Interactive Multi-Objective Reinforcement Learning in Multi-Armed Bandits for Any Utility Function

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

Luisa M. Zintgraf
University of Oxford (UK)
Vrije Universiteit Brussel (BE)

Pieter Libin
Vrije Universiteit Brussel
Brussels, Belgium

Ann Nowé
Vrije Universiteit Brussel
Brussels, Belgium

ABSTRACT

In interactive multi-objective reinforcement learning (MORL), an agent has to simultaneously learn about the environment and the preferences of the user, in order to quickly zoom in on those decisions that are likely to be preferred by the user. In this paper we study interactive MORL in the context of multi-objective multi-armed bandits. Contrary to earlier approaches to interactive MORL, we do not make stringent assumptions about the utility functions of the user, but allow for non-linear preferences. We propose a new approach called Gaussian-process Utility Thompson Sampling (GUTS), which employs non-parametric Bayesian learning to allow any type of utility function, exploits monotonicity information, and limits the number of queries posed to the user by ensuring that questions are statistically significant. We show empirically that, in contrast to earlier methods, GUTS can learn non-linear preferences, and that the regret and number of queries posed to the user are highly sub-linear in the number of arm pulls.

KEYWORDS

Multi-Objective Reinforcement Learning; Multi-objective multi-armed bandits; Interactive Online MORL

1 INTRODUCTION

Real-world decision problems often require learning about the effects of various decisions 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 *multi-armed bandit (MAB)* [2]. However, many real-world decision problems have multiple possibly conflicting objectives [26]. For example, an agent learning the best strategy to deploy ambulances from a set of alternatives may want to minimise response time, while also minimising fuel cost and the stress levels of the drivers. When the user is unable to accurately and a priori specify preferences with respect to such objectives for all hypothetically possible trade-offs, the user needs to be informed about the values of actually available trade-offs between objectives in order to make a well-informed decision. For such problems, MABs no longer suffice, and need to be extended to *multi-objective multi-armed bandits (MOMABs)* [3, 16], in which the effects of alternative decisions are measured as a vector containing a value for each objective.

Most research on MOMABs [3, 15, 41] focusses on producing *coverage sets* [26], i.e., sets of all possibly optimal alternatives given limited *a priori* information about the possible utility functions of the user(s), as the solution to a MOMAB. A minimal assumption is that the utility function is monotonically increasing in all objectives. This leads to the popular *Pareto front* concept as the coverage set.

Typically, such research assumes an offline learning setting, i.e., there is a learning phase in which there is no interaction with the user, after which the coverage set is presented to the user in order to select one final alternative. The rewards attained during learning are assumed to be unimportant; only the expected rewards of the final alternative is assumed to be relevant to the utility of the user. However, this is a limiting assumption. For example, the ambulances discussed above may well be deployed while still learning about the expected value of the deployment strategies in the different objectives. In such cases, we need to use our interactions with the environment efficiently, as we care about the rewards accrued during learning. When the learning agent can interact with the user and *elicit* preferences from the user during learning [29], the agent can focus its learning on strategies that the user finds most appealing quickly. This is thus an *online* and *interactive* learning scenario.

Roijers et al. ([2017]) recently proposed the *interactive Thompson sampling (ITS)* algorithm for online interactive reinforcement learning in MOMABs, and showed that the number of interactions with the environment in an multi-objective interactive online setting can be reduced to approximately the same amount of interactions as in single-objective state-of-the-art algorithms such as Thompson sampling. However, they make highly restrictive assumptions about the utility functions of users. Specifically, they assume that user utility is a weighted sum of the values in each objective.

In real-life multi-objective RL, it is essential to be able to deal with users that have non-linear preferences. Therefore, we propose the *Gaussian-process Utility Thompson Sampling (GUTS)* algorithm. GUTS can deal with non-linear utility functions, enabling interactive reinforcement learning in MOMABs for the full range of multi-objective RL settings. To do so, we make the following key improvements over ITS. Firstly, rather than using a pre-specified model-space for utility functions, we employ non-parametric Bayesian learning, i.e., *Gaussian Processes* (GPs) [8, 12, 24], to be able to learn arbitrary utility functions. GPs are data-efficient [14], and can thus quickly learn the relevant part of the utility function. Secondly, because GPs cannot enforce monotonicity with respect to each objective (i.e., the minimal assumption in multi-objective RL), as GPs cannot incorporate monotonicity constraints, GUTS enforces this monotonicity separately from the GP model of the utility. Finally, we propose *statistical significance testing* to only ask the user about the preferences between statistically significantly different arms, to prevent irrelevant questions in the earlier iterations, and show empirically that this does not increase regret.

2 BACKGROUND

In this section we define our problem setting, i.e., multi-objective multi-armed bandits (MOMABs), and how we model user utility. First however, we define the scalar version of a MOMAB, i.e, a MAB.

Definition 2.1. A scalar *multi-armed bandit (MAB)* [35] is a tuple $\langle \mathcal{A}, \mathcal{P} \rangle$ where

- $\mathcal{A}$ is a set of *actions* or *arms*, and
- $\mathcal{P}$ is a set of probability density functions $P_a(r) : \mathbb{R} \to [0, 1]$ over *scalar* rewards r associated with each arm $a \in \mathcal{A}$.

We refer to the the mean reward of an arm as $\mu_a = \mathbb{E}_{P_a}[r] = \int_{-\infty}^{\infty} rP_a(r)dr$, and to the optimal reward as the mean reward of the best arm $\mu^* = \max_a \mu_a$.

The goal of an agent interacting with a MAB is to minimise the expected regret.

Definition 2.2. The expected cumulative *regret* of pulling a sequence of arms for each timestep between $t = 1$ and a time horizon T (following the definition of Agrawal and Goyal, [2012]), is

$$\mathbb{E}\left[\sum_{t=1}^{T} \mu^* - \mu_{a(t)}\right],$$

where $a(t)$ is the arm pulled at time t, and $n_a(T)$ is the number of times arm a is pulled until timestep T.

At the start, the agent knows nothing about $\mathcal{P}$, and can only obtain information about the reward distributions by pulling an arm $a(t)$ each timestep, obtaining a sample from the corresponding $P_{a(t)}$.

In scalar MABs, agents aim to find the arm a that maximises the expected scalar reward. In contrast, in multi-objective problems there are n objectives that are all desirable. Hence, the stochastic rewards $\mathbf{r}(t)$ and the expected rewards for each alternative $\boldsymbol{\mu}_a$ are vector-valued.

Definition 2.3. A *multi-objective multi-armed bandit (MOMAB)* [3, 41] is a tuple $\langle \mathcal{A}, \mathcal{P} \rangle$ where

- $\mathcal{A}$ is a finite set of *actions* or *arms*, and
- $\mathcal{P}$ is a set of probability density functions, $P_a(\mathbf{r}) : \mathbb{R}^d \to [0, 1]$ over *vector-valued* rewards $\mathbf{r}$ of length d, associated with each arm $a \in \mathcal{A}$.

Rather than having a single optimal alternative as in a MAB, MOMABs can have multiple arms whose value vectors are optimal for different preferences that users may have. Following the utility-based approach to MORL [26], we assume such preferences can be expressed using a utility function, $u(\boldsymbol{\mu})$ that returns the *scalarised* value of $\boldsymbol{\mu}$. Using this utility function, the online regret for the interactive multi-objective reinforcement learning setting is as follows.

Definition 2.4. The expected cumulative *user regret* of pulling a sequence of arms for each timestep between $t = 1$ and a time horizon T in a MOMAB is

$$\mathbb{E}\left[\sum_{t=1}^{T} u(\boldsymbol{\mu}^*) - u(\boldsymbol{\mu}_{a(t)})\right],$$

where $a(t)$ is the arm pulled at time t, $\boldsymbol{\mu}^* = \arg\max_a u(\boldsymbol{\mu}_a)$, and $n_a(T)$ is the number of times arm a is pulled until timestep T.

Roijers et al. ([2017]) require that $u(\boldsymbol{\mu})$ is of the form $\mathbf{w} \cdot \boldsymbol{\mu}$, where $\mathbf{w}$ is a vector of weights per objective that sum to one. However, in many cases, users cannot be assumed to have such simple linear preferences. Instead, users will often exhibit some degree of non-linearity. To accommodate for any type of user, we want to only make the minimal assumption, i.e., that $u(\boldsymbol{\mu})$ is monotonically increasing in all objectives. This assumption implies that more is better in each objective, which follows from the definition of an objective.

Because $u(\boldsymbol{\mu})$ can in general be of any form, we do not want to restrict ourselves to a particular model space. Therefore, we model $u(\boldsymbol{\mu})$ as a *Gaussian Process* (GP) [24]. A GP can be used to approximate any function, while also capturing its uncertainties by assigning to each point in its domain a normal distribution – the mean value reflecting the expected value, and the variance reflecting the uncertainty about the function value at that point. GPs are fully specified by a mean function m and a kernel k,

$$u(\boldsymbol{\mu}) \sim GP(m(\boldsymbol{\mu}), k(\boldsymbol{\mu}, \boldsymbol{\mu}')) \ . \tag{1}$$

Before any comparisons between value vectors are posed to the user, the GP is initialised by choosing a prior mean function and the kernel function. The mean function expresses a prior estimate for the utilities of value vectors. A common non-informative prior is the zero-mean prior, $u(\boldsymbol{\mu}) = 0$, which we also use for the experiments in this paper. The kernel function defines how data points (tuples of the form $(x, u(x))$) influence each other. A common choice for the kernel, which we also use, is the squared exponential kernel:

$$k(x, x') = \exp\left(-(2\theta)^{-1}||x - x'||^2\right),$$

in which θ controls the distance over which points influence each other. The prior belief $P(u)$ about the utility function $u(\boldsymbol{\mu})$ is updated in light of observation data, C, and its likelihood $P(C|u)$ using Bayes' rule $P(u|C) \propto P(u)P(C|u)$, where $P(C|u)$ is the likelihood of the data given $u(\boldsymbol{\mu})$. The posterior $P(u|C)$ is the updated belief about the user's utility, and the current approximation of the utility function. In this paper, the data for the GP, C, is given in terms of *pairwise comparisons* between vectors,

$$C = \{\boldsymbol{\mu}_m \succ \boldsymbol{\mu}'_m\}_{m=1}^{M} \ , \tag{2}$$

where $\succ$ denotes a noisy comparison $u(\boldsymbol{\mu}_m) + \varepsilon > u(\boldsymbol{\mu}'_m) + \varepsilon'$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ accounts for noise in the user's utility (i.e., the user is allowed to make small errors). Chu and Ghahramani ([2005b]) introduce a probit likelihood for pairwise comparison data with Gaussian noise, which can be used to update the GP. Given this likelihood, the posterior becomes analytically intractable and has to be approximate (e.g., using Laplace approximation as in [8]).

The main advantages of GPs for our setting are that they do not assume a model-class of functions for $u(\boldsymbol{\mu})$, and that they are sample-efficient, i.e., the number of comparisons necessary to make accurate predictions about which $\boldsymbol{\mu}$ are preferred is small.

3 GAUSSIAN-PROCESS UTILITY THOMPSON SAMPLING

In this section, we propose our main contribution: *Gaussian-process Utility Thompson Sampling (GUTS)* (Algorithm 1). GUTS interacts

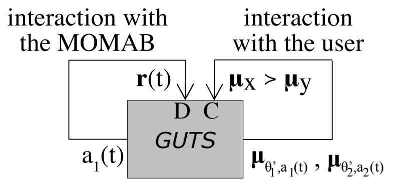

Figure 1: Schematic overview of the GUTS algorithm.

with the environment by pulling arms, and storing the resulting reward-vectors together with the associated arm numbers in a dataset D. GUTS also interacts with the user by posing comparison queries to the user, leading to a dataset C (as in Equation 2). A schematic overview of the interactions of the GUTS algorithm is provided in Figure 1.

GUTS selects the arms to pull by sampling the posterior mean reward distributions as in Thompson Sampling (TS) [35] for single-objective MABs. In contrast to single-objective MABs however, the means sampled from the posteriors in MOMABs are vector-valued (line 5). As there is typically not one arm that has the optimal reward for all objectives, minimising the regret (Definition 2.4) is only possible by also estimating the utility function, $u(\boldsymbol{\mu})$. To model this utility function, GUTS employs a Gaussian Process (GP) (Equation 1), which is a posterior distribution over utility functions, given data. Because the GP is a posterior over utility functions, GUTS can sample $u(\boldsymbol{\mu})$ from this posterior. We assume that the user's utility function does not depend on the available mean reward vectors, and can thus be learnt and sampled independently (lines 7–8). When such a sampled $u(\boldsymbol{\mu})$ is applied to the reward vectors, the action that maximises the utility according to that $u(\boldsymbol{\mu})$ can be found (lines 9–10) and executed (line 19). Aside from needing to sample a utility function from the GP to find the utility-maximising arm, learning the optimal arm to pull given the GP is thus highly similar to TS. However, the main difficulty is in learning to estimate $u(\boldsymbol{\mu})$ accurately and efficiently.

In order to learn $u(\boldsymbol{\mu}_a)$, we need data about the preferences of the user. As mentioned in Section 2 we gather data about the preferences of users in terms of binary comparisons between two value-vectors (Equation 2). This is because it is highly difficult—and thus error-prone—for humans to provide numerical utility evaluations of single items; it is known that such evaluations can differ based on a number of external factors that are not relevant to the data, making the valuations time-dependent, while pairwise comparisons (i.e., asking the question "Which of these two vectors do you prefer?") are more stable [18, 31–33]. In addition to a dataset about the value-vectors associated with the arms, D, (line 2), GUTS thus keeps a dataset of outcomes of pairwise preference queries, C, (line 1). To fill C with meaningful and relevant data, GUTS needs to pose queries to a (human) decision maker. Because a comparison consists of two arms, GUTS draws two sample sets of mean reward vectors from the posteriors of the mean reward vectors given D, θ_1 and θ_2 (line 5), and two sample utility functions from the posterior over utility functions given C (lines 7–8). When these two sets of samples, paired with their respective sampled utility function, disagree on what the optimal arm should be, a preference comparison between the sampled mean vectors of these arms are a candidate query to the user.

Algorithm 1: GP Utility Thompson Sampling

```
Input: Parameter priors on reward distributions, and GP prior
       parameters $m(\boldsymbol{\mu})$ and $k(\boldsymbol{\mu}, \boldsymbol{\mu}')$, countdown, $\alpha$
1  $C \leftarrow \emptyset$;                                   // previous comparisons
2  $D \leftarrow \emptyset$;                                   // observed reward data
3  $cd \leftarrow$ countdown
4  for $t = 1, \ldots, T$ do
5      $\theta_1, \theta_2 \leftarrow$ draw 2 sets of samples from $P(\theta|D)$
6      $\theta'_1, \theta'_2 \leftarrow$ PPrune($\theta_1$), PPrune($\theta_2$)
7      $u_1 \leftarrow$ sample utility function from GP posterior given $C$
8      $u_2 \leftarrow$ sample utility function from GP posterior given $C$
9      $a_1(t) \leftarrow \arg\max_a u_1(\boldsymbol{\mu}_{\theta'_1,a})$
10     $a_2(t) \leftarrow \arg\max_a u_2(\boldsymbol{\mu}_{\theta'_2,a})$
11     $dom \leftarrow \boldsymbol{\mu}_{\theta'_1,a_1(t)} \succ_P \boldsymbol{\mu}_{\theta'_2,a_2(t)} \vee \boldsymbol{\mu}_{\theta'_2,a_2(t)} \succ_P \boldsymbol{\mu}_{\theta'_1,a_1(t)}$
12     $significant \leftarrow$ HotellingT2($D[a_1(t)], D[a_2(t)]$) $\leq \alpha$
13     if $a_1(t) \neq a_2(t) \wedge \neg dom \wedge cd \leq 0 \wedge significant$ then
14         Perform user comparison for $\boldsymbol{\mu}_{\theta'_1,a_1(t)}$ and $\boldsymbol{\mu}_{\theta'_2,a_2(t)}$ and
             add result (in the format, $\boldsymbol{\mu}_x \succ \boldsymbol{\mu}_y$) to C
15         $cd \leftarrow$ countdown
16     else
17         $cd$--
18     end
19     $\mathbf{r}(t) \leftarrow$ play $a_1(t)$ and observe reward
20     append $(\mathbf{r}(t), a_1(t))$ to $D$
21 end
```

GUTS uses a GP to estimate $u(\boldsymbol{\mu})$ from the data in C. GPs are highly general, data-efficient [14], and as parameter-free function approximators can fit any function. However, in multi-objective decision-making we have one more piece of information that we can exploit. We know that $u(\boldsymbol{\mu})$ is monotonically increasing in all objectives; i.e., increasing the value of a mean reward vector $\boldsymbol{\mu}$ in one objective, without diminishing it in the other objectives can only have a positive effect on utility. This corresponds to the minimal assumption about utility in multi-objective decision making [26], and leads the notion of *Pareto dominance*. An arm a is Pareto-dominated, $\boldsymbol{\mu}_{a'} \succ_P \boldsymbol{\mu}_a$, when there is another arm a' with an equal or higher mean in all objectives, and a higher mean in at least one objective:

$$\boldsymbol{\mu}_{a'} \succ_P \boldsymbol{\mu}_a \overset{\text{def}}{=} (\forall i)\, \mu_{a',i} \geq \mu_{a,i} \ \wedge\ (\exists j)\, \mu_{a',j} > \mu_{a,j}.$$

Because of monotonicity, Pareto-dominated arms cannot be optimal [26]. GUTS enforces this condition on two places: first, any Pareto-dominated arms are excluded from the sets of sampled mean reward vectors, by using a pruning operator PPrune[1] resulting in two smaller sets of arms θ'_1 and θ'_2 (line 6); second, the user will not be queried w.r.t. her preferences for dominating-dominated pairs of vectors from θ'_1 and θ'_2 (line 11), as we already know the answer without posing the question to the user.

Using C we sample utility functions and maximise over the available actions in θ'_1 and θ'_2 (line 7–10). We note that because we estimate u using GPs, GUTS can directly sample values for the

[1] See e.g. [25] for a reference implementation of PPrune.

values in θ'_1 and θ'_2 without fully specifying the entire function $u(\boldsymbol{\mu})$ for all possible $\boldsymbol{\mu}$ in $\mathbb{R}^n$.

Because interactions with a decision maker are an expensive resource, we need to select preference queries between value vectors carefully. Therefore, a) these queries should be relevant with respect to finding the optimal arm (preferences between arms that are both suboptimal are less relevant), b) the arms in the queries should be statistically significantly different from each other, and c) queries should be progressively infrequent, as we assume that the user will be decreasingly motivated to provide more information to the system as time progresses. To make sure that only relevant questions are asked (a), GUTS only poses queries to the user when the two sets of samples and utility functions disagree on the arm that maximises the utility for the user. As more information comes in, we expect the two sets of samples to disagree on which arm is optimal less (c). Additionally, we also want to assure that queries only concern arms for which the difference between values is statistically significant (b). For this, we use Hotellings–T^2 test [19], which is the multi-variate version of the Students-T test. The input for this test is all the data with respect to both arms we want to compare, i.e., $a_1(t)$ and $a_2(t)$, in the data gathered so far (D). The threshold P-value with respect to when we say the difference is not insignificant, α, can be set to a low value. In the experiments we used $\alpha = 0.01$. We note that Hotellings–T^2 test applies to the rewards per arm (in D) are distributed with a multi-variate Gaussian, which we use in our experiments. If the the rewards for the arms would be distributed differently, different significance testing should be employed instead, or can even be removed (at the expense of some additional, unnecessary queries at the beginning) if necessary. [2]

Finally, one might argue that a human decision maker would not always be able to answer all the queries the algorithm might require. For example, the algorithm might simply generate too many questions for the user to answer while it continues to learn about the environment, and when the user has answered a question, the subsequent questions might already be deprecated. To simulate this effect we introduce a simple cool-down of the number of arm-pulls that will be taken before a user is again able to answer another query (line 15 and 17). We hypothesise that while more queries are desirable, GUTS will still be able to achieve sub-linear regret. In the most positive scenario, a higher cool-down could even lead to the algorithm being able to ask more informative queries because it learned more about the environment in the meantime, which could have a positive effect.

4 EXPERIMENTS

To test the performance of GUTS in terms of user regret (Definition 2.4) and the number of queries posed to the user, and to test the effects of different parameter settings, we perform experiments on two types of problems. In Section 4.1, we use a 5-arm MOMAB that requires learning non-linear preferences to show that GUTS is indeed able to handle such non-linear preferences. In Section 4.2, random MOMAB instances are used to perform more extensive experiments.

[2]Please note that assumptions about the distributions of the reward vectors are all that is required for significance testing, and that for significance testing no assumptions regarding the utility function is required.

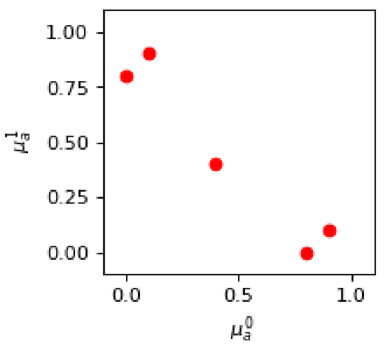

Figure 2: A 5-arm MOMAB

We compare GUTS to single-objective Thompson sampling provided with the ground truth utility functions of the user. While this is an unfair comparison (putting GUTS at a disadvantage), as our setting does not actually allow algorithms to know the ground truth utility function, it does provide insight into how much utility is lost due to having to estimate the utility function via pairwise comparison queries. Furthermore, in Section 4.1 we compare GUTS to ITS [29].

All MOMABs used in these experiments have multi-variate Gaussian reward distributions. This stands in contrast to earlier work, i.e., [29], that uses multi-variate Bernoulli-distributions. For the GPs we use the common zero-mean prior, $m(\boldsymbol{\mu}) = 0$, and squared exponential kernel [24]. When significance testing is used, we use $\alpha = 0.01$ as the significance threshold.

4.1 A 5-arm MOMAB

In this subsection, we employ a 5-arm MOMAB (depicted in Figure 2), with the following ground truth mean vectors: $(0, 0.8)$, $(0.1, 0.9)$, $(0.4, 0.4)$, $(0.8, 0.0)$, and $(0.9, 0.1)$. Note that $(0, 0.8)$ and $(0.8, 0.0)$ are Pareto-dominated. We employ a monotonically increasing utility function:

$$u_5(\boldsymbol{\mu}) = 6.25 \ \max(\mu_0, 0) \max(\mu_1, 0),$$

leading to the arm with mean vector $(0.4, 0.4)$ being optimal with utility 1. Each reward distribution is a multi-variate Gaussian with correlations 0 and in-objective variance 0.005.

To find the optimal arm, a MOMAB-learning algorithm must be able to handle the non-linearity of $u_5(\boldsymbol{\mu})$. In order to test whether GUTS indeed can, we run GUTS on the MOMAB of Figure 2, and compare it to ITS [29], and Thompson Sampling provided with the ground truth utility in Figure 3 (top). The results confirm that GUTS (for all parameter settings) can handle non-linear preferences and has a highly sub-linear regret curve. In contrast, ITS is limited to the space of linear utility functions and does not manage to attain sub-linear regret.

As previously stated, a user may not have time (or is not willing) to answer all questions posed by GUTS, as the user will need time to answer the queries, and may be preoccupied with other tasks as well. Therefore, we introduced the cool-down parameter in the GUTS algorithm to simulate the user answering a smaller portion of the questions. We compare different settings for the cool-down

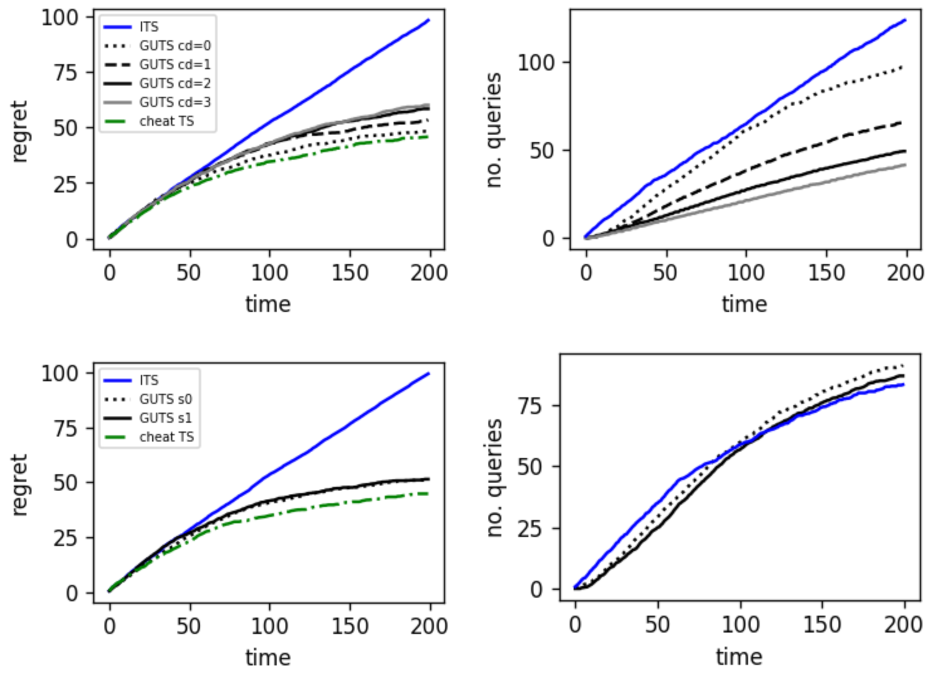

Figure 3: 5-arm MOMAB results for ITS, ground-truth TS, and GUTS with different cool-down settings and significance testing enabled (top), and with cool-down 0 and significance testing either enabled (s1) or disabled (s0) (bottom): (left) regret as a function of arm-pulls with significance testing enabled and different cool-down settings for GUTS, (right) the corresponding number of queries posed to the user as a function of arm-pulls.

parameter (Figure 3, top) we observe that, as expected, no cool-down (i.e., $cd = 0$), performs best in terms of regret, the regret closely approximates the regret of Thompson sampling supplied with the ground truth (Figure 3, top left). The number of queries to the user can be reduced by increasing cool-down (Figure 3, top right). For example, at cool-down 3, the user answers to a query once in at most 4 arm-pulls. We observe that while this comes at the cost of some regret, the learning curves remain highly sub-linear, and GUTS is still able to learn the right preferences, albeit more slowly.

We tested the effect of (disabling) significance testing (Figure 3, bottom). We found no significant effect on regret in this simple problem, and a small effect on the number of queries in favour of the doing significance testing. This is because there is very little variance in the reward distributions of the arms, so after only a few questions the difference between to sets of samples for two arms becomes significant. We show the effect on more complex MOMABs in the next subsection.

In summary, we conclude that GUTS can effectively approximate the utility function for the purpose of selecting the optimal arm in this problem.

4.2 Random MOMABs

Random MOMABs are generated randomly using the number of objectives d, and the number of arms, $|\mathcal{A}|$, as parameters, following the procedure of [29]: $|\mathcal{A}|$ samples $\boldsymbol{\mu}'_a$ are drawn from a d-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}'_a|\boldsymbol{\mu}_{rand}, \Sigma_{rand})$, where $\boldsymbol{\mu}_{rand} = \vec{1}$ (vector of ones), and Σ_{rand} is a diagonal matrix with $\sigma^2_{rand} = (\frac{1}{2})^2$ for each element on the diagonal; this set is normalised such that all $\boldsymbol{\mu}_a$ fall into the d-dimensional unit hypercube, $[0, 1]^d$. Each arm has an associated multi-variate Gaussian reward distribution, with the aforementioned true mean vectors, $\boldsymbol{\mu}_a$, and randomly drawn covariance matrices with covariances ranging from perfectly negatively correlated to perfectly positively correlated, with an average variance magnitude of $\sigma^2_{P,a} = 0.05$ unless otherwise specified.

In order to test whether GUTS can learn to identify the optimal arm, under different settings of the cool-down parameter (i.e., when a user cannot answer all the questions otherwise generated by the algorithm), we run GUTS on random 20-arm MOMABs with randomly generated polynomial utility functions of order 3. In Figure 4, we observe that the regret curves of GUTS (on the left) are all highly sub-linear. At 500 arm-pulls, GUTS with cool-down 0 has only 16.9% more cumulative regret than Thompson Sampling provided with the ground-truth utility function (cheat-TS). Furthermore, we observe that the number of questions for cool-down behaves as we expect: first queries are not yet significant; then the queries become significant leading to a higher number of queries; but the number of queries quickly becomes sub-linear as the approximation of $u(\boldsymbol{\mu})$ becomes better. We thus conclude that GUTS can adequately approximate the utility function in order to identify the optimal arm.

For cool-down 5, GUTS has 29.8% more regret than cheat-TS at 500 arm pulls, for cool-down 10, 36.0% and, for cool-down 20,

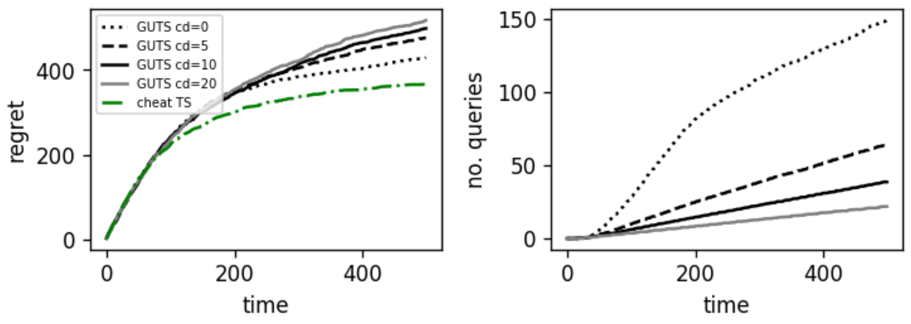

Figure 4: Regret (left) and number of queries posed to the user (right), as a function of arm-pulls, averaged over 5 random, 2-objective, 20-arm MOMAB instances with random utility functions or order 3, with noise $\varepsilon = 0.1$ on the comparisons of the user (about 2% of the optimal utility), for GUTS with different cool-downs.

41.2%. However, cool-down 5 requires only 42.8% of the queries with respect to cool-down 0, and cool-downs 10 and 20 only 25.9% resp. 14.6%. In other words, the increase in regret is less steep than the reduction in the number of queries posed to the user. We hypothesise that this is because GUTS continues learning about the arms before the next question is asked, yielding more information per query. We thus conclude that GUTS remains an effective algorithm when the user cannot answer a query at every arm-pull.

To test whether GUTS can learn effectively in random MOMABs with an increasing number of objectives, and the effect of (disabling) significance testing, we run GUTS with cooldown 10 on 2-, 4, and 6-objective 20-arm random MOMABs with randomly generated polynomial utility functions of order 3 (Figure 5). We chose the cooldown of 10 to simulate a human decision maker that is only able to answer a comparison query after every ten arm-pulls. As can be seen from the figure, the curves remain sub-linear, but less so for higher numbers of objectives, and the difference in regret with cheat-TS increases. This can be explained by the fact that GUTS suffers from the curse of dimensionality, as it has to estimate higher-dimension utility functions, while cheat-TS does not (it is provided with the ground-truth utility function, so it only has to optimise a scalar utility.) This was to be expected; cheat-TS has an unfair advantage, and is only included as a theoretical reference. For this experiment with the cool-down set to ten, the number of queries does not yet reach the point where it starts to be sub-linear. However, we do observe that while significance testing does slightly but consistently decrease the number of queries posed to the user, it has no negative effect on regret.

Finally, in order to check how GUTS handles different levels of noise in the comparisons of the user, we run GUTS on 20, 2-objective, 20-arm, random MOMAB instances with randomly generated polynomial utility functions of order 3 (Figure 6). These randomly generated utility functions have utilities ranging from about 1 for the worst arm to 5 for the best arm. We use 2%, 10%, 20% of this utility as noise, i.e., $\varepsilon = 0.1$, 0.5 and 1.0. For $\varepsilon = 0.1$ and 0.5, we observe very similar regret curves, and no significant difference in the number of queries. For $\varepsilon = 1.0$ we observe the regret and number of query curves suddenly going up; this is due to 3 of the runs, where GUTS suddenly more often samples a non-optimal arm, after a series of erroneous comparisons. However, as the number of queries also goes up, because GUTS becomes unsure what the best arm is, we observe that GUTS does converge back to the optimal arm after a while in two of the three individual runs before the run ends (in the third one the swerve happens closer to the end of the run). We thus conclude that GUTS is robust against different levels of noise.

To summarise, we conclude that: GUTS can effectively handle non-linear utility functions in order to minimise regret in online interactive MORL for MOMABs; GUTS can handle users that answer a comparison query at every arm-pull; and significance testing to prevent asking irrelevant queries in the beginning has no negative effect on the regret.

5 RELATED WORK

Several papers exist that study *offline* multi-objective reinforcement learning in MOMABs [3, 15, 16, 41]. In contrast to the *online interactive MORL*, which we study in this paper, these papers focus on producing a coverage set in a separate learning phase, without interacting with the user. After this learning phase, the interaction with the user to find the optimal alternative from the coverage set needs to be done in a subsequent selection phase. In the work we present here, we are interested in an online setting, in which we aim to minimise online user regret, by estimating the utility function of the user during learning about the MOMAB through arm-pulls.

Relative bandits [40, 43] are also a different but related setting to online interactive learning for MOMABs. Similar to our interactive online MORL for MOMABs, relative bandits assume a hidden utility function which can be queried to obtain pairwise comparisons. Contrary our setting though, the rewards (i.e., reward vectors) cannot be observed, and the comparisons are made regarding *single* arm pulls, rather than the aggregate information over all previous pulls of a given arm as in GUTS (and ITS). The closest algorithms in this field are RUCB [43] and Double Thompson Sampling [40] which keep a preference matrix to determine which *two* arms to pull in *each iteration*.

In evolutionary computing for multi-objective problems, user preferences have been used to bias the search area of the evolutionary algorithm during the optimisation process [6, 13, 34]. There is also

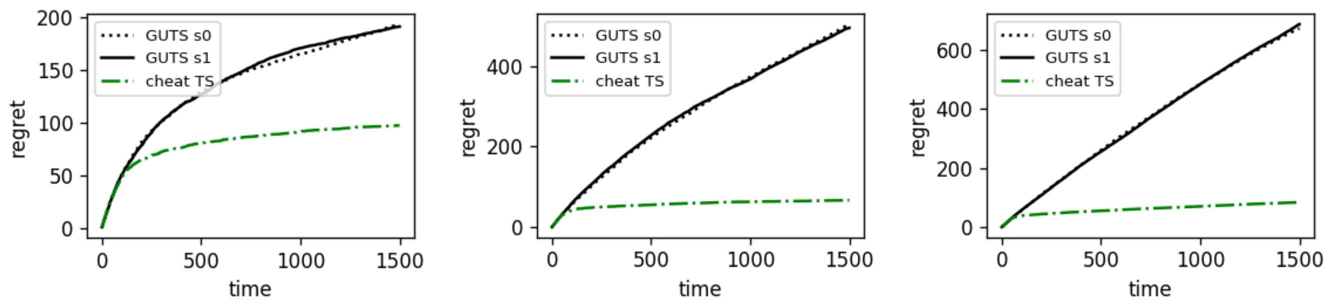

Figure 5: Regret as a function of arm-pulls averaged over 5 random, 20-arm MOMAB instances with 2 (left), 4 (middle) and 6 (right) objectives, with random utility functions or order 3, with noise $\varepsilon = 0.1$ on the comparisons of the user (about 2% of the optimal utility), for GUTS with cool-down 10, and with or without significance testing (s1 or s0).

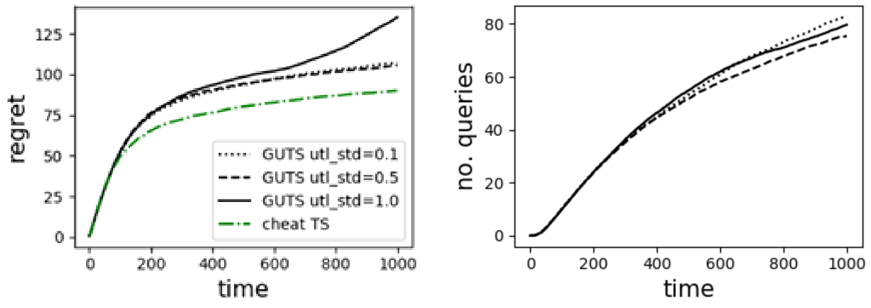

Figure 6: (left) Regret as a function of arm-pulls, (right) number of queries posed to the user as a function of arm-pulls, averaged over 20 random, 2-objective, 20-arm MOMAB instances with random utility functions or order 3, with noise $\varepsilon = 0.1$, 0.5 and 1.0 (resp. 2%, 10%, 20% of the optimal utility) on the comparisons of the user, for GUTS with cool-down 5.

some work in incorporating user guidance in sequential decision problems (e.g., [10, 37]), focusing on single-objective problems.

In multi-objective sequential planning [4], combinatorial decision-making [38], and cooperative game theory [20] preference elicitation w.r.t. (linear) utility functions has been applied as well. These methods however, apply linear programming or equation solving to induce constraints for their respective planning problems, rather than Bayesian learning. It would be interesting to extend our methods to apply to these problem classes and create Bayesian methods for learning. We will discuss multiple options for future work in the next section.

Gaussian processes for user preference elicitation was introduced by [12] and have been used, e.g., in animation design for computer graphics [7] or in learning to rank [11]. Other approaches to user preference elicitation are for example to model the user's preferences as a mixture of Gaussians [9] or as a partially observable Markov Decision Process [5]. We believe that Gaussian processes are more expressive than mixture of Gaussians, and more practical than POMDPs because they require only few data-points. We believe that partially unknown utility functions that take the multi-objective value of policies as input are a more natural way to express preferences in the context of multi-objective decision making.

6 DISCUSSION

In this paper we have proposed Gaussian-process Utility Thompson Sampling (GUTS) for interactive online multi-objective reinforcement learning in MOMABs. Contrary to earlier methods, GUTS can handle non-linear utility functions, which is an essential feature for multi-objective RL algorithms. We have shown empirically that GUTS can effectively approximate non-linear utility functions for the purpose of selecting the optimal arm, leading to only little extra regret if a query can be posed to the user at every arm pull. We further have shown experimentally that in addition to the regret, the number of queries is also sub-linear.

We also tested what the effect is of users not being able to answer a query with every arm-pull. In this case—even if the number of queries is strongly reduced—the regret remains sub-linear, albeit higher than if a query can be posed at every time-step. Therefore, we conclude that GUTS is robust against less questions being answered than generated. Furthermore, our experiments indicate that GUTS is robust against noisy comparisons.

We note that all conclusions are based on empirical evaluations rather than theoretical bounds. This is due to the usage of Gaussian Processes (GPs) that are hard to analyse theoretically w.r.t. the results we need. However, we believe using GPs is essential, because as a non-parametric function approximator GPs can fit any utility function without assuming a model-space. This, we believe, outweighs the loss of potential for a theoretical regret bound.

In future work, we aim to test GUTS on real-world problems (e.g., multi-objective decision making in epidemic mitigation [21], or traffic problems [42]), and integrate GUTS with an effective query-answering interface for human decision makers. Furthermore, we want to investigate the potential of asking more complex queries than just pairwise comparisons, such as ranking and clustering, which have recently been proven to be effective in preference elicitation on Pareto-fronts from multi-objective decision problems [42].

We also aim to extend our approach to sequential decision making settings, i.e., MOMDPs [23, 27, 36] and even MOPOMDPs [28, 39], and other problems [4, 17, 30] (such as multi-agent problems [20, 22]) where interaction with the user based on a probabilistic modelling approach with respect to user utility could be beneficial.

ACKNOWLEDGEMENTS

The first author was a postdoctoral fellow of the Research Foundation – Flanders (FWO) (grant number 12J0617N), at the time when this research was conducted. This research was in part supported by Innoviris – Brussels Institute for Research and Innovation. Pieter Libin was supported by a PhD grant of the Research Foundation – Flanders (FWO) and the VUB research council (VUB/OZR2714).

REFERENCES

[1] S. Agrawal and N. Goyal. 2012. Analysis of Thompson Sampling for the Multi-armed Bandit Problem.. In *COLT*. 39–1.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. 2002. Finite-time analysis of the multi-armed bandit problem. *Machine learning* 47, 2-3 (2002), 235–256.

[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] N. Benabbou and P. Perny. 2015. Combining Preference Elicitation and Search in Multiobjective State-Space Graphs. In *IJCAI*. 297–303.

[5] C. Boutilier. 2002. A POMDP formulation of preference elicitation problems. In *AAAI*. 239–246.

[6] Jürgen Branke and Kalyanmoy Deb. 2005. Integrating user preferences into evolutionary multi-objective optimization. In *Knowledge incorporation in evolutionary computation*. Springer, 461–477.

[7] Eric Brochu, Tyson Brochu, and Nando de Freitas. 2010. A Bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 103–112.

[8] E. Brochu, V. M. Cora, and N. De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv:1012.2599* (2010).

[9] Urszula Chajewska and Daphne Koller. 2000. Utilities as random variables: Density estimation and structure discovery. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 63–71.

[10] Weiwei Cheng, Johannes Fürnkranz, Eyke Hüllermeier, and Sang-Hyeun Park. 2011. Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 312–327.

[11] W. Chu and Z. Ghahramani. 2005. Extensions of gaussian processes for ranking: semisupervised and active learning. *Learning to Rank* (2005), 29.

[12] W. Chu and Z. Ghahramani. 2005. Preference learning with Gaussian processes. In *ICML*. 137–144.

[13] Kalyanmoy Deb, Ankur Sinha, Pekka J Korhonen, and Jyrki Wallenius. 2010. An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE Transactions on Evolutionary Computation* 14, 5 (2010), 723–739.

[14] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. 2015. Gaussian processes for data-efficient learning in robotics and control. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 37, 2 (2015), 408–423.

[15] Madalina M Drugan. 2017. PAC models in stochastic multi-objective multi-armed bandits. In *GEC*. 409–416.

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

[17] J.P. Dubus, C. Gonzales, and P. Perny. 2009. Multiobjective optimization using GAI models. In *IJCAI*. 1902–1907.

[18] J. P. Forgas. 1995. Mood and judgment: the affect infusion model (AIM). *Psychological bulletin* 117, 1 (1995), 39.

[19] Harold Hotelling. 1931. The generalization of Student's ratio. *Annals of Mathematical Statistics* ii (1931), 360–378.

[20] A. Igarashi and D. M. Roijers. 2017. Multi-Criteria Coalition Formation Games. In *Algorithmic Decision Theory*.

[21] P. Libin, T. Verstraeten, K. Theys, D. M. Roijers, P. Vrancx, and A. Nowé. 2017. Efficient evaluation of influenza mitigation strategies using preventive bandits. In *ALA*. 9 pages.

[22] P. Mannion, J. Duggan, and E. Howley. 2017. A Theoretical and Empirical Analysis of Reward Transformations in Multi-Objective Stochastic Games. In *AAMAS*. 1625–1627.

[23] Sriraam Natarajan and Prasad Tadepalli. 2005. Dynamic preferences in multi-criteria reinforcement learning. In *ICML*.

[24] C. E. Rasmussen. 2006. Gaussian processes for machine learning. (2006).

[25] D. M. Roijers. 2016. *Multi-Objective Decision-Theoretic Planning*. Ph.D. Dissertation. University of Amsterdam.

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

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

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

[29] 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.

[30] E. Rollón. 2008. *Multi-Objective Optimization for Graphical Models*. Ph.D. Dissertation. Universitat Politècnica de Catalunya.

[31] S. Siegel. 1956. Nonparametric statistics for the behavioral sciences. (1956).

[32] E. Sirakaya, J. Petrick, and H.-S. Choi. 2004. The role of mood on tourism product evaluations. *Annals of Tourism Research* 31, 3 (2004), 517–539.

[33] G. Tesauro. 1988. Connectionist Learning of Expert Preferences by Comparison Training.. In *NIPS*, Vol. 1. 99–106.

[34] Lothar Thiele, Kaisa Miettinen, Pekka J Korhonen, and Julian Molina. 2009. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary computation* 17, 3 (2009), 411–436.

[35] W. R. Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25, 3/4 (1933), 285–294.

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

[37] Aaron Wilson, Alan Fern, and Prasad Tadepalli. 2012. A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems*. 1133–1141.

[38] N. Wilson, A. Razak, and R. Marinescu. 2015. Computing Possibly Optimal Solutions for Multi-Objective Constraint Optimisation with Tradeoffs. In *IJCAI*. 815–822.

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

[40] Huasen Wu and Xin Liu. 2016. Double Thompson Sampling for Dueling Bandits. In *NIPS*. 649–657.

[41] S. Q. Yahyaa, M. M. Drugan, and B. Manderick. 2015. Thompson Sampling in the Adaptive Linear Scalarized Multi Objective Multi Armed Bandit. In *ICAART*. 55–65.

[42] L. M. Zintgraf, D. M. Roijers, S. Linders, C. M. Jonker, and Nowé A. 2018. Ordered Preference Elicitation Strategies for Supporting Multi-Objective Decision Making.. In *AAMAS*. To Appear.

[43] M. Zoghi, S. Whiteson, R. Munos, and M. De Rijke. 2014. Relative Upper Confidence Bound For The K-Armed Dueling Bandit Problem. In *ICML*. 10–18.