Interactive Thompson Sampling for Multi-Objective Multi-Armed Bandits

Diederik M. Roijers (✉), Luisa M. Zintgraf, and Ann Nowé

Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Brussels, Belgium
{droijers, luisa.zintgraf, ann.nowe}@ai.vub.ac.be

Abstract. In multi-objective reinforcement learning (MORL), much attention is paid to generating optimal *solution sets* for unknown utility functions of users, based on the stochastic reward vectors only. In *online* MORL on the other hand, the agent will often be able to elicit preferences from the user, enabling it to learn about the utility function of its user directly. In this paper, we study online MORL with user interaction employing the multi-objective multi-armed bandit (MOMAB) setting — perhaps the most fundamental MORL setting. We use Bayesian learning algorithms to learn about the environment and the user simultaneously. Specifically, we propose two algorithms: *Utility-MAP UCB (umap-UCB)* and *Interactive Thompson Sampling (ITS)*, and show empirically that the performance of these algorithms in terms of regret closely approximates the regret of UCB and regular Thompson sampling provided with the ground truth utility function of the user from the start, and that ITS outperforms umap-UCB.

1 Introduction

Many real-world decision problems require learning about the outcomes of different alternatives, either by interacting with the real world, or simulations thereof. When the outcomes 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 are further complicated by the presence of multiple (possibly conflicting) objectives [13]. For example, an agent learning the best strategy to deploy ambulances from a set of alternatives, may want to minimise average response time, while also minimising fuel cost and the stress for the drivers. For such problems, MABs can be extended to *multi-objective multi-armed bandits (MOMABs)* [3, 9].

Research on MOMABs has hitherto focussed on settings in which no preference information w.r.t. the available trade-offs between the values in different objectives of the available alternatives is provided by the user during learning [3, 18, 22]. However, in many situations, this is a limiting assumption. E.g., in the example of ambulance deployment strategies, the ambulances will have to be deployed (i.e., a strategy will have to be executed) while still learning about the expected value of the different strategies in the different objectives. By providing preferences of the responsible human decision makers, i.e., the management of the ambulance service, w.r.t. the attainable trade-offs between objectives, the agent can focus its learning on those strategies that the user finds most appealing. Other examples of multi-objective learning problems in which interaction with the user can be beneficial are: the exploration of preventive strategies for epidemics using computationally expensive simulations under objectives like minimising morbidity, the number of people infected, and the costs of the preventive strategy; and future household robots that will need to learn the preferences of their users with respect to the outcomes (performance, speed, energy usage) of alternative ways to perform a household task.

In this paper, we focus on such online learning problems, in which the agent can query the user via pairwise comparisons (following, e.g., [6, 15, 23]). Specifically, we focus on the prevalent case that a user's utility function is linear, in the context of MOMABs.

We propose two new Bayesian learning algorithms for MOMABs that learn about the environment and the user's utility function simultaneously. Specifically, we build upon two popular classes of algorithms for single-objective MABs, UCB1 [2] and Thompson sampling [16], to propose *utility-MAP UCB (umap-UCB)* and *Interactive Thompson Sampling (ITS)*. Both algorithms (umap-UCB and ITS) pose pairwise comparison queries to the user. As a UCB-algorithm, umap-UCB uses explicit exploration bonuses. To decide when to query the user, it computes the MAP of the utility function to determine the arm corresponding to the best mean estimate, and the arm corresponding to the best mean estimate plus exploration bonus. When these two best arms are different, umap-UCB queries the user by asking her to make a pairwise comparison between the estimated mean reward vectors of the two arms. ITS also elicits preferences via pairwise comparisons; it draws two sets of samples from the posteriors of the mean reward vectors of the arms, and the posterior of the utility function. When these two sets of samples have different best arms, ITS queries the user.

We test umap-UCB and ITS empirically, and find that the performance of these algorithms in terms of regret closely approximates the regret of UCB and regular Thompson sampling equipped from the beginning with the ground truth utility function of the user. When comparing umap-UCB and ITS, we find that ITS outperforms umap-UCB both in terms of minimising user regret, and in terms of minimising the number of comparisons that the user is asked to make.

2 Background

Before introducing our setting, we first provide the necessary background on scalar multi-armed bandits and multi-objective decision making in general.

2.1 Multi-armed Bandits

Definition 1. *A scalar* multi-armed bandit (MAB) *[1, 2, 16] 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, $P_a(r) : \mathbb{R} \to [0,1]$ over* scalar *rewards, r, associated with each arm $a \in \mathcal{A}$.*

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

Algorithm 1: UCB

```
$\bar{x}_a \leftarrow$ initialise with single pull, $r_a$ for each $a$
$n_a \leftarrow 1$ for each $a$
for $t = |\mathcal{A}|, ..., T$ do
    $a(t) \leftarrow \arg\max_a (\bar{x}_a + c(\bar{x}_a, n_a, t))$
    $r(t) \leftarrow$ play $a(t)$ and observe reward
    $\bar{x}_{a(t)} \leftarrow \frac{n_{a(t)}\bar{x}_{a(t)} + r(t)}{n_{a(t)}+1}$
    $n_{a(t)}$++
```

Algorithm 2: Thompson Sampling

```
Input: A prior for the reward distributions
$D \leftarrow \emptyset$;            // observed data
for $t = 1, ..., T$ do
    $\theta^t \leftarrow$ draw sample from $P(\theta^t|D)$
    $a(t) \leftarrow \arg\max_a \mathbb{E}_{P(r|a,\theta^t)}[r]$
    $r(t) \leftarrow$ play $a(t)$ and observe reward
    append $(r(t), a(t))$ to $D$
```

The goal of an agent interacting with a MAB is to maximise the expected cumulative reward, $E[\sum_{t=1}^{T} \mu_{a(t)}]$, where T is the time horizon, and $a(t)$ is the arm pulled at time t. However, 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 the MAB literature, this reward maximisation is typically defined via the minimisation of the equivalent measure of expected total regret, i.e., the amount of reward lost due to not playing the optimal arm in each step.

Definition 2. *The expected total* regret *of pulling a sequence of arms for each timestep between $t = 1$ and a time horizon T (following the definition of [1]), is*

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

where $n_a(T)$ is the number of times arm a is pulled until timestep T.

In the literature, a popular choice [1, 2] for $\mathcal{P}$ is Bernoulli distributions, i.e., distributions with only two possible outcomes: 1 (dubbed 'success'), according to a probability $p_a \in [0,1]$ or 0 (dubbed 'failure') with a probability $1 - p_a$. The expected reward for a Bernoulli distribution is $\mu_a = p_a$.

The two most popular classes of algorithms for Bernoulli-distributed MABs are UCB and Thompson Sampling. UCB [2, 7], provided in a general form in Algorithm 1, keeps estimates of the means of arms $\bar{x}_a$, and uses upper confidence bounds $c(\bar{x}_a, n_a, t)$ for exploration. I.e., at each round the arm is pulled with the highest value for the mean plus exploration bonus, $\bar{x}_a + c(\bar{x}_a, n_a, t)$. This exploration bonus is defined so that it goes down with the number of pulls of an arm, n_a, and up slowly with the total number of pulls of all arms, t. There are many variants of UCB that differ in its definition of $c(\bar{x}_a, n_a, t)$, ranging from the original UCB1 bound [2],

$$c_1(\bar{x}_a, n_a, t) = \sqrt{\frac{2 \ln t}{n_a}}, \tag{1}$$

to the upper confidence bound derived from Chernoff's bound [7]:

$$c_{ch}(\bar{x}_a, n_a, t) = \sqrt{\frac{2\bar{x}_a \ln \sqrt{t}}{n_a}} + \frac{2 \ln \sqrt{t}}{n_a}. \tag{2}$$

Thompson sampling (Algorithm 2) on the other hand, maintains posterior distributions for the parameters of the reward distributions for each arm $a \in \mathcal{A}$, and pulls arms based on samples from this posterior. Bernoulli distributions have only one parameter, μ_a. The typical prior for these μ_a are beta distributions, with a single count for the number of successes and failures: $\beta(1{,}1)$ for each arm. The posterior can be calculated by simply counting the number of successes, i.e., the number of times the reward was 1, $s_a(t)$, and the number of failures $f_a(t) = n_a(t) - s_a(t)$, leading to a posterior over the means of each arm, $\beta(s_a(t)+1, f_a(t)+1)$. We denote a sample from the joint posterior of all arms as:

$$\theta^t = \langle \theta^t_1 ... \theta^t_{|\mathcal{A}|} \rangle \sim P(\theta^t | D) = \prod_{a \in \mathcal{A}} \beta(s_a(t)+1, f_a(t)+1).$$

At each iteration, Thompson sampling draws such a sample and pulls the arm corresponding to $\arg\max_a \mathbb{E}_{P(r|a,\theta^t)}[r]$. Because these θ^t_a are samples from the posteriors of the *mean* for each arm, a, the arm corresponding to $\arg\max_a \theta^t_a$ represents the maximum expected reward (for that set of samples).

2.2 Multi-Objective Decision Making

In single-objective MABs, an agent must find the alternative a that maximises the expected reward. In multi-objective problems however, 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 (MAB) *[3, 18, 22] 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,* $P_a(\mathbf{r}) : \mathbb{R}^d \to [0{,}1]$ *over* vector-valued *rewards* $\mathbf{r}$ *of length* d*, associated with each arm* $a \in \mathcal{A}$*.*

As a result, rather than having a single optimal alternative, there can be multiple arms whose value vectors are optimal for different preferences that users may have with respect to the objectives. Such preferences can be expressed using a utility function $u(\boldsymbol{\mu}, \mathbf{w})$ that is parameterised by a parameter vector $\mathbf{w}$ and returns the *scalarized* value of $\boldsymbol{\mu}$. Following the single-objective literature, we make use of Bernoulli distributions (as a worst-case scenario of distributions with high variance). Specifically, we assume that the reward for an arm, a, is a vector of d independent Bernoulli distributions. This probability distribution can thus be compactly described with a vector of means $\boldsymbol{\mu}_a$; for each objective, samples can be drawn independently using the mean for that objective.

When the parameter vector $\mathbf{w}$ is known beforehand, it is possible to *a priori* scalarise the decision problem and apply standard single-objective algorithms like UCB or Thompson sampling. However, often we do not know $\mathbf{w}$ at the start of learning.

Much *multi-objective reinforcement learning (MORL)* research assumes that $u(\boldsymbol{\mu}, \mathbf{w})$ is unknown throughout the learning phase, and there will only be access to the user in a separate *selection phase*. We refer to this scenario as the *offline MORL decision support scenario*, depicted in Figure 1, in which an agent provides decision support to the user

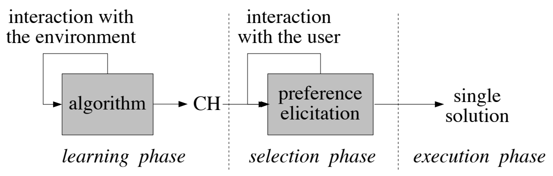

Fig. 1. The offline MORL decision support scenario.

by presenting her with a set of alternatives at the beginning of the selection phase. In the learning phase, we thus need an algorithm that computes a set of policies containing at least one arm with the maximal scalarised value for *each possible* $\mathbf{w}$. Which alternatives, i.e., arms, should be included in this set depends on what we know about the utility function u. A highly prevalent case is that u is linear.

Definition 4. *A* linear utility function *is a weighted sum of the values in each objective,* $\boldsymbol{\mu}$*, of an alternative, i.e.,*

$$u(\boldsymbol{\mu}, \mathbf{w}) = \mathbf{w} \cdot \boldsymbol{\mu}, \tag{3}$$

where $\mathbf{w}$ *is a vector of non-negative weights that sum to 1, and express the preferences of the user w.r.t. each objective. Please note that we assume that all objectives are desirable, and thus contribute positively to the utility.*

In the offline MORL decision support scenario (Figure 1) where the utility is an unknown linear utility function, a sufficient solution is the *convex hull (CH)*, the set of all undominated policies under a linear scalarisation:

$$CH(\mathcal{A}) = \{a \in \mathcal{A} \mid \exists \mathbf{w} \forall (a' \in \mathcal{A}) : \ \mathbf{w} \cdot \boldsymbol{\mu}_a \geq \mathbf{w} \cdot \boldsymbol{\mu}_{a'}\}.$$

When computation time is abundant, and there is enough time until the final decision needs to be made, it can be feasible to do MORL in an offline manner. This has the advantage that the interaction with the user can be done separately after learning, and therefore typically more efficiently (i.e., with fewer interactions with the user). However, as we have indicated in the introduction, there are also situations in which decisions have to be made on a timescale that is relatively short compared to the computation time needed to evaluate arms. Furthermore, it can also be highly important to use the available computation time as efficiently as possible. This is for example the case when evaluating and selecting different alternative preventive strategies against an emerging epidemic using computationally expensive simulations with computational epidemiological models [11], in the presence of multiple objectives like infection ratio, morbidity and economic damage. In other words, offline learning is not possible when there is no time to perform a separate learning phase before acting. In such cases, we need a different approach.

3 Online Interactive Learning with MOMABs

In this paper we focus on the case that we can have interaction with the user during learning, and that the policies executed during learning are important, i.e., accumulate

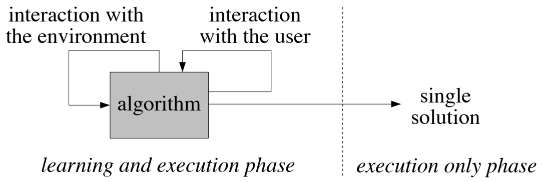
interaction with the environment
interaction with the user
algorithm
single solution
learning and execution phase
execution only phase

Fig. 2. The online interactive MORL decision support scenario.

regret. This leads to the *online interactive MORL decision support scenario*, which is schematically depicted in Figure 2. In this scenario, learning and execution of the policy happens simultaneously in the learning phase. Furthermore, we have to interact with the environment as well as the user during learning, in order to maximise the rewards (or minimise the regret). Finally, it can happen that after some amount of time (and/or number of interactions), the learning will stop, and we will move to an *execution only phase*. This happens, e.g., when the computational capacity of simulations is needed for different learning problems, and/or the user becomes unavailable for further input.

In the online interactive MORL decision support scenario, we aim to minimise *user regret*, i.e., the amount of utility that is lost due to playing suboptimal arms. We define the value of the optimal arm as

$$\boldsymbol{\mu}^* = \arg\max_a \mathbf{w}^* \cdot \boldsymbol{\mu}_a,$$

where $\mathbf{w}^*$ are the ground truth weights of a linear utility function (Definition 4). Similar to single-objective MABs, we define the expected (in our case vector-valued) regret of pulling an arm, a, once as $\boldsymbol{\Delta}_a = \boldsymbol{\mu}^* - \boldsymbol{\mu}_a$.

Definition 5. *The expected total* user regret *of pulling a sequence of arms for each timestep between $t = 1$ and a time horizon T in a MOMAB is*

$$\mathbb{E}\left[\mathbf{w}^* \cdot (\sum_{t=1}^{T} \boldsymbol{\mu}^* - \boldsymbol{\mu}_{a(t)})\right] = \sum_a (\mathbf{w}^* \cdot \boldsymbol{\Delta}_a)\, \mathbb{E}[n_a(T)],$$

where $n_a(T)$ is the number of times arm a is pulled until timestep T.

4 Algorithms

To minimise user regret (Definition 5) in the online interactive MORL setting (Figure 2), we must interact both with the environment and with the user. Similar to single-objective MABs, we need to learn about the reward, $\mathbf{r}$, but in addition, we must also learn about u and $\mathbf{w}$, as $u(\mathbb{E}[\sum_t \mathbf{r}(t)], \mathbf{w})$ is what we ultimately aim to optimise.

Following Zoghi et al. [23] — who study *relative bandits*; which is an adjacent but different model, in which the reward (vectors) cannot be observed — we assume that we can interact with the user once before (or after) pulling an arm, in the form of a pairwise comparison [6, 15, 23]. Contrary to [23] however, we present the user with (estimations of) expected reward vectors, rather than (data resulting from) single arm pulls. We thus ask users to compare two vectors, $\mathbf{x}$ and $\mathbf{y}$, and observe whether the user prefers $\mathbf{x}$ to $\mathbf{y}$, denoted $\mathbf{x} \succ \mathbf{y}$. At timestep t, we thus have access to a data set, C, of j of such preference pairs, where $j \leq t$ is the number of comparisons performed until t:

$$C = \{(\mathbf{x}_i \succ \mathbf{y}_i)\}_{i=1}^{j} . \tag{4}$$

There is no predetermined budget on the number of comparisons a user can make, other than the finite-time horizon, T, which also holds for the number of arm pulls.

Because we assume that $u(\boldsymbol{\mu},\mathbf{w})$ is a linear utility function (Definition 4), and data in the form of Equation 4, we can estimate $\mathbf{w}^*$ using *logistic regression* [5]. Specifically, as we propose Bayesian methods, we employ *Bayesian logistic regression* [5], enabling us to obtain both a *maximum a posteriori* estimate of the true weights $\mathbf{w}^*$, $\bar{\mathbf{w}}$, as well as a posterior distribution over the true weights.

Along with minimising user regret, we aim to not query the user excessively, as querying the user costs time and can be experienced as bothersome. In other words, we aim to propose algorithms in which both the expected *user regret* per timestep, as well as the expected *number of queries* posed to the user per timestep, goes down steeply as time progresses. In order to achieve this, we build on two state-of-the-art classes of algorithms: UCB (Algorithm 1), and Thompson Sampling (Algorithm 2) for single-objective bandits.

4.1 Utility MAP–UCB

Our first algorithm, that we call *utility-MAP UCB (umap-UCB)* (Algorithm 3), is built upon UCB. In UCB for single-objective MABs (Algorithm 1), actions are chosen based on the estimates of the means for each arm $\bar{x}_a$ plus an exploration bonus, which together form an upper confidence bound on the true means of the arms. When applying the same schema to MOMABs we face the following challenges: (1) that the user has a linear utility function (Equation 3) with an *unknown* weight parameter $\mathbf{w}^*$; (2) that we must decide how to select which action to play, given the current MAP estimate of the weight vector, $\bar{\mathbf{w}}$; and (3) that we want to estimate $\mathbf{w}^*$, while the *number of queries* posed to the user per timestep goes down steeply as time progresses (without sacrificing too much *user regret*).

First, let us focus on how to estimate $\mathbf{w}^*$. Because we assume a linear utility function (Equation 3) and pairwise comparisons as data (Equation 4), we use Bayesian logistic regression to estimate the weights.[1] We thus define a prior on the weights. Specifically, we use a multi-variate Gaussian prior $\mathcal{N}(\mathbf{w}|\boldsymbol{\mu}^0,\boldsymbol{\Sigma}^0)$. We use η^0 as a shorthand for $\langle\boldsymbol{\mu}^0,\boldsymbol{\Sigma}^0\rangle$. Given a prior distribution on $\mathbf{w}$, and user comparisons C as defined in

[1] We note that logistic regression based on maximum likelihood can lead to problems in earlier iterations of umap-UCB when there is little data available. We observed this empirically. Specifically, in earlier iterations umap-UCB with ML logistic regression instead of Bayesian logistic regression makes an estimate, $\bar{\mathbf{w}}$, with a sheer-infinite weight on one objective, such that no comparison will be asked from the user again. This can be prevented with a reasonable choice of prior in Bayesian logistic regression.

Algorithm 3: Utility–MAP UCB

```
Input: A parameter prior on the distribution of $\mathbf{w}$.
$C \leftarrow \emptyset$;  // previous comparisons
$\bar{\mathbf{x}}_a \leftarrow$ initialise with single pull, $\mathbf{r}_a$, for each $a$
$n_a \leftarrow 1$ for each $a$
for $t = |\mathcal{A}|, ..., T$ do
    $\bar{\mathbf{w}} \leftarrow \mathcal{P}_{simplex}(MAP(\mathbf{w}|C))$
    $\bar{a}^* \leftarrow \arg\max_a \bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a$
    $a(t) \leftarrow \arg\max_a (\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a + c(\bar{\mathbf{w}}, \bar{\mathbf{x}}_a, n_a, t))$
    $\mathbf{r}(t) \leftarrow$ play $a(t)$ and observe reward
    $\bar{\mathbf{x}}_{a(t)} \leftarrow \frac{n_{a(t)}\bar{\mathbf{x}}_{a(t)} + \mathbf{r}(t)}{n_{a(t)}+1}$
    $n_{a(t)}{+}{+}$
    if $\bar{a}^* \neq a(t)$ then
        perform user comparison for $\bar{\mathbf{x}}_{\bar{a}^*}$ and $\bar{\mathbf{x}}_{a(t)}$ and add result ( $(\bar{\mathbf{x}}_{\bar{a}^*} \succ \bar{\mathbf{x}}_{a(t)})$ or $(\bar{\mathbf{x}}_{a(t)} \succ \bar{\mathbf{x}}_{\bar{a}^*})$) to C
```

Algorithm 4: Interactive Thompson Sampling

```
Input: Parameter priors on reward distributions, and on $\mathbf{w}$ distribution.
$C \leftarrow \emptyset$;  // previous comparisons
$D \leftarrow \emptyset$;  // observed reward data
for $t = 1, ..., T$ do
    $\eta_1^t, \eta_2^t \leftarrow$ draw 2 samples from $P(\eta^t|C)$
    $\theta_1^t, \theta_2^t \leftarrow$ draw 2 samples from $P(\theta^t|D)$
    $a_1(t) \leftarrow \arg\max_a \mathbb{E}_{P(\mathbf{r},\mathbf{w}|a,\theta_1^t,\eta_1^t)}[\mathbf{w} \cdot \mathbf{r}]$
    $a_2(t) \leftarrow \arg\max_a \mathbb{E}_{P(\mathbf{r},\mathbf{w}|a,\theta_2^t,\eta_2^t)}[\mathbf{w} \cdot \mathbf{r}]$
    $\mathbf{r}(t) \leftarrow$ play $a_1(t)$ and observe reward
    append $(\mathbf{r}(t), a_1(t))$ to $D$
    if $a_1(t) \neq a_2(t)$ then
        $\tilde{\boldsymbol{\mu}}_{1,a_1(t)} \leftarrow \mathbb{E}_{P(\mathbf{r}|a_1(t),\theta_1^t)}[\mathbf{r}]$
        $\tilde{\boldsymbol{\mu}}_{2,a_2(t)} \leftarrow \mathbb{E}_{P(\mathbf{r}|a_2(t),\theta_2^t)}[\mathbf{r}]$
        perform user comparison for $\tilde{\boldsymbol{\mu}}_{1,a_1(t)}$ and $\tilde{\boldsymbol{\mu}}_{2,a_2(t)}$ and add result (($\tilde{\boldsymbol{\mu}}_{1,a_1(t)} \succ \tilde{\boldsymbol{\mu}}_{2,a_2(t)}$) or ($\tilde{\boldsymbol{\mu}}_{2,a_2(t)} \succ \tilde{\boldsymbol{\mu}}_{1,a_1(t)}$)) to C
```

Equation 4, we obtain a *maximum a posteriori (MAP)* estimate at the beginning of each iteration using Bayesian logistic regression. However, this estimate might not adhere to the simplex constraints.Therefore, we back-project the MAP estimate of the weights onto the simplex for d objectives, leading to the estimate $\bar{\mathbf{w}} = \mathcal{P}_{simplex}(MAP(\mathbf{w}|C))$. The fact that $\bar{\mathbf{w}}$ adheres to the simplex constraints is important for UCB-algorithms, as the exploration bonuses, and the regret-bounds derived from it, use the assumption that the reward samples are within the interval $[0,1]$ (after applying the utility function).

After umap-UCB obtains a $\bar{\mathbf{w}}$ at the beginning of an iteration, it can proceed to pick actions. To select which arm to play, $a(t)$, umap-UCB follows the standard UCB schema, using the expected scalarised reward, $\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a$ plus an exploration bonus c:

$$a(t) \leftarrow \arg\max_a \left(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a + c(\bar{\mathbf{w}}, \bar{\mathbf{x}}_a, n_a, t)\right).$$

We note that we can only use $\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a$ as an estimate for the scalarised means because we assume that the estimates of $\mathbf{w}^*$ are independent of the estimates of $\boldsymbol{\mu}_a$.[2] In our setting this assumption holds if the user can objectively compare two vectors, without being influenced by which arms have been pulled in previous iterations, and which comparisons have taken place in previous iterations. We believe this to be a realistic assumption for pairwise comparisons.

The exploration bonus, $c(\bar{\mathbf{w}}, \bar{\mathbf{x}}_a, n_a, t)$, can be implemented in many ways. We note that when $\mathbf{w}^*$ places all the weight on a single objective, the MOMAB becomes a scalar MAB, which is Bernoulli-distributed (the only difference being that the agent does

[2] $\mathbb{E}[\mathbf{x} \cdot \mathbf{y}] = \mathbb{E}[\mathbf{x}] \cdot \mathbb{E}[\mathbf{y}]$, iff $\mathbf{x}$ and $\mathbf{y}$ are independent.

not know $\mathbf{w}^*$). We therefore use exploration bonuses that reduce to those for single-objective MABs in this case. Specifically, we use either $c(\bar{\mathbf{w}}, \bar{\mathbf{x}}_a, n_a, t) = c_1(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a, n_a, t)$ (i.e., UCB1, Equation 1) or $c(\bar{\mathbf{w}}, \bar{\mathbf{x}}_a, n_a, t) = c_{ch}(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a, n_a, t)$ (Equation 2). We note that for weights that are more evenly distributed, tighter bounds may hold, e.g., $\mathbf{w}^*$ with equal weights for each objective, and $d \to \infty$, leads to normally distributed scalarised rewards, due to the central limit theorem. However, as the estimation of $\bar{\mathbf{w}}$ is not exact, obtaining a tighter bound is far from trivial, and we leave this open for future work.

Having defined how umap-UCB picks arms to perform, we now define when and which comparison queries umap-UCB poses to the user. We note that we want to decrease the number of queries steeply over time, but never to stop querying (as we may at any point in time, have an estimate $\bar{\mathbf{w}}$ that favours a suboptimal arm). For this reason, we tie in the querying of the user with the exploration mechanism of UCB, which has a similar purpose, i.e., it aims to pull arms so often as to keep on exploring, yet bound the regret of pulling those arms, by rapidly decreasing the number of suboptimal arm pulls over time. To achieve this, umap-UCB explicitly calculates the arm, $\bar{a}^*$, with best estimated *scalarised* mean without exploration bonus, $\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a$, at the beginning of each iteration. When $\bar{a}^*$ is different from $a(t)$, we query the user for a comparison between the estimates of the means (without exploration bonuses), $\bar{\mathbf{x}}_{\bar{a}^*}$, and $\bar{\mathbf{x}}_{a(t)}$.

Umap-UCB has two important characteristics. Firstly, the algorithm will never stop querying the user. Therefore, if the current estimate of the weights $\bar{\mathbf{w}}$ favours the wrong arm, as time — and with time the accuracy of the estimated means for each a — increases, more and more comparison data will be generated that will eventually lead the MAP estimate of the weights to favour the arm which is best for the ground truth weights, $\mathbf{w}^*$.[3] Furthermore, the expected number of queries is equal to the number of suboptimal arm-pulls, which decreases rapidly over time, and is bounded (in finite time) via the exploration bonuses inherited from the single-objective UCB algorithms that umap-UCB builds upon.

4.2 Interactive Thompson Sampling

For single-objective MABs, UCB algorithms (Algorithm 1) are in practice often outperformed [7] by Thomspon sampling [16] (Algorithm 2). Thompson sampling works according to the following schema: first, it starts with a prior distribution on the parameters of the reward distribution of each arm. Then, it gathers data by drawing samples from the posterior distributions of these parameters, and pulling the arm with the maximal expected rewards according to the sampled parameters for each arm.

We build upon Thompson sampling for MABs by not only sampling from the posteriors of the parameters of the reward distributions, but also those of the user preferences $\mathbf{w}$. We call this algorithm *Interactive Thompson Sampling (ITS)* (Algorithm 4).

ITS starts each iteration by drawing two independent samples both from the posteriors for the parameters of the reward distributions of each arm (θ_1^t and θ_2^t), and from the posterior for the parameters of the utility function (η_1^t and η_2^t). Without loss of generality, we use the first sample to determine the action to play, $a_1(t)$. We note that for our

[3] Please note that for obtaining 0 regret, it is not necessary that the MAP estimate $\bar{\mathbf{w}}$ is identical to the ground truth $\mathbf{w}^*$, as long as it leads to selecting the same arm.

assumptions (independent weights and rewards, Gaussian weights, Bernoulli reward vectors), we can compute which action to select as:

$$\arg\max_a \mathbb{E}_{P(\mathbf{r},\mathbf{w}|a,\theta_1^t,\eta_1^t)}[\mathbf{w}\cdot\mathbf{r}] = \arg\max_a \left(\tilde{\mathbf{w}}_1 \cdot \tilde{\boldsymbol{\mu}}_{1,a}\right),$$

where $\tilde{\mathbf{w}}_1$ (corresponding to η_1^t) is the sampled weights vector, and $\tilde{\boldsymbol{\mu}}_{1,a}$ (corresponding to θ_1^t) is the sampled means vector for the rewards of arm a. Again, we assume that $\tilde{\mathbf{w}}_1$ and $\tilde{\boldsymbol{\mu}}_{1,a}$ can be sampled from their resp. posterior distributions independently.

The second sample is used solely to determine whether and how to interact with the user. ITS determines which actions both samples would select, $a_1(t)$ and $a_2(t)$. If $a_2(t)$ differs from $a_1(t)$, ITS queries the user for a comparison between the expected reward according to the first sample $\tilde{\boldsymbol{\mu}}_{1,a_1(t)}$ to that of the second sample $\tilde{\boldsymbol{\mu}}_{2,a_2(t)}$.

As the posteriors of both distributions (rewards and weights) become increasingly certain, the number of suboptimal arm-pulls made by ITS goes down. Furthermore, the number of times that two sets of samples from these distributions disagree on which action to take — and thus the number of queries to the user — goes down as well.

5 Experiments

In order to test the performance of umap-UCB and Interactive Thompson Sampling, in terms of user regret (Definition 5) and the number of queries posed to the user, we compare our algorithms on two types of problems: `double circle` MOMABs in Section 5.2 and `random` MOMABs in Section 5.3. We use two variants of umap-UCB: umap-UCB1, using $c(\bar{\mathbf{w}}, \bar{\mathbf{x}}_a, n_a, t) = c_1(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a, n_a, t)$ (Equation 1), and umap-UCB-ch using $c(\bar{\mathbf{w}}, \bar{\mathbf{x}}_a, n_a, t) = c_{ch}(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_a, n_a, t)$ (Equation 2).

Besides our own algorithms, we also compare umap-UCB and ITS to single-objective UCB and Thompson sampling provided with the ground truth utility functions of the user. Note that his is an unfair comparison, in the sense that our setting does not actually allow algorithms to know the ground truth utility functions from the beginning. However, it does provide insight into how much utility is lost due to having to estimate the utility function of the user via pairwise comparisons.

5.1 Problems and Experimental Setup

To test the performance of our algorithms, we use two types of MOMABs: the `double circle` and `random`, examples of which are depicted in Figure 3. Both problems have arms, a, associated with distributions over vector-valued rewards with mean vectors $\boldsymbol{\mu}_a$. We use independent Bernoulli distributions for each objective i, with a mean μ_a^i.

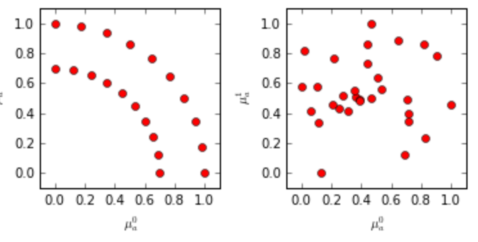

Fig. 3. Examples of `double circle` (left) and `random` (right) MOMABs.

A `double circle` is a two-objective MOMAB that is deterministically generated from two parameters: a number of ticks n_α, and a reduction parameter $p_r \in [0,1)$. A `double circle` places the mean vectors $\boldsymbol{\mu}_a$ of its arms on two quarter circles. The first is the upper quarter of the unit circle (i.e., the circle with radius 1), and the second that of a circle with radius p_r. On each of the upper quadrants of these circles, n_α arms are evenly distributed. The `double circle` for $n_\alpha = 10$ and $p_r = 0.7$ is depicted in Figure 3 (left).

A `random` instance is generated randomly using the number of objectives d, and the number of arms, $|\mathcal{A}|$, as parameters. First, $|\mathcal{A}|$ samples, $\boldsymbol{\mu}'_a$, are drawn from a d-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}'_a|\boldsymbol{\mu}_{rnd},\Sigma_{rnd})$, where $\boldsymbol{\mu}_{rnd} = \mathbf{1}$ (vector of ones), and Σ_{rnd} is a diagonal matrix with $\sigma^2_{rnd} = (\frac{1}{2})^2$ for each element on the diagonal. This set is normalised such that all means fall into the d-dimensional unit hypercube, $\boldsymbol{\mu}_a \in [0,1]^d$. Figure 3 (right) is an example `random` MOMAB with $d = 2$ and $|\mathcal{A}| = 30$.

Both umap-UCB and ITS require a prior on on the weights, $\mathbf{w}$, of the linear user utility function. We employ a multi-variate Gaussian prior, parameterised as $\eta^0 = \langle \boldsymbol{\mu}^0, \boldsymbol{\Sigma}^0 \rangle$. We use the same prior for both algorithms for all experiments. We assume $\boldsymbol{\mu}$ is the vector of equal weights, i.e., $\frac{1}{d}$ for each objective, and for the covariance matrix $\boldsymbol{\Sigma}^0$ we use a diagonal matrix with $\sigma^2_{cov} = (\frac{1}{3})^2$.

In all experiments, we measure the regret according to Definition 5, i.e., for a single run, when an algorithm pulls an arm a, $\mathbf{w}^* \cdot \boldsymbol{\Delta}_a$, is added to the total accumulated regret.

When querying the user for a comparison between two vectors, $\mathbf{x}$ and $\mathbf{y}$, we first calculate the true utility $u(\mathbf{x})$ and $u(\mathbf{y})$, to which we then add random noise $\varepsilon_\mathbf{x}$ and $\varepsilon_\mathbf{y}$, independently drawn from a normal distribution $\mathcal{N}(0, \sigma^2_{noise})$. We then compare $u(\mathbf{x}) + \varepsilon_\mathbf{x}$ to $u(\mathbf{y}) + \varepsilon_\mathbf{y}$. Unless otherwise indicated, $\sigma_{noise} = 0.001$.

5.2 Double Circles

In order to test the performance of our algorithms, we measure their regret, the number of questions they pose to the user, and the L2-norm distance between their estimated weight vectors and the ground truth weights, as a function of time (i.e., the total number of arm pulls) on a `double circle` with $p_r = 0.7$ and $n_\alpha = 10$ (Figure 4). We perform ten runs, with different levels of noise. When comparing umap-UCB and ITS, both in terms of regret and in the number of queries, ITS outperforms both UCB algorithms. However, this is not true for the approximation quality of the weights of the linear utility functions. When the noise levels are low ($\sigma_{noise} = 0.001$), ITS quickly focusses on the arms that are close to optimal, and reaches the best estimate of the weights. However, when the noise levels on the comparisons are higher ($\sigma_{noise} \geq 0.01$), the estimates of ITS first improve and then get worse again. We hypothesise that this is the result of ITS quickly focussing on the arms that are close to optimal, whose utility values lay so close that they fall inside the noise interval of the user comparisons, leading to a low signal-to-noise ratio and thus worse estimates. However, the effect on the incurred user regret of these worse estimates is minimal. We thus conclude that ITS outperforms UCB, and that poorer weight estimates are not necessarily detrimental for the incurred user regret.

We also test how much utility is lost by having to estimate the weights of the utility function rather than them being given from the start. In Figure 4 (left) the regret of the corresponding (cheating) single-objective algorithms to our algorithms are shown

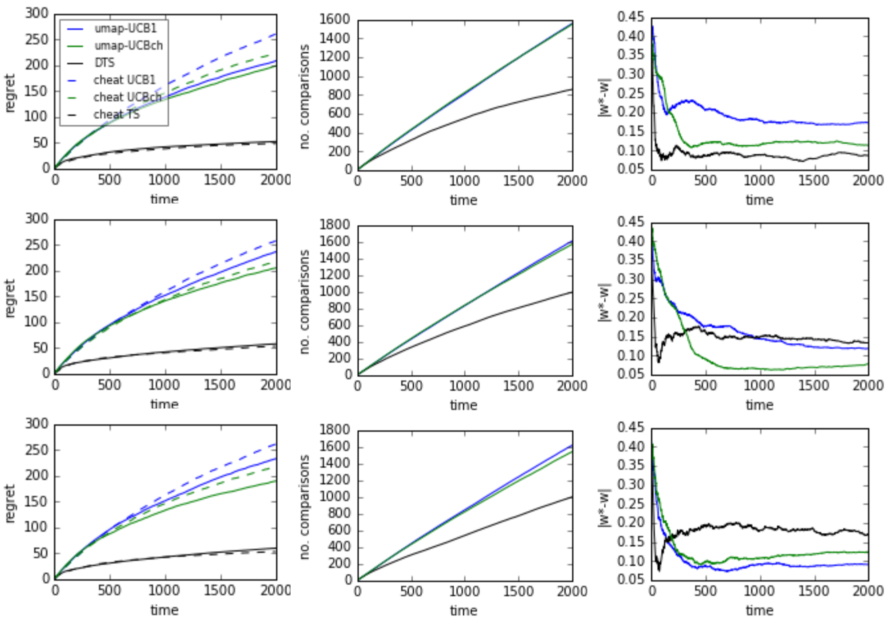

Fig. 4. The performance of umap-UCB and ITS (and UCB and Thompson sampling provided with the ground truth utility function at the start) in terms of regret (left), number of queries asked (middle), and quality of estimation of $\mathbf{w}$, on `double circle` instances with $p_r = 0.7$ and $n_\alpha = 10$, averaged over 10 runs. The rows represent the noise level in the user comparisons: $\sigma_{noise} = 0.001$(top); $\sigma_{noise} = 0.01$(middle); and $\sigma_{noise} = 0.1$(bottom). In all runs $\mathbf{w}^* = (0.2, 0.8)$.

as dashed lines. Our best algorithm — ITS — has only little more regret than single-objective Thompson sampling when provided with the ground truth weights from the beginning. Interestingly, umap-UCB1 and umap-UCB-ch perform even better than their single-objective equivalents (on most individual runs as well as on average). When inspecting the number of pulls per arm of these algorithms, umap-UCB pulls more suboptimal arms in total than single-objective UCB, but the suboptimal arms that are pulled are closer (i.e., have smaller $\mathbf{w}^* \cdot \boldsymbol{\Delta}_a$) than those of single-objective UCB. We thus conclude that there is only a small increase in regret when having to estimate $\mathbf{w}^*$.

We further note that, if the utility of the arms, $\mathbf{w}^* \cdot \boldsymbol{\mu}_a$, lay close together as in the `double circle`, umap-UCB and ITS keep on querying the user, as a result of not being able to distinguish between the optimal arm and the arms that are just below that in terms of utility (leading to over 40% of the 2000 arm pulls for our best algorithm — ITS — for the lowest noise levels). We expect the number of queries to be less in problems where arms lay further apart, as in `random` MOMAB instances.

5.3 Random MOMABs

In order to test the performance of umap-UCB on MOMABs with arms with mean vectors that lay further apart, we test them on `random` MOMABs with 30 arms each

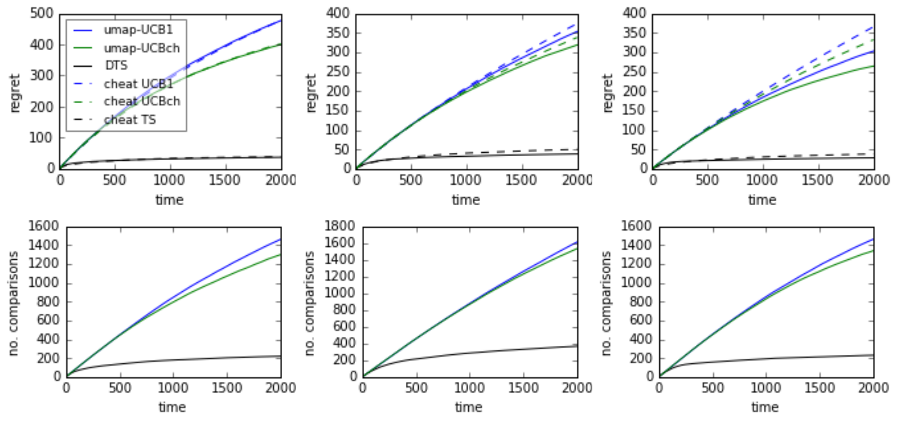

Fig. 5. The performance of umap-UCB and ITS (and UCB and Thompson sampling provided with the ground truth utility function from the start) in terms of regret (top), number of queries asked (bottom) for `random` MOMAB instances with 30 arms and user comparison noise $\varepsilon = 0.001$, randomly drawn $\mathbf{w}^*$ from a uniform distribution, for 2 (left), 4 (middle), and 6 (right) objectives, averaged over 25 runs.

and varying numbers of objectives (Figure 6). Again, we observe that the user regret for our best performing algorithm, ITS, does not attain a significantly lower regret than single-objective Thompson sampling provided with $\mathbf{w}^*$ from the start. Furthermore, ITS again outperforms both variants of umap-UCB.

Because the arms lay further apart than on the `double circle`, we observe that ITS — but not umap-UCB — is able to attain significantly lower regret. Furthermore, we observe that for ITS, the number of queries posed to the user is much less than for `double circle`, and often only increases marginally after about 1000 timesteps.

Finally, we tested the effect of the number of objectives of 30-arm `random` MOMABs on the algorithms' performances. Higher numbers of objectives seem to accrue less regret, while needing about equally many queries. This can be explained by the fact that for higher numbers of objectives, the arms lay further apart (in terms of Euclidean distance), and are thus easier to distinguish. We thus observe a blessing (rather than a curse) of dimensionality for user regret for a fixed number of arms.

In summary, we conclude that ITS outperforms umap-UCB both in terms of regret and in terms of the number of comparison queries posed to the user. Furthermore, when the expected reward vectors of the arms are sufficiently far apart, ITS is able to quickly reduce the number of questions asked per timestep, while umap-UCB cannot. We therefore conclude that ITS is the better algorithm.

6 Related Work

Several papers exist that study MOMABs [3, 9, 18, 22, a.o.]. However, to our knowledge, the previous work on MOMABs all focusses on the *offline* setting rather than the online interactive setting, as this paper does.

Related to online learning for MOMABs with pairwise comparisons by the user, are *relative bandits* [23]. Similar to our setting, relative bandits assume a hidden utility function which can be queried to obtain pairwise comparisons. Contrary to our setting however, the rewards cannot be observed, and the comparisons are made regarding *single* arm pulls, rather than the aggregate information, i.e., estimated means or posteriors over means, of all previous pulls of a given arm. The best arm is the arm with the highest probability of a single pull being preferred to a single pull of the other arms. The closest algorithms in this field are RUCB [23] and Double Thompson Sampling [21] which keep a preference matrix to determine which *two* arms to pull in *each iteration*.

In multi-objective sequential planning [4], combinatorial decision-making [20], and cooperative game theory [10] 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 adapt our methods to these problem classes to create Bayesian methods for learning in these problems, as well as for *MOMDPs* [13].

7 Conclusions and Future Work

In this paper we proposed *Utility-MAP UCB1 (umap-UCB)* and *Interactive Thompson Sampling (ITS)*, two Bayesian learning methods for *online* multi-objective reinforcement learning in which the agent can interact with its user. Both algorithms build upon state-of-the-art learning algorithms and Bayesian machine learning to learn about the environment and about the utility function of the user simultaneously. Both algorithms pose pairwise comparison queries to the user, and employ Bayesian logistic regression to learn about the linear preferences, representable with a weight vector $\mathbf{w}^*$, of its user. Umap-UCB uses explicit exploration bonuses, and elicits preferences when the best mean estimate for the current MAP estimate of the weight vector, and the best estimated mean plus exploration bonus for the same estimate, recommend different arms. ITS elicits preferences by pulling two sets of samples from both the posteriors of the means of the arms and the posterior for (the estimate of) $\mathbf{w}$, and querying the user when those two sets of samples have a different best arm.

We tested umap-UCB and ITS empirically, and showed that the performance of these algorithms in terms of user regret closely approximates the regret of UCB1 and regular Thompson sampling provided with the ground truth utility functions of the user. Umap-UCB can even perform better than UCB having access to the ground truth. Hence, we conclude that with our algorithms we can come close to the performance of state-of-the-art learning algorithms for single-objective MABs that are provided with the ground truth weights at the start. Furthermore, we conclude that ITS empirically outperforms umap-UCB both in terms of regret, and in terms of the number of queries posed to the user. We thus conclude that ITS is key for efficient multi-objective learning.

In this paper, we examined the setting of linear utility functions and multi-objective multi-armed bandits. As a next step, we aim to extend ITS to more general utility functions, e.g., additive utility functions [8], or even general-shape monotonically increasing utility functions [14]. Furthermore, we aim extend our methods to multi-objective reinforcement learning in MOMDPs and MOSGs [12, 13, 17, 19].

Acknowledgements

The first author is a postdoctoral fellow of the Research Foundation – Flanders (FWO). This research was in part supported by Innoviris – Brussels Institute for Research and Innovation.

References

1. S. Agrawal and N. Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *COLT*, pages 39–1, 2012.
2. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.
3. P. Auer, C.-K. Chiang, R. Ortner, and M. M. Drugan. Pareto front identification from stochastic bandit feedback. In *AISTATS*, pages 939–947, 2016.
4. N. Benabbou and P. Perny. Combining preference elicitation and search in multiobjective state-space graphs. In *IJCAI*, pages 297–303, 2015.
5. C. M. Bishop. *Pattern Recognition and Machine Learning*. 2006.
6. E. Brochu, N. de Freitas, and A. Ghosh. Active preference learning with discrete choice data. In *NIPS*, pages 409–416, 2008.
7. O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *NIPS*, pages 2249–2257, 2011.
8. R. T. Clemen. *Making Hard Decisions: An Introduction to Decision Analysis*. 1997.
9. M. M. Drugan and A. Nowé. Designing multi-objective multi-armed bandits algorithms: A study. In *IJCNN*, pages 1–8. IEEE, 2013.
10. A. Igarashi and D. M. Roijers. Multi-criteria coalition formation games. In *ADT*, 2017. To Appear.
11. P. Libin, T. Verstraeten, K. Theys, D. M. Roijers, P. Vrancx, and A. Nowé. Efficient evaluation of influenza mitigation strategies using preventive bandits. In *ALA*, 2017. 9 pages.
12. P. Mannion, J. Duggan, and E. Howley. A theoretical and empirical analysis of reward transformations in multi-objective stochastic games. In *AAMAS*, pages 1625–1627, 2017.
13. D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *JAIR*, 48:67–113, 2013.
14. D. M. Roijers and S. Whiteson. Multi-objective decision making. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 11(1):1–129, 2017.
15. G. Tesauro. Connectionist learning of expert preferences by comparison training. In *NIPS*, volume 1, pages 99–106, 1988.
16. W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.
17. K. Van Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of Pareto dominating policies. *JMLR*, 15(1):3483–3512, 2014.
18. K. Van Moffaert, K. Van Vaerenbergh, P. Vrancx, and A. Nowé. Multi-objective χ-armed bandits. In *IJCNN*, pages 2331–2338, 2014.
19. M. A. Wiering, M. Withagen, and M. M. Drugan. Model-based multi-objective reinforcement learning. In *ADPRL*, pages 1–6, 2014.
20. N. Wilson, A. Razak, and R. Marinescu. Computing possibly optimal solutions for multi-objective constraint optimisation with tradeoffs. In *IJCAI*, pages 815–822, 2015.
21. Huasen Wu and Xin Liu. Double thompson sampling for dueling bandits. In *NIPS*, pages 649–657, 2016.
22. S. Q. Yahyaa, M. M. Drugan, and B. Manderick. Thompson sampling in the adaptive linear scalarized multi objective multi armed bandit. In *ICAART*, pages 55–65, 2015.
23. M. Zoghi, S. Whiteson, R. Munos, and M. De Rijke. Relative upper confidence bound for the k-armed dueling bandit problem. In *ICML*, pages 10–18, 2014.