Multi-objective Reinforcement Learning for the Expected Utility of the Return

Diederik M. Roijers
Vrije Universiteit Amsterdam (NL)
Vrije Universiteit Brussel (BE)

Denis Steckelmacher
Vrije Universiteit Brussel
Brussels, Belgium

Ann Nowé
Vrije Universiteit Brussel
Brussels, Belgium

ABSTRACT

Real-world decision problems often have multiple, possibly conflicting, objectives. In multi-objective reinforcement learning, the effects of actions in terms of these objectives must be learned by interacting with an environment. Typically, multi-objective reinforcement learning algorithms optimise the utility of the expected value of the returns. This implies the underlying assumption that it is indeed the expected value of the returns (i.e., an average returns over many runs) that is important to the user. However, this is not always the case. For example in a medical treatment setting only the return of a single run matters to the patient. This return is expressed in terms of multiple objectives such as maximising the probability of a full recovery and minimising the severity of side-effects. The utility of such a vector-valued return is often a non-linear combination of the return in each objective. In such cases, we should thus optimise the expected value of the utility of the returns, rather than the utility of the expected value of the returns. In this paper, we propose a novel method to do so, based on policy gradient, and show empirically that our method is key to learning good policies with respect to the expected value of the utility of the returns.

KEYWORDS

Multi-Objective Reinforcement Learning; Policy Gradient, Expected Scalarised Return

1 INTRODUCTION

Real-world sequential decision problems often require learning about the effects of actions by interacting with an environment. When these effects can be measured in terms of a single scalar objective, such problems can be modelled as a *Markov decision process (MDP)* [18]. However, many real-world decision problems have multiple possibly conflicting objectives, leading to a *multi-objective Markov decision process (MOMDP)* [10, 25], in which the rewards are vector-valued. When the preferences of a user can be expressed as a *linear* utility function, and this function is known a priori, an MOMDP can be translated to a single-objective MDP and solved accordingly. However, when user preferences are non-linear, explicitly multi-objective methods are required, even if the utility function is in fact known a priori.

In (MO)MDPs, executing a policy leads to rewards that are accrued over time. The discounted sum of rewards resulting from a policy execution is called the *return*. In a single objective MDP the return is a scalar, and an agent aims to maximise the expected return. The expected return is called the value of a policy, and a policy that maximises the value is called an optimal policy. In an MOMDP however, the return is a vector. This makes the question of what to optimise more complex as there is typically no single policy that maximises the value of all objectives simultaneously. There are thus typically multiple policies that offer different trade-offs between the objectives. Which policy should be preferred then depends on the utility function of the user.

We follow the utility-based approach [10] and assume that there exists a utility function that maps a vector with a value for each objective to a scalar utility.[1] Specifically, we consider the *single-policy, known weights* scenario as described in [10], in which the utility function is known and possibly non-linear. When deciding what to optimise in a multi-objective MDP we need to apply this function to our vector-valued returns in some way. There are two choices for how to do this [10, 11]: *after* computing the expected value of the return of a policy, leading to the *scalarised expected returns (SER)* optimisation criterion, or *before* computing the expected value, leading to the *expected scalarised returns (ESR)* optimisation criterion. Which of these criteria is the right one depends on how the policies are applied in practice. SER is the correct criterion if a policy can be executed multiple times, and it is the average sum of rewards over multiple episodes that determines the utility for the user. ESR is the correct formulation if it is the outcome of a single policy execution that is relevant to the user. In the medical domain [8] for example, if the policy corresponds to a medical treatment plan, the treatment is only going to be executed for a single patient once, and it is the outcome of that execution that is relevant to the patient. We also note that even if the policy can be executed multiple times, it can still be the outcome of a single policy execution that is relevant to the user. For example, consider commuting to and from work; even if the average commute time and comfort levels are acceptable, it being very speedy and comfortable on some days while being highly uncomfortable and long on others will most likely have a different average utility than the utility of the average outcome.

1.1 Motivating example

Let us consider a simple multi-objective MDP. Imagine that you have been made responsible for preparing grilled fish for a feast in the evening. To prepare this, you need to start building your fire at 5PM, so you have a limited amount of time to gather fish and wood, leading to the MOMDP in Figure 2. In each state there are two available actions: either *gather* or *move*. Starting at the river (state 1), you can first obtain raw fish (objective 1) by gathering. Fishing has a stochastic outcome; you either catch a fish in a given timestep or you do not leading to a reward $(1, 0)$ with probability 0.1,

[1] We note that there is a known preference scheme that does not fit the utility-based approach, i.e., the lexicographic preference ordering, e.g., proposed by [28, 29]. In this scheme, the first objective is infinitely more important than the second, the second infinitely more important than the third, and so on. For example, this could correspond to a company that only cares about environmental impact as long as it does not diminish their profit by even one cent. We consider this an edge case, and beyond the scope of this paper.

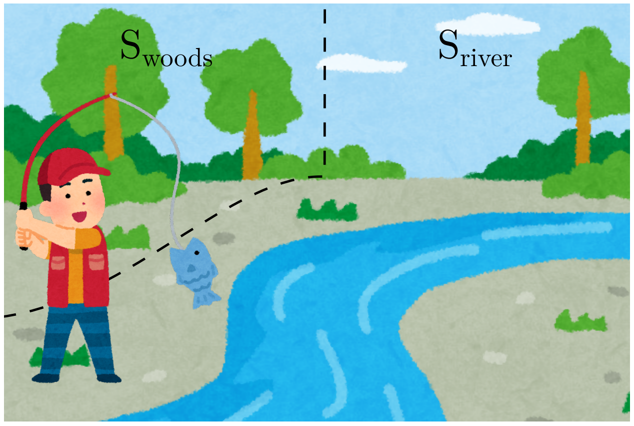

Figure 1: 2-state gathering MOMDP with 2 objectives, fish and wood, and two actions, gather and move (illustrated). This image is based on images from http://www.irasutoya.com/, who have given us permission to use these images.

and $(0,0)$ with probability 0.9. After fishing, you need to spend one timestep to move from the river to the woods. In the woods, you can gather wood to grill the fish, i.e., taking the gathering action leads to a reward of $(0,1)$ with probability 0.9 and $(0,0)$ with probability 0.1. Because two units of wood are required to grill one fish, the utility function is:

$$u(\mathbf{V}^{\pi}) = \min\left(V^{\pi}_{fish}, \left\lfloor \frac{V^{\pi}_{wood}}{2} \right\rfloor\right).$$

This is a non-linear utility function. Given this non-linear function, it is clear that a deterministic stationary policy would not suffice, i.e., any deterministic stationary policy would lead only gathering rewards in one objective, and thus to a utility of 0. Furthermore, applying a priori scalarisation by applying the utility function directly on the immediate rewards is also not possible, e.g., $u((1,0)) = 0$, which does not correspond to the utility of the user.

Instead a non-stationary policy that conditions on both state and time would lead to reasonable rewards. However, conditioning on how many fish the agent has already obtained can lead to even better policies. Furthermore, because the utility depends on the past as well as the future rewards, it no longer suffices to optimise with respect to the future rewards only. Therefore, we need methods that explicitly take both the past and future rewards into account while optimising a policy for a given state s. We make this point more formal in Section 3, and verify this experimentally in Section 4.

Finally, we note that a randomised policy that gathers only wood with a probability 0.5, and only fish with a probability of 0.5, would lead to a satisfactory scalarised value under SER. Such policies are called mixture policies and are well-known to perform well under SER [19]. However, such a policy would still yield a scalarised value of 0 under ESR. This stresses the need for different methods when we want to optimise for the ESR criterion.

1.2 Contributions

In this paper we make the following contributions. For multi-objective MDPs with non-linear utility functions and a utility based on the expected scalarised returns (ESR) optimality criterion—formalised in Section 2—we propose the Expected Utility Policy Gradient (EUPG) algorithm, which builds on policy gradient. This fills an important gap in the multi-objective literature, i.e., the non-existence of methods for the ESR formulation, which was identified as a gap in 2013 in the survey by Roijers et al. [10]. We show in Section 3 that this setting requires a reformulation of the loss function for policy gradient algorithms. Specifically, the loss function needs to contain the utility function, and the rewards gathered up until a given timestep, t, as well as from t onwards. Furthermore, we argue that while it is possible for EUPG to optimise policies that condition on different parts of the state information, on the previously gathered rewards leads to the best results. We compare EUPG to corresponding variants that omit the previously gathered rewards from the loss function, and show that our loss function is necessary to obtain good policies in MOMDPs with non-linear utility functions under the ESR criterion in Section 4. Furthermore, we show empirically that explicitly conditioning the policies indeed leads to the better policies.

2 BACKGROUND

In this section, we provide the necessary background for multi-objective MDPs, and in particular the different optimisation criteria that can be used. Furthermore, we provide background with regards to Policy Gradient for single-objective MDPs, which we use as a starting point for our main algorithmic contribution.

2.1 Multi-Objective MDPs

A *multi-objective Markov decision process* (MOMDP) [10] is a tuple $\langle \mathcal{S}, \mathcal{A}, T, \mathbf{r}, \gamma \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the transition function, γ is the discount factor that determines the relative importance of immediate rewards with respect to later rewards, and $\mathbf{r} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$ is a d-dimensional vector-valued expected immediate reward function. The reward at each timestep $\mathbf{r}_t$ resulting from taking an action a_t in a state s_t, is a random variable such that $E[\mathbf{r}_t | s_t, a_t] = \sum_{s'} T(s,a,s')\mathbf{r}(s,a,s')$. In this paper, we assume finite-horizon MOMDPs. Therefore, policies typically condition on the timestep as well as the state. All our results also hold for infinite-horizon episodic MOMDPs (with or without conditioning on the timestep).

In most multi-objective reinforcement learning research, an agent aims to compute a *policy* π that optimises the utility of the *expected return*:

$$\pi^* = \arg\max_{\pi} u(E[\sum_{t=0}^{\infty} \gamma^t \mathbf{r}_t \mid \pi, s_0]),$$

where the utility function, u, can be any monotonically increasing function in all objectives. This formulation is known as the *scalarised expected return (SER)* optimisation criterion [10]. SER is the correct criterion if a policy can be executed multiple times for the same user, and it is indeed the average sum of rewards over multiple episodes that determines the utility for the user. For example, if the user is a mining company that mines different resources that it supplies to its costumers, the average amount of the different resources is what is important to this company.

In contrast, there are many domains in which it is not the average sum of rewards over multiple episodes that determines the utility. For example, imagine that the policy corresponds to a treatment plan for a serious illness. In that case, for a single patient the policy is only ever executed once. The utility of such a policy then depends on a single rollout, and we should maximise the expected utility over single policy executions, i.e.,

$$\pi^* = \arg\max_{\pi}(E[u(\sum_{t=0}^{\infty}\gamma^t \mathbf{r}_t) \mid \pi, s_0]. \quad (1)$$

This is known as the *expected scalarised return (ESR)* optimisation criterion. The lack of methods for ESR optimisation methods for MOMDPs was stated as an important open problem in the seminal survey on MOMDPs [10]. To our knowledge, this is the first paper to address this problem.

In this paper we assume that the utility function, u, is known. We therefore require to find a single policy that optimises the expected utility of the discounted sum of rewards (ESR). This is thus a single-policy scenario, known as a *known weights setting*. In this setting, if u would be linear, we can suffice with single-objective methods. The challenge in this setting is if u is non-linear, in which we require explicitly multi-objective methods [10].

2.2 Policy Gradient

To tackle the above identified problem, we aim to construct an RL algorithm for MOMDPs under ESR. To do so, we start from single-objective policy gradient. Policy Gradient [17, 27] is a Reinforcement Learning algorithm that directly learns a parametric policy π_θ, instead of learning a value-function and inferring a policy from it [24]. After each episode, the trace of states, actions and obtained rewards is used to compute (s_t, a_t, R_t) state-action-return tuples, with $R_t = \sum_{i=0}^{\infty}\gamma^i r_{t+i}$. Then, those tuples are used to optimise the following loss:

$$\mathcal{L}(\pi) = -\sum_{t=0}^{T} R_t \log(\pi_\theta(a_t|s_t)). \quad (2)$$

Because, as a Monte Carlo method, policy gradient uses the returns as input for its updates, rather than expected returns (values), this makes it a good starting point for ESR optimisation in learning for MOMDPs.

In most cases, π_θ is represented using a neural network, with θ as weights. The optimisation process then consists of computing the gradient $\delta = \frac{\partial\mathcal{L}(\pi)}{\partial\theta}$, then moving θ one step in direction of the gradient:

$$\theta \leftarrow \theta + \alpha\delta. \quad (3)$$

Because the loss $\mathcal{L}(\pi)$, and hence its gradient, depends on the returns obtained by following the policy π_θ, only a single gradient step can be taken after each episode. Once a gradient step has been taken, another episode must be run, to obtain fresh experiences and returns from the environment. However, multiple episodes resulting from the same policy can be used for a single gradient step.

3 EXPECTED UTILITY POLICY GRADIENT

To optimise the expected scalarised returns (ESR, Equation 1), we take policy gradient for single-objective MDPs as a starting point. As a Monte-Carlo method, policy gradient explicitly uses the returns to optimise the policy. For convenience, let us split the returns into a rewards accrued until a timestep τ, $R^-_\tau = \sum_{t=0}^{\tau-1}\gamma^t \mathbf{r}_t$, and the future returns.

As any single-objective method, policy gradient exploits the fact that returns are additive and that maximisation only needs to happen with respect to the future returns, i.e., if until a given timestep τ, the accrued returns are R^-_τ, the best policy to follow thereafter is the one that optimised the returns from thereon, irrespective of R^-_τ. For multi-objective MOMDPs with a non-linear utility function u however, this is not the case, as:

$$\max_{\pi} E[u(\mathbf{R}^-_\tau + \sum_{t=\tau}^{T-1}\gamma^t \mathbf{r}_t)|\pi, s_t] \neq u(\mathbf{R}^-_\tau) + \max_{\pi} E[u(\sum_{t=\tau}^{T-1}\gamma^t \mathbf{r}_t)|\pi, s_t].$$

For example, consider the motivating example of Section 1.1, and imagine that there are five time steps left and the agent is still at the river. If the agent already has two raw fish, the agent should move to the woods, to try to obtain four pieces of wood resulting in a utility of 2. However, if the agent has been unlucky and has not caught any fish yet, it would be best to try to gather fish for one or two more time steps, hopefully resulting in one raw fish, which (in combination with gathering wood for 2 or 3 time steps) could still lead to a utility of 1. This implies that in order to optimise a policy for a state s_t at time t, we need to take the returns already accrued in the past into account.

In policy gradient, the policies are gradually adapted towards the attained utility by gradient descent with respect to the loss function (Equations 2 and 3). We thus need to redefine this loss function to reflect the expected utility over returns. To do so, we need to take both past and future rewards into account. For a given policy execution, and a timestep τ we define the (vector-valued) returns before τ as

$$\mathbf{R}^-_\tau = \sum_{t=0}^{\tau-1}\gamma^t \mathbf{r}_t,$$

and the returns from τ until the horizon T as

$$\mathbf{R}^+_\tau = \sum_{t=\tau}^{T-1}\gamma^t \mathbf{r}_t.$$

We then adapt the loss function to apply to finite-horizon multi-objective MOMDPs under the ESR optimality criterion:

$$\mathcal{L}(\pi) = -\sum_{t} u(\mathbf{R}^-_\tau + \mathbf{R}^+_\tau)\log\pi_\theta(a|s, \mathbf{R}^-_\tau, t). \quad (4)$$

Note that the the utility function is now inside the loss function, and that in addition to s, the policy can condition on the previously accrued returns $\mathbf{R}^-_\tau$ and t (because it is a finite-horizon setting). For infinite horizon settings we can remove the conditioning on t.

4 EXPERIMENTS

To test how successful EUPG is in terms of finding a policy that optimises the expected user utility of the returns (ESR, Equation 1) we perform experiments on two types of problems. In all measurements, we average over 16 runs.

Problem specifications. We consider two MOMDP: first we consider the 2-state 2-objective gathering MOMDP of Section 1.1 (Figure 2). Secondly, we consider a randomly generated MOMDP with limited underlying structure as specified in the MORL-Glue benchmark suite [21]. We use an MOMDP with $|S| = 100$ states, $|A| = 8$ actions and 4 objectives. For the first three objectives we generate strictly positive rewards, and for the fourth objective strictly negative rewards. The transition function $T(s, a, s')$ is generated using $N = 12$ possible successor states per action, with random probabilities drawn from a uniform distribution. To ensure that every state is reachable from every state, it is enforced that for every state with a number x, $x{+}1 \bmod |S|$ is one of the successor states for one of the actions. As the utility function we use:

$$u(\mathbf{v}) = v_1 + v_2^2 + v_3^2 + 0.1v_4. \tag{5}$$

Note that this function is monotonically increasing on the domain of the returns, as v_2 and v_3 are strictly positive.

Neural Networks. For these experiments we employ relatively simple neural networks. The input layer consists of one node per state, for which all values are 0, except for the current state whose value is 1 (i.e., one-hot encoding), as well as an input node for the timestep, t. If the policy conditions on the accrued rewards so far as well, there is one extra input node. Then there is a single fully connected hidden layer with 100 neurons. Finally, the output layer has one node per action.

Ablation. We perform an ablative study with respect to our two additions to just applying the utility function to the future return in Policy Gradient, i.e., adding $\mathbf{R}_\tau^-$ inside the utility function, and conditioning the policy on $\mathbf{R}_\tau^-$, leading to three alternative loss functions. Starting from our full loss function (Equation 4), we show that having $\mathbf{R}_\tau^-$ inside the utility is necessary and leads to better utility than only looking at the future returns (even if the policy does condition on the previously accrued rewards) by comparing against the following alternate loss function:

$$\mathcal{L}(\pi) = -\sum_t u(\mathbf{R}_\tau^+) \log \pi_\theta(a|s, \mathbf{R}_\tau^-, t). \tag{6}$$

Furthermore, we also investigate what happens if we remove extra policy $\mathbf{R}_\tau^-$ conditioning, with $\mathbf{R}_\tau^-$ inside the utility:

$$\mathcal{L}(\pi) = -\sum_t u(\mathbf{R}_\tau^- + \mathbf{R}_\tau^+) \log \pi_\theta(a|s, t), \tag{7}$$

as well as without:

$$\mathcal{L}(\pi) = -\sum_t u(\mathbf{R}_\tau^+) \log \pi_\theta(a|s, t). \tag{8}$$

We expect the conditioning on $\mathbf{R}_\tau^-$ to be especially important in the 2-state gathering MOMDP, as it is highly important to remember how many fish are already caught (as discussed in Section 1.1 and 3).

4.1 Results

To measure the performance of EUPG in a setting where it is highly important to remember the previously accrued rewards we run it on the 2-state gathering MOMDP specified in Section 1.1, and compare it to the ablated versions of EUPG (Figure 2). As expected, the versions with policies that condition on the previously accrued reward

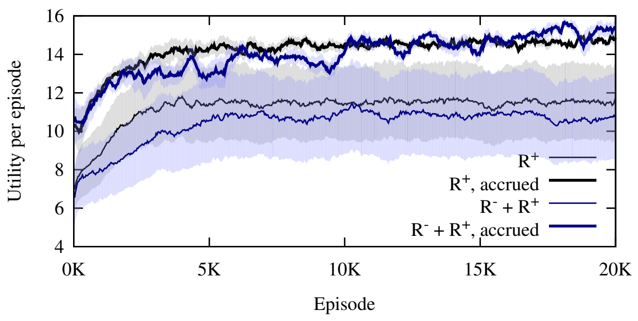

Figure 2: Learning curve (utility per episode) as a function of the episode number, averaged over 16 runs, for the 2-state gathering MOMDP of Section 1.1, for EUPG (using the loss of Equation 4, thick blue line) and the ablated versions according to Equations 6 (thick black line), 7 (thin blue line), and 8 (thin black line).

perform much better than the ones that do not. Furthermore, when we compare EUPG to the ablated version that lacks the previously accrued rewards inside the utility function (Equation 6, the thick black line in the Figure), the ablated version seems to learn a bit faster, but quickly plateaus, while EUPG continues to learn steadily, and slowly overtakes the ablated versions.

The 2-state gathering MOMDP is tailored towards it being highly important to condition on the previously accrued rewards. Furthermore, because the structure of the MOMDP is relatively simple, an agent can learn to do well, even without the exactly right version of the utility. Therefore, we also compare EUPG to its ablated versions on a random 100-state MOMDP (Figure 3). In this more complex MOMDP it is immediately obvious that it is essential to have the previously accrued rewards inside of the utility function in the loss function (the blue lines in the Figure).

When comparing the performance of EUPG to that of its ablated version in the 100-state random MOMDP, the final performance is not much different. This can be explained by the fact that the 100-state random MOMDP is ergodic, and the rewards are highly randomised. Furthermore, EUPG learns a bit slower than the ablated version, as it has extra inputs to condition on, leading to more parameters to tune in the neural network. However, we know from the gathering MOMDP that conditioning the policy on the previously accrued rewards can be essential in MOMDPs as well. We therefore conclude that all elements of EUPG are necessary to attain good utility in RL for MOMDPs under the ESR optimality criterion.

5 RELATED WORK

As mentioned above, most multi-objective reinforcement learning research employs the SER optimality criterion [10, 22, 26]. Perhaps, an important cause of this is that most MORL research follows the (older) axiomatic approach, i.e., they assume that the Pareto front is the correct solution concept, as is common in multi-objective optimisation. Assuming this axiom, there is no need to derive the

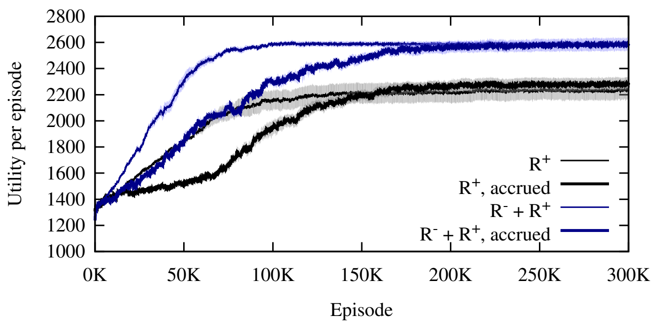

Figure 3: Learning curve (utility per episode) as a function of the episode number, averaged over 16 runs, for a 100-state Random MOMDP, for EUPG (using the loss of Equation 4, thick blue line) and the ablated versions according to Equations 6 (thick black line), 7 (thin blue line), and 8 (thin black line).

appropriate solution concept from the ways the policies are used in practice, and the distinction between SER and ESR stays hidden.

The most related SER methods are those proposed by Geibel for constrained MDPs [7] and Vamplew et al.'s steering approach for Pareto-optimal MORL [20], which have consider the use of policies conditioned on the previously accrued return, $\mathbf{R}^-_\tau$, as a means to handle non-linear utility.

One other Monte-Carlo method, that could possibly also be extended to work for ESR rather than SER, is multi-objective Monte-Carlo tree search (MO-MCTS) [23]. In this algorithm, roll-outs are performed after going down branches of a search tree, to estimate the vector-valued expected returns of the policies implied by the branches of the search tree. To reformulate this approach to apply to ESR, it should be the utility of the returns that are maximised, and the rewards attained so far should be stored inside the nodes of the search tree. It would be interesting to compare our method to such an approach, and an interesting opportunity for future work.

Rather than SER-methods for multi-objective MDPs, our approach is probably most related to preference-based RL, in which multi-objective rewards are not available but user preferences can be queried by asking the user to compare two roll-outs of (different) policies. A good example of this class of methods is *preference-based (direct) policy learning (PPL)* [1, 2]. Like our method, PPL builds off direct policy learning with policy gradient [9]. PPL generates histories, and lets a user compare these histories to provide preferences between them. PPL then tries to infer the scalar sums of rewards, i.e., the returns, that could underlie these preferences. Due to the fact that these returns are scalar, rather than vector-valued as we assume in this paper, non-linear preferences with respect to desirable features of these trajectories cannot be inferred.

Preference-based approximate policy iteration [6] employs roll-outs to learn preferences on state-action level, i.e., given a state they attempt to learn the preferred action. However, as we have shown in this paper, this way of learning preferences is fundamentally incompatible with the ESR-formulation in MORL, as the optimal action can depend on the previously accrued rewards if preferences are non-linear with respect to the objectives.

An algorithm in the class of preference-based RL algorithms that avoids estimating scalar returns is preference-based evolutionary direct policy search [4]. Here, candidate solutions are generated using an evolutionary approach, and full histories are directly compared to determine the Condorcet winner between these candidates. While this permits non-linear preferences, it does not make use of the observed vector-valued rewards in MOMDPs. We argue that in many real-world problems, while it would be hard to define scalar reward functions—as the authors also argue—it is in fact much easier to define multiple measurable objectives that represent desirable features of a solution. In such settings, preference-based RL approaches would loose this information.

6 CONCLUSION

In this paper, we proposed Expected Utility Policy Gradient. To our knowledge EUPG is the first algorithm for learning with non-linear utility functions in MOMDPs under the Expected Scalarised Returns (ESR) optimality criterion which was identified as an important gap in the literature in [10]. In order to be able to learn in this setting, we showed that the accrued returns until the current timestep needed to be included in the conditioning of the policy, and inside the utlity function inside the loss function of EUPG (this utility function needs to be inside the loss function for ESR). We showed empirically that EUPG can learn successful policies for MOMDPs under ESR, and that all elements of EUPG are necessary for doing so. We thus conclude that EUPG fills an important research gap.

In future work, we aim to significantly extend our experiments. Furthermore, this method can also be applied for learning in MOMABs [3, 5], MOPOMDPs [12, 28], and in multi-objective versions of partially observable semi-MDPs [16].

One particularly interesting setting we intend to investigate is the application of multi-objective reinforcement learning for safe RL (e.g., [15]). Particularly, we believe that in safe RL, we are typically not interested in the expected value of the safety objective to trade-off against the expected values for other objectives, but rather the safety objective should have acceptable values for each individual policy execution. In other words, we think that safe RL is a typical ESR setting.

We aim to combine our this work with interactive online multi-objective reinforcement learning approaches, in which the utility function (partially) unknown in the beginning, and must be learned from interaction with the user while simultaneously interacting with the environment [13, 14]. In the interactive online MORL setting, we ultimately aim to learn a single policy, as EUPG does. The main difference is that the utility function is highly uncertain in the beginning, adding an important additional challenge.

Another open question is how to combine the ESR setting with a multi-policy scenario, rather than a single-policy one. Specifically, following the utility-based approach, a solution set that contains at least one optimal policy for every possible utility function should be derived.

ACKNOWLEDGEMENTS

The second author is “Aspirant” with the Research Foundation – Flanders (FWO, Belgium), grant number 1129317N. The first author was “Postdoctoral Fellow” with the FWO, (grant number 12J0617N) for the first part of this research.

REFERENCES

[1] Riad Akrour, Marc Schoenauer, and Michele Sebag. 2011. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 12–27.

[2] Riad Akrour, Marc Schoenauer, and Michèle Sebag. 2012. April: Active preference learning-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 116–131.

[3] P. Auer, C.-K. Chiang, R. Ortner, and M. M. Drugan. 2016. Pareto front identification from stochastic bandit feedback. In *AISTATS*. 939–947.

[4] Róbert Busa-Fekete, Balázs Szörényi, Paul Weng, Weiwei Cheng, and Eyke Hüllermeier. 2014. Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm. *Machine Learning* 97, 3 (2014), 327–351.

[5] M. M. Drugan and A. Nowé. 2013. Designing multi-objective multi-armed bandits algorithms: A study. In *IJCNN*. IEEE, 1–8.

[6] Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. 2012. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning* 89, 1 (2012), 123–156.

[7] Peter Geibel. 2006. Reinforcement learning for MDPs with constraints. In *European Conference on Machine Learning*. Springer, 646–653.

[8] D. J. Lizotte, M. Bowling, and S. A. Murphy. 2010. Efficient reinforcement learning with multiple reward functions for randomized clinical trial analysis. In *ICML*.

[9] Jan Peters and Stefan Schaal. 2008. Reinforcement learning of motor skills with policy gradients. *Neural networks* 21, 4 (2008), 682–697.

[10] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. 2013. A Survey of Multi-Objective Sequential Decision-Making. *JAIR* 48 (2013), 67–113.

[11] D. M. Roijers and S. Whiteson. 2017. Multi-Objective Decision Making. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 11, 1 (2017), 1–129.

[12] Diederik Marijn Roijers, Shimon Whiteson, and Frans A Oliehoek. 2015. Point-Based Planning for Multi-Objective POMDPs.. In *IJCAI*. 1666–1672.

[13] D. M. Roijers, L. M. Zintgraf, P. Libin, and A. Nowé. 2018. Interactive Multi-Objective Reinforcement Learning in Multi-Armed Bandits for Any Utility Function. In *ALA workshop at FAIM*. 8. To Appear.

[14] D. M. Roijers, L. M. Zintgraf, and A. Nowé. 2017. Interactive Thompson Sampling for Multi-objective Multi-armed Bandits. In *Algorithmic Decision Theory*. 18–34.

[15] Jonathan Serrano-Cuevas, Eduardo F. Morales, Pablo Hernandez-Leal, Daan Bloembergen, and Michael Kaisers. 2018. Learning on a Budget Using Distributional RL. In *ALA workshop at FAIM*. 6. To Appear.

[16] Denis Steckelmacher, Diederik M Roijers, Anna Harutyunyan, Peter Vrancx, Hélène Plisnier, and Ann Nowé. 2018. Reinforcement learning in POMDPs with memoryless options and option-observation initiation sets. In *AAAI*.

[17] Richard Sutton, D McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Neural Information Processing Systems (NIPS)* (2000), 7. https://doi.org/10.1177/004728757601400384

[18] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[19] P. Vamplew, R. Dazeley, E. Barker, and A. Kelarev. 2009. Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks. In *Advances in Artificial Intelligence*. 340–349.

[20] Peter Vamplew, Rustam Issabekov, Richard Dazeley, Cameron Foale, Adam Berry, Tim Moore, and Douglas Creighton. 2017. Steering approaches to Pareto-optimal multiobjective reinforcement learning. *Neurocomputing* 263 (2017), 26–38.

[21] Peter Vamplew, Dean Webb, Luisa M. Zintgraf, Diederik M. Roijers, Richard Dazeley, Rustam Issabekov, and Evan Dekker. 2017. MORL-Glue: A Benchmark Suite for Multi-Objective Reinforcement Learning. In *BNAIC*. 389–390.

[22] K. Van Moffaert and A. Nowé. 2014. Multi-objective reinforcement learning using sets of Pareto dominating policies. *JMLR* 15, 1 (2014), 3483–3512.

[23] Weijia Wang and Michele Sebag. 2012. Multi-objective monte-carlo tree search. In *Asian conference on machine learning*, Vol. 25. 507–522.

[24] Christopher Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[25] C Ch White and Kwang W Kim. 1980. Solution procedures for vector criterion Markov decision processes. *Large Scale Systems* 1, 4 (1980), 129–140.

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

[27] Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning* 8, 3 (1992), 229–256. https://doi.org/10.1023/A:1022672621406

[28] Kyle Hollins Wray and Shlomo Zilberstein. 2015. Multi-Objective POMDPs with Lexicographic Reward Preferences.. In *IJCAI*. 1719–1725.

[29] Kyle Hollins Wray, Shlomo Zilberstein, and Abdel-Illah Mouaddib. 2015. Multi-Objective MDPs with Conditional Lexicographic Reward Preferences.. In *AAAI*. 3418–3424.