Multi-Objectivization of Reinforcement Learning Problems by Reward Shaping

Tim Brys, Anna Harutyunyan, Peter Vrancx, Matthew E. Taylor, Daniel Kudenko and Ann Nowé

Abstract— **Multi-objectivization is the process of transforming a single objective problem into a multi-objective problem. Research in evolutionary optimization has demonstrated that the addition of objectives that are correlated with the original objective can make the resulting problem easier to solve compared to the original single-objective problem. In this paper we investigate the multi-objectivization of reinforcement learning problems. We propose a novel method for the multi-objectivization of Markov Decision problems through the use of multiple reward shaping functions. Reward shaping is a technique to speed up reinforcement learning by including additional heuristic knowledge in the reward signal. The resulting composite reward signal is expected to be more informative during learning, leading the learner to identify good actions more quickly. Good reward shaping functions are by definition correlated with the target value function for the base reward signal, and we show in this paper that adding several correlated signals can help to solve the basic single objective problem faster and better. We prove that the total ordering of solutions, and by consequence the optimality of solutions, is preserved in this process, and empirically demonstrate the usefulness of this approach on two reinforcement learning tasks: a pathfinding problem and the Mario domain.**

I. Introduction

A COMMON PROBLEM in reinforcement learning is that the naive algorithms require too many interactions with the environment to be useful in complex domains [1], [2]. The goal in reinforcement learning is to maximize the expected return of a reward signal that indicates how well the solver is performing, but this reward signal is often very sparse, e.g. a flat landscape with a single peak value when the goal is achieved. Learning in such a context is very hard, and for such situations, reward shaping [3] was proposed as a way to speed up learning. Reward shaping is the addition of an extra reward signal that encodes some heuristic knowledge of the system designer or domain expert, that can encourage the learning agent to explore parts of the state space that are believed to contain good solutions. Reward shaping has been successfully applied to speed up reinforcement learning techniques in complex domains [4], [5]. However, with the exception of one paper [4], all work on reward shaping has only considered the addition of a single shaping signal.[1]

Tim Brys, Anna Harutyunyan, Peter Vrancx and Ann Nowé are with the AI Lab at the Vrije Universiteit Brussel, Belgium ({timbrys, anna.harutyunyan, pvrancx, anowe}@vub.ac.be). Matthew E. Taylor is head of the Intelligent Robot Learning Lab at Washington State University, WA (email: taylorm@eecs.wsu.edu). Daniel Kudenko is head of the Reinforcement Learning Group at York University (email: daniel.kudenko@york.ac.uk)

In this paper, we explicitly formulate the addition of multiple reward shaping signals to a single-objective reinforcement learning problem as the multi-objectivization of this problem, and prove that this process does not alter the optimality of policies from the original problem. We start by giving some background knowledge in the next section on reinforcement learning, reward shaping and multi-objective reinforcement learning. We discuss related work on multi-objectivization in Section III. Then we formulate the multi-objectivization of a reinforcement learning problem by reward shaping in Section IV, and discuss some theoretical properties thereof. We discuss solution techniques in Section V. After that, we introduce two problem domains (a pathfinding problem and the Infinite Mario domain) in Section VI, after which we evaluate how the multi-objectivization of these problems can result in much faster learning in Section VII. We conclude and give directions for future work in Section VIII.

II. Background Knowledge

A. Reinforcement Learning

Reinforcement learning (RL) [6] is a framework that allows an agent to optimize its behaviour by interacting with its environment and learning from these interactions. The environment is typically described as a Markov Decision Process (MDP), which is formulated as follows. Let $S = \{s_1, s_2, \ldots\}$ be the (potentially infinite) set of states, and $A = \{a_1, a_2, \ldots\}$ the action set available to the learning agent. Each combination of current state s, action choice $a \in A$ and next state s' has an associated transition probability $T(s'|s,a)$ and immediate reward $R(s,a,s')$. The goal is to learn a policy π that probabilistically maps states to actions in such a way that the expected accumulated future discounted reward J^π is maximized:

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

where $\gamma \in [0,1]$ is the discount factor and expectations are taken over stochastic rewards, transitions and action

[1]At least explicitly. Some make the reward signal itself richer by adding auxiliary goals, e.g. a bonus for an agent bicycling towards a goal for actually driving in the direction of the goal, without realizing they are applying reward shaping. Care must be taken with such additions to the reward signal itself, as these can change the optimal policy, e.g. resulting in the agent never arriving at the goal, but making circles to collect the extra reward for driving towards the goal over and over [2]. Reward shaping signals must always be potential-based to guarantee no change in the optimal policy [3], see Section II-B.

selection, with t representing timesteps. 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:

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

One of the best known RL algorithms is Q-Learning [7], proposed by Watkins to iteratively approximate Q^*. In the Q-learning algorithm, a Q-table containing state-action pairs is stored. Each entry contains a value for $\hat{Q}(s,a)$, which is the learner's current estimate for the actual value of $Q^*(s,a)$. The $\hat{Q}$-values are updated according to the following update rule:

$$\begin{aligned} \hat{Q}_t(s,a) \leftarrow \quad & (1-\alpha_t)\hat{Q}_{t-1}(s,a) \\ & +\alpha_t[R(s,a,s') + \gamma \max\nolimits_{a'} \hat{Q}_{t-1}(s',a')] \end{aligned}$$

where α_t is the learning rate at time step t and $R(s,a,s')$ is the reward received for performing action a in state s, resulting in state s'. Provided that all state-action pairs are visited infinitely often, and a suitable schedule for the learning rate is chosen, the estimates, $\hat{Q}$, will converge to the optimal values, Q^* [8].

A greedy policy can easily be derived from these estimates by taking the action with highest $\hat{Q}$-value in every state. But, as these estimates may not yet be correct during learning, a trade-off needs to be found between exploring unknown states and actions, and exploiting known good (partial) policies. Therefore, an exploration mechanism such as ϵ-greedy is often used. With a probability ϵ, a random action is chosen during operation, while with probability $1-\epsilon$, the greedy action with respect to the current estimates is chosen.

Many practical reinforcement learning problems have very large state spaces and/or continuous state variables, making basic tabular learning methods impractical or impossible to use. A very popular way to overcome this problem is to use tile-coding as function approximator [9]. This overlays the state space with multiple axis-parallel tilings, allowing for a discretization of the state-space, while the overlapping tilings guartantee a certain degree of generalization. The Q-function can then be approximated by learning weights that map the tiles activated by the current state s to an estimated Q-value:

$$\hat{Q}_t(s,a) = \theta_{t,a}^T \phi_s$$

ϕ_s is the feature vector representing state s, i.e. the tiles activated by this state, and θ is the parameter vector that needs to be learned to approximate the actual Q-function.

Eligibility traces [10] are records of past occurrences of state-action pairs, which can be used to speed up learning, by not only updating the Q-value of the current state-action pair, but also past state-action pairs, rewarding these inversely proportional to the time since they were experienced. A replacing eligibility trace [11] $e_t(s,a)$ for state s and action a is updated as follows:

$$e_{t+1}(s,a) = \begin{cases} 1 & s = s_t, a = a_t \\ \gamma\lambda e_t(s,a) & \textit{otherwise} \end{cases}$$

It is set to 1 if (s,a) is the current state-action pair (s_t,a_t), and otherwise it is decayed by $\gamma\lambda$, with γ the discounting factor and λ the specific eligibility trace decay. This update is performed after every action, and thus traces decay over time. The trace is then included in the Q update rule:

$$\begin{aligned} \hat{Q}_t(s,a) \leftarrow \quad & (1-\alpha_t)\hat{Q}_{t-1}(s,a) \\ & +\alpha_t e_t(s,a)[R(s,a,s') \\ & +\gamma \max\nolimits_{a'} \hat{Q}_{t-1}(s',a')] \end{aligned}$$

Instead of only updating $Q(s,a)$, we update the Q-value of every state-action pair where the elibility trace is non-zero. This effectively allows us to immediately propagate reward into the past, rewarding actions that led to the current reward, significantly reducing the learning time. Otherwise, this reward propagates only by means of the $\gamma \max_{a'} \hat{Q}_{t-1}(s',a')$ part in the update-rule.

B. *Reward Shaping*

With reward shaping, an RL agent is given an extra reward $F(s,a,s')$ on top of the reward from the environment $R(s,a,s')$. This reward is aimed at steering the exploration behaviour of the learning agent, incorporating heuristic knowledge of the system designer on the problem domain. The shaping function F is included in the Q-learning update rule as follows:

$$\begin{aligned} \hat{Q}_t(s,a) \leftarrow \quad & (1-\alpha_t)\hat{Q}_{t-1}(s,a) \\ & +\alpha_t[R(s,a,s') + F(s,a,s') \\ & +\gamma \max\nolimits_{a'} \hat{Q}_{t-1}(s',a')] \end{aligned}$$

If F is implemented as the difference of some potential function Φ over the state space, and incorporates γ, the discount factor, as follows, then the shaping function is guaranteed to not alter the optimality of policies [3]:

$$F(s,a,s') = \gamma\Phi(s') - \Phi(s) \qquad (1)$$

For example, if one wants to make an agent learn to traverse a crowd without bumping into other agents, one possible potential function $\Phi(s)$ to incorporate in a shaping could be the distance to the nearest agent. Encouraging the agent to maximize this distance will help him avoid running into other agents.

Potential-based reward shaping has been successfully applied in such complex domains as RoboCup KeepAway soccer [4] and StarCraft [5], improving agent performance significantly.

C. *Multi-Objective Reinforcement Learning*

Multi-objective reinforcement learning [12] (MORL) is an extension to standard reinforcement learning, where the environment is a multi-objective MDP, or MOMDP, and the feedback signal returns a vector rather than a single scalar value, i.e.:

$$\mathbf{R}(s,a,s') = (R_1(s,a,s'), \ldots, R_m(s,a,s'))$$

where m represents the number of objectives. In MORL, the solution concept is also a policy π, which is evaluated by its expected return $\mathbf{J}^\pi$, a vector containing the expected discounted return for each objective:

$$\mathbf{J}^\pi \equiv \Bigg[E\left[\sum_{t=0}^{\infty} \gamma^t R_1(s_t, a_t, s_{t+1})\right], \ldots, E\left[\sum_{t=0}^{\infty} \gamma^t R_m(s_t, a_t, s_{t+1})\right] \Bigg]$$

Since the environment now consists of multiple objectives, and conflicts may exist between them, there is typically no total order over policies. Hence the notion of Pareto-optimality. A policy x_1 is said to strictly Pareto dominate another policy x_2, i.e. $x_1 \succ x_2$, if for each objective, x_1 performs at least as well as x_2, and it performs strictly better on at least one objective. In the case where x_1 improves over x_2 on some objective, and x_2 improves over x_1 on some other objective, the two solutions are said to be incomparable. The set of non-dominated policies is referred to as the *Pareto optimal set* or *Pareto front*.

In such an environment, Q-values can be learned for every objective in parallel, storing $\hat{\mathbf{Q}}$-vectors, with a $\hat{Q}$-value for each objective [13], [14]:

$$\hat{\mathbf{Q}}(s,a) = \Bigg[\hat{Q}_1(s,a), \ldots, \hat{Q}_m(s,a) \Bigg]$$

The most commonly used techniques in MORL are *scalarizations* of the multi-objective problem [12], [14], [15], either before the fact by reducing the dimensionality of the problem to a single scalar objective, or after the fact, by applying them to $\hat{\mathbf{Q}}$-vectors. Scalarization functions typically assign a weight to each objective, allowing the user to put more or less emphasis on each of the objectives, giving him some control over the trade-off policies the learner will converge upon. This trade-off is parametrized by $w_o \in [0, 1]$ for objective o, with $\sum_{o=1}^{m} w_o = 1$. In most cases, a linear combination of the objectives is employed, i.e. $\sum_{o=1}^{m} w_o \cdot \hat{Q}_o(s,a)$, although setting these weights a priori to achieve a particular trade-off is hard and unintuitive [16], often requiring a lot of parameter tuning.

III. Multi-Objectivization

To reiterate, the multi-objectivization of a problem is the conversion of a single-objective problem into a multi-objective problem in order to improve performance on the original objective, as measured by solution quality, time to solution, or some other measure [17]. This idea has mainly been studied in the evolutionary computation literature, and there exist two main approaches for the multi-objectivization of a single-objective problem: either by decomposing the single objective [17], [18], [19], or by adding extra objectives [20], [21]. Examples are training decision trees using the misclassification of each individual class separately instead of the total misclassification [22], using the number of recursive calls to a procedure and the number of iterations in loops to optimize running time when generating programming competition tasks [23], or turning a constrained problem into an unconstrained one with extra objectives encoding those constraints [24], [25].

The approach we propose in this paper falls in the second category, i.e. it adds extra objectives to the problem. Such additional objectives often encode heuristic information or expert knowledge of the problem. Since the only goal is to optimize the single problem-inherent objective, the extra objectives should not introduce conflicts, but rather, they should correlate with the original objective. Ideally, the Pareto front of the multi-objective problem should be a single point, corresponding to the optimal solution(s) of the original problem. Some theoretical results exist on exact auxiliary functions for evolutionary algorithms, which are additional objectives whose optimal solutions coincide with those of the target objective [26].

Furthermore, if the additional objectives are to improve solution quality, they should possess some properties that make their function landscape easier to navigate, e.g. smoothing out local optima, or providing gradient where there is none in the original objective. Empirical results, again from the evolutionary computation literature, show that using the Pareto operator can help in genetic algorithms [27], and that additional objectives can reduce the number of local optima in the search space [17], but also that search can become harder [21], as the additional objectives can make solutions incomparable when the extra objectives do not correlate well with the main objective for those solutions. Some interesting work on multi-objectivization shows how one can improve performance on multi-objectivized problems by making every optimization decision based on feedback from only a single of the correlated objectives [20]. In [23], [28], Buzdalova et al. show how this choice can be made adaptively using reinforcement learning, trained on feedback from the actual target objective.

From this brief literature survey, we identify two requirements for a good multi-objectivization:

1) The optimality of solutions is preserved; no conflicts are introduced, no suboptimal solutions in the original problem become Pareto-optimal in the multi-objectivized problem.
2) The search space becomes easier to navigate; more information is present in the multi-objective problem, such that optimization becomes easier.

IV. Multi-Objectivization by Shaping

In this section, we describe the multi-objectivization of a single-objective MDP by using multiple reward shaping functions, and show how the requirements for a good multi-objectivization can be satisfied. Importantly, we theoretically prove that the first requirement, namely that the optimality of policies is preserved, is always satisfied if the shaping functions are potential-based.

To turn MDP M into MOMDP M' using m reward shaping functions F_i, the reward vector $\mathbf{R}$ of M' is constructed as follows:

$$\mathbf{R}(s,a,s') = \begin{array}{l}(R(s,a,s') + F_1(s,a,s'), \ldots, \\ R(s,a,s') + F_m(s,a,s'))\end{array} \qquad (2)$$

where R is the reward function of M. Thus, we copy the base reward of M several times, and add a different shaping function to each.

We will prove that this formulation preserves the total ordering, and thus also the optimality, of policies between M and M', provided the shapings are potential-based. That is, that multi-objectivization by reward shaping does not introduce conflicts.

Theorem 1 *Let M be a given (finite, discrete) MDP, $M = (S, A, T, R)$. We say that the MOMDP M' is a shaping-based multi-objectivization of M, iff $M' = (S, A, T, \mathbf{R})$ with*

$$\mathbf{R}(s,a,s') = \begin{array}{l}(R(s,a,s') + F_1(s,a,s'), \ldots, \\ R(s,a,s') + F_m(s,a,s'))\end{array}$$

If all shaping functions $F_i,\ i = 1, \ldots, m$ are potential-based, as defined in Equation 1, we have the following properties:

- *Any policy π^* which is an optimal policy for M, is a Pareto optimal policy for M'.*
- *No other Pareto optimal policies for M' exist, i.e. if π is not an optimal policy for M, π is not Pareto optimal in M'.*

Proof: The proof follows from the results in [3]. There, Ng et al. proved that if a policy is optimal for an MDP with reward function R, it is also optimal for the shaped MDP with rewards $R + F$ (and vice versa), provided that F is a potential-based shaping function. So, any policy π^* that is optimal for MDP M will also be optimal for each of the shaped rewards $R + F_i$. Since π^* maximises the returns for all objectives, no policy which Pareto dominates π^* can exist (since such a policy would have to perform strictly better on at least one objective) and π^* must be part of the Pareto front for M'. Now suppose a policy π exists, which is part of the Pareto front of M', but which is not optimal in M. Since π is suboptimal in M, according to [3] it must also be suboptimal for each of the $R + F_i$ objectives. However, this means that any policy π', that is optimal in M,[2] will achieve a strictly higher return for all objectives. Thus, π' Pareto dominates π and π cannot be part of the Pareto optimal set for M', which contradicts our original assumption.

Corollary. *Since all optimal policies of M' (and thus also of M) achieve the highest expected return for each objective in M', the Pareto front of M' consists of a single point. Moreover, since Ng et al. [3] actually prove that the total order of policies is preserved when using potential-based shaping, and not just optimality, MOMDP M' also has a total order over all policies. These all lie on a single line in the multi-objective space, see Figure 1.*

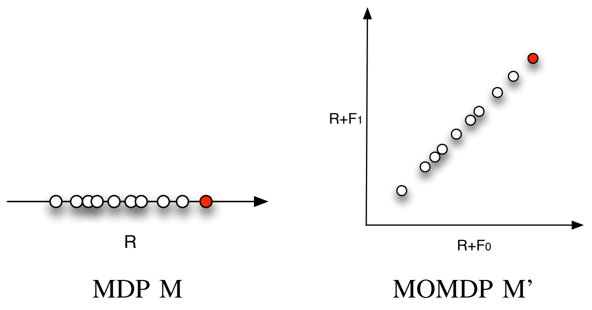

Fig. 1. An example of the quality of policies given their return in MDP M, and it's multi-objectivized version M'. The total order over policies is preserved.

This shows that the first important requirement for a good multi-objectivization is always satisfied if the shaping functions are potential-based. The second requirement, namely that it makes the search space easier to navigate, depends on the formulation of the shaping functions themselves. Good shaping functions correlate with V^*, the value function of the problem.[3] The more they correlate with V^*, the more they will help solve the problem. While it is unlikely that the system designer can define a single shaping that correlates well with V^* throughout the search space (which would amount to solving the problem), it is more likely that he can define several shapings that correlate well with V^* in different parts of the state space, i.e. several rules of thumb for different situations,[4] or rules of thumb that weakly correlate with V^* throughout the state space. If these shapings are used to multi-objectivize an MDP, one can then attempt to strategically combine these signals, e.g. by identifying when which signal is most informative [30].

Now, this multi-objectivization of course opens up the possibility to use multi-objective techniques to find solutions to a single-objective problem. For example, given the convergence guarantees of tabular Q-learning, learning on any linear scalarization of this MOMDP will converge to an optimal policy. In such a case, the only improvement multi-objectivization can yield lies in the speed of convergence. For learning algorithms not guaranteed to converge to an optimal policy, or settings where the necessary assumptions are violated, as in the case of Q-learning with function approximation for example, both speed of convergence and solution quality may be improved through multi-objectivization.

[2] At least one such optimal policy must exist, see e.g. [29]

[3] $V^*(s) = \max_a Q^*(s,a)$.

[4] E.g. shaping using kinetic (speed) or potential energy (height) in the Mountain Car domain [11], a problem where an underpowered car needs to build up momentum to climb a hill. These are opposite forces in this domain, as the car trades speed for height and vice versa, yet each is useful in a different situation: the car needs to focus on gaining speed when it can no longer gain height, and focus on gaining height when speed is already high.

V. Solution Methods

Although we could use any technique that solves general multi-objective MDPs, not all techniques will actually be useful. Multi-objective optimization techniques typically focus on finding trade-offs between the different objectives, with as goal a specific trade-off, or finding a diverse set of Pareto optimal trade-offs which can then be presented to the user. The multi-objective problem we have constructed here on the other hand does not require trade-offs to be found; the optimal solutions for each objective are the same. It is a specific sub-class of multi-objective problems that we believe requires a different type of techniques. That is, techniques that can combine the different correlated signals in such a way that optimization is sped up, and/or the optimal solutions are better approximated.

In this paper, we illustrate the usefulness of this multi-objectivization by applying a linear scalarization of the objectives, i.e. scaling the reward vector back down to a single scalar signal, which can then be solved by regular single-objective techniques. Of course, this is equivalent to creating a single super shaping F' that is a weighted sum of the different shapings defined (if $\sum_i w_i = 1$).

$$\sum_i w_i(R + F_i) = R + \sum_i w_i F_i = R + F'$$

This is the same approach Devlin et al. [4] used. Yet, we believe better techniques can be constructed that actually benefit from keeping the signals separate, since dimensionality reduction techniques such as a scalarization always lose information. For example, useful information that would be thrown away is the amount of agreement between the shapings, something which could be used to make more informed action selection decisions. A set of techniques that does exactly this is ensemble systems for reinforcement learning [31], [32]. These will likely prove to be good solution methods for this kind of problem, as they are built to combine the suggestions of different RL algorithms learning the same task. That is, these combine different predictors for the same signal. Another candidate solution technique is adaptive objective selection [30], which is a technique developed specifically for multi-objective MDPs with correlated objectives. It determines for every state, during learning, which of the objectives to use for action selection, based on a measurement of confidence in its own estimates.

We leave the evaluation of these more advanced techniques for future work, and only focus on the basic scalarization techniques in this paper.

VI. Problem domains

In this section, we describe the two problem domains we consider in this paper to illustrate how a learning agent may solve a problem faster and better by multi-objectivizing it. The first domain is a simple gridworld in which an agent needs to find the quickest path to the goal location. The second problem is the much more complex domain of Infinite Mario Bros, a public reimplementation of Nintendo's Super Mario Bros®, where the agent controls Mario, who needs to traverse a level while collecting coins and avoiding to get killed.

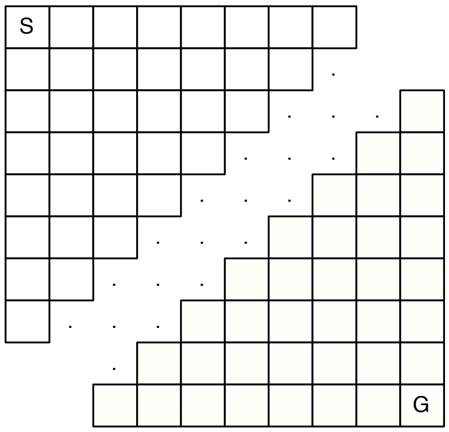

Fig. 2. The pathfinding gridworld. The start-state is top-left, the goal state is bottom-right.

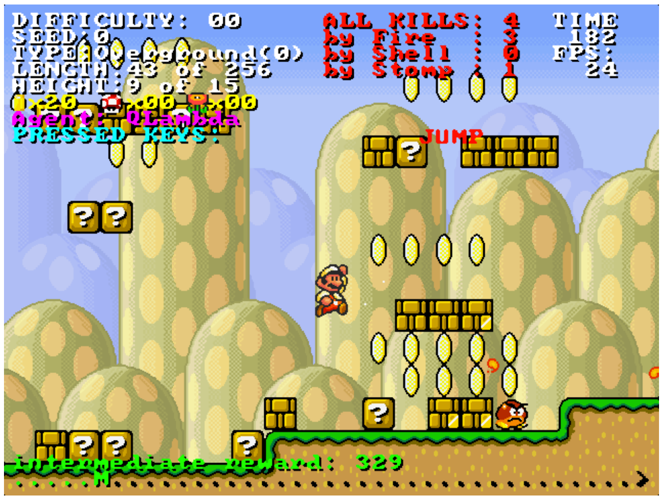

Fig. 3. A screenshot of Mario during the level used in this paper.

A. Pathfinding

The pathfinding problem is situated in a 20×20 gridworld, with a starting location at $(0, 0)$ and goal location at $(19, 19)$, see Figure 2. The agent needs to find the goal location as fast as possible by moving in the four cardinal directions, only receiving a reward of 1000 when arriving at the goal location. Otherwise, the agent receives a reward of 0. This is a very sparse signal, without gradient throughout the search space, except for the single peak at the goal location.

B. Mario

The Mario benchmark problem [33] is based on Infinite Mario Bros, which is a public reimplementation of the original 80's game Super Mario Bros®. In this task, Mario needs to collect as many points as possible, which are attributed for killing an enemy (10), devouring a mushroom (58) or a fireflower (64), grabbing a coin (16), finding a hidden block (24), finishing the level (1024), getting hurt by a creature (-42) or dying (-512). The actions available to Mario correspond to the buttons on the NES controller, which are (left, right, no direction), (jump, don't jump), and (run, don't run). One action from each of these groups can be taken simultaneously, resulting in 12 distinct combined or 'super' actions. The state space in Mario is quite complex, as Mario observes the exact locations of all enemies on the screen and their type, he observes all information pertaining to himself, such as what mode he is in (small, big, fire), and furthermore he is surrounded by a gridlike receptive field in which each cell indicates what type of object is in it (such as a brick, a coin, a mushroom, a goomba (enemy), etc.).

VII. Experiments

Here we describe experimental results that demonstrate the usefulness of multi-objectivization by reward shaping.

A. Setup

1) Pathfinding: The learning algorithm used for this problem is regular tabular ϵ-greedy Q-learning. The state of the agent consists only of its x and y coordinates. We define two shaping functions, F_x and F_y, that respectively encourage increasing the x and y coordinates. Their potential functions are:[5]

$$\Phi_x(s) = \frac{x}{100}$$
$$\Phi_y(s) = \frac{y}{100}$$

These shapings provide gradient information that is not present in the actual problem.

The parameters for Q-learning are $\epsilon = 0.1$, $\alpha = 1.0$ and $\gamma = 0.99$.

2) Mario: The learning algorithm used for the Mario domain is $Q(\lambda)$, which is Q-learning with eligibility traces. We use tile-coding for function approximation, as the full state information of Mario is far too complex to condition an RL agent on. We take a very simple approach and give the agent only information on (1) the relative x and y coordinates of the closest enemy, so that he can avoid or kill it, (2) the height of the obstacle right in front of Mario, so he knows he has to jump over it, and (3) the time left in the level, so that he can know when to hurry up to finish the level. These are discretized using tile-coding, height with uniform tiles of width 3, the x and y with logarithmic tiles, symmetric around 0 ($sgn(x)log(|x|)$), and time also with logarithmic tiles ($log(time)$). We use logarithmic tiles for x and y because enemies only have a large impact close to Mario, so we generalize more when an enemy is further away. We do the same for time, since Mario does not need to hurry too much when a lot of time is left, while seconds start to matter when time is running out. Additional binary features are: (4) whether or not Mario is able to shoot fireballs, (5) the direction he is facing, and (6) whether he is on the ground, making for 6 variables in total.

We implement two shaping functions. F_r, to encourage moving to the right, as the level unfolds from left to right, and F_h, to encourage gaining height, as being higher up can help overcome obstacles and killing enemies by dropping on them. Their potentials are:

$$\Phi_r(s) = 100 \times x$$

$$\Phi_h(s) = 100 \times y$$

The parameters for $Q(\lambda)$ are $\epsilon = 0.1$, $\alpha = \frac{0.01}{32}$, $\lambda = 0.9$ and $\gamma = 1.0$, with 32 tilings for tile coding.

The Mario levels are generated procedurally, and we use the level generated with seed 0 and difficulty 0 in our experiments.

B. Results

We show the results of a comparison between using no shaping, using a single of the shapings, and multi-objectivizing the problem and subsequently solving this MOMDP by scalarizing it using uniform weights ($w_0 = w_1 = 0.5$).

1) Pathfinding: For the pathfinding problem, we ran 100 experiments, each 10^4 episodes long. The performance of the learning agent is measured by the number of steps to goal, and the average performance over all these experiments is shown in Figure 4. Table I summarizes these results. Using either the x shaping or y shaping alone already yields a dramatic improvement compared to without shaping, but the combination of the two shapings yields even faster learning. Note that the final performance can not be improved as Q-learning is guaranteed to converge to the optimal policy in this case.

2) Mario: Since the pathfinding problem is a trivial problem, we perform a similar experiment in the Mario domain, which is much more complex. We ran 100 experiments of 1000 episodes each. Figure 5 and Table II show the average results of this experiment. Both right and height shaping result in worse final performance compared to without shaping, although the right shaping does show faster learning initially. Height shaping has a surprisingly catastrophic effect on performance, resulting in the agent not advancing in the level. Even more surprisingly, the multi-objectivization of the problem using these two shapings, each separately resulting in worse performance, does yield better performance, with very fast learning initially, and convergence to similar final performance as without shaping.

[5]How to optimally set the scaling of a shaping function with respect to the reward function remains an open question. It should not overpower the reward function such that the learner gets distracted from its actual goal in states where the shaping function is not fully correlated with the V^* function (which is possible given suboptimal learning parameters). On the other hand, if too small, the shaping function becomes negligible and has no effect on exploration. We tuned the magnitude of the shapings to achieve best performance.

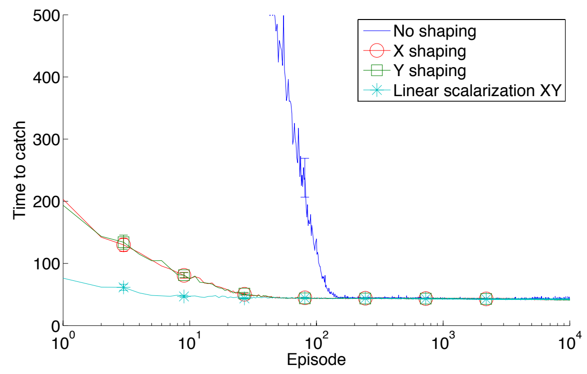

Fig. 4. Results for the pathfinding problem. Using a single shaping function already dramatically imrpoves performance, and the combination of both shapings improves performance even more. Error bars indicate the 95% confidence interval.

Variant	Cumulative performance	Final performance
No shaping	105038.05 ± 1847.869	$\mathbf{42.07 \pm 0.624}$
X shaping	45049.3 ± 127.445	$\mathbf{41.7 \pm 0.516}$
Y shaping	45053.03 ± 116.567	$\mathbf{42.27 \pm 0.601}$
Linear combination	$\mathbf{43825.69 \pm 64.139}$	$\mathbf{41.83 \pm 0.554}$

TABLE I

RESULTS FOR THE PATHFINDING PROBLEM. THE COMBINATION OF SHAPINGS RESULTS IN MUCH BETTER CUMULATIVE PERFORMANCE (MEASURED FOR THE FIRST 1000 EPISODES), I.E. FASTER LEARNING, THAN WITH A SINGLE SHAPING OR WITHOUT SHAPING. THE BEST RESULTS AND THOSE NOT SIGNIFICANTLY DIFFERENT FROM THE BEST ARE INDICATED IN BOLD (STUDENT'S T-TEST, $\alpha = 0.05$). 95% CONFIDENCE INTERVAL IS INDICATED.

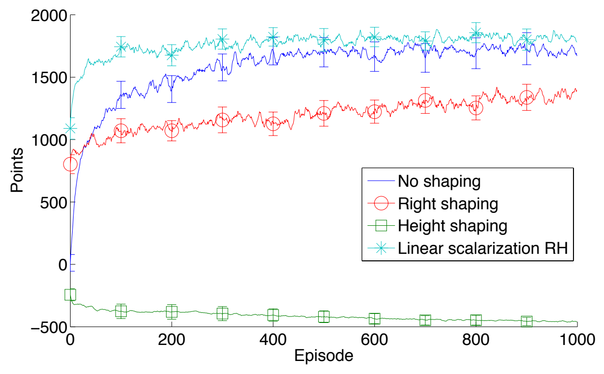

Fig. 5. Results for the Mario domain. Using either the right or height shaping alone does not improve performance compared to no shaping. It is only with the combination of shapings that performance (especially early performance) can significantly be improved. Error bars indicate the 95% confidence interval.

Variant	Cumulative performance	Final performance
No shaping	1579780.6 ± 91824.2	$\mathbf{1670.3 \pm 130.2}$
Right shaping	1197782.3 ± 75215.3	1395.2 ± 105.3
Height shaping	-414637.4 ± 44596.2	-462.1 ± 29.8
Linear combination	$\mathbf{1765874.5 \pm 56815.5}$	$\mathbf{1780.2 \pm 79.7}$

TABLE II

RESULTS FOR THE MARIO DOMAIN. THE COMBINATION OF SHAPINGS RESULTS IN MUCH BETTER CUMULATIVE PERFORMANCE, I.E. FASTER LEARNING, AND SIMILAR FINAL PERFORMANCE AS WITHOUT SHAPING. THE BEST RESULTS AND THOSE NOT SIGNIFICANTLY DIFFERENT FROM THE BEST ARE INDICATED IN BOLD (STUDENT'S T-TEST, $\alpha = 0.05$). 95% CONFIDENCE INTERVAL IS INDICATED.

VIII. Conclusions and Future Work

We propose the multi-objectivization of reinforcement learning problems by reward shaping, a process that turns a single-objective problem into a multi-objective problem, in order to speed up learning and improve performance. We prove that the optimality of solutions is preserved in the process, and empirically show that learning with a scalarization of this multi-objectivized problem can improve learning a lot.

In the experimental section of this work, we confined ourselves to solving a multi-objectivized problem using a simple scalarization with uniform weights, which is equivalent to not multi-objectivizing and combining the shapings into a single super shaping. The following step in this line of research is to consider more advanced techniques for the combination of correlated objectives, i.e. techniques that benefit from keeping these signals apart. Adaptive objective selection [30], a technique introduced specifically for the combination of correlated objectives, and ensemble systems for reinforcement learning [31], [32] are promising candidates for this.

Acknowledgements

Tim Brys is funded by a PhD scholarship of the Research Foundation Flanders (FWO). Anna Harutyunyan is supported by the IWT-SBO project MIRAD (grant nr. 120057).

References

[1] M. J. Mataric, "Reward functions for accelerated learning." in *ICML*, vol. 94, 1994, pp. 181–189.

[2] J. Randlov and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, pp. 463–471.

[3] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287.

[4] S. Devlin, M. Grześ, and D. Kudenko, "An empirical study of potential-based reward shaping and advice in complex, multi-agent systems," *Advances in Complex Systems*, vol. 14, no. 02, pp. 251–278, 2011.

[5] K. Efthymiadis and D. Kudenko, "Using plan-based reward shaping to learn strategies in starcraft: Broodwar," in *Conference on Computational Intelligence in Games, IEEE CIG*, 2013.

[6] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998, vol. 1, no. 1.

[7] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, 1989.

[8] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994.

[9] J. Albus, *Brains, behavior and robotics*. McGraw-Hill, Inc., 1981.

[10] A. H. Klopf, "Brain function and adaptive systems: a heterostatic theory," Air Force Cambridge Research Laboratories, Bedford, MA, Tech. Rep. AFCRL-72-0164, 1972.

[11] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine learning*, vol. 22, no. 1-3, pp. 123–158, 1996.

[12] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[13] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria reinforcement learning," in *International Conference on Machine Learning*, 1998.

[14] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Scalarized Multi-Objective Reinforcement Learning: Novel Design Techniques," in *2013 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. IEEE, 2013.

[15] 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.

[16] I. Das and J. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems," *Structural optimization*, vol. 14, no. 1, pp. 63–69, 1997.

[17] J. D. Knowles, R. A. Watson, and D. W. Corne, "Reducing local optima in single-objective problems by multi-objectivization," in *Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 269–283.

[18] J. Handl, S. C. Lovell, and J. Knowles, "Multiobjectivization by decomposition of scalar cost functions," in *Parallel Problem Solving from Nature–PPSN X*. Springer, 2008, pp. 31–40.

[19] M. Jähne, X. Li, and J. Branke, "Evolutionary algorithms and multi-objectivization for the travelling salesman problem," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 595–602.

[20] M. T. Jensen, "Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 323–347, 2005.

[21] D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler, "Do additional objectives make a problem harder?" in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 765–772.

[22] J. E. Fieldsend, "Optimizing decision trees using multi-objective particle swarm optimization," in *Swarm Intelligence for Multi-objective Problems in Data Mining*. Springer, 2009, pp. 93–114.

[23] A. Buzdalova, M. Buzdalov, and V. Parfenov, "Generation of tests for programming challenge tasks using helper-objectives," in *Search Based Software Engineering*. Springer, 2013, pp. 300–305.

[24] S. Watanabe and K. Sakakibara, "Multi-objective approaches in a single-objective optimization environment," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2. IEEE, 2005, pp. 1714–1721.

[25] D. K. Saxena and K. Deb, "Trading on infeasibility by exploiting constraints criticality through multi-objectivization: A system design perspective," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 919–926.

[26] Y. G. Yevtushenko and V. Zhadan, "Exact auxiliary functions in optimization problems," *USSR Computational Mathematics and Mathematical Physics*, vol. 30, no. 1, pp. 31–42, 1990.

[27] S. J. Louis and G. J. Rawlins, "Pareto optimality, GA-easiness and deception," in *ICGA*, 1993, pp. 118–123.

[28] A. Buzdalova and M. Buzdalov, "Increasing efficiency of evolutionary algorithms by choosing between auxiliary fitness functions with reinforcement learning," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 1. IEEE, 2012, pp. 150–155.

[29] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. Wiley. com, 2009, vol. 414.

[30] T. Brys, K. Van Moffaert, A. Nowé, and M. E. Taylor, "Adaptive objective selection for correlated objectives in multi-objective reinforcement learning," in *The 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014.

[31] M. A. Wiering and H. van Hasselt, "Ensemble algorithms in reinforcement learning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 4, pp. 930–936, 2008.

[32] V. Marivate and M. Littman, "An ensemble of linearly combined reinforcement-learning agents," in *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[33] S. Karakovskiy and J. Togelius, "The mario ai benchmark and competitions," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 55–67, 2012.