Scalarized Multi-Objective Reinforcement Learning: Novel Design Techniques

Kristof Van Moffaert
Department of Computer Science
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium
Email: kvmoffae@vub.ac.be

Madalina M. Drugan
Department of Computer Science
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium
Email: mdrugan@vub.ac.be

Ann Nowé
Department of Computer Science
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium
Email: anowe@vub.ac.be

***Abstract*—In multi-objective problems, it is key to find compromising solutions that balance different objectives. The linear scalarization function is often utilized to translate the multi-objective nature of a problem into a standard, single-objective problem. Generally, it is noted that such as linear combination can only find solutions in convex areas of the Pareto front, therefore making the method inapplicable in situations where the shape of the front is not known beforehand, as is often the case. We propose a non-linear scalarization function, called the Chebyshev scalarization function, as a basis for action selection strategies in multi-objective reinforcement learning. The Chebyshev scalarization method overcomes the flaws of the linear scalarization function as it can (i) discover Pareto optimal solutions regardless of the shape of the front, i.e. convex as well as non-convex , (ii) obtain a better spread amongst the set of Pareto optimal solutions and (iii) is not particularly dependent on the actual weights used.**

I. Introduction

Many real-life problems involve dealing with multiple objectives. For example, in network routing the objectives could consist of energy, latency, and channel capacity. When the system engineer wants to optimize more than one objective, it is not always clear which objectives are correlated and how they influence each other upon initially inspecting the problem at hand. Also, it is not sufficient to try to optimize just one objective in the routing problem 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. So deciding a priori on the importance of the different criteria might be difficult. The goal of MOO is to search the policy space and eventually find policies that simultaneously optimize one or more objectives.

A popular approach to solving MOO problems is to transform the multi-objective problem into a single-objective problem by employing *scalarization* functions. These functions provide a single score indicating the quality over a combination of objectives, which allows a simple and fast ordering of the candidate solutions. In many cases, a linear combination of the objectives is utilized, but as noted in [1], this mechanism only allows Pareto optimal solutions to be found amongst convex[1] areas of the Pareto front.

Main contributions. In our work, we analyze and argue the limitations of the linear scalarization function on both convex and non-convex environments. We will show that the linear scalarization function is unsuitable for action selection purposes in on-line reinforcement learning. Instead, we propose a novel non-linear scalarization function, i.e. the weighted *Chebyshev* scalarization function, that improves the linear scalarization function on three aspects. We empirically analyse the non-linear scalarization function and note that it allows to (i) discover Pareto optimal solutions regardless of the shape of the front, (ii) obtain a better spread amongst the set of Pareto optimal solutions and (iii) is less dependent on the actual weights used, compared to the linear scalarization function.

Outline. In Sections II and III, we discuss in more detail previous work on multi-objective optimization and reinforcement learning. Furthermore, we elaborate on our contributions in Section IV and empirically analyze them in Section VI using the experimental setting of Section V. In Section VII we draw conclusions.

II. Preliminaries

A. Multi-Objective Optimization

A multi-objective optimization problem optimizes a vector function whose elements represent the objectives. A maximization multi-objective problem is $\max \mathbf{F}(x) = \max\{f^1(x), f^2(x), ..., f^m(x)\}$, where m is the number of objectives, and f^o is the value for the o-th objective. A solution x_1 is said to Pareto *dominate* another solution x_2, $\mathbf{F}(x_2) \prec \mathbf{F}(x_1)$, iff for all objectives j, $f^j(x_2) \leq f^j(x_1)$, and there exists an objective i, for which $f^i(x_2) < f^i(x_1)$.

B. Multi-Objective Reinforcement Learning

Reinforcement learning (RL) involves an agent operating in a certain environment and receiving reward or punishment for its behaviour. In single-objective learning the goal of the agent is to find a mapping from states to actions that maximizes

[1]An object is convex if for every pair of points within the object, every point on the straight line segment that joins them is also within the object. If not, the object is non-convex or concave.

the reward received over time. In the following sections, we give a brief overview of reinforcement learning and how it is extended to multi-objective environments.

Markov decision process. The principal structure for RL is a Markov Decision Process (MDP). An MDP can be described 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 agent. Each combination of starting 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 discounted reward J^π is maximized:

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

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.

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

So in order to find the optimal policy, one can learn this Q-function and then use greedy action selection over these values in every state. Watkins described an algorithm to iteratively approximate Q^*. In the Q-learning algorithm [2] a table consisting of state-action pairs is stored. Each entry contains the 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 accordingly to following update rule:

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

where α_t is the learning rate at time step t and r 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^*.

Multi-objective MDPs. In MOO, the principal structures are multi-objective MDPs or MO-MDPs [3]. These extend MDPs by replacing the single reward signal by a vector of reward signals, i.e. $\vec{R}(s_i, a_i) = (R_1(s_i, a_i), \ldots R_m(s_i, a_i))$, where m represents the number of objectives. Since the reward vector consists of multiple components, each representing different objectives, it is very likely that conflicts arise when trying to optimize one or more objectives. In such case, trade-offs between these objectives have to be learned, resulting in a set policies. The set of optimal policies for each objective or a combination of objectives is referred to as the *Pareto optimal set*.

III. Related work

There are several multi-objective reinforcement learning (MORL) approaches proposed in literature. For instance, [4] suggests a MORL method that uses a lexicographic ordering of the objectives and by placing minimal thresholds on certain objectives, policies are discovered that take into account these constraints. Furthermore, [5] suggests a batch Convex Hull Value Iteration algorithm that learns all policies in parallel, defining the convex hull of the Pareto optimal set. Additionally, [6] also proposes a batch MORL approach, based on the linear scalarization function, to identify which actions are favoured in which parts of the objective space. Notwithstanding their results, they all consists of *off-line* algorithms, which involve sweeps over a set of collected data. Therefore, the aspects of these algorithms on using and adapting their policy during the learning process (i.e. *on-line* learning) were not studied.

IV. Contributions

In this section, we elaborate on our contributions to the MORL domain. More precisely, we analyze the drawbacks of the linear scalarization function as a basis for an action selection strategy on two benchmarks. To address these shortcomings, we propose the weighted Chebyshev scalarization function as an alternative mechanism. Later, we further present a general framework for MORL, based on Q-learning, in which any scalarization function can be incorporated.

A. The linear scalarization function

In single-objective learning, the agent's table is used to store the expected reward for the combination of state s and action a, i.e. $\hat{Q}(s, a)$. In a multi-objective setting, the Q-table is extended to incorporate objectives, i.e. $\hat{Q}(s, a, o)$. Thus, the expected rewards for each state, action and objective can be stored, retrieved and updated separately.

An important aspect of multi-objective optimization consists of how the actions are selected, based on different objectives that can be complementary, conflicting or independent. A common approach is to employ scalarization functions as a scoring mechanism for action selection strategies. More precisely, these functions transform a multi-objective problem into a single objective problem by performing a function over the objectives to obtain a combined score for an action a for different objectives o. This single score can then be used to evaluate the particular action a. Given these scores, one can utilize the standard action selection strategies of single-objective reinforcement learning, such as ϵ-greedy and Boltzmann, to decide which action to select. Most scalarization functions 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 greater or lesser 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 $\sum_{o=1}^{m} w_o = 1$. The most common function is the linear scalarization function [7] (LS) because of its simplicity and straightforwardness. More precisely, for a multi-objective solution x, a weighted-sum is performed over

each objective function (i.e. f_o with $o = 1 \dots m$) and their corresponding weights to obtain the score of x, i.e.

$$LS(x) = \sum_{o=1}^{m} w_o \cdot f_o(x) \tag{4}$$

In the case of multi-objective reinforcement learning, the objective functions f are considered the $\hat{Q}(s, a, o)$-values. As a result of applying the scalarization, scalarized Q-values or SQ-values are obtained:

$$SQ(s, a) = \sum_{o=1}^{m} w_o \cdot \hat{Q}(s, a, o) \tag{5}$$

The action corresponding to the largest weighted-sum or SQ-values is considered the greedy action in state s, formulated in Eq. 6.

$$greedy_{a'}(s) = \max_{a'} SQ(s, a') \tag{6}$$

The linear scalarization function has a fundamental limitation as it can only find policies that lie in convex regions of the Pareto optimal set [1]. Taken into account the fact that in most cases the shape of the optimal set is not known beforehand or the fact that some convex Pareto optimal sets have local concavities, applying this function for action selection purposes, would imply that some Pareto dominating actions will not be discovered.

B. The Chebyshev scalarization function

Our novel alternative as a mechanism to evaluate actions with multiple objectives consists of using L_p metrics [8]. In detail, L_p metrics measure the distance between a point in the multi-objective space and a utopian point z^*. In our setting, we measure this distance to the value of the objective functions f for each objective o of the multi-objective solution x, i.e.

$$\min_{x \in \mathbb{R}^n} L_p(x) = \big(\sum_{o=1}^{m} w_o |f_o(x) - z_o^*|^p \big)^{1/p} \tag{7}$$

, where $1 \leq p \leq \infty$. In the case of p = ∞, the metric is also called the weighted L_∞ or the Chebyshev metric and is of the form:

$$\min_{x \in \mathbb{R}^n} L_\infty(x) = \max_{o=1 \dots m} w_o |f_o(x) - z_o^*| \tag{8}$$

In terms of action selection mechanism, the objective function values f are replaced by $\hat{Q}(s, a, o)$-values to obtain the scalarized Q-value or SQ-value, for state s and action a:

$$SQ(s, a) = \max_{o=1 \dots m} w_o \cdot |\hat{Q}(s, a, o) - z_o^*| \tag{9}$$

Resulting from the formula in Eq. 8, using the Chebyshev metric, the action corresponding to the minimal SQ-value is considered the greedy action in state s, i.e. $greedy_{a'}(s')$:

$$greedy_{a'}(s) = \min_{a'} SQ(s, a') \tag{10}$$

The reference point z^* is a parameter that is being constantly adjusted during the learning process by recording the best value so far for each objective o, plus a small constant τ, i.e. $z_o^* = f_o^{best}(x) + \tau$. The Chebyshev method has already proven its effectiveness in the evolutionary computation domain [8], where it is considered more powerful that the linear scalarization function, but was, until now, not evaluated in reinforcement learning.

Fig. 1. Scalarized ϵ-greedy strategy, *scal*-ϵ-greedy()

1: $SQList \leftarrow \{\}$
2: **for** each action $a_i \in A$ **do**
3: $\quad \vec{o} \leftarrow \{\hat{Q}(s, a_i, o_1), \dots, \hat{Q}(s, a_i, o_m)\}$
4: $\quad SQ(s, a) \leftarrow scalarize(\vec{o})$ ▷ Scalarize $\hat{Q}$-values
5: $\quad$ Append $SQ(s, a)$ to $SQList$
6: **end for**
7: **return** ϵ-greedy($SQList$)

C. Scalarized MORL framework

In the previous sections, we have presented the linear and Chebyshev scalarization functions as mechanisms to evaluate actions in multi-objective reinforcement learning environments. We will now present how every scalarization function can be incorporated into a general Q-learning algorithm for multi-objective purposes.

In Fig. 1, we present a general outline for action evaluation in multi-objective environments using scalarization functions. At line 3, we retrieve the $\hat{Q}$-values for each objective o of action a. Additionally, at line 4, the $scalarize$ function can be instantiated by any scalarization function (i.e. a linear or non-linear function) to obtain a single score for the quality of the combination of state s and action a, i.e. the $SQ(s, a)$ value. Thus, we transform the multi-objective problem into a single-objective problem and store the individual evaluations in $SQList$. Furthermore, an action selection from single-objective learning is utilized to decide which action a_i to select in state s, based on the obtained scores. At line 7, we specify this by adopting the ϵ-greedy strategy.

The new multi-objective Q-learning algorithm (MO Q-learning) is presented in Fig. 2. At line 1, the $\hat{Q}$-values for each triple of states, actions and objectives are initialized. Each episode, the agent starts in state s (line 3) and chooses an action based on the multi-objective action selection strategy of Fig. 1 at line 5. Upon taking action a, the environment transitions the agent into the new state s' and provides the vector of rewards $\vec{r}$.

As the Q-table has been extended to incorporate a separate value for each objective, these values are updated for each objective individually. The single-objective Q-learning update rule is extended for a multi-objective environment at line 10 of Fig. 2, where α represents the learning rate and γ the discount factor. More precisely, the $\hat{Q}$-values for each triple of state s, action a and objective o are updated using the corresponding reward for each objective, $\vec{r}(s, a, o)$, into the direction of the $\hat{Q}$-value of the best scalarized action of the next state s', i.e. $greedy_{a'}(s')$. Convergence is guaranteed as long as each action and state is sufficiently sampled.

Fig. 2. MO Q-learning algorithm

Initialize $\hat{Q}(s, a, o)$ arbitrarily
for each episode T **do**
    Initialize state s
    repeat
        Choose action a from s using policy derived from $\hat{Q}$-values (e.g. *scal*-ϵ-greedy)
        Take action a and observe state $s' \in S$ and reward vector $\vec{r} \in \mathbb{R}$
        $greedy_{a'}(s'), \leftarrow$ Call scal. greedy action selection
        for each objective o **do**
            $\hat{Q}(s, a, o) \leftarrow \hat{Q}(s, a, o) + \alpha[\vec{r}(s, a, o) + \gamma\hat{Q}(s', greedy_{a'}(s'), o) - \hat{Q}(s, a, o)]$
        end for

        $s \leftarrow s'$ ▷ Proceed to next state
    until s is terminal
end for

V. Experimental setting

Recently, [9] proposed empirical evaluation techniques for multi-objective RL, together with a few benchmark instances. In the following sections, we will perform an empirical analysis on two of these benchmark environments using the multi-objective Q-learning algorithm, employed with the linear and Chebyshev functions as scoring mechanisms for the action selection strategy.

A. Benchmark environments

More precisely, [9] proposed the *Deep Sea Treasure* and the *Multi-Objective Mountain Car* benchmark environments for multi-objective reinforcement learning. Together with the description of the environments, they provided the Pareto optimal sets which can be used to evaluate the scalarization functions in detail.

Deep Sea Treasure world. The Deep Sea Treasure world concerns 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 increasing values as the distance from the starting location increases. 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[2] and maximized, respectively. This world is particularly suited for multi-objective reinforcement learning as the optimal path to each treasure is an element of the Pareto optimal set. As a result, the shape of the Pareto optimal set is entirely concave. A visualization of the environment is depicted in Figure 3.

Multi-Objective Mountain Car. The single-objective Mountain Car world is a famous benchmark for reinforcement learning algorithms. In this world, a car is required to escape a one-dimensional valley. As the car's engine is less powerful than gravity, the vehicle must perform a series of acceleration and reversal actions to build enough potential energy to escape the valley. The action space consists of three actions: full throttle forward, full throttle backward, and zero throttle. The multi-objective version of this benchmark is challenging as it consists of three objectives that are to be optimized. The three objectives are the time required to escape the valley and the number of acceleration and reversal actions, which are all to be minimized. The Pareto optimal set contains 470 dominating policies and the maximum amount of steps allowed to reach the goal location is 500 steps. It is important to note that the shape of the Pareto optimal set has a significant portion of locally convex and non-convex areas.

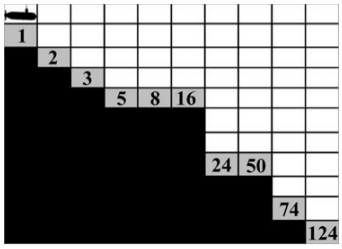

Fig. 3. A visual representation of the Deep Sea Treasure world. The agent starts each episode in the top-left position. The treasures are indicated by the grey cells, while black cells represent the seabed. Figure taken from [9].

B. Parameter setting

In the experiments, presented below, we relied on identical parameter settings for each of the testing environments. We applied an ϵ-greedy exploration strategy with ϵ set to 0.1 and the $\hat{Q}$-values were initialized optimistically for each objective. The learning rate α was set to 0.1 and the discount factor γ to 0.9. The τ parameter[3] of the Chebyshev mechanism was specified to 4.0 and 1.0 for the Deep Sea Treasure world and the MO Mountain Car world, respectively. Results are collected and averaged over 50 trials of each 500 iterations.

[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 also into a maximization problem.

[3]These values were found in a parameter sweep to create the best-performing algorithm, though the results are not very sensitive to particular values for τ.

TABLE I
THE WILCOXON RANK TEST DENOTED A SIGNIFICANT DIFFERENCE BETWEEN THE HYPERVOLUME OF THE OBTAINED POLICIES OF THE TWO ALGORITHMS ON BOTH BENCHMARK PROBLEMS.

	avg. hv linear	avg. hv Chebyshev	p-value	Significant?
DST	762	938.29	$4.4306e^{-19}$	$\surd$
MC	15727946.18	23028392.94	$2.6528e^{-14}$	$\surd$

VI. EXPERIMENTS

In the following experiments, we will analyze the performance of both scalarizations in the MO Q-learning framework, presented in Section IV-C. First, we provide results for both scalarization functions when weights are varied across the entire range of $[0, 1]$ in Section VI-A and VI-B. Thus, we analyze the ability of each algorithm to discover all elements of the Pareto optimal sets, when the time needed to conduct the parameter sweep is no issue.

Furthermore, in Section VI-C, we examine the case when it is not feasible to evaluate the performance of an algorithm multiple times under different circumstances, one after another. For example, imagine the case where a robot is operating in a real-life multi-objective environment. Then, it is not doable to recompile and test its program with multiple weight settings before an acceptable set of policies is learned. Moreover, we determine the ability of both scalarization functions to discover multiple Pareto optimal solutions when we keep the weight parameter fixed.

In Section VI-D, we elaborate on the diversity of the policies that are obtained by both scalarization functions. More diverse policies imply that the end-user has a larger choice of distinct possibilities to solve a particular problem.

A. Quality indicator comparison

In multi-objective research, quality indicator studies are a popular approach for conducting algorithm comparisons. The performance indicators utilized are the (inverted) generational distance , the generalized spread indicator, the cardinality and the hypervolume distance. The first three are to be minimized, while the last two are to be maximized.

The generational distance and the inverted generational distance were both proposed by [10]. The former measures how far the elements in the set of Pareto approximations, learned by the algorithms, are from those in the Pareto optimal set. The latter calculates how far each element in the Pareto optimal set is from those in the set of Pareto approximations. The spread indicator [11] on the other hand is a diversity metric that measures the extent of spread achieved amongst the obtained solutions. The cardinality measure simply counts the number of elements found in the Pareto set. The hypervolume metric is a commonly accepted quality measure for comparing approximations of Pareto sets in the field of MOO [12]. In our case, the elements are policies and denoted by their collected reward for each objective throughout the episode. The hypervolume metric calculates the volume of the area between a reference point and the Pareto set obtained by a specific algorithm. In Fig. 4, a visual representation of the hypervolume measure is provided. For three bi-objective solutions S_1, S_2 and S_3 the area is calculated, given a reference point r.

TABLE II
THE REFERENCE POINTS USED TO CALCULATE THE HYPERVOLUME (HV) INDICATOR AS QUALITY ASSESSMENT TOOL FOR EACH ENVIRONMENT.

	Deep Sea Treasure	**MO-Mountain Car**
Reference point	$(-25, 0)$	$(-350, -250, -500)$

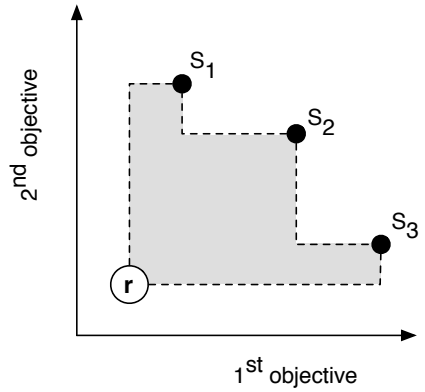

Fig. 4. The grey area represents the hypervolume obtained for the three solutions in a bi-objective environment, given the reference point r.

Table II presents the reference points for the hypervolume metric, used to quantify the learned policies in each of the benchmark environments. These values were determined by examining the bounds on the reward structure of each testing environment in a straightforward way. To be more precise, in case of a maximization problem, one can determine the minimal values for each objective and subtract a small constant to determine a suitable reference point. The results on the final policies, i.e. the Pareto approximation set obtained at the end of the learning phase, are presented in Table III.

TABLE III
FIVE QUALITY INDICATOR ON THE TWO BENCHMARKS. THE FIRST THREE INDICATORS ARE TO BE MINIMIZED, WHILE THE LATTER TWO ARE TO BE MAXIMIZED. THE BEST VALUES ARE DEPICTED IN BOLD FACE.

		Linear	**Chebyshev**
Inverted generational distance	**DST**	0.128	**0.0342**
	MC	0.012	**0.010**
Generalized spread	**DST**	**$3.14e^{-16}$**	0.743
	MC	**0.683**	0.808
Generational distance	**DST**	**0**	**0**
	MC	0.0427	**0.013824**
Hypervolume	**DST**	762	**938.29**
	MC	15727946	**23028392**
Cardinality	**DST**	2	**8**
	MC	25	**38**

Deep Sea Treasure results. On the Deep Sea Treasure (DST) world, the Chebyshev method learned 8 distinct, optimal policies out of 10 optimal policies in the Pareto optimal set. The linear scalarization function only found two elements of the Pareto optimal set. The Q-learner with the Chebyshev-based strategy obtained the best value for the hypervolume

metric. Furthermore, each of the 8 policies were an element of the Pareto optimal set, so that the generational distance is 0. Given the fact that we ran only 10 distinct weight tuples or agents, the cardinality and generational distance results are impressive. Therefore, also the distance from the policies found to the elements in the Pareto optimal set is minimized, which is being reflected in the inverted generational distance values for the Chebyshev scalarization function.

MO Mountain Car results. On the Mountain Car (MC) world, the Chebyshev method found 38 distinct Pareto dominating policies by only 64 distinct agents. As a result, it also obtained the best results for the hypervolume metric. The policies obtained by the Chebyshev method are also close to the Pareto optimal set, as is shown by both distance indicators. The linear scalarization function struggled on this complex environment and learned only 25 non-dominated policies. Additionally, the results for both distance indicators were also inferior to the results of the Chebyshev method.

A very important quality indicator, that we did not discus before, is the generalized spread. In multi-objective optimization, the goal of an algorithm is not only to find a certain amount of (optimal) policies, but also to ensure a significant spread in the multi-objective space. When the obtained results are scattered across the objective space, the end-user has a larger choice amongst very different policies. On the contrary, when policies are clustered into particular areas, often the differences between those policies are minimal and they do not provide the end-user with a diverse set of choices. For this metric, we note that Q-learner with the linear scalarization function obtained the best results as depicted by the spread indicator in Table III. This indicator could provide a distorted picture to the reader stating that the Chebyshev method is incapable of obtaining policies with a lot of diversity. The reason behind this result is the fact that the generalized spread indicator exclusively takes into account the bounding solutions of the Pareto optimal set and averages the distance to every non-dominated policy obtained by the algorithm. It is important to note that these *extreme* elements of the Pareto optimal set are the least interesting for the end-user, i.e. they maximize only one objective regardless of the values for the other objectives. What this metric lacks to take into account are the real multi-objective solutions, i.e. the trade-off or compromising solutions. Therefore, in Section VI-D, we conduct additional experiments that focus on the diversity level of the policies. Moreover, we will see that the linear scalarization function is biased towards these extreme solutions. Thus minimizing its generalized spread value, but lacking the ability to learn policies with a large variety in the objective space. First, we will examine the average performance of both scalarization functions over a range of weights.

B. Combined performance of weights

An important aspect of a multi-objective learning algorithm is its average performance for a range of weights. This experiment denotes the ability of each scalarization function to discover the entire Pareto front, when combining the results for a range of weight settings. In Fig. 5 and 6, we present the learning curve for both scalarization functions on the Deep Sea Treasure and MO Mountain Car benchmark, respectively. More precisely, for every iteration, we denote the hypervolume of the learned policies for each of the 11 and 64 agents in the environment with two and three objectives, respectively. In the Deep Sea world (Fig. 5), the Q-learner utilizing the linear scalarization function stagnates after 100 iterations and keeps on finding the same policies over each of the 50 trials (i.e. the standard deviation becomes zero). The algorithm employing the Chebyshev function learns faster and finds improved policies than the linear scalarization function. In the

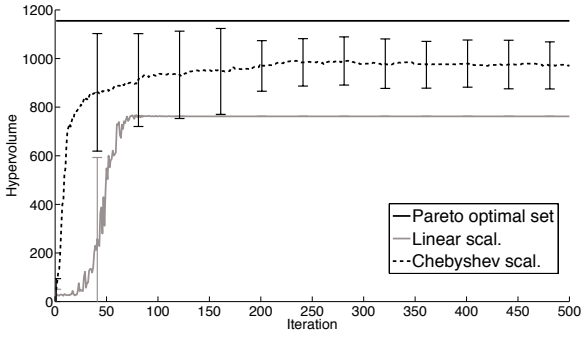

Fig. 5. The learning curve of the Q-learners using the linear and Chebyshev scalarization functions as their action evaluation methods on the DST world. The Pareto optimal set is depicted in black.

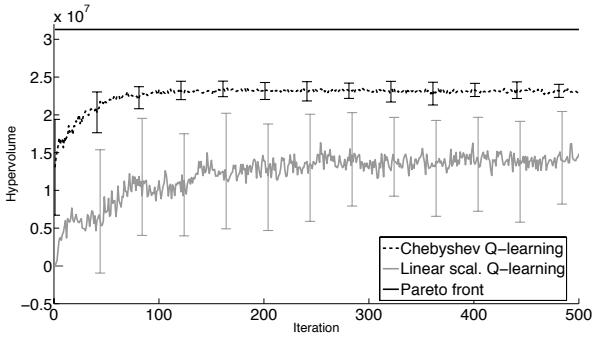

Fig. 6. The learning curve for both scalarization functions in action selection strategies on the multi-objective Mountain Car world. The linear scalarization strategy slowly learns to optimize its policies, with a lot of deviation. The Chebyshev scalarization mechanism learns faster and more smoothly.

complex three-objective mountain car world, the differences between the two scalarization methods are even clearer (Fig. 6). The Chebyshev-based Q-learner is learning faster and smoother than the Q-learner using the linear function. The learning curve of the Chebyshev function smooths out after a few hundred iterations, while the algorithm using the linear scalarization function is still fluctuating a lot. In Table I, we relied on the Wilcoxon rank [13] test to determine whether the results between the linear and the Chebyshev scalarization functions are significant.

We conclude this experiment by stating that the Chebyshev scalarization function is able to learn a wide variety of Pareto optimal solutions when providing it with different weights. Therefore, it approaches the Pareto optimal set much closer than the linear scalarization function.

C. Individual performance of weights

Combining policies over different weighting tuples, as we did in the previous experiment, is interesting to draw conclusions about an algorithm's performance when the time needed to perform the experiments is no issue. In some cases, however, it might not be possible to run a parameter sweep and collect the results. For example, for a robot is operating in a real-life environment, it is not feasible to change its parameters all the time and let it learn from scratch before a wide set of (optimal) policies is collected. On the contrary, in a multi-objective real-life setting, it would be interesting if the agent could learn a diverse set of policies for fixed parameter values (i.e. weights on objectives) without any manual intervention. Once these policies are learned, the agent can then chose from a set of high-quality, non-dominated policies to perform its multi-objective task.

In the following experiments, this is exactly what we analyze, i.e. we show the number of distinct policies learned for a fixed scalarization weight. The specific weight was chosen to focus on trade-off solutions in the Deep Sea Treasure world and was specified to $(0.4, 0.6)$, where the first coordinate represents the emphasis on the *time* objective, while the second coordinate represents the weight on the *treasure* objective. In Fig. 7(a), we depict the hypervolume of the ten dominating policies of the Pareto optimal set in the Deep Sea Treasure world, represented by points in a bi-objective space. The area in blue corresponds to the hypervolume of these ten optimal policies and is the maximal area that can be obtained in the Deep Sea Treasure world. In Fig. 7(b), the linear scalarization function always obtained the same solutions, i.e. $(-19, 124)$ for the fixed weight, resulting in the small red area. Learning once using the Chebyshev scalarization function obtained, for the same weighting coefficient, a significantly larger set of optimal policies (Fig. 7(c)). More precisely, the set consisted of 6 elements. The green area represents the hypervolume corresponding to these 6 distinct policies. The green area of (Fig. 7(c)) is clearly larger than the red volume (Fig. 7(b)), obtained using the linear scalarization function, and approaches the blue area of the Pareto optimal set (Fig. 7(a)).

We conclude that the Chebyshev scalarization function is less biased by the particular weight setting, defined a priori. In contrast to the linear scalarization function, learning once with the Chebyshev selection method for a fixed weight already allows to obtain an acceptable spread in the multi-objective space and thus also much larger set for the robot to select from.

In Fig. 8, we extend the above experiment and show the hypervolume obtained for each of the 11 scalarization weights for both scalarization functions. We clearly note that the linear scalarization function for weights 1 to 7 and 8 to 11,

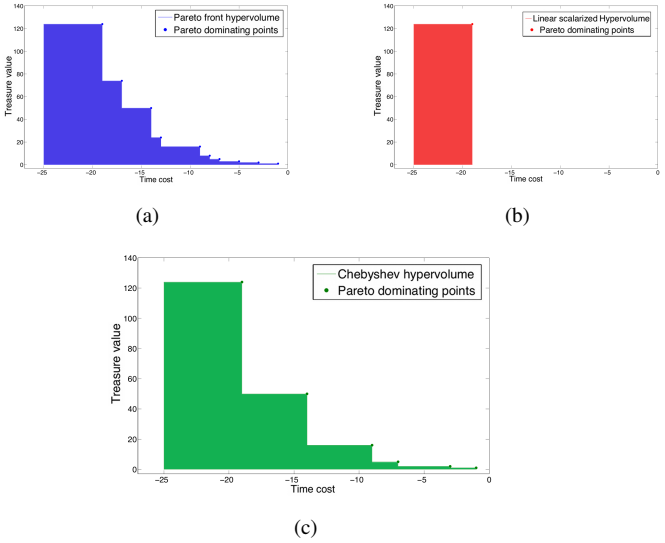

Fig. 7. An illustrative visualisation of the hypervolume measure on the Pareto optimal set (Fig. 7(a)). The linear strategy (Fig. 7(b)) found only one policy for a fixed weight, while the Chebyshev (Fig. 7(c)) function obtained 6 solutions, resulting in a larger hypervolume. Also in this experiment, the reference point r for the hypervolume calculations was set to $(-25, 0)$.

exclusively finds the same solutions. The Chebyshev function learns a large collection of policies for each individual weight. Only for weight 11, a single policy was found. This was the case when large emphasis was defined to minimize the $time$ objective and therefore the algorithm opted to always take the path to the closest (and smallest) treasure value. More precisely, beginning from the starting position (Fig. 3), the agent decides to go for the action that takes it one cell down.

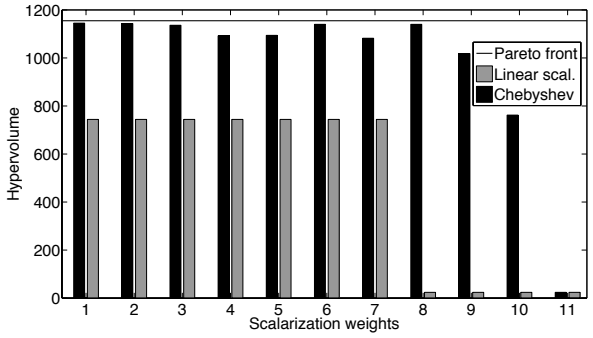

Fig. 8. For each of the 11 weights from 0 to 1, the hypervolume of the learned policies is depicted. The Chebyshev strategy uses its weight coefficients in a much better way than the linear scalarization function and found very diverse policies for almost every weight configuration on the DST world.

Similar to Fig. 8 for the Deep Sea Treasure world, we examined the performance of both scalarization functions for the 64 individual agents in the MO Mountain Car benchmark. In Fig. 9, the hypervolume for the set of obtained policies for each individual scalarization weight is depicted. Note that only for a very small set of weights, the learning algorithm utilizing

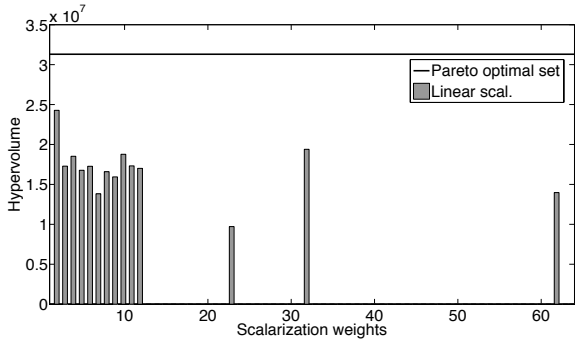

Fig. 9. The obtained hypervolume corresponding to the policies learned for each of the 64 weight tuples using the linear linear scalarization. For a large amount of weights no policies were found on the MC world.

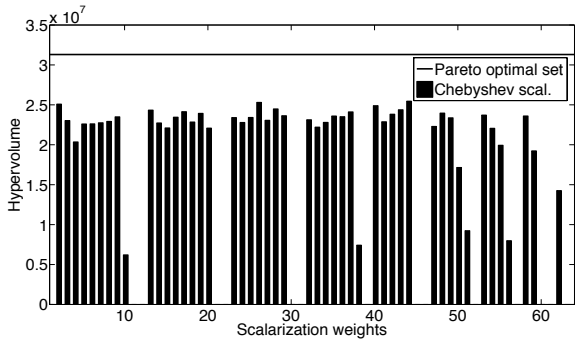

Fig. 10. For a large part of the weight tuples, the Chebyshev scalarization mechanism obtained good policies. Although, for some weight combinations, i.e. the gaps in the figure, no policies were learned.

the linear scalarization function was able to learn policies that led to escaping the valley. In the other cases, the learner did not succeed to reach the goal location within the restricted time frame of 500 time steps. The Chebyshev method was much less affected by the specified weights and was more robust in escaping the valley in almost every circumstance (Fig. 10). Although, for a particular number of weights, neither of the both methods was able to reach the top of the hill in the constrained time interval. After looking into which particular set of weights was being used in these cases, we found out that these results occurred when assigning a low weight (i.e. < 0.2) to the $time$ objective of the environment. This outcome is in fact very logical as when allocating a minimal credit for this objective, the algorithm will pay very little attention to optimizing the time needed to escape the valley, but will only focus on minimizing the number of acceleration and reversal actions. Thus, for these particular sets of weights, the time constraint of 500 steps to reach the goal will not be satisfied, resulting in a hypervolume of zero.

D. Spread experiment

In a final experiment, we analyze in depth the ability of each scalarization function to learn policies for different Pareto optimal solutions. Recall that in the quality indicator study of Section VI-A, the linear scalarization function yielded the best results for the generalized spread indicator. We have noted that this indicator was however biased to particular areas of the multi-objective space and therefore does not allow a fair comparison. In the experiment below, we present alternative experiments to evaluate each method's diversity.

In Fig. 11, we collect the learned policies during the entire learning phase of the experiment in Section VI-B and plot a frequency distribution over the 10 goals, i.e. treasure locations, in the Deep Sea Treasure world. It is important to note that each of the collected policies to these treasures were Pareto optimal policies. We see that the Chebyshev method is able to obtain improved performance over the linear scalarization function in terms of discovering Pareto optimal actions. Out of the 10 possible treasures, 9 are found on a frequent basis, with increased frequencies towards the extreme solutions. In case of the Deep Sea Treasure world, extreme solutions consist of solutions that either minimize the time or maximize the treasure value, but not both at the same time. The treasure with value 24 is the only one that was not frequently discovered by the algorithm using the Chebyshev scalarization function, i.e. in only 0.001% of the cases. We believe this is the case because that particular treasure is located at only one timestep from the treasure with value 50. The difference in time cost needed to reach the latter treasure is only 1, where the advantage in terms of treasure value compared to the former is $(50-24)$. Therefore, the step size of the corresponding weight interval, i.e. steps of size 0.1 in $[0, 1]$, did not allow to assign significant credit to the treasure with value of 24. We also see that the linear scalarization always forces the algorithm towards the extreme treasures and no policies towards other treasures were learned. This finding confirms our interpretation of the results on the generalized spread indicator of Table III.

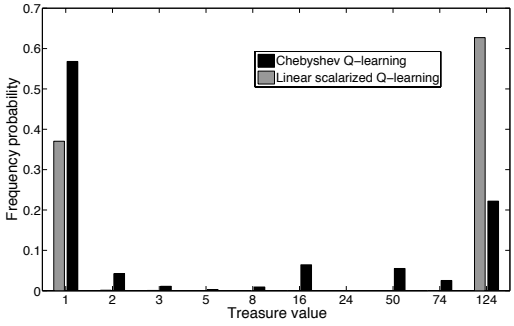

Fig. 11. A plot indicating the sampling frequency of both scalarization functions on the Deep Sea Treasure world. The x-axis represents the values of each of the treasures in the environment.

The MO Mountain Car world consists of three objectives that need to be optimized. Therefore a 3D plot is constructed that presents the Pareto dominating solutions found by the two learning algorithms and the Pareto optimal set in Fig. 12. Learning using the linear scalarization function allowed to obtain only 25 Pareto dominating solutions, where the Chebyshev scalarization obtained 38. Note that both algorithms were run only 64 times (i.e. using 64 weight tuples) and

the number of Pareto dominating solutions found using the Chebyshev function could be enlarged easily by performing additional runs. We also notice that the policies obtained by the linear scalarization method are, as expected, located in convex areas of the Pareto optimal set, while the Chebyshev function learned policies that are situated in both convex and non-convex regions. Fig. 13 is a sliced representation of the 3D plot where we focus on only two objectives. Note that the Chebyshev method obtains also in this complex world a larger spread in the multi-objective space, where the results of the linear scalarization function remain clustered into particular regions.

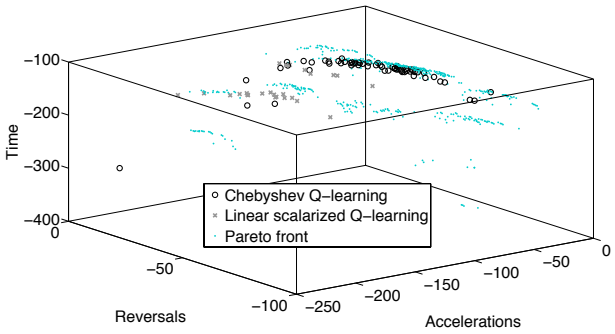

Fig. 12. A 3D representation of the elements in the Pareto optimal set and the solutions found by the two learning algorithms.

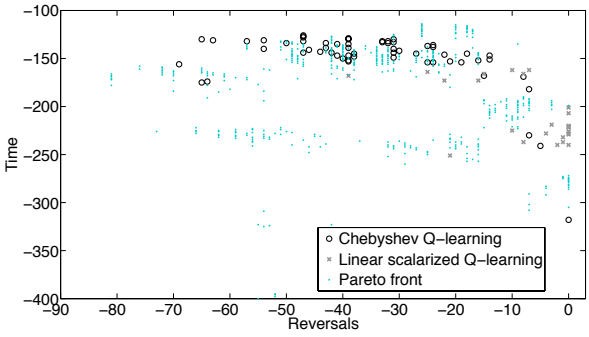

Fig. 13. A sliced representation of only two objectives of Fig. 12, i.e. the reversal and the time objective. The Chebyshev scalarization obtains solutions that are spread in the objective space, while the solutions of the linear method are clustered together.

VII. Conclusions

In this paper, we have elaborated on the drawbacks of the linear scalarization function as a mechanism to evaluate actions in multi-objective environments. Through quality indicators, we have noted the linear scalarization function is not suitable as a basis for an exploration strategy as it is biased towards particular actions, while ignoring other Pareto dominating actions. We proposed an alternative action selection strategy, based on the weighted Chebyshev function, that improves the linear scalarization function on three aspects. More precisely, we experimentally demonstrated that the Chebyshev scalarization method can (i) discover Pareto optimal solutions regardless of the shape of the front, (ii) obtain a better spread amongst the set of Pareto optimal solutions and (iii) is less dependent of the weights used. The reason behind its superior performance is found by the fact that it is a non-linear scalarization function, taking into account a reference point that is gradually adjusted as learning proceeds.

We also generalized scalarized multi-objective RL by introducing a multi-objective Q-learning framework, which can incorporate any scalarization function, e.g. linear or non-linear.

In future work, we will investigate towards alternative multi-objective solvers that overcome the drawbacks of scalarization functions, i.e. defining emphasis a priori in terms of weights. Therefore, other methods should be analyzed that search directly into the multi-objective space without applying scalarizations. A possible solution would be to incorporate the Pareto dominance rules directly as action selection mechanism.

Acknowledgement

This research is supported by the IWT-SBO project PERPETUAL (grant nr. 110041).

References

[1] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems," *Structural and Multidisciplinary Optimization*, vol. 14, pp. 63–69, 1997.

[2] C. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, University of Cambridge,England, 1989.

[3] M. A. Wiering and E. D. de Jong, "Computing Optimal Stationary Policies for Multi-Objective Markov Decision Processes," in *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. IEEE, Apr. 2007, pp. 158–165.

[4] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria reinforcement learning," in *ICML*, J. W. Shavlik, Ed. Morgan Kaufmann, 1998, pp. 197–205.

[5] L. Barrett and S. Narayanan, "Learning all optimal policies with multiple criteria," in *Proceedings of the 25th international conference on Machine learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 41–47.

[6] D. J. Lizotte, M. Bowling, and S. A. Murphy, "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, pp. 695–702.

[7] S. Gass and T. Saaty, "The computational algorithm for the parametric objective function," *Naval Research Logistics Quarterly*, vol. 2, p. 39, 1955.

[8] T. Voß, N. Beume, G. Rudolph, and C. Igel, "Scalarization versus indicator-based selection in multi-objective cma evolution strategies." in *IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 3036–3043.

[9] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Machine Learning*, vol. 84, no. 1-2, pp. 51–80, 2010.

[10] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," 1998.

[11] K. D. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm : NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[12] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[13] J. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*, ser. Statistics, Textbooks and monographs. Marcel Dekker, 2003.