (incorrectly) not observing a resource can be set, false positives do not occur. In this case, instead of a MOMDP, we have a *multi-objective partial-observable MDP* (MOPOMDP) [12, 14].

Continuity – For continuous problems, the agent and resource locations are no longer discrete but can be any value within the field size such that coordinates are defined as $(x, y) \in [0, width] \times [0, height]$. In addition, actions are replaced by a 2-dimensional continuous value from $[-1, 1]$ where one value indicates horizontal movement and the other vertical movement. Resources are collected when the Manhattan distance between the agent and the resource is less than 1. A visual representation of a continuous environment is seen in Figure 1 (right).

[3] In the code, the state also contains the resource positions and types.

5.4. Availability

The benchmark implementation can be used in two fashions. As it is written in Java, the project can be included in any other Java project. This gives programmers complete control over the specific use of the benchmark, which is particularly useful when it is desired to perform specific tests rather than the standard experiment. The source code can be found on the Git repository at `https://github.com/tvkanters/MORL-benchmark`.

Alternatively, other programming languages can be connected through MORL-Glue, which is the multi-objective extension of RL-Glue[4]. By writing an agent with the interface of the MORL-Glue codec implemented, it can receive observations and rewards while responding with actions. Running the benchmark implementation will provide the environment on the other end which handles all transitions based on the received actions.

6. Experiments

In order to demonstrate our benchmark, we implemented two algorithms designed to solve MOMDPs: Multi-Objective Monte-Carlo Tree Search[5] (MOMCTS) [17] and Convex Hull Value Iteration (CHVI) [1]. We test and compare these algorithms using different problem instances of our generalised benchmark (with known PFs and CHs), judging performance across different state spaces, resources and action spaces. In addition, we create a problem to highlight that CHVI can quickly identify the CH but not the PF. With the exception of the last problem, we let each algorithm run ten times for 1000 episodes where each episode ends after 1000 steps or reaching the goal state. After each run, the resource positions are shuffled for generalisation purposes.

6.1. Problem sizes

The first experiment involves testing both algorithms with different problem sizes. For this we use three different problems of sizes 4 by 4, 7 by 7 and 10 by 10. These problems have 2, 4 and 5 resources to pick up respectively. In Figure 2 (left), we can observe the different behaviour of the algorithms by looking at the additive ε-metric (from which the MUL can be calculated for Lipschitz-continuous scalarisation functions). CHVI often finds a (near) optimal solution faster than MOMCTS. Regardless of problem sizes however, MOMCTS has a better policy for a period during the search before getting caught up by CHVI.

6.2. Action spaces

In order to judge how well the algorithms handle different action spaces, we test both with all available action spaces (Section 5.2) in the previously used 4 by 4 problem. The results, as seen in Figure 2 (right), show that while the behaviour is similar to that across different problem sizes, MOMCTS is able to handle large state spaces better in the early stage; the large action space performance indicates that it achieves relatively better performance early on.

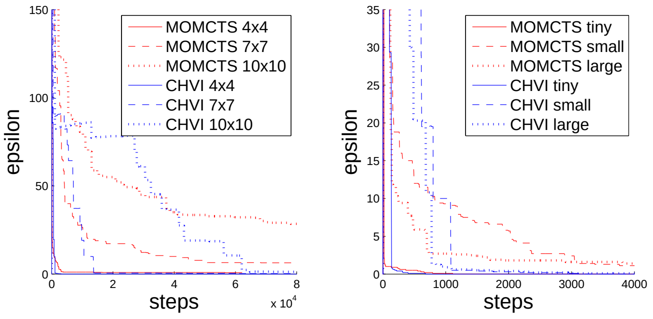

Figure 2. The additive ε-indicator over steps across different problem sizes (left) and different action spaces (right), averaged over the 10 runs.

6.3. Convex hull

Finally, we test the algorithm against a problem that is meant to indicate behaviour of finding the convex hull rather than the complete Pareto front, as CHVI does. To do this, an environment as seen in Figure 3 (left) is used. In contrary to the default setting, this problem has more than 4 actions (the agent can also move diagonally) and the agent can only collect one resource. The rewards for collecting each resource are shown per location in Figure 3. N.B., the time cost of 1 per step is still active. This results in the convex hull $\{(-3,0,1),(-2,1,0)\}$ and the Pareto front $\{(-3,0,1),(-2,1,0),(-3,0.4,0.4)\}$.

(0,1)	(1,0)	goal
start		(0.4,0.4)

Figure 3. The field of the convex hull problem.

When looking at the average rewards in Figure 4, we see that even though CHVI's solution lacks a point from the Pareto front, as opposed to MOMCTS's, the linear scalarisation function gives both the same reward. This is because with linear scalarisation, a convex hull is already an optimal solution. To show that there is in fact a difference between the two solutions in terms of potential rewards, we define a monotonically increasing scalarisation function:

$$f(\mathbf{V}^\pi, \mathbf{w}) = \min_{i=1,\dots,n-1}(V_i^\pi) + \sum_{j=1}^{n} \mathbf{w}_j V_j^\pi$$

In the context of this problem, using this function means that the agent gets an extra reward if it collects the same amount of each resource type. In our case, there are only two resource types to collect, giving us $f(\mathbf{V}^\pi, \mathbf{w}) = \min_{i=1,2}(V_i^\pi) + \sum_{j=1}^{3} \mathbf{w}_j V_j^\pi$. Using this function, we see

[4]`http://glue.rl-community.org/wiki/Main_Page`

[5]For computational reasons, our implementation of MOMCTS differs slightly in that it uses the projection on the hypercubes.

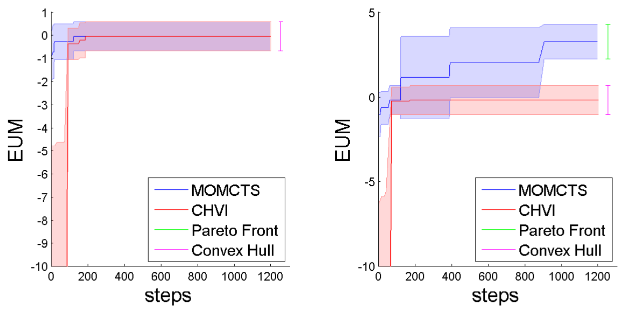

Figure 4. The expected utility and its standard deviations over time with a linear scalarisation function (left) and a monotonically increasing (nonlinear) scalarisation function (right).

in Figure 4 (right) that once MOMCTS has found a better solution set than the convex hull, it achieves a higher reward than CHVI.

7. Conclusion

In this paper we proposed two methods to assess the quality of an approximate solution with respect to the (maximal) utility loss and a benchmark for multi-objective reinforcement learning which is publicly available and can be used cross-language through MORL-Glue. Our benchmark is parameterisable and creates a whole class of problems rather than focusing on a specific one. We discussed the current standard of quality evaluation metrics and the drawbacks when rigorously applying them in a MORL setting. To perform a more suitable evaluation, we advocated the *expected utility metric* and showed how to use the ε-metric [22] to estimate the *maximal utility loss* of Lipschitz-continuous functions.

To demonstrate some of the capabilities of our benchmark we implemented and compared two state-of-the-art algorithms: MOMCTS and CHVI. Our benchmark has proven to be capable of illustrating key differences between algorithms in terms of learning speed, solution quality and solution set characteristics.

In future work, we intend to compare different planning [6, 19] and learning [16, 20] algorithms for MO(PO)MDPs, and provide other generalised benchmarks based on different scenarios.

Acknowledgements

We thank Rustam Issabekov and Peter Vamplew for sharing MORL-Glue with us. This research is supported by the NWO DTC-NCAP (#612.001.109) project and in part by NWO Innovational Research Incentives Scheme Veni (#639.021.336).

References

[1] L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *ICML*, pages 41–47, 2008.

[2] C. Bone and S. Dragicevic. GIS and intelligent agents for multiobjective natural resource allocation: A reinforcement learning approach. *Trans. in GIS*, 13(3):253–272, 2009.

[3] L. Buşoniu, R. Munos, and R. Babuška. A survey of optimistic planning in Markov decision processes, 2012.

[4] E. Falkenauer. On method overfitting. *Journal of Heuristics*, 4(3):281–287, 1998.

[5] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *STOC*, pages 681–690, 2008.

[6] C. Kooijman, M. de Waard, M. Inja, D. M. Roijers, and S. Whiteson. Pareto local policy search for MOMDP planning. In *ESANN*, pages 53–58, 2015.

[7] O.-A. Maillard and R. Munos. Online learning in adversarial Lipschitz environments. In *ECML*, pages 305–320. 2010.

[8] R. Marinescu. Efficient approximation algorithms for multi-objective constraint optimization. In *ADT*, pages 150–164. 2011.

[9] P. Perny, P. Weng, J. Goldsmith, and J. P. Hanna. Approximation of lorenz-optimal solutions in multiobjective markov decision processes. In *AAAI*, 2013.

[10] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.*, 48:67–113, 2013.

[11] D. M. Roijers, S. Whiteson, and F. A. Oliehoek. Computing convex coverage sets for faster multi-objective coordination. *J. Artif. Intell. Res.*, 52:399–443, 2015.

[12] D. M. Roijers, S. Whiteson, and F. A. Oliehoek. Point-based planning for multi-objective POMDPs. In *IJCAI*, 2015. To appear.

[13] J. R. Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Technical report, 1995.

[14] H. Soh and Y. Demiris. Evolving policies for multi-reward partially observable markov decision processes (MR-POMDPs). In *GECCO*, pages 713–720, 2011.

[15] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84(1-2):51–80, 2011.

[16] K. van Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *JMLR*, 15:3483–3512, 2014.

[17] W. Wang, M. Sebag, et al. Multi-objective monte-carlo tree search. In *Asian conference on machine learning*, 2012.

[18] S. Whiteson, B. Tanner, M. E. Taylor, and P. Stone. Generalized domains for empirical evaluations in reinforcement learning. In *ICML*, 2009.

[19] M. A. Wiering and E. D. De Jong. Computing optimal stationary policies for multi-objective markov decision processes. In *ADPRL*, pages 158–165, 2007.

[20] M. A. Wiering, M. Withagen, and M. M. Drugan. Model-based multi-objective reinforcement learning. In *ADPRL*, pages 1–6, 2014.

[21] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. 1999.

[22] E. Zitzler, J. Knowles, and L. Thiele. Quality assessment of pareto set approximations. In *Multiobjective Optimization*, pages 373–404. 2008.