Thompson Sampling in the Adaptive Linear Scalarized Multi Objective Multi Armed Bandit

Saba Q. Yahyaa, Madalina M. Drugan and Bernard Manderick
Department of Computer Science, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium

Keywords: Multi-armed Bandit Problems, Multi-objective Optimization, Linear Scalarized Function, Scalarized Function Set, Thompson Sampling Policy.

Abstract: In the stochastic multi-objective multi-armed bandit (MOMAB), arms generate a vector of stochastic normal rewards, one per objective, instead of a single scalar reward. As a result, there is not only one optimal arm, but there is a set of optimal arms (Pareto front) using Pareto dominance relation. The goal of an agent is to find the Pareto front. To find the optimal arms, the agent can use linear scalarization function that transforms a multi-objective problem into a single problem by summing the weighted objectives. Selecting the weights is crucial, since different weights will result in selecting a different optimum arm from the Pareto front. Usually, a predefined weights set is used and this can be computational inefficient when different weights will optimize the same Pareto optimal arm and arms in the Pareto front are not identified. In this paper, we propose a number of techniques that adapt the weights on the fly in order to ameliorate the performance of the scalarized MOMAB. We use genetic and adaptive scalarization functions from multi-objective optimization to generate new weights. We propose to use Thompson sampling policy to select frequently the weights that identify new arms on the Pareto front. We experimentally show that Thompson sampling improves the performance of the genetic and adaptive scalarization functions. All the proposed techniques improves the performance of the standard scalarized MOMAB with a fixed set of weights.

1 INTRODUCTION

Multi-Objective Optimization (MOO) problem with conflicting objectives is present everywhere in the real-world. For instance, in shipping firm, the conflicting objectives could be consist of the shipping time and the cost. At the same time, shorten shipping time is needed in order to improve customer satisfaction, while also reducing the number of used ships to reduce the operating cost. It is obvious that adding more ships will reduce the needed shipping time but will increase the operating cost. The goal of the MOO with conflicted objectives is to tradeoff the conflicting objectives. The Multi-Objective Multi-Armed Bandit (MOMAB) problem (Drugan and Nowe, 2013; S. Q. Yahyaa and Manderick, 2014b) is the simplest approach to representing the MOO problem.

MOMAB problem is a sequential stochastic learning problem. At each time step t, an agent pulls one arm i from an available set of arms A and receives a reward vector $\boldsymbol{r}_i$ from the arm i with D dimensions (or objectives) as feedback signal. The reward vector is drawn from a probability distribution vector, for example from a normal probability distribution $N(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$, where $\boldsymbol{\mu}_i$ is the true mean vector and $\boldsymbol{\sigma}_i^2$ is the covariance matrix parameters of the arm i. The reward vector $\boldsymbol{r}_i$ that the agent receives from the arm i is independent from all other arms and independent from the past reward vectors of the selected arm i. Moreover, the mean vector of the arm i has *independent D* distributions, i.e. $\boldsymbol{\sigma}^2$ is a *diagonal covariance matrix*. We assume that the true mean vector and covariance matrix of each arm i are unknown parameters to the agent. Thus, by drawing each arm i, the agent maintains estimations of the true mean vector and the diagonal covariance matrix (or the variance vector) which are known as $\hat{\boldsymbol{\mu}}_i$ and $\hat{\boldsymbol{\sigma}}_i^2$, respectively.

The MOMAB problem has a set of Pareto optimal arms (Pareto front) A^*, that are incomparable, i.e. can not be classified using a designed partial order relations (Zitzler and et al., 2002). The agent has to figure out the optimal arms to minimize the total Pareto loss of not pulling the optimal arms. At each time step t, the Pareto loss (or Pareto regret) is the distance between the set mean of Pareto optimal arms and the mean of the selected arm (Drugan and Nowe, 2013).

Yahyaa S., Drugan M. and Manderick B..
Thompson Sampling in the Adaptive Linear Scalarized Multi Objective Multi Armed Bandit.
DOI: 10.5220/0005184400550065
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 55-65
ISBN: 978-989-758-074-1

Thus, the total Pareto regret is the cumulative summation of the Pareto regret over t time steps.

The Pareto front A^* can be found for example, by using linear scalarized function (f) (Eichfelder, 2008). Linear scalarized function is simple and intuitive. Given a predefined set weight $\boldsymbol{w}$, the linear scalarized function f weighs each value of the mean vector of an arm i, converts the multi-objective space to a single-objective one by summing the weighted mean values and selects the optimal arm i^* that has the maximum scalarized function. However, solving a multi-objective optimization problem means finding the Pareto front A^*. Thus, we need various linear scalarized functions $\boldsymbol{F}$, each scalarized function f^s, $f^s \in \boldsymbol{F}$, $s = 1,\cdots,S$ has a corresponding set of weight $\boldsymbol{w}^s$, to generate the variety of elements belonging to the Pareto front. The predefined total weight set $\boldsymbol{W}, \boldsymbol{W} = \{\boldsymbol{w}^1,\cdots,\boldsymbol{w}^S\}$ is uniformly random spread sampling in the weighted space (Das and Dennis, 1997). However, there is no guarantee that the total weight set $\boldsymbol{W}$ can find all the optimal arms in the Pareto front A^*. To improve the performance of the linear scalarized function (S. Q. Yahyaa and Manderick, 2014a) have used Knowledge Gradient (KG) policy (I. O. Ryzhov and Frazier, 2011) in the MOMAB problem, resulting Linear Scalarized Knowledge Gradient Function (LS-KG-F).

In this paper, we improve the performance of the linear scalarized knowledge gradient function LS-KG-F by introducing techniques from multi-objective optimization that redefine the weights for the weight set $\boldsymbol{w}$. We either generate a new weight set $\boldsymbol{w}$ by using genetic operators that change the weights directly (Drugan, 2013) or adapt the weights by using the arms in the Pareto front like in (J. Dubois-Lacoste and Stutzle, 2011). We propose also the Thompson sampling policy (Thompson, 1933) to select from the total weight set $\boldsymbol{W}$, the weight set $\boldsymbol{w}$ that identifies a larger set of optimal arms from the Pareto front A^*.

The rest of the paper is organized as follows: In Section 2 we introduce the multi-objective multi-armed bandit problem. In Section 3 we present the linear scalarized functions and the scalarized multi-objective bandits algorithm. In Section 4 we introduce algorithms to determine the weight set, the standard, the adaptive and the genetic algorithms. In Section 5 we introduce the adaptive scalarized multi-objective bandits algorithm that uses Thompson sampling policy to select the appropriate weight set. In Section 6, we describe the experiments set up followed by experimental results. Finally, we conclude and discuss future work.

2 MULTI OBJECTIVE MULTI ARMED BANDITS PROBLEM

Let us consider the MOMABs problems with $|A| \geq 2$ arms and with *independent D* objectives per arm. At each time step t, the agent selects one arm i and receives a reward vector $\boldsymbol{r}_i$. The reward vector $\boldsymbol{r}_i$ is drawn from a corresponding normal probability distribution $N(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$ with unknown mean parameter vector $\boldsymbol{\mu}_i$, $\boldsymbol{\mu}_i = [\mu_i^1,\cdots,\mu_i^D]^T$ and unknown variance parameter vector $\boldsymbol{\sigma}_i^2$, $\boldsymbol{\sigma}_i^2 = [\sigma_i^{2,1},\cdots,\sigma_i^{2,D}]^T$, where T is the transpose. Thus, by drawing each arm i, the agent maintains estimate of the mean parameter vector $\hat{\boldsymbol{\mu}}_i$ and the variance $\hat{\boldsymbol{\sigma}}_i^2$ parameter vector, and computes the number of times N_i arm i is drawn. The agent updates the estimated mean $\hat{\mu}_i^d$, the estimated variance $\hat{\sigma}_i^{2,d}$ of the selected arm i in each objective $d, d \in D$ and the number of times N_i arm i has been selected as follows (Powell, 2007):

$$N_{i+1} = N_i + 1 \tag{1}$$

$$\hat{\mu}_{i+1}^d = (1 - \frac{1}{N_{i+1}})\,\hat{\mu}_i^d + \frac{1}{N_{i+1}} r_{t+1}^d \tag{2}$$

$$\hat{\sigma}_{i+1}^{2,d} = \frac{N_{i+1}-2}{N_{i+1}-1}\,\hat{\sigma}_i^{2,d} + \frac{1}{N_{i+1}}(r_{t+1}^d - \hat{\mu}_i^d)^2 \tag{3}$$

where N_{i+1} is the updated number of times arm i has been selected, $\hat{\mu}_{i+1}^d$ is the updated estimated mean, and $\hat{\sigma}_{i+1}^{2,d}$ is the updated estimated variance of the arm i in the objective d and r_{t+1}^d is the observed reward of the arm i in the objective d.

When the objectives are conflicting with one another then the mean component μ_i^d of arm i corresponding with objective d, $d \in D$, can be better than the component μ_j^d of another arm j but worse if we compare the components for another objective d': $\mu_i^d > \mu_j^d$ but $\mu_i^{d'} < \mu_j^{d'}$ for objectives d and d', respectively. The agent has a set of optimal arms (Pareto front) A^* which can be found by the Pareto dominance relation (or Pareto partial order relation).

The Pareto dominance relation finds the Pareto front A^* directly in the multi-objective MO space (Zitzler and et al., 2002). It uses the following relations between the mean vectors of two arms. We use i and j to refer to the mean vector (estimated mean vector or true mean vector) of arms i and j, respectively:

Arm i dominates or is better than j, $i \succ j$, if there exists at least one objective d for which $i^d \succ j^d$ and for all other objectives d' we have $i^{d'} \succeq j^{d'}$. Arm i is incomparable with j, $i \parallel j$, if and only if there exists at least one objective d for which $i^d \succ j^d$ and there exists another objective d' for which $i^{d'} \prec j^{d'}$. Arm i is not dominated by j, $j \not\succ i$, if and only if there exists

at least one objective d for which $j^d \prec i^d$. This means that either $i \succ j$ or $i \parallel j$.

Using the above relations, Pareto front $A^*, A^* \subset A$ be the set of arms that are not dominated by all other arms. Moreover, the optimal arms in A^* are incomparable with each other.

In the MOMAB, the agent has to find the Pareto front A^*, therefore, the performance measure is the Pareto regret (Drugan and Nowe, 2013). *The Pareto regret measure* (R_{Pareto}) measures the distance between a mean vector of an arm i that is pulled at time step t and the Pareto front A^*. Pareto regret R_{Pareto} is calculated by finding firstly the virtual distance dis^*. The virtual distance dis^* is defined as the minimum distance that is added to the mean vector of the pulled arm $\boldsymbol{\mu}_t$ at time step t in each objective to create a virtual mean vector $\boldsymbol{\mu}_t^*$, $\boldsymbol{\mu}_t^* = \boldsymbol{\mu}_t + \boldsymbol{\varepsilon}^*$ that is incomparable with all the arms in Pareto set A^*, i.e. $\boldsymbol{\mu}_t^* || \boldsymbol{\mu}_i \, \forall_{i \in A^*}$. Where $\boldsymbol{\varepsilon}^*$ is a vector, $\boldsymbol{\varepsilon}^* = [dis^{*,1}, \cdots, dis^{*,D}]^T$. Then, the Pareto regret R_{Pareto}, $R_{Pareto} = dis(\boldsymbol{\mu}_t, \boldsymbol{\mu}_t^*) = dis(\boldsymbol{\varepsilon}^*, \mathbf{0})$ is the distance between the mean vector of the virtual arm $\boldsymbol{\mu}_t^*$ and the mean vector of the pulled arm $\boldsymbol{\mu}_t$ at time step t, where dis, $dis(\boldsymbol{\mu}_t, \boldsymbol{\mu}_t^*) = (\sum_{d=1}^{D} (\mu_t^{*,d} - \mu_t^d)^2)^{(1/2)}$ is the Euclidean distance. Thus, the regret of the Pareto front is 0 for optimal arms, i.e. the mean of the optimal arm coincides itself.

3 THE SCALARIZED MULTI-OBJECTIEVE BANDITS

Linear scalarization function converts the multi-objective into single-objective optimization (Eichfelder, 2008). However, solving a multi-objective optimization problem means finding the Pareto front A^*. Thus, we need a set of scalarized functions $\boldsymbol{F}$, $\boldsymbol{F} = \{f^1, \cdots, f^s, \cdots, f^S\}$ to generate a variety of elements belonging to the Pareto front A^*. Each scalarized function f^s, $f^s \in \boldsymbol{F}$ has a corresponding predefined set of weight $\boldsymbol{w}^s$, $\boldsymbol{w}^s \in \boldsymbol{W}$, where $\boldsymbol{W} = (\boldsymbol{w}^1, \cdots, \boldsymbol{w}^S)$.

The linear scalarization function assigns to each value of the mean vector of an arm i a weight w^d and the result is the sum of these weighted mean values. Given a predefined set of weights $\boldsymbol{w}^s$, $\boldsymbol{w}^s = (w^1, \cdots, w^D)$ such that $\sum_{d=1}^{D} w^d = 1$, the linear scalarized across mean vector is:

$$f^s(\boldsymbol{\mu}_i) = w^1 \mu_i^1 + \cdots + w^D \mu_i^D \tag{4}$$

where $f^s(\boldsymbol{\mu}_i)$ is a linear scalarized function s, $s \in S$ over the mean vector $\boldsymbol{\mu}_i$ of the arm i. After transforming the multi-objective problem to a single-objective problem, the linear scalarized function f^s selects the arm $i_{f^s}^*$ that has the maximum linear scalarized function value:

$$i_{f^s}^* = \underset{1 \le i \le A}{\operatorname{argmax}} \, f^s(\boldsymbol{\mu}_i)$$

The linear scalarization is very popular because of its simplicity. However, it can not find all the optimal arms in the Pareto front A^* (Das and Dennis, 1997). To improve the performance of the linear scalarized function, (S. Q. Yahyaa and Manderick, 2014b) have extended Knowledge Gradient (KG) policy (I. O. Ryzhov and Frazier, 2011) to the MOMAB problem, resulting linear scalarization function knowledge gradient. (S. Q. Yahyaa and Manderick, 2014b) have proposed two variants of linear scalarized KG, linear saclarized KG across arms (LS1-KG) and linear saclarized KG across dimensions (LS2-KG). Since LS1-KG performs better than LS2-KG, we will use linear scalarized KG across arms LS1-KG.

The linear scalarized-KG across arms (LS1-KG) converts the multi-objective estimated mean $\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\mu}}_i = [\hat{\mu}_i^1, \cdots, \hat{\mu}_i^D]^T$ and estimated variance $\hat{\boldsymbol{\sigma}}_i^2, \hat{\boldsymbol{\sigma}}_i^2 = [\hat{\sigma}_i^{2,1}, \cdots, \hat{\sigma}_i^{2,D}]^T$ of each arm to one-objective, then computes the corresponding bound (or term) ExpB_i. At each time step t, LS1-KG weighs both the estimated mean vector $\hat{\boldsymbol{\mu}}_i$ and estimated variance vector $\hat{\boldsymbol{\sigma}}_i^2$ of each arm i, converts the multi-objective vectors to one-objective values by summing the elements of each vector. Thus, we have one-objective multi-armed bandits problem. The KG policy calculates for each arm, a bound which depends on all available arms and selects the arm that has the maximum estimated mean plus the bound. The LS1-KG is as follows:

$$\widetilde{\mu}_i = f^s(\hat{\boldsymbol{\mu}}_i) = w^1 \hat{\mu}_i^1 + \cdots + w^D \hat{\mu}_i^D \qquad \forall_i \tag{5}$$

$$\widetilde{\sigma}_i^2 = f^s(\hat{\boldsymbol{\sigma}}_i^2) = w^1 \hat{\sigma}_i^{2,1} + \cdots + w^D \hat{\sigma}_i^{2,D} \qquad \forall_i \tag{6}$$

$$\widetilde{\widetilde{\sigma}}_i^2 = \widetilde{\sigma}_i^2 / N_i \qquad \forall_i \tag{7}$$

$$v_i = \widetilde{\widetilde{\sigma}}_i \, g\left(-\left| \frac{\widetilde{\mu}_i - \max\limits_{j \neq i, \, j \in A} \widetilde{\mu}_j}{\widetilde{\widetilde{\sigma}}_i} \right| \right) \qquad \forall_i \tag{8}$$

where f^s is a linear scalarization function that has a predefined set of weight $\boldsymbol{w}^s = (w^1, \cdots, w^D)$, $\widetilde{\mu}_i$, and $\widetilde{\sigma}_i^2$ are the modified estimated mean, and the modified estimated variance of an arm i, respectively which are one-objective values and $\widetilde{\widetilde{\sigma}}_i$ is the modified Root Mean Square Error (RMSE) of an arm i. The v_i is the KG index of an arm i. The function $g(\zeta)$, $g(\zeta) = \zeta\Phi(\zeta) + \phi(\zeta)$, where Φ, and ϕ are the cumulative distribution, and the density of the standard normal density $N(0,1)$, respectively. Linear scalarized-KG

across arms selects the optimal arm $i^*_{LS_1KG}$ according to:

$$i^*_{LS_1KG} = \underset{i=1,\cdots,|A|}{\text{argmax}} \ (\widetilde{\mu}_i + \text{ExpB}_i) \quad (9)$$

$$= \underset{i=1,\cdots,|A|}{\text{argmax}} \ (\widetilde{\mu}_i + (L-t) * |A| D * v_i) \quad (10)$$

where ExpB_i is the bound of arm i, $|A|$ is the number of arms, D is the number of objectives, L is the horizon of an experiments, i.e. length of trajectories and t is the current time step.

The algorithm. The pseudocode of the Scalarized Multi-objectieve Multi-Armed Bandit (SMOMAB) algorithm is given in Figure 1. The linear scalarized-KG across arms LS1-KG function is f. The scalarized function set is $\boldsymbol{F} = (f^1, \cdots, f^S)$, where each LS1-KG function f^s has a predefined weight set, $\boldsymbol{w}^s = (w^{1,s}, \cdots, w^{D,s})$ and the number of scalarized function is $|S|$, $|S| = D+1$.

The algorithm in Figure 1 plays each arm for each scalarized function s, *Initial* plays (step: 2)[1]. N^s is the number of times the scalarized function s is pulled and N^s_i is the number of times the arm i under the scalarized function s is pulled. $(\boldsymbol{r}_i)^s$ is the reward of the pulled arm i under the scalarized function s which is drawn from a normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2_r)$, where $\boldsymbol{\mu}$ is the unknown true mean vector and $\boldsymbol{\sigma}^2_r$ is the unknown true variance vector of the reward. $(\hat{\boldsymbol{\mu}}_i)^s$ and $(\hat{\boldsymbol{\sigma}}_i)^s$ are the estimated mean and standard deviation vectors of the arm i under the scalarized function s, respectively. After initial playing, the algorithm chooses uniformly at random one of the scalarized function s (step: 4). The algorithm determines the corresponding weight set $\boldsymbol{w}^s$ such that $\sum_{d=1}^{D} w^{d,s} = 1$ (step: 5). The weight set $\boldsymbol{w}^s$ can be specified by using standard algorithm (Das and Dennis, 1997), adaptive algorithm (J. Dubois-Lacoste and Stutzle, 2011), or genetic algorithm (Drugan, 2013), we refer to Section 4 for more details. If the SMOMAB algorithm uses the standard algorithm to set the weights, then the total weight set $\boldsymbol{W} = (\boldsymbol{w}^1, \cdots, \boldsymbol{w}^S)$ is fixed until the end of playing L steps. However, if the SMOMAB algorithm uses the adaptive or the genetic algorithm, then the total weight set $\boldsymbol{W} = (\boldsymbol{w}^1, \cdots, \boldsymbol{w}^S)$ will change at each time step. The SMOMAB algorithm uses the predefined total weight set $\boldsymbol{W}$ till the end of playing *Initial* steps, then at each time step the adaptive and the genetic algorithms generate new weights. The algorithm selects the optimal arm $(i^*)^s$ that maximizes the LS1-KG function (step: 6) and simulates the selected arm $(i^*)^s$ to observe the reward vector $(\boldsymbol{r}_{i^*})^s$ (step: 7). The estimated mean vector $(\hat{\boldsymbol{\mu}}_{i^*})^s$, estimated standard deviation vector $(\hat{\boldsymbol{\sigma}}_{i^*})^s$, and the number $N^s_{i^*}$ of the selected arm and the number of the pulled scalarized function N^s are updated (step: 8). Finally, the algorithm computes the Pareto regret (step: 9). This procedure is repeated until the end of playing L steps which is the horizon of an experiment.

Note that, the algorithm in Figure 1 is an adapted version of the scalarized MOMABs from (Drugan and Nowe, 2013), but here the reward is drawn from normal distribution and the weight set $\boldsymbol{w}^s$ is determined.

1. **Input:** Horizon of an experiment L;number of arms $|A|$;number of objectives D;number of scalarized functions $|S| = D+1$;reward vector $\boldsymbol{r} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2_r)$.

2. **Initialize:** Total Weight set $\boldsymbol{W} = (\boldsymbol{w}^1, \cdots, \boldsymbol{w}^{D+1})$
   For each scalarized function $s = 1$ to S
   Play: each arm i, *Initial* steps
   Observe: $(r_i)^s$
   Update: $N^s \leftarrow N^s + 1; N^s_i \leftarrow N^s_i + 1;$ $(\hat{\mu}_i)^s; (\hat{\sigma}_i)^s$
   End
3. **Repeat**
4. Select a function s uniformly at random
5. Compute the weight set $w^s \leftarrow$ *Weight*
6. Select the optimal arm $(i^*)^s$ that maximizes the scalarized function f^s
7. Observe: reward vector $(\boldsymbol{r}_{i^*})^s, (\boldsymbol{r}_{i^*})^s = ([r^1_{i^*}, \cdots, r^D_{i^*}]^T)^s$
8. Update: $(\hat{\boldsymbol{\mu}}_{i^*})^s; (\hat{\boldsymbol{\sigma}}_{i^*})^s; N^s_{i^*} \leftarrow N^s_{i^*} + 1; N^s \leftarrow N^s + 1$
9. Compute: Pareto regret
10. **Until** L

11. Output: Pareto regret

Figure 1: The scalarized multi-objective multi-armed bandit (SMOMAB) algorithm.

4 ADAPTIVE WEIGHTS FOR THE SCALARIZED MOMAB

In this section, we provide different algorithms to identify the weight set $\boldsymbol{w}^s$.

Fixed set of weights. The standard algorithm (Das and Dennis, 1997) defines a fixed total weight set $\boldsymbol{W}$, $\boldsymbol{W} = (\boldsymbol{w}^1, \cdots, \boldsymbol{w}^s, \cdots, \boldsymbol{w}^S)$ that is uniformly random spread sampling in the weighted space. For example, the bi-objective multi-armed bandit with number of scalarized function $|S|$. The weight $w^{1,s}$ of the scalarized function s in the objective d, $d = 1$ is set to $1 - \frac{s-1}{|S|-1}$ and the weight $w^{2,s}$ of the scalarized function s in the objective d, $d = 2$ is set to $1 - w^{1,s}$.

Note that, this algorithm performs a uniform sampling in the weight space. However, there is no

[1] We use s to refer the scalarized function f^s that has a predefined weight set $\boldsymbol{w}^s = (w^1, \cdots, w^d, \cdots, w^D)$.

guarantee that the resulting arms will give a uniform spread in the objective space. The weight set $\boldsymbol{w}^s$, $\boldsymbol{w}^s \in \boldsymbol{W}$ is sequentially ordered, therefore, when the scalarized MOMAB algorithm (the algorithm in Figure 1) is stopped prematurely, it will not have sampled all the part of the weight space, possibly leaving a part of the Pareto front A^* undiscovered.

Adaptive Direction. The adaptive algorithm from (J. Dubois-Lacoste and Stutzle, 2011) takes into account the shape of the Pareto front A^* in order to obtain a well-spread set of the non-dominated arms. For a scalarized function s, this algorithm defines a norm (Euclidean distance) to specify the largest gap in the coverage of the Pareto front A^*. The largest gap is the gap between the maximum norm $\max_{i\in A}||\hat{\boldsymbol{\mu}}_i^s||$ and the minimum norm $\min_{i\in A}||\hat{\boldsymbol{\mu}}_i^s||$ of the estimated mean $\hat{\boldsymbol{\mu}}_i^s$ of an arm i, $i \in A$ under the scalarized function s. For example, the bi-objective multi-armed bandit with scalarized function s. The adaptive new weight $w^{1,s}$ of the objective d, $d = 1$ is perpendicular to the segment defined by the maximum arm $i_{\max}$, $i_{\max} = \text{argmax}_{i\in A}||\hat{\boldsymbol{\mu}}_i^s||$ and the minimum arm arm $i_{\min}$, $i_{\min} = \text{argmin}_{i\in A}||\hat{\boldsymbol{\mu}}_i^s||$ in the objective space, that is:

$$w^{1,s} = \frac{\hat{\mu}_{i_{\max}}^{2,s} - \hat{\mu}_{i_{\min}}^{2,s}}{\hat{\mu}_{i_{\max}}^{2,s} - \hat{\mu}_{i_{\min}}^{2,s} + \hat{\mu}_{i_{\min}}^{1,s} - \hat{\mu}_{i_{\max}}^{1,s}}$$

where $\hat{\mu}_{i_{\max}}^{2,s}$ and $\hat{\mu}_{i_{\min}}^{2,s}$ are the estimated mean of the arms $i_{\max}$ and $i_{\min}$ in the objective 2 under scalarized function s, respectively . And, $\hat{\mu}_{i_{\max}}^{1,s}$ and $\hat{\mu}_{i_{\min}}^{1,s}$ are the estimated mean of the arms $i_{\max}$ and $i_{\min}$ in the objective 1 under scalarized function s, respectively. The new weight $w^{2,s}$ of the second objective $d = 2$ under scalarized function s is set to $1 - w^{1,s}$. Note that, for number of objectives $D > 2$, the weight w^1 in the objective $d = 1$ is calculated by using the estimated mean of the objectives $d = 1$ and $d = 2$. The weight w^2 in the objective $d = 2$ is calculated by using the estimated mean of the objectives $d = 2$ and $d = 3$, and so on. While the weight w^D in the objective D is set to $1 - (w^1 + \cdots + w^{D-1})$.

This algorithm defines new weights based on the shape of the Pareto front. Therefore, if the shape of the Pareto front is irregular, then the new weight will not discover all the optimal arms in the Pareto front. This operator adapt the weights of only two objectives at the time.

Genetic Operators. The scalarized local search algorithm (Drugan, 2013) generates new weights for scalarized function s using real-coded genetic operators. The new weights are different from the parenting weights, therefore, it could explore the parts of the Pareto front A^* that are undiscovered.

The *mutation* operator, mutates each weight of a scalarization function independently using a normal distribution. The *recombination* operator generates a new weight $\boldsymbol{w}$ from two or more scalarized functions, each scalarized function has a predefined weight set $\boldsymbol{w}^s = (w^1, \cdots, w^D)$. The *translation recombination* operator translates the main set of scalarized function S with a normally distributed variable. The *rotation recombination* operator, considers that the scalarized functions S are positioned on a S-dimensional hypersphere. The generated new scalarized function s also belongs to this hypersphere around the main scalarized functions S, that is rotated with a small normally distributed angle. For more details, we refer to (Drugan and Thierens, 2010).

Since the mutation operator is the easiest one to implement, we will use it in our comparison. Given the weight set $\boldsymbol{w}^s(t)$ of the scalarized function s at time step t, the mutated new set of weight $\boldsymbol{w}^s(t+1)$ at time step $t+1$ is calculated as follows;

$$\boldsymbol{w}^s(t+1) = \boldsymbol{w}^s(t) + \boldsymbol{I}\mathbf{1}$$

where $\boldsymbol{I}$ is a diagonal matrix of size $D \times D$ with normally distributed variables and $\mathbf{1}$ is a vector of size D with 1 variables. After calculating the new weight set $\boldsymbol{w}^s(t+1)$, we can either replace the old weight set $\boldsymbol{w}^s(t)$ with the new weight set, i.e. $\boldsymbol{w}^s(t) \leftarrow \boldsymbol{w}^s(t+1)$ (mutation) or at each time step t, we generate new set of weight that is independent from the previous one (mutation without replacement).

5 THOMPSON SAMPLING IN THE SCALARIZED MOMAB ALGORITHM

In this section, we design an algorithm that frequently selects the appropriated scalarized function set of weights $\boldsymbol{w}^s$, $\boldsymbol{w}^s \in \boldsymbol{W}$, where the total weight set $\boldsymbol{W}$ is either determined by using standard algorithm, adaptive algorithm or genetic algorithm. The appropriate scalarized function is the one that improves the performance of the algorithm by identifying new Pareto optimal arms.

In the Bernoulli one-objective, Multi-Armed Bandits (MABs), the reward is a stochastic scalar value, and there is only one optimal arm. The reward r_i, $r_i \sim B(p_i)$ for an arm i is either 0, or 1 and generated from a Bernoulli distribution B with unknown probability of success p_i. The goal of an agent is to minimize the loss of not pulling the best arm i^* over L time steps. The loss (or the total regret) is $R_L = Lp^* - \sum_{t=1}^{L} p_i(t)$, where $p^* = \max_{i=1,\cdots,A} p_i$ is the probability of suc-

cess of the best arm i^*, and p_i is the probability of success of the selected arm i at time step t. To minimize the total regret, at each time step t, the agent has to trade-off between selecting the optimal arm i^* (exploitation) to minimize the regret[2] and selecting one of the non-optimal arm i to increase the confidence in the estimated probability of success $\hat{p}_i$, $\hat{p}_i = {}^{\alpha_i}/_{(\alpha_i+\beta_i)}$ of the arm i (exploration). Where α_i is the number of successes (the number of receiving reward equals 1) and β_i is the number of failures (the number of receiving reward equals 0) of the arm i.

Thompson Sampling Policy. (Thompson, 1933) assigns to each arm i, $i \in A$ a random probability of selection P_i to trade-off between exploration and exploitation. The random probability of selection P_i of each arm i is generated from Beta distribution, i.e. $P_i = \text{Beta}(\alpha_i, \beta_i)$, where α_i is the number of successes and β_i is the number of failures of the arm i. The random probability of selection P_i of an arm i depends on *the performance of the arm i*, i.e. the unknown probability of success p_i of the arm i. It will be high value if the arm i has high probability of success p_i value. With Bayesian priors on the Bernoulli probability of success p_i of each arm i, Thompson sampling assumes initially the number of successes, α_i and the number of failures, β_i for each arm i is 1. At each time t, Thompson sampling samples the probability of selection P_i for each arm i, $i \in A$ (the probability that an arm i is optimal) from Beta distribution, i.e. $P_i = \text{Beta}(\alpha_i, \beta_i)$. Beta distribution generates random values, therefore, probably, at time step t, the optimal arm i^*, $i^* = \text{argmax}_{i \in A}\, p_i$ has high probability of selection P_{i^*}, while at time step $t+1$ the suboptimal arm j, $j \in A$, $j \neq i^*$ has high probability of selection P_j.

Thompson sampling selects the optimal arm i^*_{TS} that has the maximum probability of selection $P_{i^*_{TS}}$, i.e. $i^*_{TS} = \text{argmax}_{i \in A} P_i$ and observes the reward $r_{i^*_{TS}}$. If $r_{i^*_{TS}} = 1$, then Thompson sampling updates the number of successes $\alpha_{i^*_{TS}} = \alpha_{i^*_{TS}} + 1$ for the arm i^*_{TS}. As a result, the estimated probability of success $\hat{p}_{i^*_{TS}}$ of the arm i^*_{TS} will increase. If $r_{i^*_{TS}} = 0$, then Thompson sampling updated the number of failures $\beta_{i^*_{TS}} = \beta_{i^*_{TS}} + 1$ for the arm i^*. As a result, the estimated probability of success $\hat{p}_{i^*_{TS}}$ of the arm i^*_{TS} will decrease.

Since, Thompson sampling is very easy to implement, we will use it to select the scalarized function s, $s \in S$. We assume that each scalarized function s has unknown probability of success p_s and when we select s, we either receive reward 1 or 0. We call the algorithm that uses Thompson sampling to select the weight set "Adaptive Scalarized Multi-Objective Multi-Armed Bandit" (adaptive-SMOMAB). Note that, adaptive-SMOMAB uses Thompson sampling to select the weight set, while scalarized multi-objective multi-armed bandit (MOMAB) selects uniformly at random one of the weight set $\boldsymbol{w}^s$, $\boldsymbol{w}^s \in \boldsymbol{W}$.

The Adaptive-SMOMAB Algorithm. As in the case of MABs, Thompson sampling uses random of beta distribution $\text{Beta}(\alpha_s, \beta_s)$ to assign a probability of selection P_s for each scalarized function s. Where α_s is the number of successes of the scalarized function s and β_s is the number of failures of the scalarized function s. We consider that each scalarized function s has unknown probability of success p_s and by playing each scalarized function s, we can estimate the corresponding probability of success. At each time step t, we maintain value $V_s(t)$ for each scalarized function s, where $V_s(t) = \max_{i \in A} f^s((\hat{\boldsymbol{\mu}}_i)^s)$ is the value of the optimal arm i^*, $i^* = \text{argmax}_{i \in A}\, f^s((\hat{\boldsymbol{\mu}}_i)^s)$ under scalarized function s and $(\hat{\boldsymbol{\mu}}_i)^s$ is the estimated mean vector of the arm i under the scalarized function s. If we select the scalarized function s at time step t and the value of this scalarized function $V_s(t)$ is greater or equal than the value at the previous selection, $V_s(t) \geq V_s(t-1)$, then this scalarized function s performs well because it has the ability to select the same optimal arm or to select another optimal arm that has higher value. Otherwise, the scalarized function s does not perform well.

The pseudocode of the adaptive-SMOMAB algorithm is given in Figure 2. The linear scalarized-KG across arms LS1-KG function f is used to convert the multi-objective to a single one. The number of scalarized function is $|S|$, $|S| = D+1$, where D is the number of objectives. The horizon of an experiment is L steps. The algorithm in Figure 2 plays each arm for each scalarized function s, *Initial* plays. The scalarized function set is $\boldsymbol{F} = (f^1, \cdots, f^{|S|})$, each scalarized function s has a corresponding predefined weight set, $\boldsymbol{w}^s = (w^{1,s}, \cdots, w^{D,s})$. N^s is the number of times the scalarized function s is pulled and N^s_i is the number of times the arm i under the scalarized function s is pulled. $(\boldsymbol{r}_i)^s$ is the reward vector of the pulled arm i under the scalarized function s which is drawn from a normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2_r)$, where $\boldsymbol{\mu}$ is the true mean vector and $\boldsymbol{\sigma}^2_r$ is the true variance vector of the reward. $(\hat{\boldsymbol{\mu}}_i)^s$ and $(\hat{\boldsymbol{\sigma}}_i)^s$ are the estimated mean and standard deviation vectors of the arm i under the scalarized function s, respectively. V_s, $V_s = \max_{i \in A} f^s((\hat{\boldsymbol{\mu}}_i)^s)$ is the value of each scalarized function s after playing each arm i *Initial* steps, where $f^s((\hat{\boldsymbol{\mu}}_i)^s)$ is the value of the LS-KG for the arm i under scalarized function s. The number of successes α_s, and the number of failures β_s for each scalarized function s are set to 1 as (Thompson, 1933), therefore, the estimated probability $\hat{p}_s$, $\hat{p}_s = {}^{\alpha_s}/_{(\alpha_s+\beta_s)}$ of success is 0.5. The prob-

[2] At each time step t, the regret equals $p^* - p_i(t)$.

1. **Input:** Horizon of an experiment L;number of arms $|A|$;number of objectives D;number of scalarized functions $|S| = D+1$;reward vector $\boldsymbol{r} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}_r^2)$.

2. **Initialize:**Total wight set $\boldsymbol{W} = (\boldsymbol{w}^1, \cdots, \boldsymbol{w}^{D+1})$
   For each scalarized function $s = 1$ to $|S|$
   Play: each arm i, *Initial* steps
   Observe: $(\boldsymbol{r}_i)^s$
   Update: $N^s \leftarrow N^s + 1; N_i^s \leftarrow N_i^s + 1$
   $(\hat{\boldsymbol{\mu}}_i)^s; (\hat{\boldsymbol{\sigma}}_i)^s$
   Compute: $V_s(t) = \max_{1 \le i \le A} f^s((\hat{\boldsymbol{\mu}}_i)^s)$
   Set: $\alpha_s = 1;\ \beta_s = 1;\ \hat{p}_s = 0.5;\ P_s = \frac{1}{|S|}$
   End

3. **Repeat**
4. For each scalarized function $s = 1, \cdots, S$
5. Sample P_s from Beta(α_s, β_s)
6. End for
7. $s^* = \text{argmax}_s\ P_s$
8. Compute: the new weight set $\boldsymbol{w}^{s^*} \leftarrow$ *Weight*
9. Select: the optimal arm i^* that maximizes scalarized function f^{s^*} that has the new weight set $\boldsymbol{w}^{s^*}$
10. Compute: $V_{s^*}(t) = \max_{1 \le i \le A} f^{s^*}((\hat{\boldsymbol{\mu}}_i)^{s^*})$
11. If $V_{s^*}(t) \ge V_{s^*}(t-1)$
12. $\alpha_{s^*} = \alpha_{s^*} + 1$
13. Else
14. $\beta_{s^*} = \beta_{s^*} + 1$
15. End
16. $V_{s^*}(t-1) \leftarrow V_{s^*}(t)$
17. Observe: reward vector $(\boldsymbol{r}_{i^*})^{s^*} = ([r_{i^*}^1, \cdots, r_{i^*}^D]^T)^{s^*}$
18. Update: $(\hat{\boldsymbol{\mu}}_{i^*})^{s^*}$; $(\hat{\boldsymbol{\sigma}}_{i^*})^{s^*}$;
    $N_{i^*}^{s^*} \leftarrow N_{i^*}^{s^*} + 1;\ N^{s^*} \leftarrow N^{s^*} + 1$
19. Compute: Pareto regret

20. **Until** L

21. **Output:** Pareto regret

Figure 2: Adaptive Scalarized MOMAB.

ability of selection P_s each scalarized function s is $\frac{1}{|S|}$ (step: 2).

After initial playing, the algorithm computes the probability of selection P_s of each scalarized function s, the probability of selection P_s is sampled from beta distribution Beta(α_s, β_s)(step: 4). The algorithm selects the optimal scalarized function s^*, the one that has a max probability of success (step: 7). The algorithm determines the weight set $\boldsymbol{w}^{s^*}$ for the optimal scalarized function s^* (step: 8). The weight set $\boldsymbol{w}^{s^*}$ is determined either by using adaptive algorithm, genetic algorithm or standard algorithm, Section 4. The algorithm selects the optimal arm i^* under the optimal scalarized function s (step: 9) and computes the value of the optimal scalarized function s^* (step: 10) which is the value of the optimal arm i^*. If the value $V_{s^*}(t)$ of the optimal scalarized function s^* at time step t, is greater or equal than the value of the of the optimal scalarized function s^* at time step $t-1$, then the optimal scalarized function s^* performs well. The number of successes α_{s^*} is increased. Other wise, the number of failures β_{s^*} is increased (steps: 11-15). Then, the algorithm updates the value V_{s^*} of the optimal scalarized function s^* (step: 16). The algorithm simulates the optimal arm i^* of the optimal scalarized function s^*, observes the corresponding reward vector $(\boldsymbol{r}_{i^*})^{s^*}$ and updates the estimated mean vector $(\hat{\boldsymbol{\mu}}_{i^*})^{s^*}$, the estimated standard deviation vector $(\hat{\boldsymbol{\sigma}}_{i^*})^{s^*}$ of the arm i^* and updates the number $N_{i^*}^{s^*}$ of the selected arm and the number of the pulled scalarized function N^{s^*} (steps: 17-18). Finally, the algorithm computes the Pareto regret (step: 19). This procedure is repeated until the end of playing L steps.

Note that, if the adaptive-SMOMAB algorithm uses the standard algorithm to set the weights, then the total weight set $\boldsymbol{W} = (\boldsymbol{w}^1, \cdots, \boldsymbol{w}^S)$ is fixed until the end of playing L steps. However, if the adaptive-SMOMAB algorithm uses the adaptive or the genetic algorithm, then the total weight set $\boldsymbol{W} = (\boldsymbol{w}^1, \cdots, \boldsymbol{w}^S)$ will change at each time step. The adaptive-SMOMAB algorithm uses a predefined total weight set $\boldsymbol{W}$ till the end of playing *Initial* steps, then at each time step the adaptive and the genetic algorithms generate new weights.

6 EXPERIMENTS

In this section, we firstly compare the performance of the adaptive scalarized multi-objective, multi-armed bandit (adaptive-SMOMAB) algorithm, the algorithm in Figure 2 and the performance of the scalarized multi-objective multi-armed bandit (SMOMAB) algorithm, the algorithm in Figure 1. We use the genetic, the adaptive, and the standard algorithms, Section 4 to set the weight set $\boldsymbol{w}^s$ for each linear scalarized knowledge gradient across arms (LS1-KG) s. Secondly, we experimentally compare the standard, the adaptive and the genetic algorithms, using the adaptive-SMOMAB algorithm. The performance measures are:

1. The Pareto regret, Section 2 at each time step t which is the average of M experiments.
2. The cumulative Pareto regret, Section 2 at each time step t which is the average of M experiments.

The number of experiments M is 1000. The horizon of each experiment L is 1000. The reward vectors $\boldsymbol{r}_i$ of each arm i are drawn from corresponding normal distribution $N(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_{i,r}^2)$ where $\boldsymbol{\mu}_i = [\mu_i^1, \cdots, \mu_i^D]^T$ is the true mean vector and $\boldsymbol{\sigma}_{i,r} = [\sigma_{i,r}^1, \cdots, \sigma_{i,r}^D]^T$ is the true

standard deviation vector of the reward of the arm i. The true means and the true standard deviations of arms are unknown parameters to the agent. The LS1-KG needs the estimated variance $\hat{\sigma}_i^2$ for each arm i, therefore, each arm is played initially 2 times which is the minimum number to estimate the variance.

At each time step t, the mutation operator mutates the weight set $\boldsymbol{w}^s(t)$ of each scalarized function s at time step t independently using a normal distribution $N(\mu,\sigma^2)$ to generate new weight set $\boldsymbol{w}^s(t+1)$, we set the mean μ to 0 and the variance σ^2 to 0.05 as (Drugan, 2013). We can either replace the old weight $\boldsymbol{w}^s(t)$ set with the new weight set $\boldsymbol{w}^s(t+1)$, i.e. $\boldsymbol{w}^s(t) \leftarrow \boldsymbol{w}^s(t+1)$ or at each time step t, we generate new set of weight that is independent from the previous one. We compare the two different setting and we find out that the replacement setting performs better, therefore, we use this in our comparison.

6.1 Bi-Objective

Example 1. We used the same example in (Drugan and Nowe, 2013) because it simple to understand and contains non-convex mean vector set. The number of arms $|A|$ equals 6, the number of objectives D equals 2. The standard deviation for arms in each objective is 0.1. The true mean set vector is ($\boldsymbol{\mu}_1 = [0.55,0.5]^T$, $\boldsymbol{\mu}_2 = [0.53,0.51]^T$, $\boldsymbol{\mu}_3 = [0.52,0.54]^T$, $\boldsymbol{\mu}_4 = [0.5,0.57]^T$, $\boldsymbol{\mu}_5 = [0.51,0.51]^T$, $\boldsymbol{\mu}_6 = [0.5,0.5]^T$). Note that, the Pareto front is $A^* = (a_1^*, a_2^*, a_3^*, a_4^*)$, where a_i^* refers to the optimal arm i^*. The suboptimal a_5 is not dominated by the two optimal arms a_1^* and a_4^*, but a_2^* and a_3^* dominates a_5 while a_6 is dominated by all the other mean vectors. Figure 3 shows a set of 2-objective true mean with a non-convex set.

First, we compare the performance of the SMOMAB and adaptive-SMOMAB algorithms. We use either the standard algorithm, the adaptive algorithm, or the genetic algorithm to set the weights. Figure 4 gives the comparison of the SMOMAB and adaptive-SMOMAB algorithms using the standard, the adaptive, and the genetic algorithms. The standard deviation $\sigma_{i,r}^d$ of reward for each arm i in each objective d is set to 0.1. The x-axis gives the time step. The y-axis is the cumulative Pareto regret which is the average of M experiments at each time step t. Figure 4 shows that the SMOMAB algorithm performs better than the adaptive-SMOMAB algorithm using the adaptive, and the genetic algorithms, since the cumulative Pareto regret is decreased. While, the adaptive-SMOMAB algorithm performs slightly better than the SMOMAB algorithm using the standard algorithm.

Second, we compare the performance of the standard, the adaptive, and the genetic algorithms using the adaptive-SMOMAB algorithm. Figure 5 gives the cumulative Pareto regret. The x-axis is the time step. The y-axis is the cumulative Pareto regret which is the average of M experiments at each time step t. Figure 5 shows that the standard algorithm is the best algorithm and the adaptive algorithm is the worst one. The mutation algorithm performs better than the adaptive algorithm and slightly worse than the standard algorithm.

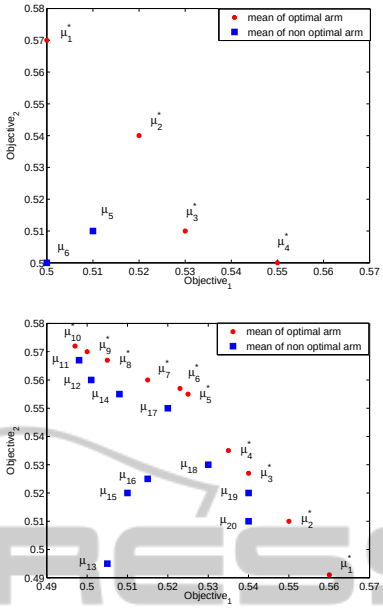
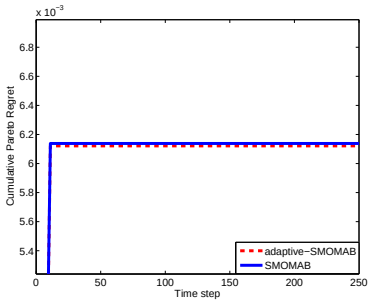


Figure 3: Non-convex and convex mean vector sets with bi-objective. Upper figure shows a non-convex set with 6-armed. Lower figure shows a convex set with 20-armed.

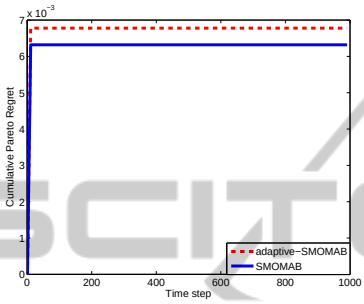
In Example 1, the Pareto front A^* contains optimal arms that are far from the non-optimal arms according to the Euclidean distance and the number of the optimal arms $|A^*|$, $|A^*| = 4$ is larger than the number of the non-optimal arms which is equal 2. Therefore, the SMOMAB algorithm almost performs better than the adaptive-SMOMAB because it selects uniformly at random one of the scalarized function. And, the standard algorithm performs better than the adaptive and the genetic algorithms because they generate new weights that are nearest to each other to explore more the optimal arms.

6.2 Triple-Objective

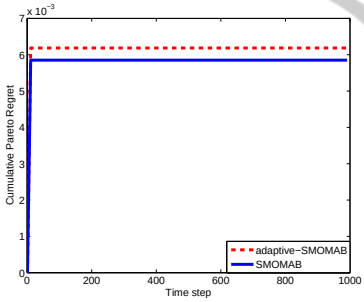
Example 2. With number of objectives D equals 2, number of arms $|A|$ equals 20 and convex Pareto mean set, ($\boldsymbol{\mu}_1 = [.56,.491]^T, \boldsymbol{\mu}_2 = [.55,.51]^T, \boldsymbol{\mu}_3 = [.54,.527]^T, \boldsymbol{\mu}_4 = [.535,.535]^T, \boldsymbol{\mu}_5 = [.525,.555]^T, \boldsymbol{\mu}_6 = [.523,.557]^T, \boldsymbol{\mu}_7 = [.515,.56]^T$,



a. Using the standard algorithm



b. Using the adaptive algorithm



c. Using the genetic algorithm

Figure 4: Performance comparison of the SMOMAB algorithm with the adaptive-SMOMAB algorithm on 2-objective, 6-armed problem. The weights are set using the standard algorithm in figure *a*, using the adaptive algorithm in figure *b*, and using the genetic algorithm in figure *c*.

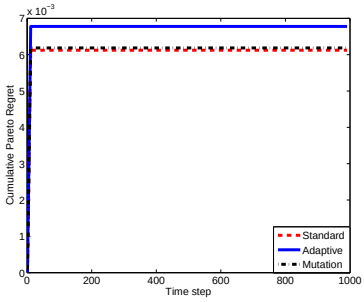
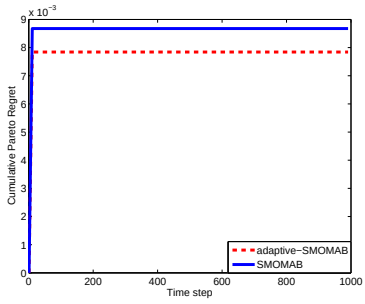


Figure 5: Performance comparison of the standard, the adaptive, and the genetic algorithms on 2-objective, 6-armed problem using the adaptive-SMOMAB algorithm.



a. Using the adaptive algorithm

Figure 6: Performance comparison of the SMOMAB algorithm with the adaptive-SMOMAB algorithm on 3-objective, 20-armed problem. The weights are set using the adaptive algorithm.

$\boldsymbol{\mu}_8 = [.505, .567]^T, \boldsymbol{\mu}_9 = [.5, .57]^T, \boldsymbol{\mu}_{10} = [.497, .572]^T, \boldsymbol{\mu}_{11} = [.498, .567]^T, \boldsymbol{\mu}_{12} = [.501, .56]^T, \boldsymbol{\mu}_{13} = [.505, .495]^T, \boldsymbol{\mu}_{14} = [.508, .555]^T, \boldsymbol{\mu}_{15} = [.51, .52]^T, \boldsymbol{\mu}_{16} = [.515, .525]^T, \boldsymbol{\mu}_{17} = [.52, .55]^T, \boldsymbol{\mu}_{18} = [.53, .53]^T, \boldsymbol{\mu}_{19} = [.54, .52]^T, \boldsymbol{\mu}_{20} = [.54, .51]^T$), the standard deviation for arms in each objective is set to 0.1. The Pareto front A^* contains 10 optimal arms, $A^* = (a_1^*, a_2^*, a_3^*, a_4^*, a_5^*, a_6^*, a_7^*, a_8^*, a_9^*, a_{10}^*)$ (S. Q. Yahyaa and Manderick, 2014c). Note that, the number of the optimal arms $|A^*|$, $|A^*| = 10$ is equal to the number of non-optimal arms and the optimal arms are close to the non-optimal arm according to the Euclidean distance. Figure 3 shows a set of 2-objective convex true mean vector set. We add extra objective to Example 2, resulting in 3-objective, 20-armed bandit problem. The Pareto front A^* still contains 10 optimal arms and the optimal arms are closer to non-optimal arms compared to Example 2 according to the Euclidean distance.

First, we compare the performance of the SMOMAB and adaptive-SMOMAB algorithms. We use either the standard algorithm, the adaptive algorithm, or the genetic algorithm to determine the weight set. The adaptive-SMOMAB algorithm performs better than the SMOMAB algorithm for the standard, adaptive and genetic algorithms. Figure 6 gives the comparison of the SMOMAB and adaptive-SMOMAB algorithms using the adaptive algorithm. Figure 6 shows that the adaptive-SMOMAB algorithm performs better than the SMOMAB algorithm using the adaptive algorithm to set the weight set.

Second, we compare the performance of the standard, the adaptive, and the genetic algorithms using the adaptive-SMOMAB algorithm. Figure 7 gives the cumulative Pareto regret. Figure 7 shows that the standard algorithm is the worst algorithm. The adaptive algorithm is the best one and performs slightly better than the mutation algorithm. The mutation al-

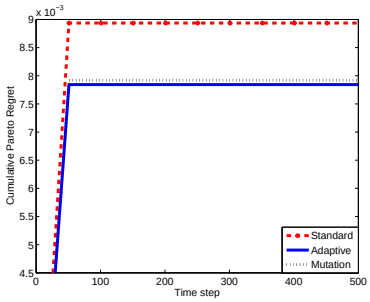


Figure 7: Performance comparison of the standard, the adaptive, and the genetic algorithms on 3-objective, 20-armed problem using the adaptive-SMOMAB algorithm.

gorithm performs better than the standard algorithm and worse than the adaptive algorithm.

From the above experiment, we see that when we increase the number of objectives, the adaptive-SMOMAB algorithm performs better than the SMOMAB algorithm. Also, we see that the adaptive and the genetic algorithms perform better than the standard algorithm.

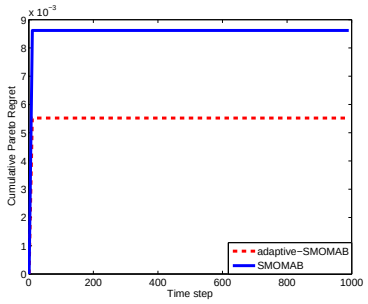
6.3 5-Objective

We add extra 2 objectives to Example 2, resulting in 5-objective, 20-armed bandit problem, leaving the Pareto front A^* unchanged. The optimal arms in the Pareto front A^* are still closer to the non-optimal arms according to the Euclidean distance. The standard deviation for arms in each objective is set to 0.01.

First, we compare the performance of the SMOMAB and adaptive-SMOMAB algorithms. We use either the standard algorithm, the adaptive algorithm, or the genetic algorithm to set the weights. The adaptive-SMOMAB algorithm performs better than the SMOMAB algorithm for all the weight setting. Figure 8 gives the comparison of the SMOMAB and adaptive-SMOMAB algorithms using the standard algorithm. Figure 8 shows that the adaptive-SMOMAB algorithm performs better than the SMOMAB algorithm using the standard algorithm.

Second, we compare the performance of the standard, the adaptive, and the genetic algorithms using the adaptive-SMOMAB algorithm. Figure 9 gives the cumulative Pareto regret. Figure 9 shows that the standard algorithm is the worst algorithm and the mutation algorithm is the best algorithm. The adaptive algorithm performs better than the standard algorithm and worse than the mutation algorithm.

From the above experiment, we see that when we increase the number of objectives, i.e. $D > 3$ the performance of the adaptive-SMOMAB algorithm is increased. We also see the performance of the genetic



a. Using the adaptive algorithm

Figure 8: Performance comparison of the SMOMAB algorithm with the adaptive-SMOMAB algorithm on 5-objective, 20-armed problem. The weights are set using the adaptive algorithm.

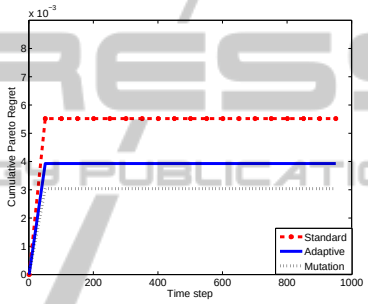


Figure 9: Performance comparison of the standard, the adaptive, and the genetic algorithms on 5-objective, 20-armed problem using the adaptive-SMOMAB algorithm.

and adaptive algorithms are increased, where the cumulative Pareto regret is decreased.

Discussion. from the above experiments, we see that with minimum number of objectives D, $D = 2$, the SMOMAB algorithm performs better than the adaptive-SMOMAB algorithm. While, for number of objectives D, $D > 2$ larger than 2, the performance of the adaptive-SMOMAB algorithm is better than the performance of the SMOMAB algorithm. As the number of objectives is increased, the performance of the adaptive-SMOMAB algorithm increases. We also see that, for small number of objectives D, $D = 2$, the standard algorithm performs better than the adaptive and the genetic algorithms using the adaptive-SMOMAB algorithm. While as the number of objectives is increased, the adaptive and genetic algorithms perform better than the standard algorithm using the adaptive-SMOMAB algorithm. Where, the adaptive algorithm performs better than the genetic algorithm for number of objectives equals 3 and the genetic algorithm performs better than the adaptive algorithm for number of objectives equals 5. The intuition is that for small number of objectives D, $D = 2$ and small number of arms $|A|$, $|A| = 6$, the small num-

ber of scalarized functions S, $|S| = D+1$ was able to identify almost all the optimal arm in the Pareto front A^*. While, for large number of objectives D, $D \geq 2$ and large number of arms $|A|$, $|A| = 20$, the adaptive and genetic algorithms generate new weights, and the new weights explore all the optimal arms in the Pareto front A^*. We also see that the figures have a flat performance and this is because of the calculation of the Pareto regret. Pareto regret adds minimum distance (virtual distance) to the selected suboptimal arm to create an optimal arm, therefore the added distance will be the same if the suboptimal arms are close to each other.

7 CONCLUSIONS

We presented multi-objective, multi-armed bandit (MOMAB) problem and the regret measures in the MOMAB. We presented the linear scalarized function and the linear scalarized-KG that transform the multi-objective problem into a single problem by summing the weighted objectives to find the optimal arms. Usually, the scalarized multi-objective (SMOMAB) algorithm selects uniformly at random the scalarized function s. We proposed to use techniques from the multi-objective optimization in the SMOMAB algorithm to adapt the weights online. We use the genetic operators to generate new weights in the proximity of the current weight sets, and we adapt the weights to be perpendicular on the set of Pareto optimal solutions. We propose the adaptive scalarized multi-armed bandit (adaptive-SMOMAB) algorithm that uses Thompson sampling policy to select the scalarized s. We experimentally compared the SMOMAB algorithm and the adaptive-SMOMAB algorithm using the proposed algorithms: the fixed weights, the adaptive weights, and the genetic weights. We conclude that when the number of objective D is increased $D > 2$, the adaptive-SMOMAB performs better than the SMOMAB algorithm. The adaptive and the mutation algorithms perform better than the standard algorithm (fixed weights).

REFERENCES

Das, I. and Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69.

Drugan, M. (2013). Sets of interacting scalarization functions in local search for multi-objective combinatorial optimization problems. In *IEEE Symposium Series on Computational Intelligence (IEEE SSCI)*.

Drugan, M. and Nowe, A. (2013). Designing multi-objective multi-armed bandits algorithms: A study. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

Drugan, M. and Thierens, D. (2010). Geometrical recombination operators for real-coded evolutionary mcmcs. *Evolutionary Computation*, 18(2):157–198.

Eichfelder, G. (2008). *Adaptive Scalarization Methods in Multiobjective Optimization*. Springer-Verlag Berlin Heidelberg, 1st edition.

I. O. Ryzhov, W. P. and Frazier, P. (2011). The knowledge-gradient policy for a general class of online learning problems. *Operation Research*.

J. Dubois-Lacoste, M. L.-I. and Stutzle, T. (2011). Improving the anytime behavior of two-phase local search. In *Annals of Mathematics and Artificial Intelligence*.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley and Sons, New York, USA, 1st edition.

S. Q. Yahyaa, M. D. and Manderick, B. (2014a). Empirical exploration vs exploitation study in the scalarized multi-objective multi-armed bandit problem. In *International Joint Conference on Neural Networks (IJCNN)*.

S. Q. Yahyaa, M. D. and Manderick, B. (2014b). Knowledge gradient for multi-objective multi-armed bandit algorithms. In *International Conference on Agents and Artificial Intelligence (ICAART)*, France. International Conference on Agents and Artificial Intelligence (ICAART).

S. Q. Yahyaa, M. D. and Manderick, B. (2014c). Multivariate normal distribution based multi-armed bandits pareto algorithm. In *the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. In *Biometrika*.

Zitzler, E. and et al. (2002). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7:117–132.