AI Lab Technical Report AI-TR-14-69

Efficient Weight Space Search in Multi-Objective Reinforcement Learning

Kristof Van Moffaert[1], Tim Brys[1], and Ann Nowé[1]

Artificial Intelligence Lab,
Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussels, Belgium
{kvmoffae,timbrys,anowe}@vub.ac.be

Abstract. Linear scalarization is, due to its simplicity, by far the most commonly used method to solve multi-objective problems, as it permits the use of single-objective solution techniques by combining multiple objectives into a single scalar value by means of a weighted sum. This technique is also extensively used in multi-objective reinforcement learning (MORL) specifically, but the usefulness of scalarization depends on the shape of the Pareto optimal set, and the values assigned to the weights. For each policy in the convex multi-objective hull of the Pareto optimal set, there exist weight combinations that modify the Markov Decision Process (MDP), such that that specific policy is optimal. Setting the weights appropriately allows the system designer to select which policy the learning agent will converge to, but this assignment of weights is not trivial. In this paper, we propose an algorithm that efficiently searches the weight space to identify the policies that are included in the convex hull of the Pareto optimal set, and the associated regions in the weight space that make those policies optimal. The technique is experimentally validated on several MORL benchmark problems.

Keywords: Multi-objective reinforcement learning, linear scalarization, weight space search.

1 Introduction

Many optimization problems that need to be solved nowadays are in essence tasks that involve more than one objective. For example, in network routing, the objectives to be optimized are packet latency and energy consumption. When the system engineer wants to optimize more than one objective at the same time, it is not always clear from the problem description (if any) how to achieve this, and how the objectives influence each other. Also, it is often not sufficient to

try to optimize just one objective without considering the effect that this maximization has on the other objectives in the system. In such cases, we are dealing with a genuine multi-objective optimization problem. Formally, multi-objective optimization (MOO) is the process of simultaneously optimizing multiple objectives which can be complementary, conflicting as well as independent. The goal of MOO is to search the policy space and eventually find policies that provide different trade-offs between objectives.

Usually, a (weighted) linear *scalarization* function is used to translate the original multi-objective problem into a single-objective problem that can be solved by single-objective techniques. The weight parameters $w_o \in [0, 1]$ are *preference* factors that identify the relative importance of objectives o, with $\sum_{o=0}^{m-1} w_o = 1$ for m objectives. This transforms the original problem into a single-objective optimization problem, which is a weighted convex sum of the original objectives: $f(x) = \sum_{o=0}^{m-1} w_o f_o(x)$. The main disadvantages of this approach are (i) that it is limited to finding policies that are located on convex parts of the Pareto optimal set, and (ii) that the correspondence between weights and policies is not clear, i.e. a uniform sampling of weights usually does not result in a uniform sampling of (convex parts of) the Pareto optimal set. The former problem can be solved by applying non-linear scalarization techniques, such as the Chebyshev metric [1]. The latter problem requires system designers to spend a lot of excessive computation to the problem of fine-tuning the linear scalarization weights, in order to find desired trade-off policies from the Pareto optimal set.

Main contributions. In this paper, we analyse the mapping of the weight-space to policies on the convex hull of the Pareto optimal set, found by reinforcement learning agents for several convex MORL benchmark problems. We show that setting scalarization weights based on intuition or a uniform sampling of the weight-space is very inefficient, since often very few Pareto optimal policies have large basins of attraction in the weight space, with other policies limited to very small regions in the weight-space. We therefore present a simple tree-based search algorithm on top of a value-iteration algorithm that steers the weight configuration of the learning agent by analyzing the learned policies. The algorithm is particularly suited for being used in an *on-line* context, while being easily applied to problems with any number of objectives, in contrast to other techniques that are often limited to two or three, due to complex (geometrical) calculations. We perform an empirical evaluation of the method on several multi-objective environments and show it to be superior both to uniform sampling and a popular non-linear scalarization method.

Outline. The contents of the paper is organized as follows. In Section 2, we describe related concepts such as reinforcement learning and multi-objective optimization. In Section 3, we present our tree-based algorithm for steering the underlying value-iteration algorithm. Furthermore, in Section 4, we conduct an empirical evaluation of the method on several benchmark instances. In Section 5, we summarize the paper and form conclusions.

2 Related work

In this section, we describe some of the basic concepts related concepts to multi-objective reinforcement learning (MORL) and current techniques in MORL for the approximation of convex Pareto optimal sets.

2.1 Reinforcement learning

A reinforcement learning [2] environment is typically described as a Markov Decision Process (MDP), which is formulated as follows. Let $S = \{s_1, \ldots, s_N\}$ be the state space of a finite Markov chain $\{x_l\}_{l \geq 0}$ and $A = \{a_1, \ldots, a_r\}$ the action set available to the learning agent. Each combination of current state s_i, action choice $a_i \in A_i$ and next state s_j has an associated transition probability $T(s_j|s_i, a_i)$ and immediate reward $R(s_i, a_i)$. The goal is to learn a policy π, which maps each state to an action so that the expected accumulated future discounted reward J^π is maximized:

$$J^\pi \equiv E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right] \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor and expectations are taken over stochastic rewards and transitions. This goal can also be expressed using Q-values which explicitly store the expected discounted reward for every state-action pair. The optimal Q^*-values are defined as follows.

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} T(s'|s, a) \max_{a'} Q^*(s', a') \tag{2}$$

The most well known RL algorithm is Q-Learning [3], described by Watkins to iteratively approximate Q^*. In the Q-learning algorithm, a Q-table consisting of state-action pairs is stored. Each entry contains a value for $\hat{Q}(s, a)$ which is the learner's current estimate about the actual value of $Q^*(s, a)$. The $\hat{Q}$-values are updated according to the following update rule:

$$\hat{Q}(s, a) \leftarrow (1 - \alpha_t)\hat{Q}(s, a) + \alpha_t[R(s, a) + \gamma \max_{a'} \hat{Q}(s', a')] \tag{3}$$

where α_t is the learning rate at time step t and $R(s, a)$ is the reward received for performing action a in state s. Provided that all state-action pairs are visited infinitely often, and a suitable evolution for the learning rate is chosen, the estimates, $\hat{Q}$, will converge to the optimal values, Q^* [4].

2.2 Multi-objective reinforcement learning

In multi-objective optimization, the objective space consists of two or more dimensions. Therefore, regular MDPs are extended to multi-objective MDPs or

MOMDPs. MOMDPs are by definition MDPs that provide a vector of rewards instead of a single reward, i.e:

$$\mathbf{R}(s_i, a_i) = (R_0(s_i, a_i), \dots R_{m-1}(s_i, a_i)) \tag{4}$$

where m represents the number of objectives. Since reinforcement learning traditionally concerns a maximization problem, we also describe multi-objective reinforcement learning (MORL) as a maximization problem. In the case of MORL, a solution is a policy π and is evaluated by its expected return $\mathbf{J}^\pi$, which is a vector of expected returns for each objective. Thus,

$$\mathbf{J}^\pi \equiv \left[E\left[\sum_{t=0}^{\infty} \gamma^t R_0(s_t, \pi(s_t)) \right], \dots, E\left[\sum_{t=0}^{\infty} \gamma^t R_{m-1}(s_t, \pi(s_t)) \right] \right] \tag{5}$$

Since the environment now consists of multiple objectives, conflicts could arise when trying to simultaneously optimize the objectives. In such a case, trade-offs between these objectives have to be made, resulting in a set of incomparable policies. The set of non-dominated, optimal policies for each objective or a combination of objectives is referred to as the *Pareto optimal set*[1]. A solution x_1 is said to strictly dominate another solution x_2, i.e. $x_2 \prec x_1$, if each objective in x_1 is not strictly less than the corresponding objective of x_2 and at least one criterion is strictly greater. In the case where x_1 improves x_2 on some criterion and x_2 also improves x_1 on one or more objectives, the two solutions are said to be incomparable.

Current approaches in MORL often use *scalarization* functions [5] to reduce the dimensionality of the underlying multi-objective environment to a single dimension. Scalarization functions often imply that an objective o is associated with a weighted coefficient, which allows the user some control over the nature of the policy found by the system, by placing more or less emphasis on each of the objectives. In a multi-objective environment, this trade-off is parametrized by $w_o \in [0, 1]$ for objective o and $\sum_{o=0}^{m-1} w_o = 1$.

In [1], a general framework for MORL algorithms was proposed that extends the scalar $\hat{Q}$-values to $\hat{\mathbf{Q}}$-vectors that store a $\hat{Q}$-value for each objective, i.e.

$$\hat{\mathbf{Q}}(s, a) = \left[\hat{Q}_0(s, a), \dots, \hat{Q}_{m-1}(s, a) \right] \tag{6}$$

The proposed framework uses a scalarization function to perform the multi-objective action selection, based on the $\hat{\mathbf{Q}}$-vector of each state-action pair. More precisely, by performing a scalarization function over the $\hat{\mathbf{Q}}$-vector of action a in state s, a scalarized Q-value or SQ-value is obtained. Commonly, the *linear* scalarization function is used that performs a weighted-sum over the Q-value of each objective and its corresponding weight, i.e.

$$SQ(s, a) = \sum_{o=0}^{m-1} w_o \cdot \hat{Q}_o(s, a) \tag{7}$$

[1] Also known as the *Pareto front*

The framework also allows to incorporate non-linear scalarization functions, such as the Chebyshev scalarization function. This particular method was found to be very effective on non-convex Pareto optimal sets, as it always tries to minimize the weighted distance from the current solution to a utopian point, i.e.

$$SQ(s,a) = \max_{o=0...(m-1)} w_o \cdot |\hat{Q}(s,a,o) - z^*_o| \tag{8}$$

Using the Chebyshev metric, the action corresponding to the minimal SQ-value is considered the greedy action in state s, i.e. $greedy_{a'}(s')$:

$$greedy_{a'}(s) = \min_{a'} SQ(s,a') \tag{9}$$

2.3 Related work on methods to identify the convex hull

There are few value-iteration algorithms that approximate the convex parts of the Pareto optimal set. The work by Barrett and Narayanan [6] is a model-based method that aims to learn every deterministic policy that lies on the convex hull of the Pareto optimal set. More precisely, their algorithm learns in parallel all optimal policies under a set of fixed linear weight combinations of the reward components. Their convex hull value iteration algorithm uses a series of translation, scaling and summing operations to retrieve the Q-values that are maximal for some set of linear preferences to identify the convex hull. These operations are very costly in terms of computational complexity which makes extensions to larger and higher dimensional environments computationally very demanding.

In [7], an algorithm is proposed that calculates a piecewise linear spline representation of the value function over the actions in a bi-objective environment. More precisely, it calculates a list of trade-off weights together with a list of their policies from batch data. The algorithm was recently extended to environments with three objectives in [8], using triangulation geometry. Extending their work to more than three objectives is definitely not straightforward. The authors conjecture that *extended algebraic decision diagrams* might work for higher dimensional spaces, but they state themselves that there is no guarantee these methods would provide a computationally feasible solution.

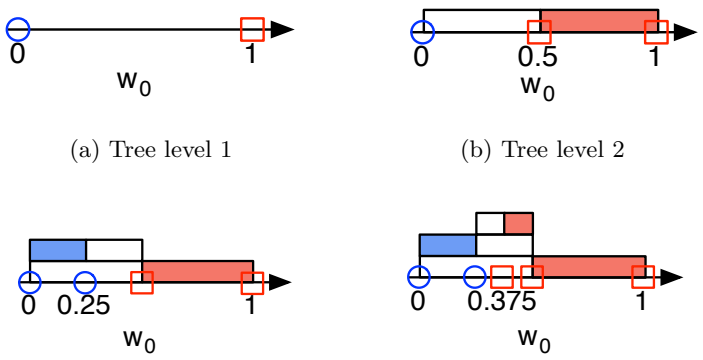
3 Efficient weight search

Current techniques that aim to identify or approximate convex regions in the Pareto optimal set are limited to model-based or batch learning approaches, which are furthermore not easily extendible to higher dimensional spaces, i.e. environments with three or more objectives. In this section, we present a tree-based search algorithm that efficiently explores the weight space for linear scalarization in order to identify all policies located on convex regions of the Pareto optimal set.

Our tree-search algorithm exploits two properties of convex linear combinations. First, we know that the weight space for linear scalarization is always $(m-1)-$dimensional, with m the number of objectives. This is due to the fact that all weights must sum to 1, and therefore setting all but one weight is enough to also fix the value for the last weight. Selecting different weight assignments $\mathbf{w} = \{w_0, w_1, ..., w_{m-1}\}$ transforms the MOMDP into MDPs with different Pareto optimal policies as optimal policy.

Another important aspect of convex Pareto optimal sets is that the set of policies that is optimal for a weight $w_o \in [0, 1]$ are precisely those policies whose line-representations lie on the upper convex envelope of the Q-functions [9], i.e. if for two different weight assignments a policy is optimal, it is also optimal for the weight assignments on the line between them.

The aim of our method is to take benefit from these properties and exploit the structure of convex hulls to identify the weight space regions that are associated with a particular Pareto optimal policy, and specifically by recursively refining the boundaries of these regions.



(a) Tree level 1 (b) Tree level 2

(c) Tree level 3 (d) Tree level 4

Fig. 1. A vizualization of search through the weight space for a bi-objective problem. Search is only performed on weight w_0 (x-axis), since $w_1 = 1 - w_0$. Different symbols and colors identify different Pareto optimal policies.

The algorithm starts by solving the MOMDP using linear scalarized Q-learning with the weight combinations that define the boundaries of the entire weight space, which are e.g. for a bi-objective problem (and therefore one-dimensional weight space), the two extrema: $(w_0 = 0; w_1 = 1)$ and $(w_0 = 1; w_1 = 0)$. As a running example, we visualize the search process for a bi-objective toy problem in Figure 1(a), with different symbols and colors identifying different Pareto optimal policies. If the discounted cumulative rewards of the learned

policies (with learning converged) are the same, our search stops, and we can conclude that there is only one Pareto-optimal policy, therefore no concave regions in the Pareto optimal set, and that the objectives are not conflicting. Note that different policies can be optimal for a given weight assignment, but their discounted cumulative rewards must be the same.

If the discounted cumulative rewards for the policies found are different, as is the case in our toy problem, the region bounded by these weight assignments is split into equal parts. In the one-dimensional weight space case, the interval is split in two, and the scalarized MOMDP is again solved with the weight values at the split point, see Figure 1(b). Thus, we build a tree (a binary tree in the bi-objective case) of weight space regions. With breadth-first search, we efficiently search the tree to identify the weight space regions associated with different policies, by only expanding nodes (splitting regions) if some of the boundary points yield different policies. After four iterations on our toy problem, two Pareto optimal policies are identified, with the intervals currently estimated at $[0, a]$ and $[a, 1]$ respectively, with a somewhere between 0.25 and 0.375 . Further refinements of the boundary may reveal new intervals lying in the $(0.25, 0.375)$, leading to other Pareto-optimal policies, or the boundary may simply be refined ad infinitum.

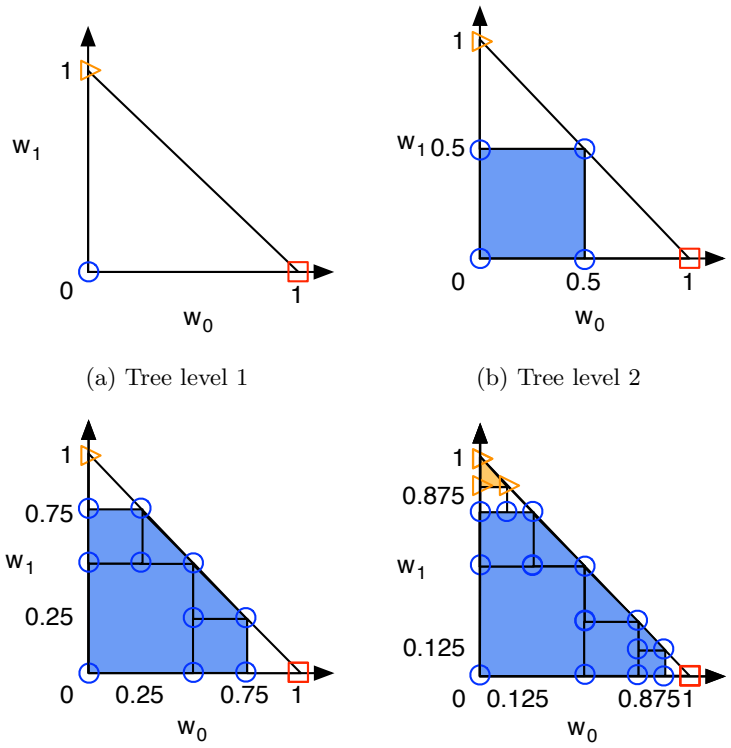
For higher dimensional spaces ($m > 2$), the binary tree becomes a quad-tree, octree, etc. Whereas for splitting a one-dimensional interval, evaluating one new point is enough, the number of split points increases with the number of objectives. For a three objective problem, the number of split points is 5 (one in the middle of each edge of the square, and one in the middle). An example search in a three objective environment is visualized in Figure 2. Note that the upper-right part of the weight space is infeasible, since in that region $w_0+w_1 > 1$.

In general, for an m-objective environment, the number of split points of an $(m-1)-$dimensional hypercube is:

$$1 + \sum_{l=1}^{m-2} 2^{m-1-l} \binom{m-1}{l} = 1 + \sum_{l=1}^{m-2} 2^{m-1-l} \frac{(m-1)!}{l!(m-1-l)!} \tag{10}$$

which is basically counting the lower dimensional geometrical figures ($l = 1$ for edges, $l = 2$ for faces, $l = 3$ for cells, etc.) of which the $(m-1)-$dimensional hypercube consists. Note that instead of having m in the equation, we have $m-1$ because the weight-space has one dimension less than the objective-space. For example, for an environment with four objectives, the search tree is an octree and the geometrical figures we are splitting are cubes. In that case, the number of split points to be evaluated is equal to the number of edges (one split point in the middle of each edge), plus the number of faces (one split point in the middle of each face), plus 1 split point for the middle of the cube:

$$1 + 12 + 6 = 19 \tag{11}$$



(a) Tree level 1 (b) Tree level 2

(c) Tree level 3 (d) Tree level 4

Fig. 2. A vizualization of search through the weight space for a tri-objective problem. Search is only performed on weights w_0 (x-axis) and w_1 (y-axis), since $w_2 = 1 - w_0 - w_1$. Different symbols and colors identify different Pareto optimal policies.

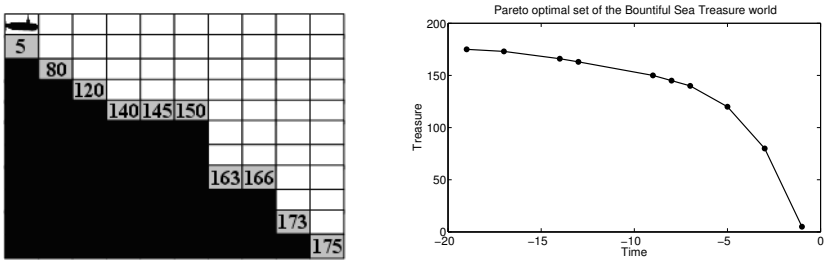
4 Experiments

In this section, we will perform an empirical evaluation of the search algorithm on several MORL benchmark instances.

4.1 Testing environments

The Bountiful Sea Treasure world. The Bountiful Sea Treasure (BST) environment is an adaptation of the Deep Sea Treasure world [10], with the reward structure altered to yield a convex Pareto optimal set. The problem is an episodic task where an agent controls a submarine, searching for undersea treasures. The world consists of a 10 x 11 grid where 10 treasures are located, with larger treasures as the distance from the starting location increases. A visualization of the environment and its Pareto optimal set is shown in Fig. 3. At each time step, the agent can move into one of the cardinal directions (up, down, left,

right). The two objectives are the time needed to reach the treasure and the treasure value itself, which are to be minimized and maximized, respectively[2]. The shortest path to each treasure is an element of the Pareto optimal set, and the shape of the Pareto optimal set is entirely convex (Fig.3(b)). The world is named bountiful because the convexification of the Pareto optimal set required assigning higher treasure values to most of the treasure locations as compared to the original Deep Sea Treasure world.



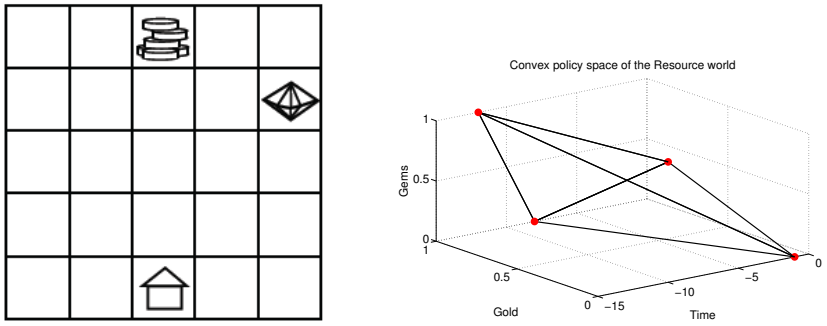
(a) The Bountiful Sea Treasure world (b) The Pareto optimal set of policies

Fig. 3. Bountiful Sea Treasure world

The Bountiful Sea Treasure 3D world. Additionally, we extended the two-dimensional BST by adding a third objective, called *pressure*. This pressure objective relates to the depth of the submarine and a penalty is given linear to this depth, i.e. at every step, the agent receives a reward of −depth. As a result, the Pareto optimal set still consists of 10 optimal policies, but instead of allowing for any shortest path to a treasure (Manhattan distance), the submarine must travel as long as possible in the shallowest water before making its descent to a particular treasure.

The Resource world. In this environment, adapted from [6], the agent goes on a quest to bring gold and gems to its home. Upon returning to its starting location, the goods he acquired are analyzed. The possible gathering outcomes are bringing home nothing, gold, gems or gold and gems. The three objectives are time, gold, and gems, and there are 4 Pareto optimal policies in this environment, i.e. a policy that returns to the home location as fast as possible without carrying anything, a policy that gathers the gold and returns as fast as possible, a similar one for the gems objective, and a policy that gathers both resources as fast as possible. Figure 4 vizualizes the Resource world and depicts the convex hull these policies represent.

[2] Traditionally, single-objective reinforcement learning solves a maximization problem. If the problem at hand concerns a minimization of one of the objectives, negative rewards are used for that objective to transform it into a maximization problem.



(a) The Resource world (b) The Pareto optimal set of policies, a convex hull

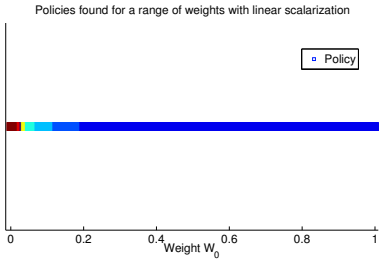
Fig. 4. Resource world

4.2 Results

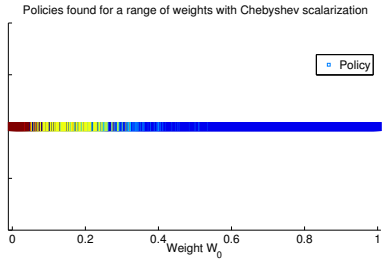
We now present the results of our search technique on these testing environments. We compare uniform random weight selection for linear scalarization and a popular non-linear scalarization function, i.e. the Chebyshev metric, which typically obtains good results for concave regions in the Pareto optimal set [1, 11], as well as our tree-based search with linear and Chebyshev scalarization. We analyze the results by looking at the number of elements obtained from the Pareto optimal set as the search process progresses. We do not compare with the batch learning method of [7], which requires complex geometrical calculations, and is not applicable to multi-objective problems with more than three objectives.

In a first experiment, we explore the bi-objective Bountiful Sea Treasure world. To give an idea about the mapping between weights and Pareto optimal policies, we show in Fig. 5(a) and (b) a sweep through the weight space for linear and Chebyshev scalarization respectively, identifying different policies with different colors. Note that for linear scalarization, there are clearly delineated intervals in the weight space for which specific policies are optimal. Yet, a large part of the weight space leads to the same policy being optimal (dark blue), and the other policies are only optimal in restricted areas of the weight space, showing that uniform sampling of the weight space will be biased towards this one policy. The mapping between the weight space and policies using Chebyshev scalarization is very fragmented, resulting from the non-linear Chebyshev metric not yielding a deterministic mapping. Instead of finding clearly bounded intervals for each policy, we see most policies scattered through the weight space.

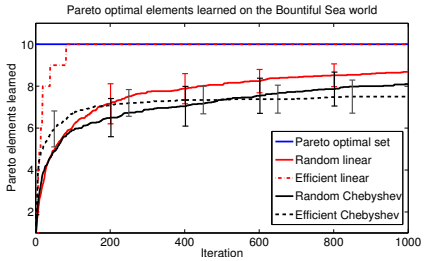
In Fig. 5(c), we depict a running sum of the number of Pareto optimal policies obtained during the search process, averaged over 50 runs. Note that our (deterministic) method efficiently searches the weight space and was able to obtain every element of the Pareto optimal set in under 100 iterations (one iteration equals solving an MDP with a single weight combination). The random search method performs worse as it only finds 8 Pareto optimal policies after 1000 iterations. The fact that the random method is still able to obtain 8 poli-



(a) Weight space with policies found using linear scalarization



(b) Weight space with policies found using Chebyshev scalarization

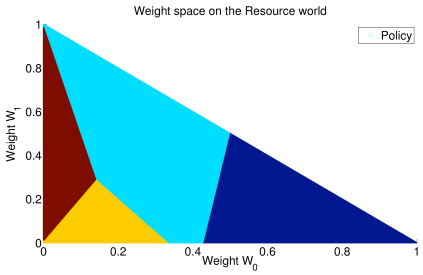


(c) Search performance (number of Pareto optimal policies identified as a function of the number of weight combinations tried)

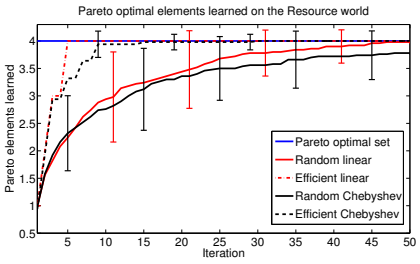
Fig. 5. Bi-objective Bountiful Sea world

cies is due to the still somewhat reasonable sizes of most of the weight space regions leading to different policies. In the three objective version of this world, we will see that the difference with the random method becomes much larger, as the size of most regions shrinks dramatically. The Chebyshev scalarization function however was not able to find every element on the convex hull and found around 7 optimal policies on average. Also, applying the tree-search method on top of the Chebyshev scalarization function is not very effective. Paradoxically, because Chebyshev scalarization is quite robust with respect to specific weight settings [1], potentially finding various policies for the same weight assignment in different runs (usually a good property, achieving greater spread), it cannot be relied on in a search procedure as that proposed in this paper, since the search procedure relies on a deterministic relationship between weights and policies.

In the three-objective Resource world, the weight space in Fig. 6(a) is two-dimensional, as only w_0 and w_1 need to be set, with the assignment of w_2 subsequently being fixed, as $\sum_{o=0}^{m} w_o = 1$. From this figure, we can clearly see the four large, distinct regions associated with the four Pareto optimal policies. We hypothesize that uniform random sampling the weight space will be much more efficient in this world. In Fig. 6(b) we summarize the search performance on this benchmark instance. We notice that, although there are no dramatic differences in size between the various regions, the random search on average neetds more than 50 iterations. On the other hand, the tree-based search algorithm only re-



(a) Weight space with associated Pareto optimal policies



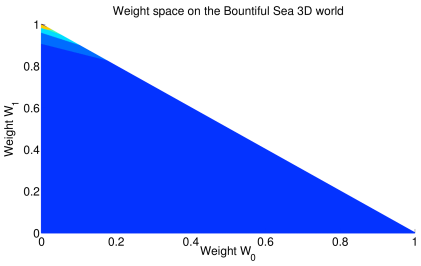
(b) Search performance (number of Pareto optimal policies identified as a function of the number of weight combinations tried)

Fig. 6. Resource world

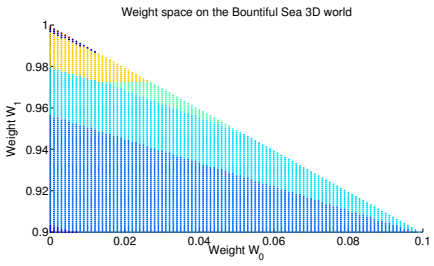
quires 5 iterations. We also note that applying the search method on top of the non-linear Chebyshev function is able to improve the search performance of the algorithm (although not over the linear scalarization). This is the case because the world is in essence relatively easy and the areas under which the different policies are optimal are even large enough for a random search technique to perform well. Nevertheless, our smart method efficiently found all elements at a minimum cost of resources.

The Bountiful Sea Treasure 3D world is the most challenging environment of the three discussed here. As is clear from the weight space in Fig. 7(a), a single policy is optimal in 95% of the space, while the other optimal policies are only obtained for a very specific set of weights (see the zoomed-in representation in Fig. 7(b)). Even using a brute-force search of the weight space, trying all weight combinations with increments of 0.001, only 7 of the 10 Pareto optimal policies are identified. This again shows that uniformly sampling the weight space to identify the convex parts of the Pareto optimal set can be a very inefficient procedure indeed.

In Fig. 8(a), we show the search performance of the different search algorithms. It is clear that a random search strategy, be it using a linear or Chebyshev scalarization function, only finds a limited subset of the convex hull. The combination of the Chebyshev learner with the tree-search method again improves the search process, but on average only 7 out of 10 elements are learned. Once more, our tree-based search algorithm is able to identify all Pareto optimal policies, although requiring many more evaluations than on previous worlds, due to the extremely small size of the weight space regions associated with some of the Pareto optimal policies (especially the three that are never found by brute-force, random, or tree-based Chebyshev search). Note that there is some stochasticity present in the search process, as learning with weight combinations near boundaries has not always completely converged, resulting in a misclassification of the weight combination.



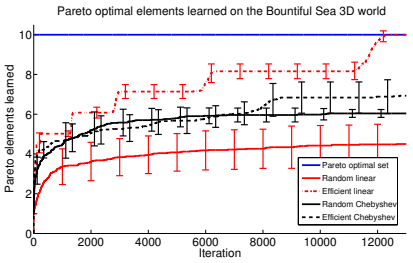
(a) Weight space with associated Pareto optimal policies. Only 7 of 10 Pareto optimal policies are identified using an exhaustive brute-force search.



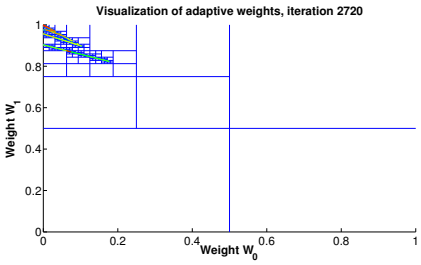
(b) Weight space zoomed in.

Fig. 7. Bountiful Sea Treasure 3D

Fig. 8(b) shows the resulting quad-tree, with color indicating the depth of the tree, red being the deepest levels. The figure clearly shows that the search algorithm is able to concentrate on the regions containing multiple policies, while avoiding those other regions that will not contain as yet unidentified policies. Figure 9 also confirms this, by showing which weight combinations are evaluated by the tree-based search algorithm. The algorithm intensively explores the boundaries between different regions, and ignores the interiors of these regions, as they will not yield new policies. Refining the boundaries between regions can help find new (very small) regions that are associated with other Pareto optimal policies.

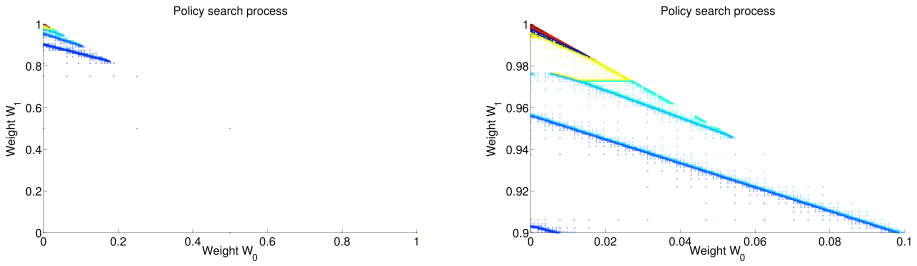


(a) Search performance (number of Pareto optimal policies identified as a function of the number of weight combinations tried)



(b) Resulting Quad-tree

Fig. 8. Bountiful Sea Treasure 3D



(a) Weight space search (each point is a weight combination that is evaluated).

(b) Weight space search zoomed in (exploration is focused on the boundaries between regions)

Fig. 9. Bountiful Sea Treasure 3D

5 Conclusions

In this paper, we proposed a search algorithm that can automate the fine-tuning of weights to identify the Pareto optimal set's entire convex hull, using a multi-objective reinforcement learning technique. We showed that a uniform sampling of the weight space often does not lead to a uniform sampling of the convex parts of the Pareto optimal set, and that therefore system designers can benefit from a search algorithm, instead of relying on intuition and wasting computation power when tuning weights by hand, as is so often done. Our algorithm performs breadth-first search on a tree-based partitioning of the weight space, and explores those regions that contain the boundaries between already identified policies. We experimentally showed it to outperform a uniform random sampling of the weight space, and that it can automate the process of tuning weights to find policies that achieve a desired trade-off between the different objectives. Even though the number of split points for the algorithm increases exponentially in the number of objectives, it is still by far preferable over uniform random sampling, since uniform sampling of high dimensional spaces is notoriously unrepresentative of the space. Directed search is always to be preferred over random sampling, even on the easiest of problems, as seen on the Resource world. Our tree-search technique is furthermore easily extendible to environments with more than three objectives in contrast to the methods in [6, 7], as our method only requires an additional number of weight points to be evaluated and their results stored in the tree.

In future work, we intend to investigate heuristic search processes that can improve over the current breadth-first search process, by helping decide which nodes of the tree are most likely to contain regions leading to undiscovered Pareto optimal policies, looking at distance metrics for policies. Furthermore, we are interested in leveraging information learned in initial iterations of the search algorithm, to speed up learning in later iterations. For example by storing the MDP reward and transition samples in the first few iterations, and building a Dyna-Q model to speed up learning in later iterations. This methodology

would allow the search algorithm to transition from initially learning completely on-line (which might be expensive), to model-based and batch learning later on in the search process. Another alternative to the process described here, is to use k-d trees, instead of quad-trees, octrees, etc., and not splitting regions symmetrically, but based on an inspection of the bounding weight assignments, and neighbouring regions.

6 Acknowledgement

Kristof Van Moffaert is supported by the IWT-SBO project PERPETUAL (grant nr. 110041). Tim Brys is funded by a Ph.D grant of the Research Foundation Flanders (FWO).

References

1. Van Moffaert, K., Drugan, M.M., Nowé, A.: Scalarized Multi-Objective Reinforcement Learning: Novel Design Techniques. In: 2013 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, IEEE (2013)
2. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. Adaptive Computation and Machine Learning. Mit Press (1998)
3. Watkins, C.: Learning from Delayed Rewards. PhD thesis, University of Cambridge,England (1989)
4. Tsitsiklis, J.: Asynchronous stochastic approximation and q-learning. Journal of Machine Learning **16**(3) (1994) 185–202
5. Vamplew, P., Yearwood, J., Dazeley, R., Berry, A.: On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In: Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence. AI '08, Berlin, Heidelberg, Springer-Verlag (2008) 372–378
6. Barrett, L., Narayanan, S.: Learning all optimal policies with multiple criteria. In: Proceedings of the 25th international conference on Machine learning. ICML '08, New York, NY, USA, ACM (2008) 41–47
7. Lizotte, D.J., Bowling, M., Murphy, S.A.: Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In: Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML). (2010) 695–702
8. Lizotte, D.J., Bowling, M., Murphy, S.A.: Linear fitted-q iteration with multiple reward functions. Journal of Machine Learning Research **13** (2012) 3253–3295
9. Ehrgott, M.: Multicriteria Optimization. Lectures notes in economics and mathematical systems. Springer (2005)
10. Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., Dekker, E.: Empirical evaluation methods for multiobjective reinforcement learning algorithms. Machine Learning **84**(1-2) (2010) 51–80
11. Van Moffaert, K., Drugan, M.M., Nowé, A.: Hypervolume-based multi-objective reinforcement learning. Lecture Notes in Computer Science, Evolutionary Multi-Criterion Optimization (EMO 2013) (2013)