Multi-Objective $\mathcal{X}$-Armed Bandits

Kristof Van Moffaert, Kevin Van Vaerenbergh, Peter Vrancx and Ann Nowé

***Abstract*— Many of the standard optimization algorithms focus on optimizing a single, scalar feedback signal. However, real-life optimization problems often require a simultaneous optimization of more than one objective. In this paper, we propose a multi-objective extension to the standard $\mathcal{X}$-armed bandit problem. As the feedback signal is now vector-valued, the goal of the agent is to sample actions in the Pareto dominating area of the objective space. Therefore, we propose the multi-objective *Hierarchical Optimistic Optimization* strategy that discretizes the continuous action space in relation to the Pareto optimal solutions obtained in the multi-objective objective space. We experimentally validate the approach on two well-known multi-objective test functions and a simulation of a real life application, the filling phase of a wet clutch. We demonstrate that the strategy allows to identify the Pareto front after just a few epochs and to sample accordingly. After learning, several multi-objective quality indicators indicate that the set of sampled solutions by the algorithm very closely approximates the Pareto front.**

I. Introduction

Many engineering applications inherently consist of multiple, possibly conflicting, objectives. Generally, multi-objective problems are hard because it is not clear how the objectives influence each other, i.e. what the effect is of optimizing one objective over the other objectives. Moreover, often trade-offs between the different objectives must be made, as it is generally not possible to have optimal strategies for all objectives at the same time. For example, in waste water treatment plants contaminated water is purified by a sequence of electrical aeration pumps. Depending on the amount of aeration, the water quality reaches a certain level and sludge is produced. Therefore, the goal of the engineer is not only to maximize the water quality, but also to minimize the operational costs of the plant and the amount of sludge being produced.As these objectives are conflicting, there usually exists no single optimal solution. In such cases, we are interested in a set of trade-off solutions between the objectives. More precisely, we want to obtain the set of best trade-off solutions, i.e. the set of solutions that Pareto dominate the other solutions and are incomparable amongst each other. i.e. a solution x_1 is said to strictly dominate another policy x_2, if each objective of x_1 is not strictly less than the value for the corresponding objective of x_2 and at least one objective of x_1 is strictly greater than the corresponding objective of x_2. The set of non-dominated policies is also referred to as the *Pareto front*.

Kristof Van Moffaert, Kevin Van Vaerenbergh, Peter Vrancx and Ann Nowé are with the Department of Computer Science, Vrije Universiteit Brussel, Brussels, Belgium (email: {kvmoffae, kevvaere, pvrancx, anowe}@vub.ac.be). The first two authors have equal contribution to this paper.

In essence, there are two main approaches for dealing with multi-objective problems. The simplest way is to use a scalarization function to transform the multi-objective problem to a single-objective problem, which can be solved by traditional techniques. This approach is called a *single-policy* mechanism as each run only converges to a single solution. In order to find a set of trade-off solutions, the approach would have to be repeated with different scalarization weights without having any guarantees that this process will result in a good approximation of the Pareto front. Examples of this class are the algorithms by [1], [2] and [3] that use scalarization functions in combination with preference parameters in order to converge to a single-solution that is optimal with regard to the desired preference.

Another class of algorithms are *multi-policy* algorithms. In contrast to focussing only a single solution at a time, a multi-policy algorithm searches for a set of solutions at once. A well-known example of this class of algorithms are evolutionary multi-objective (EMO) algorithms, such as SPEA2 [4] and NSGA-II [5], which evolve a population of multi-objective solutions. These algorithms are amongst the most powerful algorithms for multi-objective optimization in static environments where the sampling cost is not one of the main issues.

In this paper, we focus on bandit problems for multi-objective environments. In the standard bandit setting, a gambler is faced with a set of slot machines, i.e. arms, and he has to decide which lever to pull at each time step [6]. Upon playing a particular arm, a reward is drawn from that action's specific distribution and the goal of the gambler is to maximize his long-term scalar reward. In the case of a multi-objective bandit, the gambler is provided with a vector of feedback signals, one for each objective. Hence, the goal of the gambler is to sample actions that yield rewards located in the Pareto front. So far, multi-objective bandit problems have particularly been focussing on a discrete set of arms [7]. In this paper, we tackle the optimization of a multi-objective $\mathcal{X}$-armed bandit problem, i.e. a bandit with a continuous, but finite-countable, action space. More precisely, we extend the respected *Hierarchical Optimistic Optimization* for multi-objective optimization.

The outline of the paper is as follows. In Section II we briefly present single-objective reinforcement learning and $\mathcal{X}$-armed bandits. Subsequently, in Section III, we present our novel multi-objective algorithm for multi-objective $\mathcal{X}$-armed bandit problems and conduct an in-depth discussion. In Section IV, we empirically evaluate the algorithm on a series of test functions and real-life application simulating the filling phase of a wet clutch. Finally, in Section V we form conclusions and present outlooks for future research.

II. $\mathcal{X}$-Armed Bandits

In the classical bandit problem, a gambler wants to maximize his reward by selecting the best possible action from a finite set of arms with unknown reward distributions. The on-line aspect is very important, i.e. in order to maximize his gain the gambler needs to find a balance between *exploring* uncharted territory and *exploiting* his current knowledge. Thus, by maximizing the cumulative pay-off the total regret is minimized, i.e the difference between the total cumulative pay-off of the gambler and the best possible arm of each round is minimized.

Recently, this problem was generalized to environments where the action space is continuous but finite-dimensional, i.e. the gambler is faced with an continuous set of arms. This problem is called an $\mathcal{X}$-armed bandit [8] problem and has a strong resemblance to control problems which involve the tuning of several parameters. More precisely, a stochastic bandit problem $\mathcal{B}$ consists of a pair of $\mathcal{B} = (\mathcal{X}, M)$, where $\mathcal{X}$ is a measurable space of arms and M determines the distribution of rewards associated with each arm [8].

One of the algorithms that effectively solves a $\mathcal{X}$-armed bandit problem is the *Hierarchical Optimistic Optimization* (HOO) strategy [8]. As we are interested in sampling near the maxima of the pay-off function, the HOO strategy focusses on estimating this function near its maxima while leaving the other, less interested parts of the function, less explored. Internally, the HOO strategy comprises a binary tree that is incrementally refined. A node n in the tree represents an area over the action space $\mathcal{X}$ and is represented by (h, i), where i is the index of node n at depth h. Hence, the root node is represented by $(0, 1)$. Following the terminology of [8], $(h + 1, 2i - 1)$ and $(h + 1, 2i)$ refer to the child nodes of the node located at (h, i). Let $\mathcal{P}_{h,i} \subset \mathcal{X}$ be the area of the action space corresponding to node (h, i), given these two conditions:

$$\mathcal{P}_{0,1} = \mathcal{X} \tag{1}$$

$$\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h+1,2i}, \text{for all } h \geq 0 \text{ and } 1 \leq i \leq 2^h \tag{2}$$

A node stores an optimistic estimate of the quality of its subtree, i.e. a U-value. The $U_{h,i}(n)$-value is an initial estimate of the maximum value of the pay-off function in the corresponding region of the action space $\mathcal{X}$:

$$\mathcal{U}_{h,i}(n) = \begin{cases} \hat{\mu}_{h,i}(n) + \sqrt{\frac{2 \ln n}{\mathcal{T}_{h,i}(n)}} + \nu_1 \rho^h, \text{if } \mathcal{T}_{h,i}(n) > 0 \\ +\infty, \text{otherwise} \end{cases}$$

When the action was not sampled before, it receives the highest optimistic value, i.e. $+\infty$. In the other cases, two terms are added to the mean rewards. The second term is the standard exploration term, while the third term takes into consideration the maximum possible variation of the pay-off function. Given the U-value of a node n, its B-value is computed by taking into account its own U-value and the B-values of its two child nodes. The B-value is designed to put a tighter upper-bound on the best possible value that can be obtained in that region of $\mathcal{X}$. The $\mathcal{B}$-value estimate is:

$$\mathcal{B}_{h,i} \leftarrow \min\{\mathcal{U}_{h,i}, \max\{\mathcal{B}_{h+1,2i-1}, \mathcal{B}_{h+1,2i}\}\} \tag{3}$$

Thus, the root node stores an (optimistic) estimate of the quality of the entire action space, where its left child stores a more accurate estimate for the interval $[\inf \mathcal{X}, \frac{\sup \mathcal{X}}{2}]$ of the action space, i.e. between the infimum of $\mathcal{X}$ and half of its supremum given the divide-and-conquer approach. Similarly, the right child of the root stores equivalent information for $[\frac{\sup \mathcal{X}}{2}, \sup \mathcal{X}]$. Selecting an action is performed by traversing the subtree with the largest B-value until a leaf value is reached. Thereafter the action is sampled and the estimates of the nodes on the traversed path are updated and refined by considering the observed reward and the number of times the subtree was visited in previous plays. Bubeck et al. prove this algorithm to converge to the mean pay-off function around its maxima if the function is weakly *Lipschitz*.

III. Multi-Objective $\mathcal{X}$-Armed Bandits

Transforming the single-objective HOO strategy to multi-objective environments requires several adjustments to the algorithm's internal workings. Below, we tackle the required modifications for constructing the Multi-Objective Hierarchical Optimistic Optimization (MO-HOO) algorithm:

Sampling purpose. A first crucial aspect of transforming a single-objective on-line algorithm to a multi-objective environment is defining its purpose. In the standard problem the goal of the agent is to maximize the average reward being received, which is the logical approach as there is a total order between the scalar rewards. Dealing with multiple objectives, with the Pareto dominance relationship, there only is a partial order, i.e. only dominating and dominated vectors can be compared but for instance two Pareto dominating vectors are *incomparable*. As a consequence, if one would average the reward vectors being obtained by the agent, we are not guaranteed that the average is an actual attainable solution for a problem. Take for instance, a very easy multi-objective problem where the agent can only take two action a and b where the deterministic rewards are $[1, 0]$ and $[0, 1]$, respectively. If one would take action a with probability x and action b with probability $(1 - x)$, the average reward vector would be $[x, 1 - x]$. Thus, although there are only two possible outcomes for the problem, considering average reward vectors implicates that we are no longer sampling on the Pareto front but on the convex hull of the Pareto front . The convex hull is an infinite set of points that can be expressed as convex combinations of the points in the Pareto front. Therefore the hull does not represent actual solution points in the objective space.

In control problems we are not interested the stochastic *mixture* policies obtained from the convex hull of the Pareto front, but rather in sampling a trade-off in every execution of the policy. This means that we want to identify actually realizable control policies that lie on the Pareto front and thereby offer a fixed trade-off between the different control objectives. However, despite the fact that our main goal is to

identify the Pareto front, we do not want to neglect the sampling efficiency of our method. Our motivating assumption here is that sampling policies inherently has a certain cost associated with it, either in time, in resources or both. Often, these costs are associated with evaluating a control strategy on a real system. Furthermore, we assume that solutions which lie far from the Pareto front, are suboptimal in at least one of the target objectives and thus typically have higher costs associated with them, e.g. because they are less efficient, take longer to reach a goal, or might even violate safety constraints. Therefore, our goal is to develop an efficient method for identifying the Pareto front, which minimizes the the amount of sampling done in regions far from the Pareto front.

The shift from simple on-line optimization of the average reward to identification of the Pareto front requires some fundamental adaptations to the HOO approach. Since our main interest is now the sampling of the Pareto optimal solutions, rather than from the convex hull of the front, we no longer consider the U-values in their original form. Recall that the U-value of a node represents the average quality for its subtree. Therefore, in the multi-objective case, we only propagate the U-vector of leaf nodes as they are only (average) samples that are not averaged out over subtrees.

Knowledge representation. Recall that in the single-objective HOO strategy, the agent's goal is to maximize the scalar reward obtained by sampling close to the optimum of the pay-off function f. Therefore, the algorithm propagates an estimate of the maximum of f from leaf nodes to the root node in terms of B-values. In a multi-objective problem, however, there usually is no single-optimum but there are multiple Pareto optimal solutions that are *incomparable*. Thus, a scalar estimate or B-value is insufficient to store the quality of a region of f. The solution we propose is to consider sets of estimates or B-sets. The elements of the sets are $\mathbf{B}$-vectors that store an estimate for each objective. The most optimistic upper bound is now a vector $\mathbf{u}$ that is always a member of the Pareto front without excluding other Pareto dominating vectors, e.g. $\mathbf{u} = [-\infty, +\infty]$ for an environment with two objectives.

Information propagation. In Eq. 3, the HOO strategy performs a backward computation to adjust the estimate of a node by taking into account its U-value and the B-value of its two child nodes. To determine the B-value of a node, the algorithm determines the highest upper bound of its child nodes by a $\max$ operator. In the multi-objective variant, we replace this operator by the operator $ND(\bigcup_{\mathbf{B}}(\mathbf{B}_{h+1,2i-1}(n), \mathbf{B}_{h+1,2i}(n))$ which yields every non-dominated B-vector of the left and right child of node n. The $\min$ operator in Eq. 3 assures a tighter bound of the B-value of node n. As we are not focussing on the average Pareto front or convex hull, we no longer consider the U-value of non-leaf nodes. Therefore, we omit a $\min$ operator in our MO-HOO implementation. If one would be interested in the average sampling performance, a multi-objective $\min$ operator should be proposed, which is far from trivial. The specification of a multi-objective $\min$ operator could place a stricter bound on the $\mathbf{B}$-vectors in order to make them less optimistic. However, defining such a $\min$ operator that would work in a multi-objective setting with multiple incomparable solutions is currently an open question.

Pseudo code. An algorithmic outline of the MO-HOO strategy for an m-objective problem can be found in Algorithm 1.

IV. Experiments

The experimental evaluation of the MO-HOO algorithm is divided into two parts. First, we analyze the strategy on two common multi-objective test functions which allow us to compare the obtained solutions to the actual solutions in the Pareto front. We then gradually increase the difficulty of the test environments to highlight the properties of the MO-HOO strategy.

A. Schaffer 1 function

The first test function is the bi-objective Schaffer no 1 function [9] that is defined by

$$\text{maximize } f(x) = \begin{cases} f_1(x) = -x^2 \\ f_2(x) = -(x-2)^2 \end{cases}$$

where $x \in [-10, 10]$.[1] The function is depicted in Fig. 2 (a) where the convex Pareto front is indicated by green dots.

In Fig. 2 (b) the sampling performance of the MO-HOO strategy is compared to a random sampling strategy. Recall that our goal is to sample as fast as possible as close as possible on the Pareto front. As the Pareto front is in this case continuous, we determine whether the vector valued return of the sampled action is within a distance of ϵ of a discrete approximation of the true Pareto front [2]. This discrete approximation of the true Pareto front was obtained by collecting the Pareto optimal outcomes of many independent random runs of the algorithm. From the figure, we note that the MO-HOO strategy gradually increases its performance by sampling closer to the Pareto front as the learning time increases. After 500 epochs, the probability of sampling a Pareto optimal action approaches 1.

In Fig. 2 (c) and (d), we plot three visualization graphs for random and MO-HOO strategy, respectively. In a first subplot, we denote the sample in the normalized action space. We clearly see the exploration-exploitation of the MO-HOO strategy. In a second subplot, we show the depth of the tree in relation to the action space. Subtrees that are more expanded have a greater depth and focus on a finer part of the action space compared to small subtrees. Where we see that the random strategy expands each subtree uniformly, the MO-HOO strategy roughly explores some parts of the actions space, whereafter it focusses on the region corresponding by the Pareto optimal area in the objective space. The third

[1] Originally, these test functions were minimization problems but we negated them to obtain a maximization problem as this is what our reinforcement learning algorithm assumes.

[2] In our experiments, ϵ was set to 0.2.

Algorithm 1: The MO-HOO strategy
Parameters: Two real numbers $\nu_1 > 0$ and $\rho \in (0,1)$, a sequence $(\mathcal{P}_{h,i})_{h\geq 0, 1\leq i\leq 2^h}$ of subsets of $\mathcal{X}$ satisfying the conditions (1) and (2).
Auxiliary function LEAF($\mathcal{T}$): outputs a leaf of $\mathcal{T}$.
Initialization: $\mathcal{T} = \{(0,1)\}$ and $\mathbf{B}_{1,2} = [+\infty, \ldots, -\infty]$ and $\mathbf{B}_{2,2} = [-\infty, \ldots, +\infty]$. ▷ Use combination of ∞ to make incomparable m-dimensional vectors

for $n = 1, 2, \ldots$ **do** ▷ Strategy MO-HOO in round $n \geq 1$
  $(h,i) \leftarrow (0,1)$ ▷ Start at the root
  $P \leftarrow \{(h,i)\}$ ▷ P stores the path traversed in the tree
  while $(h,i) \in \mathcal{T}$ **do** ▷ Search the tree $\mathcal{T}$
    $\mathbf{B}_{\text{random}} \in \mathbf{B}_{h,i}$ ▷ Select random Pareto dominant $\mathbf{B}$-vector
    if $\mathbf{B}_{\text{random}} \in \mathbf{B}_{h+1,2i-1}$ **then**
      $(h,i) \leftarrow (h+1, 2i-1)$
    else
      $(h,i) \leftarrow (h+1, 2i)$
    end if
    $P \leftarrow P \cup \{(h,i)\}$
  end while
  $(H,I) \leftarrow (h,i)$ ▷ The selected node
  Choose arm X in $\mathcal{P}_{H,I}$ and play it ▷ Arbitrary selection of an arm
  Receive corresponding reward vector $\mathbf{Y}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{(H,I)\}$ ▷ Extend the tree
  $\hat{\mu}_{H,I} \leftarrow \big(1 - 1/T_{H,I}\big)\hat{\mu}_{H,I} + \mathbf{Y}/T_{H,I}$ ▷ Calculate the mean vector $\hat{\mu}_{\mathbf{H},\mathbf{I}}$ of new added leaf (H,I)
  for all $(h,i) \in \mathcal{T}$ **do** ▷ Update the statistics U stored in the tree
    $\mathbf{U}_{h,i} \leftarrow \hat{\mu}_{h,i} + \sqrt{(2\ln n)/T_{h,i}} + \nu_1 \rho^h$ ▷ Update the U–value set of node (h,i)
  end for
  $\mathbf{B}_{H+1,2I-1} \leftarrow [+\infty, \ldots, -\infty]$ ▷ m-dimensional incomparable $\mathbf{B}$–vectors for the children of the new leaf
  $\mathbf{B}_{H+1,2I} \leftarrow [-\infty, \ldots, +\infty]$
  $\mathcal{T}' \leftarrow \mathcal{T}$ ▷ Local copy of the current tree $\mathcal{T}$
  while $\mathcal{T}' \neq \{(0,1)\}$ **do** ▷ Backward computation of the B–values
    $(h,i) \leftarrow \text{LEAF}(\mathcal{T}')$ ▷ Take any remaining leaf
    if $\mathbf{B}_{h,i}$ *is* LEAF **then**
      $\mathbf{B}_{h,i} \leftarrow \mathbf{U}_{h,i}$
    else
      $\mathbf{B}_{h,i} \leftarrow ND \bigcup_{\mathbf{B}} \big(\mathbf{B_{h+1,2i-1}}, \mathbf{B_{h+1,2i}}\big)$ ▷ Backward computation of non-dominated $\mathbf{B}$'s of children
    end if
    $\mathcal{T}' \leftarrow \mathcal{T}' \setminus \{(h,i)\}$ ▷ Drop updated leaf (h,i)
  end while
end for

Fig. 1. The MO-HOO strategy

subplot depicts the sampled points in the objective space. The color of the points indicate the epoch number, where red dots denote points obtained in later epochs. The random strategy explores the entire objective space where MO-HOO samples almost exclusively the Pareto front.In Table I, we show the result for both approaches using two multi-objective quality indicators.

1) *Cardinality* This quality indicator filters the set of obtained solutions Y using the non-dominated (ND) operator and counts the number of solutions that are retained, i.e.

$$Cardinality(Y) = |ND(Y)| \tag{4}$$

The higher the cardinality, the larger the set of compromise solution the user can choose from.

2) *Inverted Generational Distance* (IGD) [16]. This quality indicator measures the average Euclidean distance d from a set of obtained solution vectors Y to the true Pareto front X:

$$IGD(X,Y) = \frac{1}{|Y|}\Big(\sum_{i=1}^{|Y|} d(y_i, X)^2\Big) \tag{5}$$

The lower the IGD value, the closer the obtained elements are to the Pareto front.

Based on the results, we note MO-HOO acquires more than 6 times the amount of Pareto optimal points obtained by the

random method and the total set of sampled points is much closer to the Pareto front.

	Random	MO-HOO
Cardinality	69.5	**436.2**
IGD	0.0012	**0.0005**

TABLE I

QUALITY INDICATORS ON THE SCHAFFER 1 FUNCTION FOR RANDOM EXPLORATION AND MO-HOO. THE GOAL IS TO MAXIMIZE THE NUMBER OF PARETO OPTIMAL POINTS AND TO MINIMIZE THE IG DISTANCE.

B. Fonseca and Flemming function

The second problem is the Fonseca and Fleming function: [10].

$$\text{maximize } f(\mathbf{x}) = \begin{cases} f_1(\mathbf{x}) = -(1 - e^{-\sum_{i=1}^{n}(\mathbf{x}_i - \frac{1}{\sqrt{n}})^2}) \\ f_2(\mathbf{x}) = -(1 - e^{-\sum_{i=1}^{n}(\mathbf{x}_i + \frac{1}{\sqrt{n}})^2}) \end{cases}$$

where $\mathbf{x}$ is a two-dimensional input vector with $\mathbf{x}_i \in [-4, 4]$ for objective i. This bi-objective function has an entirely non-convex and large Pareto front which makes it especially appealing for multi-objective optimisation algorithms to test their ability to find a close and uniform approximation of the Pareto front. The function and its corresponding Pareto front can be found in Fig. 3 (a). To test the effectiveness of the MO-HOO strategy on noisy environments, we also added Normal noise with $\sigma = 0.1$ to the values of each of the objectives.

In Fig. 3 (b) we see that also on the harder Fonseca and Flemming function the percentage of ϵ-Pareto optimal actions increases significantly over time. When comparing the random strategy to MO-HOO in Fig. 3 (c) and Fig. 3 (d), respectively, we see the MO-HOO tree focussing on the optimal part of the action space and gradually increasing its sampling accuracy in later learning epochs.

The cardinality and IGD indicators in Table II also denote MO-HOO's improved performance on this test function.

C. Wet clutch

Finally, we test the MO-HOO strategy on a motivation example of a multi-objective control problem. To do this we use a realistic simulation environment of a wet clutch setup, i.e. a clutch where the friction plates are immersed in oil, in order to smooth the transmission and increase the lifetime of the plates. Wet clutches are typically used in heavy duty transmission systems, such as those found in off-road vehicles and tractors. Figure 4 offers a schematic representation

	Random	MO-HOO
Cardinality	8.1	**16.22**
IGD	0.0056	**0.0019**

TABLE II

QUALITY INDICATORS ON THE FONSECA AND FLEMMING FUNCTION FOR RANDOM EXPLORATION AND MO-HOO.

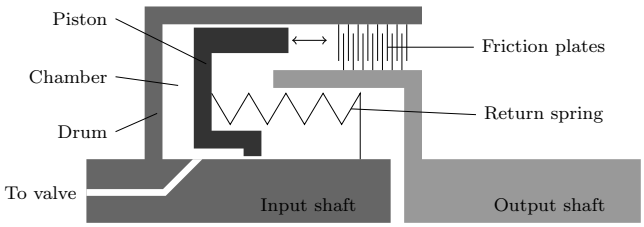


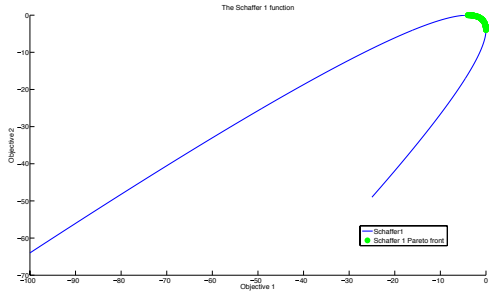
Fig. 4. Schematic representation of a wet clutch (from [11]).

of its functioning. Two sets of friction plates are connected to the input and the output shaft, respectively, such that, when they are pressed together, the torque generated by the motor, connected to the input shaft via a set of gears, is transmitted to the output shaft, which is connected to the wheels of the vehicle. The two sets of plates are free to translate axially, and a return spring keeps the clutch open: to close the clutch, the plates are pressed together by a hydraulic piston, which can be pushed against the plates by increasing the pressure of the oil in the clutch chamber. This pressure can in turn be controlled by the current input of an electromechanical servo-valve: by opening the valve, the clutch and supply line are filled up with oil, and the pressure increases until it is high enough to overcome the return spring force. As a result, the piston starts to move towards the plates. During this first part of the engagement, called the *filling phase*, no torque is transferred. The transmission only commences as the piston makes contact with the plates. The clutch then enters the *slip phase*, as the slip, defined as the difference in rotational speeds between the shafts, decreases. When the pressure is high enough, the output shaft is accelerated until it rotates synchronously with the input, and the slip reaches zero.

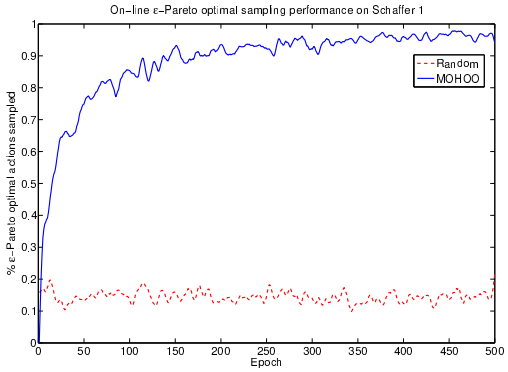
There currently is no reliable model of the whole engagement: an approximate model[3] is available for the filling phase, until the piston touches the plates, but not for the following slip phase, which would allow to simulate and optimize the resulting torque loss. However, it has been observed that the torque loss depends on the speed of the piston when this reaches the plates: the lower the speed, the lower the torque loss, and the smoother the engagement. In other words, there is a trade-off among the two objectives: on one hand, a very slow piston movement will result in a smooth engagement, which takes a long time; on the other hand, a fast piston will engage in a short time, but it will also cause a jerky engagement, with a large torque dip, and possibly damage the setup. These two objectives are conflicting and make a good testbed for our multi-objective technique.

A parametric form of such a signal has been adopted in [12], for a genetic-based optimization approach, and it is displayed in Fig. 5. First, a large current pulse is sent to the valve, in order to generate a high pressure level in the oil chamber of the clutch, which will allow the piston

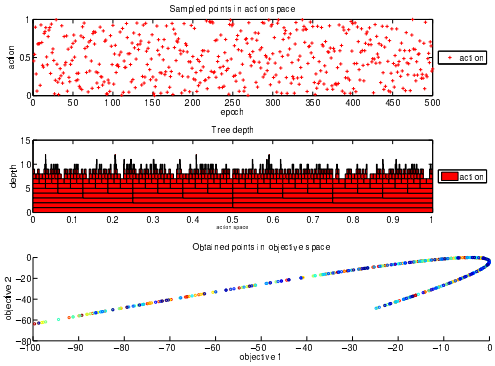
[3]Model developed in Simulink® by Julian Stoev, Gregory Pinte (FMTC), Bert Stallaert and Bruno Depraetere (PMA, Katholieke Universiteit Leuven).



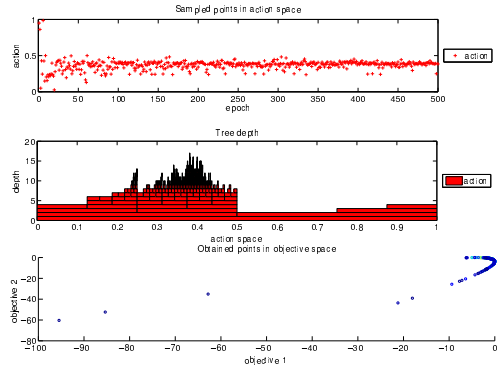
(a) The Schaffer 1 objective space



(b) Percentage optimal actions



(c) Sampling performance of random selection



(d) Sampling performance of MO-HOO

Fig. 2. Schaffer 1 results. Subfigure (a) denotes the objective space of the Schaffer 1 function and its Pareto front. In (b), the sampling performance MO-HOO is compared to a naive random strategy, while additional information on the size of the three and the sampled points in the action space are presented in (c) and (d), respectively.

to overcome the resistance of the preloaded return spring, and accelerate towards the friction plates. After this pulse, a lower constant current is sent out, in order to decelerate the piston as it approaches the friction plates: before reaching this low value, the current decreases shortly to an even lower value, creating a small dent in the signal, which should act as a "brake", limiting the speed of the piston. Finally, the current signal grows following a ramp with a low slope, closing the clutch smoothly. The signal is parametrized as follows: the first parameter (a) controls the duration of the initial peak, whose amplitude is fixed at the maximum level, while the last (d) controls the low current level, just before the engagement begins. The remaining parameters (b, c) control the shape of the "braking" dent, while the final slope during the engagement phase is fixed. The above mentioned parametric input signal is created to be used for the whole engagement of the clutch. Focussing only on the first phase of the engagement (filling phase), we simplify the signal to a 2-dimensional signal were the width of the first pulse (a-parameter) and the height of the "braking" dent (c-parameter) are adjustable. The remaining parameters are respectively set to 0 and the value of the c-parameter.

The results on the wet clutch environment can be found below. In contrast to the test functions, where the identification of the true Pareto front was relatively easy, this is not the case for the wet clutch environment. In Fig 6 (a), we denote the obtained points in the objective space after a parameter sweep. We normalized the objectives and transformed both into objectives to be maximized. We note that many of the parameters lead to policies located in clustered suboptimal areas of the objective space. The optimal policies, which are located in the top-right corner of the objective space, are

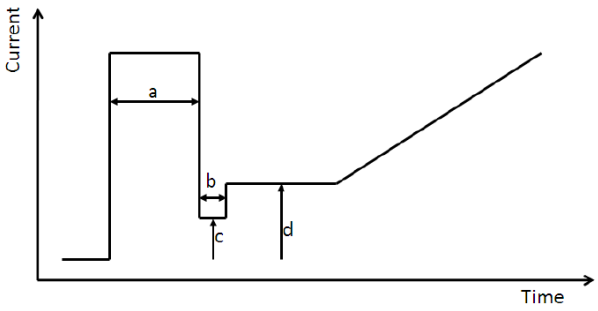
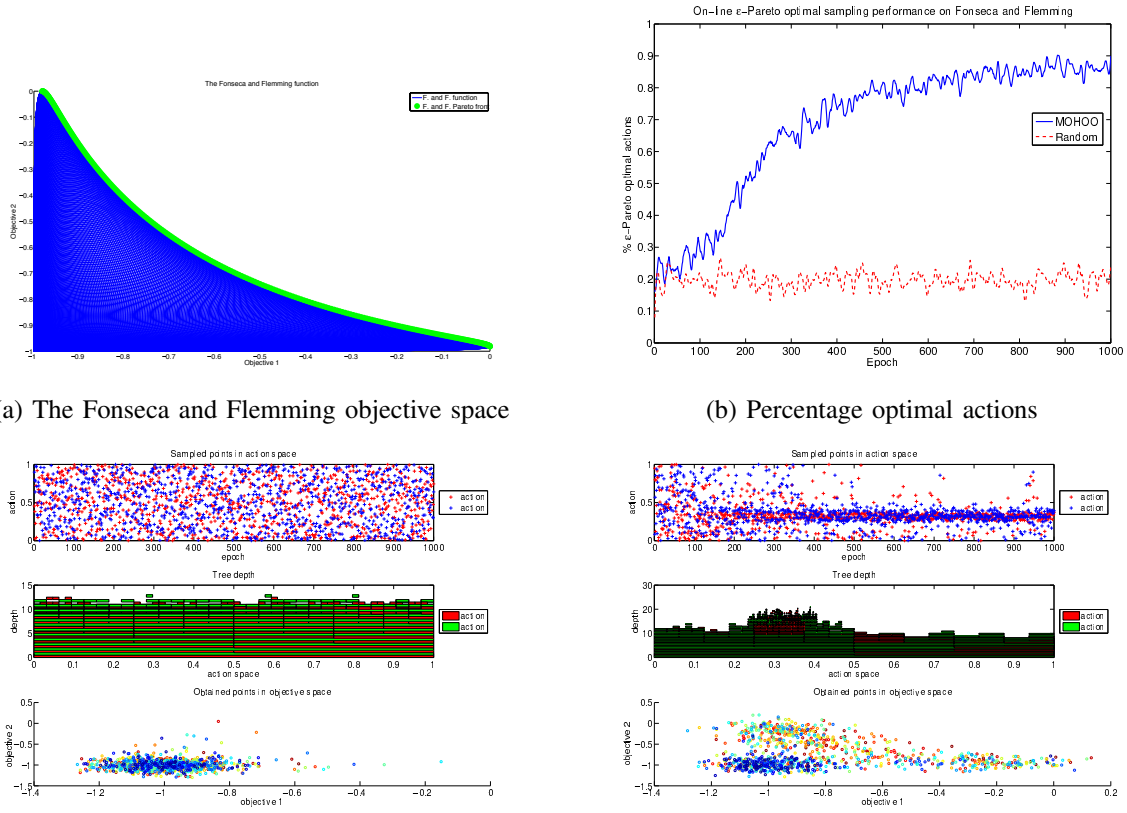


Fig. 5. Parametric input signal from [12], with four parameters (a, b, c, d). In our implementation, all parameter ranges are mapped to the unit interval $[0, 1]$.



(a) The Fonseca and Flemming objective space

(b) Percentage optimal actions

(c) Sampling performance of random selection

(d) Sampling performance of MO-HOO

Fig. 3. Fonseca and Flemming results. Subfigure (a) denotes the objective space of the Fonseca and Flemming function and its Pareto front. In (b), the sampling performance MO-HOO is compared to a naive random strategy, while additional information on the size of the three and the sampled points in the action space are presented in (c) and (d), respectively.

much harder to find as small changes in parameter values imply big gaps between the policies. As a consequence, our exhaustive sweep only allowed to obtain 8 Pareto optimal policies, denoted in blue, of the continuous Pareto front. This is an indication of the level of difficulty of the problem.

In Fig. 6 (b), we depict the average percentage of ϵ-Pareto optimal actions selected by both the random and MO-HOO strategy. On this hard environment, randomly sampling the action space almost never leads to a Pareto optimal action where the MO-HOO strategy approaches 55%. As denoted by Fig. 6 (d), we note that the algorithm is not converged yet as no clear action emerged after 5000 epochs. In Fig. 4 (c), we see that randomly sampling the action space only focusses on the dominated part of the objective space while MO-HOO is able to escape the local optima and approach the true Pareto front within the limited epochs. Eventually, in Table III, MO-HOO obtained 222 Pareto optimal solutions on average.

	Random	MO-HOO
Cardinality	7.14	**222.56**
IGD	0.0349	**0.0219**

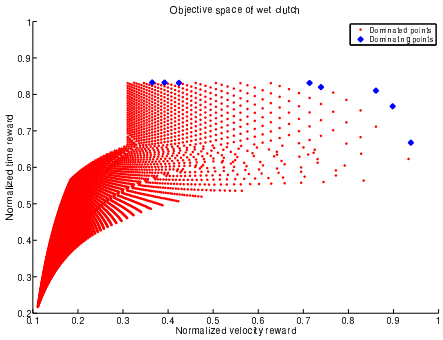
TABLE III

QUALITY INDICATORS ON THE WET CLUTCH ENVIRONMENT FOR RANDOM EXPLORATION AND MO-HOO

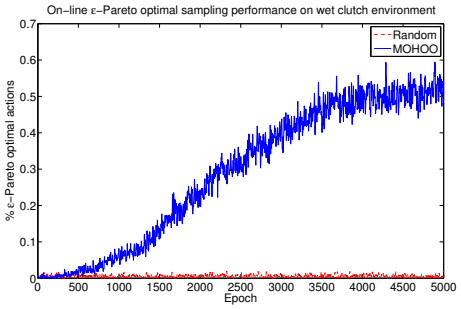
V. CONCLUSIONS

In this paper, we have argued the need for multi-objective optimization algorithms for real-life control applications. Therefore, we have proposed a new problem class, i.e. the multi-objective $\mathcal{X}$-armed bandit, where an agent is faced with a continuous finite-dimensional action space where the goal of the agent is to sample actions in the Pareto front based on a vector-valued feedback signal. Our novel algorithm, the multi-objective hierarchical optimistic optimization strategy builds an infinite binary tree where each node represents an optimistic estimate of the corresponding area in the objective space. More precisely, each node stores a set of optimistic estimates of the Pareto front attainable from sampling its sub-tree. Future research will focus on reducing these optimistic bounds by placing stricter bounds.

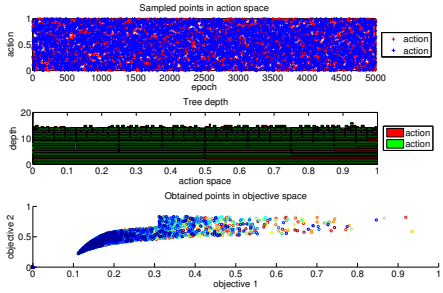
Based on the experimental validation, we have seen that the MO-HOO strategy obtained satisfying results on both the deterministic and noisy test functions and the wet clutch simulation environment. We have seen that after a few hundred epochs the algorithm already focusses on sampling the Pareto front while leaving less interested areas of the objective space untouched. As expected, results indicated



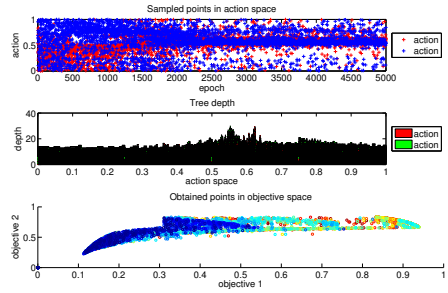
(a) Wet clutch objective space



(b) Percentage optimal actions



(c) Sampling performance of random selection



(d) Sampling performance of MO-HOO

Fig. 6. Wetclutch results. Subfigure (a) denotes the objective space of the Wetclutch simulation environment and its Pareto front. In (b), the sampling performance MO-HOO is compared to a naive random strategy, while additional information on the size of the three and the sampled points in the action space are presented in (c) and (d), respectively.

by several multi-objective quality indicators taught us that the MO-HOO strategy sampled much more Pareto optimal points than a random strategy and the resulting set of sampled points were much closer to the true Pareto front. Most multi-objective techniques work off-line and only identify the Pareto front or sample the convex hull. Therefore we would like to extensively compare the MO-HOO strategy to evolutionary multi-objective approaches such as NSGA-II and SPEA2. After the promising results on the simulation model of the filling phase, we will also analyze the behaviour and the performance of the MO-HOO strategy on the real wet clutch setup.

Acknowledgements

This work has been carried out within the framework of the Perpetual project (grant nr. 110041) of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Together with the Multi-criteria Reinforcement Learning project (grant G.0878.14N) of the Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO).

References

[1] Z. Gabor, Z. Kalmar, and C. Szepesvari, "Multi-criteria reinforcement learning," in *International Conference on Machine Learning (ICML-98)*, Madison, WI, 1998.

[2] K. Van Moffaert, M. Drugan, and A. Nowé, "Scalarized Multi-Objective Reinforcement Learning: Novel Design Techniques," in *2013 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. IEEE, 2013.

[3] S. Mannor and N. Shimkin, "A geometric approach to multi-criterion reinforcement learning," *Journal of Machine Learning Research*, vol. 5, pp. 325–360, 2004.

[4] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, K. Giannakoglou *et al.*, Eds. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

[5] K. D. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm : NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[6] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 952.

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

[8] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits," *CoRR*, vol. abs/1001.4475, 2010.

[9] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 93–100. [Online]. Available: http://dl.acm.org/citation.cfm?id=645511.657079

[10] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization," in *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 416–423. [Online]. Available: http://dl.acm.org/citation.cfm?id=645513.657757

[11] B. Depraetere, G. Pinte, and J. Swevers, "Iterative optimization of the filling phase of wet clutches," in *The 11th International Workshop on Advanced Motion Control (AMC 2010)*. IEEE, 2010, pp. 94–99.

[12] Y. Zhong, B. Wyns, R. D. Keyser, G. Pinte, and J. Stoev, "Implementation of genetic based learning classifier system on wet clutch system," in *30th Benelux Meeting on Systems and Control — Book of Abstracts*. Gent, Belgium: Universiteit Gent, 2011, p. 124.