Interactive Multi-Objective Reinforcement Learning in Multi-Armed Bandits with Gaussian Process Utility Models

Diederik M. Roijers ✉[1,2], Luisa M. Zintgraf[3], Pieter Libin[2], Mathieu Reymond[2], Eugenio Bargiacchi[2], and Ann Nowé[2]

[1] HU University of Applied Sciences Utrecht, The Netherlands
`diederik.yamamoto-roijers@hu.nl`
[2] Vrije Universiteit Brussel, Belgium
`{pieter.libin,mathieu.reymond,eugenio.bargiacchi,ann.nowe}@vub.ac.be`
[3] University of Oxford, United Kingdom
`luisa.zintgraf@cs.ox.ac.uk`

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 that force the utility of the user to be expressed as a weighted sum of the values for each objective, we do not make such stringent a priori assumptions. Specifically, we not only allow non-linear preferences, but also obviate the need to specify the exact model class in the utility function must fall. To achieve this, we propose a new approach called Gaussian-process Utility Thompson Sampling (GUTS). GUTS employs parameterless 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 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.[1]

Keywords: Multiple Objectives · Multi-Armed Bandits · Thompson Sampling · Reinforcement Learning

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 [18]. may want to minimise response time,

[1] A preliminary version of this work was presented at the ALA workshop in 2018 [20].

while also minimising fuel cost and the stress levels of the drivers. For example, an agent learning the best strategy to control epidemics may want to minimise number of infections and minimise the burden on society [13]. 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)* [7, 3], in which the effects of alternative decisions are measured as a vector containing a value for each objective.

Most research on MOMABs [28, 3, 8] focusses on producing *coverage sets* [18], i.e., sets of all possibly optimal alternatives given limited *a priori* information about the possible utility functions of the user, as the solution to a MOMAB. A minimal assumption is that utility functions are 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 [19], 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. [19] recently proposed the *interactive Thompson sampling (ITS)* algorithm for online interactive reinforcement learning in MOMABs, and show 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. First, rather than using a pre-specified model-space for utility functions, we employ parameterless Bayesian learning, i.e., *Gaussian processes* (GPs) [6, 15, 5], to be able to learn arbitrary utility functions. GPs are data-efficient, 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 by filtering provably dominated actions. 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 1. *A scalar* multi-armed bandit (MAB) *[25] 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 distributions* $P_a(r) : \mathbb{R} \rightarrow [0,1]$ *over* scalar *rewards* r *associated with each arm* $a \in \mathcal{A}$.

We refer to the the mean reward of an arm as $\mu_a = \mathbb{E}_{P_a}[r] = \int_{-\infty}^{\infty} r P_a(r) dr$*, 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. *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 et al. [1], is*

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

where $a(t)$ *is the arm pulled at time* 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 3. *A* multi-objective multi-armed bandit (MOMAB) *[3, 28] 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 distributions,* $P_a(\mathbf{r}) : \mathbb{R}^d \rightarrow [0,1]$ *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 [18], 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 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.

For the aforementioned ITS algorithm, Roijers et al. [19] require that $u(\boldsymbol{\mu})$ is of the form $\mathbf{w} \cdot \boldsymbol{\mu}$, where $\mathbf{w}$ is a vector of positive weights per objective that sum to one. However, in many cases, users cannot be assumed to have such simple linear preferences. Instead, users often exhibit some degree of non-linearity. To accommodate any 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.

2.1 Modelling User Preference using Gaussian Processes

Because the user's utility $u(\boldsymbol{\mu})$ can be of any form, we use a general-purpose *Gaussian process* (GP) [15] due to its flexibility for a wide range of tasks. GPs capture uncertainty by assigning to each point a normal distribution – the mean reflecting the expected value, and the variance reflecting the uncertainty at that point. GPs are specified by a mean function m and a kernel k,

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

where the kernel defines how data points influence each other. Since we do not have any prior information on the shape of the utility function of the user, we use the common zero-mean prior, $m(\boldsymbol{\mu}) = 0$. We assume the user's utility function to be smooth (i.e., it does not make sudden large jumps if the reward vector $\boldsymbol{\mu}$ changes only slightly), and therefore use the squared exponential kernel [15], $k(x, x') = \exp\left(-(2\phi)^{-1}||x - x'||^2\right)$, which is appropriate for modelling smooth functions, with ϕ controlling the distance over which neighbouring points influence each other. For large ϕ, the expected values of the data points are going to be highly correlated, while for small ϕ the GP can have larger fluctuations.

Given observation data C about the user's preferences, we update our prior belief $P(u)$ about the utility function $u(\boldsymbol{\mu})$ 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). We opt for binary comparison 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 [9, 21, 22, 24].

To update the GP using this data, we need to define a likelihood function. Chu et al. [6] 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 [5]).

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.

Note that strictly enforcing monotonicity constraints on Gaussian processes is not possible. Some approximate solutions exist [16, 11, 26], however, none of these directly apply to the probit likelihood [6] function we use. In this paper, we instead use the prior knowledge that Pareto-dominated arms cannot be optimal and enforce the monotonicity constraint by using a pruning strategy (introduced in more detail in the next section).

3 Gaussian-process Utility Thompson Sampling

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

GUTS selects the arms to pull by sampling the posterior mean reward distributions as in Thompson Sampling (TS) [25] for single-objective MABs. In contrast to single-objective MABs, 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 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 values of the arms, 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 $u(\boldsymbol{\mu})$ can be found (lines 9–10) and executed (line 18). 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. The main difficulty is in learning to estimate $u(\boldsymbol{\mu})$ accurately and efficiently.

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

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 $t = 1, ..., T$ do
5      $\theta_D, \leftarrow$ draw set of samples from $P(\theta|D)$
6      $\theta'_D \leftarrow$ PPrune($\theta_D$)
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      $a_1(t) \leftarrow \arg\max_a \; u_1(\boldsymbol{\mu}_{\theta'_D,a})$
10     $a_2(t) \leftarrow \arg\max_a \; u_2(\boldsymbol{\mu}_{\theta'_D,a})$
11     $significant \leftarrow$ HotellingT2($D[a_1(t)], D[a_2(t)]) \leq \alpha$
12     if $a_1(t) \neq a_2(t) \;\wedge\; cd \leq 0 \;\wedge\; significant$ then
13         Perform user comparison for $\boldsymbol{\mu}_{\theta'_D,a_1(t)}$ and $\boldsymbol{\mu}_{\theta'_D,a_2(t)}$ and add result (in
           the format, $\boldsymbol{\mu}_x \succ \boldsymbol{\mu}_y$) to C
14         $cd \leftarrow$ countdown
15     else
16         $cd$--
17     end
18     $\mathbf{r}(t) \leftarrow$ play $a_1(t)$ and observe reward
19     append $(\mathbf{r}(t), a_1(t))$ to $D$
20 end
```

comparisons between two value-vectors (Equation 2). 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 one sample set of mean reward vectors from the posteriors of the mean reward vectors given D, θ_D (line 5), and subsequently samples 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. Please note that this sampling is different from ITS [19], where the mean reward vectors are sampled twice as well; we argue that this is not necessary as the utility function is independent from the reward vectors, and we are only interested in possible differences between sampled utilities for a single given set of mean reward vectors.

GUTS uses a GP to estimate $u(\boldsymbol{\mu})$ from the data in C. GPs are highly general, data-efficient, and as non-parametric 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 [18], and leads the notion of *Pareto domination.* 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 [18]. GUTS enforces this condition as follows: Pareto-dominated arms are excluded from the sets of sampled mean reward vectors, by using a pruning operator PPrune[2] resulting in a smaller sets of arms θ'_D (line 6). This ensures that the user will not be queried w.r.t. her preferences for dominating-dominated pairs of vectors from θ'_D, 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'_D (line 7–10). We note that because we estimate u using GPs, GUTS can directly sample values for the values in θ'_D 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 is 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 sampled and utility functions disagree on the arm that maximises the utility for the user. As more information comes in, we expect the two sampled utility functions 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, when we are dealing with multi-variate Gaussian-distributed rewards, we can use Hotellings–T^2 test [10], 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 $\alpha = 0.01$.

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

[2] See e.g. [17] for a reference implementation of PPrune.

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 14 and 16). 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 before it asks the queries.

4 Experiments

To test the performance of GUTS in terms of user regret (Definition 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 we use a 5-arm real-world inspired benchmark with Poisson-distributed rewards to show that GUTS can handle different reward distributions as well as utility functions. In Section 4.3, random MOMAB instances are used to perform more extensive experiments. The code for all experiments can be found at `https://github.com/Svalorzen/morl_guts`.

We compare GUTS to single-objective Thompson sampling provided with the ground truth utility functions of the user. We refer to this baseline as cheat-TS. 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 [19].

We note that cheat-TS does not have the same regret guarantees as Thompson Sampling for e.g., single-objective Gaussian-distributed rewards. This is because the ground-truth utility function is non-linear. So even though the reward vectors may be distributed as multi-variate Gaussian, and therefore the posteriors over mean reward vectors are also multi-variate Gaussians, the posterior distribution over utilities that result from these means is not Gaussian. As the theoretical regret bounds for Thompson sampling rely on the underlying reward distributions of the arms, cheat-TS does not inherit these regret bounds.

In Sections 4.1 and 4.3 the MOMABs have multi-variate Gaussian reward distributions, as common in many real-world environments. This stands in contrast to earlier work, i.e., [19], that uses independent Bernoulli-distributions for each objective. For multi-variate Gaussian distributions, given the rewards $r_1, \dots r_n$, the posterior distribution is also a multi-variate Gaussian [4]:

$$\mu \sim \mathcal{N}(\frac{\mu_0 \kappa_0 + n\bar{r}}{n + \kappa_0}, (\Lambda(n + \kappa_0))^{-1}),$$

where Λ is Wishart-distributed:

$$\Lambda \sim \mathcal{W}i(n_0 + \frac{n}{2}, \Lambda_0 + \frac{1}{2}[\bar{\Sigma} + \frac{\kappa_0}{\kappa_0 + n}(\bar{r} - \mu_0)(\bar{r} - \mu_0)^T])$$

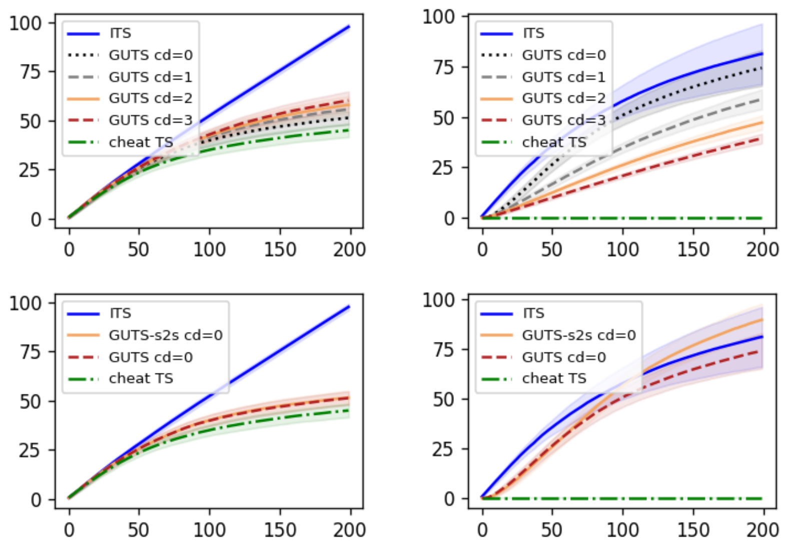

Fig. 2: 5-arm MOMAB results for ITS, ground-truth (cheat) TS, and GUTS with different count-down settings and significance testing enabled (top), and with count-down 0 comparing the GUTS sampling strategy versus that of ITS, i.e., sampling two sets of means (s2s) (bottom): regret (left) and the corresponding number of queries posed (right), as a function of arm-pulls. The error regions indicate $\pm 1\sigma$.

In Section 4.2, we use a real-world inspired MOMAB that has independent Poisson distributions for each objective, to test whether GUTS performs well for differently distributed reward vectors.

For the GPs we use the common zero-mean prior, $m(\boldsymbol{\mu}) = 0$, and squared exponential kernel [15]. When significance testing is used, we use $\alpha = 0.01$ as the significance threshold.

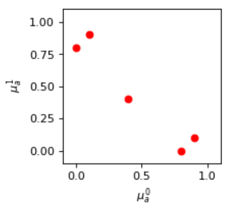

Fig. 1: A 5-arm MOMAB

4.1 A 5-arm MOMAB

In this subsection, we employ a 5-arm MOMAB (depicted in Figure 1), 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 can indeed handle this, we run GUTS on the MOMAB of Figure 1, and compare it to ITS [19], and Thompson Sampling provided with the ground truth utility function in Figure 2 (top). The results confirm that GUTS 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 count-down parameter in the GUTS algorithm to simulate the user answering a smaller portion of the questions. We compare different settings for the count-down parameter (Figure 2, top) we observe that, as expected, no count-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 2, top left). The number of queries to the user can be reduced by increasing count-down (Figure 2, top right). For example, at count-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 further test the difference in sampling strategy between ITS and GUTS. Specifically, we test how GUTS performance is affected if it does not sample the mean rewards once, but twice, as in ITS. The results in Figure 2 bottom-left, indicate that the regret is not affected by this difference. This is as expected, as the shape of the utility function (the user) and the values of the mean rewards (the environment) are independent. However, Figure 2 bottom-right shows that the number of queries is significantly improved by using our sampling strategy. We therefore conclude that our sampling strategy is superior to that of ITS.

We tested the effect of (disabling) significance testing. 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. We did observe a slight increase in the number of queries when significance testing was disabled. 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. Furthermore, we conclude that sampling the means of the rewards for each arm once (the GUTS sampling strategy) is better than sampling them twice (as in ITS).

4.2 Organic MOMAB

To test GUTS on different reward distributions and utility functions, we adapt a real-world inspired benchmark environment [12]. This environment is motivated by a research question that stems from organic agriculture, i.e., to investigate strategies that maximize the prevalence of certain insect species on farmland to predate on a pest insect species [23]. Maximising the occurrence of predatory insect species will reduce pest species and will benefit crop yield, without using artificial pesticides. This prevalence distribution follows a Poisson distribution [23].We consider the case where there are two species of insects that need to

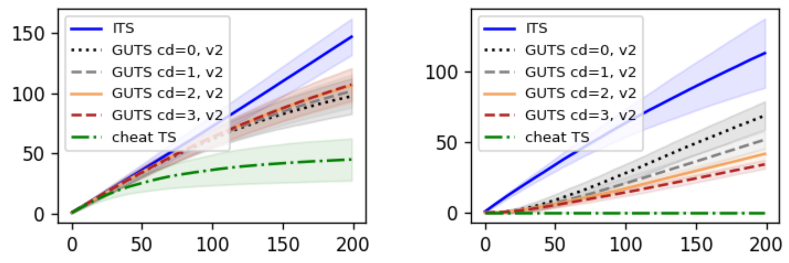

Fig. 3: Regret (left) and number of queries posed (right), as a function of arm-pulls, averaged over 30 trials of the MOMAB of Fig. 1, with count-downs $0, 1, 2, 3$.

be controlled, and therefore we aim to maximize the prevalence of two species that predate on the insect species that is to be controlled. As such, the utility function is defined as:

$$u_{\min}(\boldsymbol{\mu}) = \min(\mu_0, \mu_1)$$

The benchmark environment consists of a 5-arm MOMAB, with the same mean vectors as our first experiment (Section 4.1). Each reward distribution consists of independent Poisson distributions per objective. For a Poisson distribution, the conjugate Jeffreys prior is a gamma distribution [14], $\mathcal{G}\text{amma}(\alpha_0 = 0.5, \beta_0 = 0)$. Given rewards $\mathbf{r} = \langle r_1, ..., r_n \rangle$, this leads to posterior

$$\mu \sim \mathcal{G}\text{amma}(\alpha_0 + \sum_{i=1}^{n} r_i, \beta_0 + n).$$

As $\beta_0 = 0$, this posterior needs to be initialized one time for it be proper.

For this experiment, not only does GUTS need to cope with the nonlinearity of $u_{\min}(\boldsymbol{\mu})$, it also needs to learn on a more complicated problem due to each reward following a Poison distribution (where the variance equals the mean). Despite that, the results on Figure 3 show that GUTS has a sub-linear regret curve, regardless of the cooldown used. Moreover, the number of queries required is lower than ITS, even without any cooldown.

We thus conclude that GUTS can cope with different reward distributions, as we experimented with both Gaussian and Poisson distributions.

4.3 Random MOMABs

Random MOMABs are generated using the number of objectives d, and the number of arms, $|\mathcal{A}|$, as parameters, following the procedure of [19]: $|\mathcal{A}|$ 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

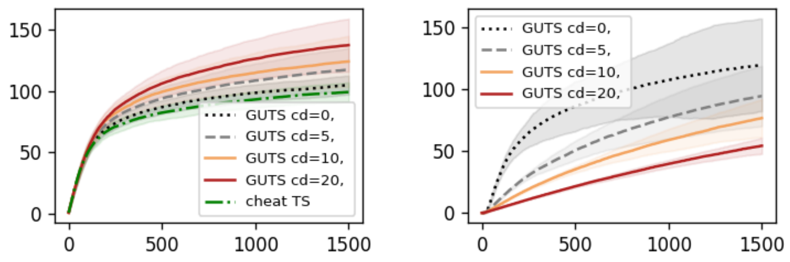

Fig. 4: Regret (left) and number of queries posed to the user (right), as a function of arm-pulls, averaged over 30 random, 2-objective, 20-arm MOMAB instances with random utility functions or order 3, with noise $\varepsilon = 0.1$ on the comparisons of the user (about 2% of the optimal utility), for GUTS with different count-downs.

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.

To test whether GUTS can learn to identify the optimal arm, under different settings of the count-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 sub-linear, and only a small amount worse than Thompson Sampling provided with the ground-truth utility function (cheat-TS); at 1500 arm-pulls, GUTS with count-down 0 has only 5.8% more cumulative regret than cheat-TS. Furthermore, we observe that the number of questions for count-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 goes down 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 count-down 5, GUTS has 18.6% more regret than cheat-TS at 500 arm pulls, for count-down 10, 25.3% and, for count-down 20, 38.8%. However, count-down 5 requires only 78.9% of the queries with respect to count-down 0, and count-downs 10 and 20 only 64.1% resp. 45.4%. I.e., the increase in regret is less steep than the reduction in the number of queries. 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 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). As can be seen, the curves remain sub-linear, but less so for more ob-

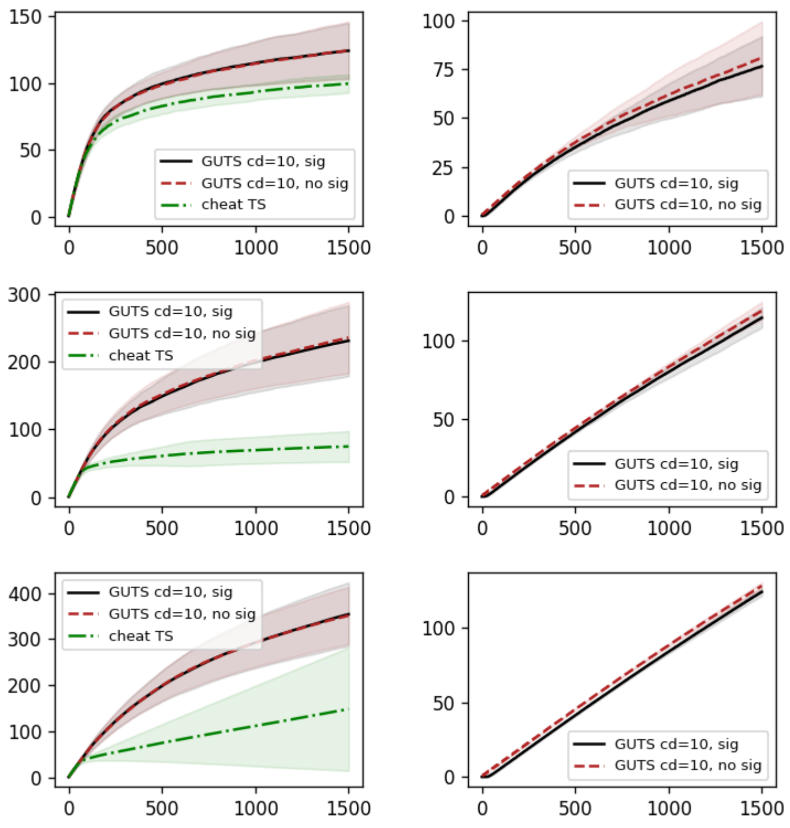

Fig. 5: Regret (left) and number of queries posed (right) as a function of arm-pulls, averaged over 30 random, 20-arm MOMAB instances with 2 (top), 4 (middle) and 6 (bottom) objectives, with random utility functions or order 3, with noise $\varepsilon = 0.1$ on the comparisons of the user ($\sim 2\%$ of the optimal utility), for GUTS with count-down 10, and with or without significance testing (s1/s0).

jectives, and the difference in regret with cheat-TS increases. This is because 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. We observe that for 6 objectives, there is a lot of variance in the performance of cheat-TS. This is due to several runs where the utility function is steeper (has a higher gradient) at the mean of a suboptimal arm than that of the optimal arm, so that smaller sampling differences in the mean reward vector get amplified by the utility function. This is a clear example of the theoretical regret bounds for TS for Gaussian reward distributions not applying to cheat-TS. In other words, the MOMAB with non-linear utility function

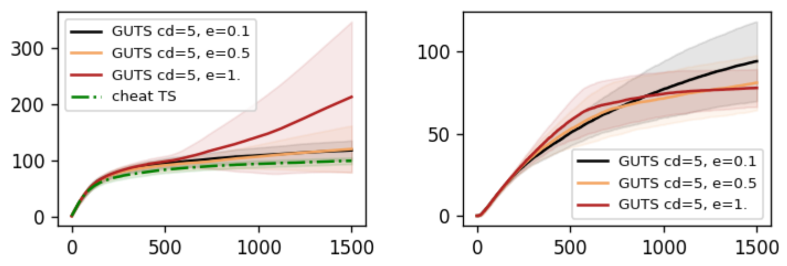

Fig. 6: (left) Regret as a function of arm-pulls, (right) number of queries posed to the user as a function of arm-pulls, averaged over 30 random, 2-objective, 20-arm MOMABs with random utility functions or order 3, with noise $\varepsilon = 0.1$, 0.5 and 1.0 (resp. 2%, 10%, 20% of the optimal utility) on the comparisons, for GUTS with count-down 5.

over multi-variate Gaussian reward distributions can be a considerably harder problem than a single-objective MAB with Gaussian reward distributions.

Finally, 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 because in many of the runs GUTS suddenly samples a non-optimal arm more often after a series of erroneous comparisons. In one of these runs we observe that GUTS does converge back to the optimal arm after a while before the run ends. We thus conclude that GUTS is robust against reasonable levels of noise, but becomes unstable when the noise in the comparisons is too large.

5 Related work

Several papers exist that study *offline* multi-objective reinforcement learning in MOMABs [28, 3, 8]. These papers focus on producing a coverage set in a separate learning phase, without user interaction. After this learning phase, the interaction with the user needs to be done in a separate selection phase. In contrast, we study an online interactive setting, and aim to minimise online regret by estimating the user's utility function during learning.

Relative bandits [30, 27] are a related setting to online interactive learning for MOMABs. Similar to interactive online MORL for MOMABs, relative bandits assume a hidden utility function which can be queried with pairwise comparisons. Contrary our setting, these rewards (i.e., reward vectors) cannot be observed, and the comparisons are made regarding *single* arm pulls, rather than the aggregate information over all pulls of a given arm as in GUTS (and ITS).

6 Discussion

In this paper we 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 MORL 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 w.r.t. Thompson Sampling when provided a priori with the ground-truth utility function. We show the effects of limiting the number of user queries. In this case—even if the number of queries is strongly reduced—the regret remains sub-linear, albeit higher than when queries 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 reasonable levels of noise in the user comparisons.

We note that all conclusions in this paper are based on empirical evaluations rather than theoretical bounds. This is due to the usage of Gaussian Processes (GPs) which makes a theoretical analysis hard. However, we believe it essential to use GPs, because GPs can fit any utility function without assuming a model-space. A limitation of our regret metric is that it assumes the utility is derived from the expected values ($\boldsymbol{\mu}$) at each timestep. We intend to provide new regret metrics in future work. We also aim to test GUTS on real-world problems, and integrate the algorithm with an effective query-answering interface for decision makers. Furthermore, we aim to investigate the potential of asking more complex queries than just pairwise comparisons, such as ranking and clustering [29].

Acknowledgments

This research received funding from the Flemish Government (AI Research Program). Pieter Libin and Eugenio Bargiacchi were supported by a PhD grant of the FWO (Fonds Wetenschappelijk Onderzoek – Vlaanderen).

References

1. Agrawal, S., Goyal, N.: Analysis of Thompson sampling for the multi-armed bandit problem. In: COLT. pp. 39–1 (2012)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning **47**(2-3), 235–256 (2002)
3. Auer, P., Chiang, C.K., Ortner, R., Drugan, M.M.: Pareto front identification from stochastic bandit feedback. In: AISTATS. pp. 939–947 (2016)
4. Bishop, C.M.: Pattern Recognition and Machine Learning (2006)
5. Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv:1012.2599 (2010)

6. Chu, W., Ghahramani, Z.: Preference learning with Gaussian processes. In: ICML. pp. 137–144 (2005)
7. Drugan, M.M., Nowé, A.: Designing multi-objective multi-armed bandits algorithms: A study. In: IJCNN. pp. 1–8. IEEE (2013)
8. Drugan, M.M.: PAC models in stochastic multi-objective multi-armed bandits. In: GEC. pp. 409–416 (2017)
9. Forgas, J.P.: Mood and judgment: the affect infusion model (aim). Psychological bulletin **117**(1), 39 (1995)
10. Hotelling, H.: The generalization of Student's ratio. Annals of Mathematical Statistics **ii**, 360–378 (1931)
11. Lampinen, J.: Gaussian processes with monotonicity constraint for big data (2014)
12. Libin, P., Verstraeten, T., Roijers, D.M., Wang, W., Theys, K., Nowé, A.: Bayesian anytime m-top exploration. In: ICTAI. pp. 1422–1428 (2019)
13. Libin, P.J., Verstraeten, T., Roijers, D.M., Grujic, J., Theys, K., Lemey, P., Nowé, A.: Bayesian best-arm identification for selecting influenza mitigation strategies. In: ECML-PKDD. pp. 456–471 (2018)
14. Lunn, D., Jackson, C., Best, N., Thomas, A., Spiegelhalter, D.: The BUGS book: A practical introduction to Bayesian analysis. CRC press (2012)
15. Rasmussen, C.E.: Gaussian processes for machine learning (2006)
16. Riihimäki, J., Vehtari, A.: Gaussian processes with monotonicity information. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 645–652 (2010)
17. Roijers, D.M.: Multi-Objective Decision-Theoretic Planning. Ph.D. thesis, University of Amsterdam (2016)
18. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. JAIR **48**, 67–113 (2013)
19. Roijers, D.M., Zintgraf, L.M., Nowé, A.: Interactive Thompson sampling for multi-objective multi-armed bandits. In: Algorithmic Decision Theory. pp. 18–34 (2017)
20. Roijers, D.M., Zintgraf, L.M., Libin, P., Nowé, A.: Interactive multi-objective reinforcement learning in multi-armed bandits for any utility function. In: ALA workshop at FAIM (2018)
21. Siegel, S.: Nonparametric statistics for the behavioral sciences. (1956)
22. Sirakaya, E., Petrick, J., Choi, H.S.: The role of mood on tourism product evaluations. Annals of Tourism Research **31**(3), 517–539 (2004)
23. Soulsby, R.L., Thomas, J.A.: Insect population curves: modelling and application to butterfly transect data. Methods in Ecology and Evolution **3**(5), 832–841 (2012)
24. Tesauro, G.: Connectionist learning of expert preferences by comparison training. In: NeurIPS. vol. 1, pp. 99–106 (1988)
25. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25**(3/4), 285–294 (1933)
26. Ustyuzhaninov, I., Kazlauskaite, I., Ek, C.H., Campbell, N.D.: Monotonic Gaussian process flow. arXiv preprint arXiv:1905.12930 (2019)
27. Wu, H., Liu, X.: Double thompson sampling for dueling bandits. In: NeurIPS. pp. 649–657 (2016)
28. Yahyaa, S.Q., Drugan, M.M., Manderick, B.: Thompson sampling in the adaptive linear scalarized multi objective multi armed bandit. In: ICAART. pp. 55–65 (2015)
29. Zintgraf, L.M., Roijers, D.M., Linders, S., Jonker, C.M., Nowé, A.: Ordered preference elicitation strategies for supporting multi-objective decision making. In: AAMAS. pp. 1477–1485 (2018)
30. Zoghi, M., Whiteson, S., Munos, R., De Rijke, M.: Relative upper confidence bound for the k-armed dueling bandit problem. In: ICML. pp. 10–18 (2014)