Pareto Front Identification from Stochastic Bandit Feedback

Peter Auer
Montanuniversität Leoben
auer@unileoben.ac.at

Chao-Kai Chiang
Montanuniversität Leoben
chaokai@gmail.com

Ronald Ortner
Montanuniversität Leoben
rortner@unileoben.ac.at

Madalina M. Drugan
Vrije Universiteit Brussel
madalina.drugan@gmail.com

Abstract

We consider the problem of identifying the Pareto front for multiple objectives from a finite set of operating points. Sampling an operating point gives a random vector where each coordinate corresponds to the value of one of the objectives. The Pareto front is the set of operating points that are not dominated by any other operating point in respect to all objectives (considering the mean of their objective values). We propose a confidence bound algorithm to approximate the Pareto front, and prove problem specific lower and upper bounds, showing that the sample complexity is characterized by some natural geometric properties of the operating points. Experiments confirm the reliability of our algorithm. For the problem of finding a sparse cover of the Pareto front, we propose an asymmetric covering algorithm of independent interest.

1 INTRODUCTION

Multi-objective optimization is a natural extension of single-objective problems and has various applications, also within machine learning (Jin, 2006). Recently, multi-objective optimization has become of increasing importance in reinforcement learning contexts as well, ranging from multi-objective Markov decision processes to algorithms (Drugan et al., 2014) and applications (Moffaert and Nowé, 2014; Lizotte et al., 2010).

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41.

In multi-objective optimization, one is usually interested in the *Pareto front*, the set of all operating points that are not dominated by any other operating point. Thus we consider the sample complexity of identifying the Pareto front in such a multi-objective setting, where the learning algorithm sequentially chooses an operating point to sample, and receives feedback only from this point. This can be seen as an active learning problem, or as a generalization of multi-armed bandit problems, where instead of identifying the single optimal arm, the set of Pareto optimal operating points has to be identified.

Sample complexity bounds for best arm identification in multi-armed bandit problems have recently received significant attention (Audibert et al., 2010; Gabillon et al., 2012; Kaufmann et al., to appear). In our setting, operating points attaining the optimal value for a particular objective are also Pareto optimal, such that in the course of identifying the Pareto front, one also has to determine the best operating point for each objective. Thus, our problem is also related to the multi-armed multi-bandit problem where the learner has to identify the optimal arms in several independent bandit problems (Gabillon et al., 2011; Bubeck et al., 2013). However, our setting is combinatorially more involved, as each sample gives information about all objectives of the chosen operating point, and — more importantly — there are operating points in the Pareto front that are not optimal for any objective.

The question of identifying the Pareto front in a multi-objective bandit setting has been considered before by Zuluaga et al. (2013). However, in this work it is assumed that the operating points are generated by a Gaussian process, which induces a strong regularity that can be exploited by the suggested PAL algorithm and is reflected in the theoretical results. Our algorithm and analysis do not depend on any such assumptions. An empirical comparison of PAL to our

algorithm also shows that PAL sometimes is too optimistic with respect to the regularity of the data. For more details we refer to Section 8.

The goal of this paper is to characterize the sample complexity for finding the Pareto front, and our main contribution is to identify the relevant problem parameters and prove their correctness. We introduce these parameters in Section 2, also providing some intuition about them. As the points on the Pareto front might be indistinguishable from points close to it, we propose two ways to approximate the Pareto front in Section 3. We first consider the problem of finding all Pareto optimal points, possibly together with some additional, almost optimal points. In Section 4, we present an algorithm that returns all points in the Pareto front (and some more close to it), and we give its analysis in Section 5. An almost matching lower bound is shown in Section 6. Then Section 7 considers the natural problem of finding a minimal set of optimal or almost optimal points that cover the Pareto front. Since the underlying distance measure is asymmetric, the algorithm we introduce for that purpose is of independent interest. Finally, Section 8 presents experiments to illustrate the validity of our approach, comparing our algorithm to the PAL algorithm of Zuluaga et al. (2013).

2 PRELIMINARIES

We consider a multi-objective optimization problem with D objectives and a finite set of operating points indexed by $i = 1, \dots, K$. The mean values of the objectives for an operating point i are denoted by $\boldsymbol{y}_i = (y_i^1, ..., y_i^D) \in \mathbb{R}^D$. When an operating point i is sampled, then a random vector $\mathbf{X}_i \in \mathbb{R}^D$ with $\mathbb{E}[\mathbf{X}_i] = \boldsymbol{y}_i$ is received.

Definition 1. *An operating point i is* weakly dominated *by an operating point j (denoted by $i \preceq j$), if $y_i^d \le y_j^d$ for all objectives $d = 1, \dots, D$. An operating point i is* dominated *by an operating point j (denoted by $i \prec j$), if $i \preceq j$ and $y_i^d < y_j^d$ for at least one objective d. An operating point i is* strongly dominated *by an operating point j (denoted by $i \precneqq j$), if $y_i^d < y_j^d$ for all objectives $d = 1, \dots, D$.*

An operating point is called Pareto optimal if it is not dominated, and the set of Pareto optimal points is called the Pareto front,

$$P^* = \{i \,|\, \nexists j : i \prec j\}.$$

Remark 2. *For a sampling based algorithm, the distinction between dominance and weak or strong dominance is immaterial, as a sharp distinction is not possible due to sampling error. We keep this distinction only for formal correctness of the following definitions.*

To measure by how much a point i is dominated by a point j, we define the gap[1]

$$\begin{aligned} m(i,j) &= \min\{s \ge 0 \,|\, \exists d : y_i^d + s \ge y_j^d\} \\ &= \max\{0, \min_d (y_j^d - y_i^d)\} \end{aligned}$$

as the amount by which the objective values of point i have to be increased such that i would not be strongly dominated by j. We have $m(i,j) > 0$ if and only if $i \precneqq j$. The distance of a point i to the Pareto front is then denoted by

$$\Delta_i^* = \max_{j \in P^*} m(i,j).$$

This is the amount by which the objective values of i have to be increased such that i would not be strongly dominated by any other point. If i is Pareto optimal then $\Delta_i^* = 0$. We also define the gap

$$\begin{aligned} M(i,j) &= \min\{s \ge 0 \,|\, \forall d : y_i^d \le y_j^d + s\} \\ &= \max\{0, \max_d (y_i^d - y_j^d)\} \end{aligned}$$

as the amount by which the values of j have to be increased such that i would be weakly dominated by j. We have $M(i,j) = 0$ if and only if $i \preceq j$. The gap function $M(\cdot,\cdot)$ is an asymmetric distance function that satisfies the triangle inequality.

Lemma 3. $M(i,k) \le M(i,j) + M(j,k)$.

Proof. The statement follows since $y_j^d + s \ge y_i^d$ and $y_k^d + t \ge y_j^d$ imply $y_k^d + s + t \ge y_i^d$. □

Based on these gap functions, we characterize the complexity of identifying the Pareto front. We consider the required estimation accuracy for the objective values of an operating point i to determine the Pareto optimality of point i itself and also of other points j. By an estimation error above Δ_i^*, a suboptimal point i may appear Pareto optimal. Thus we define the required accuracy for suboptimal points as

$$\Delta_i = \Delta_i^* \quad \text{for } i \notin P^*. \tag{1}$$

For Pareto optimal points the situation is more complicated. For two Pareto optimal points i and j, i may appear suboptimal if underestimated by $M(i,j)$, and j may appear suboptimal if i is overestimated by $M(j,i)$. Thus we define

$$\Delta_i^+ = \min_{j \in P^* \setminus i} \min\{M(i,j), M(j,i)\}.$$

For a Pareto optimal point i and a suboptimal point j, we consider a slightly modified and Pareto optimal operating point j' that might replace point j and is defined as $y_{j'}^d = y_j^d + 2\Delta_j^*$. By an estimation error of

[1] See appendix in the supplementary material for an illustration of the quantities introduced in this section.

$M(j,i)+2\Delta_j^*$ for point i, the Pareto optimal point j' may appear dominated by point i. As it may be difficult for an algorithm to distinguish between these objective values for j and j', we define

$$\Delta_i^- = \min_{j \notin P^*} [M(j,i) + 2\Delta_j^*].$$

As the required estimation accuracy for a Pareto optimal point i we now define

$$\Delta_i = \min\{\Delta_i^+, \Delta_i^-\}. \tag{2}$$

In Section 5, the estimation accuracy will be used for the analysis of our algorithm. This definition is also reflected in the lower bound of Section 6.

3 SETTING AND PARETO FRONT APPROXIMATIONS

We are interested in the sample complexity of identifying the Pareto front from random samples for a given set of operating points with unknown mean objective values $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K \in \mathbb{R}^D$. The sample complexity is the total number of samples a learning algorithm requests, until it outputs an approximation P of the Pareto front P^*. When operating point i is sampled the n-th time, the algorithm receives a random vector $\mathbf{X}_{i,n} \in \mathbb{R}^D$ with $\mathbb{E}[\mathbf{X}_{i,n}] = \boldsymbol{y}_i$. We assume that the random vectors $\mathbf{X}_{i,n}$ are mutually independent for $i = 1, \ldots, K$ and $n \geq 1$. The coordinates of a vector $\mathbf{X}_{i,n} = (X_{i,n}^1, \ldots, X_{i,n}^D)$ for fixed i and n may be correlated. We also assume that the random variables $X_{i,n}^d - y_i^d$ are 1-subgaussian. This allows to use Hoeffding's inequality in the analysis and holds e.g. when $X_{i,n}$ is Gaussian with variance $\sigma^2 \leq 1$ or if $X_{i,n}$ is bounded as $|X_{i,n}| \leq 1$.

As the learning is subject to sampling noise, it cannot be expected to return the perfectly correct Pareto front. The following example depicts that identifying the Pareto front from samples can be difficult if points are close to the Pareto front.

Example 1. *Consider two operating points with mean value vectors* $\boldsymbol{y}_1 = (\frac{3}{4}, \frac{1}{2})$ *and* $\boldsymbol{y}_2 = (\frac{1}{2}, \frac{3}{4})$. *Then a third point* $\boldsymbol{y}_3 = (\frac{1}{2}+\theta, \frac{1}{2}+\theta)$ *would be on the Pareto front if* $\theta > 0$, *while it would be suboptimal for* $\theta \leq 0$. *If* θ *is very close to 0, then a moderately sized random sample will not suffice to settle the status of the point.*

Therefore, we propose two possible conditions for approximating the Pareto front. For each of the two conditions we provide algorithms (in Sections 4 and 7) that meet these conditions with high probability.

Success Condition 1:
The first condition requires that the algorithm, given the required precision $\epsilon_0 > 0$, outputs a set of operating points P, such that P contains all Pareto optimal points and possibly some suboptimal points that are close to the Pareto front:

(1.a) $P^* \subseteq P$,
(1.b) $\forall i \in P : \Delta_i^* \leq \epsilon_0$.

Success Condition 2:
The second condition requires, given $\epsilon_0 > 0$ and $\epsilon > 0$, a sparse cover P of the Pareto front, such that all Pareto optimal points are close to a point in P, and all points in P are distant from each other:

(2.a) $\forall i \in P^* \; \exists j \in P : M(i,j) \leq \epsilon_0$,
(2.b) $\forall i \in P : \Delta_i^* \leq \epsilon_0$,
(2.c) $\forall i, j \in P$ with $i \neq j : M(i,j) \geq \frac{\epsilon_0}{2} - \epsilon$.

It is no accident that the factor $\frac{1}{2}$ appears in (2.c). The example below shows that for any $\frac{\epsilon_0}{2} < \beta$ there are operating points such that no subset P of the operating points satisfies (2.a) and $M(i,j) \geq \beta$ for all $i, j \in P$. The additional precision parameter ϵ in (2.c) is necessary to account for sampling errors.

Table 1: For $\frac{\epsilon_0}{2} < \alpha < \beta$ there is no β-sparse ϵ_0-cover of the Pareto front.

$\boldsymbol{y}_i$	y_i^1	y_i^2	y_i^3	y_i^4	y_i^5
$\boldsymbol{y}_1$	1	1	1	$1-\alpha$	$1-2\alpha$
$\boldsymbol{y}_2$	1	1	$1-\alpha$	$1-2\alpha$	1
$\boldsymbol{y}_3$	1	$1-\alpha$	$1-2\alpha$	1	1
$\boldsymbol{y}_4$	$1-\alpha$	$1-2\alpha$	1	1	1
$\boldsymbol{y}_5$	$1-2\alpha$	1	1	1	$1-\alpha$

Example 2. *Consider the operating points* $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_5$ *in Table 1. All these operating points are Pareto optimal. Further,* $M(1,2) = M(2,3) = M(3,4) = M(4,5) = M(5,1) = \alpha < \beta$ *and* $M(i,j) = 2\alpha > \epsilon_0$ *for all other* $i \neq j$. *Thus a subset* P *satisfying (2.a) must contain for each point* i *the point* i *itself or its "successor" with* $M(i,j) = \alpha$. *It follows that such a subset* P *contains at least three points, which implies that it contains a pair with* $M(i,j) = \alpha < \beta$.

In Section 7, we show that for any finite set P^* with asymmetric distance function M, and any $\epsilon_0 > 0$, a set $P \subseteq P^*$ can be calculated satisfying (2.a) and (2.c). Note that for $D = 1$, Success Condition 2 is quite trivial, as any single ϵ_0-optimal point constitutes a cover.

4 ALGORITHM FOR SUCCESS CONDITION 1

In this section, we present Algorithm 1 that meets Success Condition 1 with high probability. The set P output by this algorithm will also be used in Section 7 to calculate a sparse cover of the Pareto front meeting Success Condition 2.

Algorithm 1 for finding all Pareto optimal points

Input: set of operating points $\{1,\dots,K\}$, accuracy parameter $\epsilon_0 > 0$, confidence parameter $\delta > 0$.

Initialize $A \leftarrow \{1,\dots,K\}$ and $P \leftarrow \emptyset$.

Repeat until $A = \emptyset$:

1. Sample each operating point $i \in A$ once.
2. $A_1 \leftarrow \{i \in A \,|\, \forall j \in A : \hat{m}(i,j) \le \beta_i + \beta_j\}$
3. $P_1 \leftarrow \{i \in A_1 \,|\, \forall j \in A_1 \setminus \{i\} : \hat{M}_{\epsilon_0}(i,j) \ge \beta_i + \beta_j\}$.
4. $P_2 \leftarrow \{j \in P_1 \,|\, \nexists i \in A_1 \setminus P_1 : \hat{M}_{\epsilon_0}(i,j) \le \beta_i + \beta_j\}$.
5. $A \leftarrow A_1 \setminus P_2$ and $P \leftarrow P \cup P_2$.

Output: P

The algorithm is an elimination algorithm based on confidence intervals, that maintains a set A of active operating points. It eliminates points that are suboptimal with high probability (step 2), or points that are (almost) Pareto optimal with high probability (set P_1 in step 3). Optimal points in P_1 are not removed, though, if they are needed to establish the status of other points they may dominate (steps 4 and 5). The algorithm stops when no active operating points remain and outputs the set of (almost) Pareto optimal points.

The algorithm uses the empirical estimates $\hat{y}_i^d$ of the mean objective values y_i^d to calculate estimates of the gap functions as

$$\begin{aligned} \hat{m}(i,j) &= \min\{s \ge 0 \,|\, \exists d : \hat{y}_i^d + s \ge \hat{y}_j^d\}, \\ \hat{M}_{\epsilon_0}(i,j) &= \min\{s \ge 0 \,|\, \forall d : \hat{y}_i^d + \epsilon_0 \le \hat{y}_j^d + s\}. \end{aligned}$$

The latter definition reflects success condition (1.b) that allows to return points that are not dominated by more than ϵ_0. We will also use the estimates

$$\hat{M}(i,j) = \hat{M}_0(i,j)$$

with $\hat{M}(i,j) \le \hat{M}_{\epsilon_0}(i,j) \le \hat{M}(i,j) + \epsilon_0$. The width of the confidence intervals for the $\hat{y}_i^d$ is defined as

$$\beta_i = \sqrt{\frac{2\log(4KDn_i^2/\delta)}{n_i}}, \tag{3}$$

where n_i is the number of samples from operating point i so far.

Algorithm 1 is analyzed in the next section, proving Theorem 4 as the first main result of the paper. The constant in the sample complexity bound is quite loose, though.

Theorem 4. *When Algorithm 1 is run on a set of operating points with parameters ϵ_0 and δ, then with probability $1-\delta$ it outputs a set of operating points P satisfying Success Condition 1, using at most a total number of*

$$\sum_i \frac{4320}{(\Delta_i^{\epsilon_0})^2} \log\Big(\frac{12KD}{\delta\Delta_i^{\epsilon_0}}\Big)$$

samples, where $\Delta_i^{\epsilon_0} = \max\{\Delta_i, \epsilon_0\}$ and Δ_i is as defined in equations (1) and (2).

For $D = 1$, Theorem 4 recovers known bounds for the multi-armed bandit problem as given e.g. by Mannor and Tsitsiklis (2004), Gabillon et al. (2011). Note that the sample complexity of an algorithm that simply compares any two points will depend on the smallest distance between any two points (with respect to any objective), which in general can be much larger than the quantities that appear in Theorem 4.

5 ANALYSIS OF ALGORITHM 1

We start with some simple results about the confidence intervals and their relation to the number of samples.

Lemma 5. *With probability $1-\delta$, $|\hat{y}_i^d - y_i^d| < \beta_i$ for all i and d and any number of samples n_i from the operating points.*

Proof. By Hoeffding's inequality and a union bound. □

For simplicity we assume in the following that the confidence bounds of Lemma 5 hold. Thus some of the following lemmas hold only with probability $1-\delta$.

Lemma 6. *When $n_i \ge \frac{30\log(KD/(\delta\Delta))}{\Delta^2}$, then $\beta_i \le \Delta$.*

Proof. By the definition of β_i and simple calculus. □

Lemma 7. *For any i, j, and d, $|(\hat{y}_i^d - \hat{y}_j^d) - (y_i^d - y_j^d)| < \beta_i + \beta_j$, $|m(i,j) - \hat{m}(i,j)| < \beta_i + \beta_j$, and $|M(i,j) - \hat{M}(i,j)| < \beta_i + \beta_j$.*

Proof. By the confidence bounds of Lemma 5. □

The following lemmas show that with high probability the algorithm does not miss any point of the Pareto front.

Lemma 8. *Only suboptimal points are removed in Step 2 of the algorithm.*

Proof. If $\hat{m}(i,j) > \beta_i + \beta_j$ then $m(i,j) > 0$ by Lemma 7. Thus $i \prec j$ and i is not Pareto optimal. □

Lemma 9. *At any stage of the algorithm, $P^* \subseteq A \cup P$.*

Proof. Initially the statement holds since $A = \{1,\dots,K\}$. Since points are removed from $A \cup P$ only by Step 2, the statement follows from Lemma 8. □

Lemma 10. *Let $i \prec j$ and $j \in P^*$ with $m(i,j) = \Delta_i^* \geq \epsilon_0$. If $i \in A$ then also $j \in A$. Furthermore, $i \notin P_1$ and $i \notin P$.*

Proof. The proof is by induction on the iterations of the algorithm. Initially the statement holds since $A = \{1, \ldots, K\}$ and $P = \emptyset$. Assume that i is not removed in Step 2 and $i \in A_1$ in the current iteration. By Lemma 8 and induction assumption also $j \in A_1$. Since $m(i,j) \geq \epsilon_0$ we have for all d that $y_i^d - y_j^d \leq -\epsilon_0$, and by Lemma 7 that $\hat{y}_i^d - \hat{y}_j^d + \epsilon_0 < y_i^d - y_j^d + \epsilon_0 + \beta_i + \beta_j \leq \beta_i + \beta_j$. Thus $\hat{M}_{\epsilon_0}(i,j) < \beta_i + \beta_j$. Hence $i \notin P_1$ and $j \notin P_2$, such that j remains in A and i is not added to P. □

Lemma 11. *If $i \in P \setminus P^*$, then $\Delta_i^* \leq \epsilon_0$.*

Proof. Immediate from Lemma 10. □

By Lemmas 9 and 11 we have established that the output of the algorithm satisfies conditions (1.a) and (1.b). To bound the sample complexity, we lower bound β_i for $i \in A$. Note that by definition of the algorithm, $\beta_i = \beta_j$ for all $i, j \in A$.

Lemma 12. *If in some iteration of the algorithm $\beta_i \leq \frac{\epsilon_0}{4}$ for $i \in A_1$, then $P_2 = A_1$ and the algorithm stops.*

Proof. By Step 2 of the algorithm we have that for all $i, j \in A_1$, there is a d with $\hat{y}_j^d - \hat{y}_i^d \leq \beta_i + \beta_j$. Hence $\hat{y}_i^d - \hat{y}_j^d + \epsilon_0 \geq \epsilon_0 - \beta_i - \beta_j \geq \frac{\epsilon_0}{2} \geq \beta_i + \beta_j$ and $\hat{M}_{\epsilon_0}(i,j) \geq \beta_i + \beta_j$. Thus by Step 3 we have $P_1 = A_1$, and by Step 4 also $P_2 = P_1 = A_1$. □

Lemma 13. *If $i \notin P^*$, $\Delta_i^* \geq \epsilon_0$, and $i \in A_1$, then $\beta_i \geq \Delta_i/4$.*

Proof. By Lemma 10 there is a $j \in A$ with $m(i,j) = \Delta_i^*$. Then $\Delta_i = \Delta_i^* = m(i,j) \leq \hat{m}(i,j) + \beta_i + \beta_j \leq 2\beta_i + 2\beta_j = 4\beta_i$ by Lemma 7 and Step 2. □

Lemma 14. *If $i \in P^*$ and $i \in A_1 \setminus P_1$, then $\beta_i \geq \Delta_i/4$.*

Proof. If $i \in A_1$ and $i \notin P_1$, then by Step 3 there is a $j \in A_1$, $i \neq j$, with $\hat{M}_{\epsilon_0}(i,j) < \beta_i + \beta_j$. Thus $M(i,j) < \hat{M}(i,j) + \beta_i + \beta_j \leq \hat{M}_{\epsilon_0}(i,j) + \beta_i + \beta_j < 2\beta_i + 2\beta_j = 4\beta_i$. If $j \in P^*$, then by definition $\Delta_i \leq M(i,j) \leq 4\beta_i$. If $j \notin P^*$, then there is a $j^* \in P^*$ with $j \prec j^*$ such that by Lemma 3 $\Delta_i \leq M(i,j^*) \leq M(i,j) + M(j,j^*) = M(i,j) \leq 4\beta_i$. □

Lemma 15. *If $j \in P^*$, $j \in P_1 \setminus P_2$, and $\beta_j \geq \epsilon_0/4$. Then $\beta_j \geq \Delta_j/12$.*

Proof. If $j \in P_1 \setminus P_2$, then by Step 4 there is an $i \in A_1$, $i \neq j$, with $\hat{M}_{\epsilon_0}(i,j) \leq \beta_i + \beta_j$ and thus $M(i,j) \leq 4\beta_j$. If $\Delta_i^* < \epsilon_0$, then $\Delta_i^* \leq 4\beta_j$. If $\Delta_i^* \geq \epsilon_0$, then by Lemma 10 there is an $i^* \in A$ with $m(i,i^*) = \Delta_i^*$. Thus by Step 2 we also get $\Delta_i^* = m(i,i^*) \leq \hat{m}(i,i^*) + \beta_i + \beta_{i^*} \leq 2\beta_i + 2\beta_{i^*} = 4\beta_j$. Since by definition $\Delta_j \leq M(i,j) + 2\Delta_i^*$, we have $\Delta_j \leq 12\beta_i$. □

Proof of Theorem 4. Since the algorithm terminates when $A = \emptyset$, Success Condition 1 is satisfied by Lemmas 9 and 11. Combining Lemmas 12–15, we find that $i \in A_1 \setminus P_2$ only if $\beta_i \geq \Delta_i^{\epsilon_0}/12$. By Lemma 6, this gives the sample complexity bound of the theorem.

6 LOWER BOUND

The following Theorem 17 provides a lower bound corresponding to the upper bound in Theorem 4. For the proof see the appendix (supplementary material).

Definition 16. *We call an algorithm a* learning algorithm for Success Condition 1 on $[0,1]^D$, *if for any set of distributions $\mathcal{D}_i$ on $[0,1]^D$ for the operating points $i = 1, \ldots, K$, it outputs with probability $1 - \delta$ a set of operating points P satisfying Success Condition 1.*

Theorem 17. *For any set of operating points $\boldsymbol{y}_i \in [\frac{1}{4}, \frac{3}{4}]^D$, $i = 1, \ldots, K$, there exist distributions $\mathcal{D}_i$ such that any learning algorithm for Success Condition 1 on $[0,1]^D$ requires*

$$\Omega\Big(\sum_{i=1}^{K} \frac{1}{(\tilde{\Delta}_i^{\epsilon_0})^2} \log\Big(\frac{1}{\delta}\Big)\Big)$$

samples to identify the Pareto front with precision $\epsilon_0 \leq \frac{1}{8}$, where $\tilde{\Delta}_i^{\epsilon_0} = \max\{\Delta_i^*, \epsilon_0\}$ for $i \notin P^*$ and $\tilde{\Delta}_i^{\epsilon_0} = \max\{\Delta_i^+, \epsilon_0\}$ for $i \in P^*$.

Remark 18. *We note that the lower bound may not match the upper bound for $i \in P^*$ with $\Delta_i = M(j,i) + 2\Delta_j^* < \Delta_i^+$ for some $j \notin P^*$. But since in this case $\Delta_i > \Delta_j^*$, the missing Δ_i-term can be compensated by the Δ_j^*-term, as long as the same j is not needed in too many such definitions of Δ_i, $i \in P^*$.*

7 ALGORITHM AND ANALYSIS FOR SUCCESS CONDITION 2

While the set P that is output by Algorithm 1 with high probability satisfies Success conditions (2.a) and (2.b), for achieving (2.c) we have to compute a sparse cover of P. In order to do so, we first check which operating points are too close to each other. This may require additional samples from points in P. Thus, our proposed Algorithm 2 maintains a set of points Q that still have to be sampled (step 2). The algorithm however only samples the point with the widest confidence interval (step 3). When two close points have been identified, an edge is put between them (step 1). Note that edges are directed since the measure $\hat{M}(j,i)$ is not symmetric. When the status of all pairs of points has been identified, the algorithm outputs a directed graph whose vertices are the points in P with edges between close points. Note that Algorithm 2 stops in particular when all $\beta_i \leq \frac{\epsilon}{4}$ such that $Q = \emptyset$ by definition.

Algorithm 2 for identifying close points

Input: the set P output by Algorithm 1
Initialize $E \leftarrow \emptyset$ and $Q \leftarrow P$.
Repeat until $Q = \emptyset$:

1. $E' \leftarrow \{(i,j) \in Q^2 \mid \hat{M}(j,i) \leq \frac{\epsilon_0}{2} - \beta_i - \beta_j, i \neq j\}$.
2. $Q \leftarrow \Big\{ i \in Q \mid \exists j \in Q, j \neq i :$
   $\frac{\epsilon_0}{2} - \beta_i - \beta_j < \hat{M}(i,j) \leq \frac{\epsilon_0}{2} - \epsilon + \beta_i + \beta_j$
   or $\frac{\epsilon_0}{2} - \beta_i - \beta_j < \hat{M}(j,i) \leq \frac{\epsilon_0}{2} - \epsilon + \beta_i + \beta_j \Big\}$.
3. Sample some $i \in Q$ with $\beta_i = \max_{j \in Q} \beta_j$ once.
4. $E \leftarrow E \cup E'$.

Output: the graph $G = (P, E)$

The algorithm returns an edge (i,j) if $M(j,i) < \frac{\epsilon_0}{2} - \epsilon$, and it returns no edge (i,j) if $M(j,i) > \frac{\epsilon_0}{2}$. This follows from the construction of the algorithm and is argued formally in the following sections.

7.1 Asymmetric Covering

In order to compute a sparse cover of P satisfying Success Condition 2 we propose an algorithm that eliminates redundant operating points from the graph G output by Algorithm 2. As we are not aware of any algorithm that computes sparse covers for asymmetric distance relations (as $M(i,j)$ in our application), this algorithm is of independent interest. For any finite set P^* with an asymmetric distance function M and any $\epsilon_0 > 0$, it returns a cover $P \subseteq P^*$ that satisfies (2.a) and (2.c).

The proposed Algorithm 3 first breaks up cycles in the graph (step 1) and then removes redundant vertices in the remaining graph (step 2) until no edges are left in the graph.

Algorithm 3 for finding a sparse cover

Input: a directed graph $G = (V, E)$ (e.g., as produced by Algorithm 2)

1. Repeat until there is no directed cycle in G:
   Pick an arbitrary vertex i on some cycle, and mark i.[2]For any successor j of i, if j belongs to any cycle, then remove j together with incident edges.
2. Repeat until there is no edge in G:
   Pick a vertex i without predecessor, remove all successors j of i together with edges incident with j.

Output: the set V' of remaining vertices

Lemma 19. *For any two distinct vertices $i, j \in V'$ it holds that $M(i,j) \geq \frac{\epsilon_0}{2} - \epsilon$ and $M(j,i) \geq \frac{\epsilon_0}{2} - \epsilon$.*

Proof. If $i, j \in V'$ then i and j have not been connected by an edge in G, since otherwise this edge and one of the two vertices would have been deleted by Algorithm 3. Hence, by Algorithm 2, we always have $\hat{M}(i,j) > \frac{\epsilon_0}{2} - \beta_i - \beta_j$ and $\hat{M}(j,i) > \frac{\epsilon_0}{2} - \beta_i - \beta_j$. At some point either i or j is removed from Q. Assume without loss of generality that i is removed before or at the same step as j from Q since $\hat{M}(i,k) > \frac{\epsilon_0}{2} - \epsilon + \beta_i + \beta_k$ and $\hat{M}(k,i) > \frac{\epsilon_0}{2} - \epsilon + \beta_i + \beta_k$ hold for all $k \in Q$ and in particular for $k = j$. Then by Lemma 7, $M(i,j) > \hat{M}(i,j) - \beta_i - \beta_j > \frac{\epsilon_0}{2} - \epsilon$ and similarly $M(j,i) > \frac{\epsilon_0}{2} - \epsilon$. □

Lemma 20. *(a) If $(i,j) \in E$, then $M(j,i) \leq \frac{\epsilon_0}{2}$.*
(b) If (i,ℓ) and (ℓ,j) are in E, then $M(j,i) \leq \epsilon_0$.

Proof. For $(i,j) \in E$, by Algorithm 2 we have $\hat{M}(j,i) \leq \frac{\epsilon_0}{2} - \beta_j - \beta_i$. Then by Lemma 7, $M(j,i) < \hat{M}(j,i) + \beta_j + \beta_i \leq \frac{\epsilon_0}{2}$, which proves (a). This also implies (b), as by Lemma 3 we have $M(j,i) \leq M(j,\ell) + M(\ell,i)$. □

Lemma 21. *Let $j \in V$ be a vertex that is removed. Then there is a vertex $i \in V'$ such that $M(j,i) \leq \epsilon_0$.*

Proof. First, we note that no marked vertex ℓ will belong to any cycle, since every successor of ℓ belonging to a cycle has been removed. Thus no marked vertex is removed in Step 1. Also when a vertex j is removed in Step 2, there is a predecessor i without predecessor, such that i will never be removed.

Now for any removed vertex j, one of the following three cases applies: Either (i) vertex j is removed in Step 2, or (ii) it is removed in Step 1 and has a marked predecessor belonging to V', or (iii) it is removed in Step 1 and has a marked predecessor ℓ that is removed in Step 2. In cases (i) and (ii), as argued before, there is a vertex i in V' which is the predecessor of the deleted vertex. In case (iii) we have $(\ell,j) \in E$ and since none of the marked vertices is removed in Step 1, there is an $i \in V'$ such that $(i,\ell) \in E$. Applying Lemma 20 we obtain the lemma. □

7.2 Sample Complexity Bound for Success Condition 2

Lemma 22. *Consider Algorithm 2 and let $i \in Q$. If $|M(i,j) - (\frac{\epsilon_0}{2} - \epsilon)| > 2(\beta_i + \beta_j)$ and $|M(j,i) - (\frac{\epsilon_0}{2} - \epsilon)| > 2(\beta_i + \beta_j)$ holds for all $j \in Q$, $j \neq i$, then $i \notin Q$ in the next iteration.*

[2]Marking vertices helps with the proof, but is immaterial to the algorithm.

Proof. Note that $\frac{\epsilon_0}{2}-\beta_i-\beta_j < \hat{M}(i,j) \le \frac{\epsilon_0}{2}-\epsilon+\beta_i+\beta_j$ implies $|\hat{M}(i,j)-\frac{\epsilon_0}{2}+\epsilon| \le \beta_i+\beta_j$. Hence, by Lemma 7, if i remains in Q in the next iteration, then in the current iteration $\exists j \in Q$ such that $|M(i,j)-\frac{\epsilon_0}{2}+\epsilon| \le 2(\beta_i+\beta_j)$ or $|M(j,i)-\frac{\epsilon_0}{2}+\epsilon| \le 2(\beta_i+\beta_j)$, and the claim follows by contraposition. □

Theorem 23. *The output V' of Algorithm 3 (when using the graph produced by Algorithm 2 as input) satisfies Success Condition 2. Furthermore, the sample complexity for computing V' is*

$$O\left(\sum_i \frac{1}{(\Delta_i^{\epsilon_0,\epsilon})^2}\log\left(\frac{KD}{\delta\Delta_i^{\epsilon_0,\epsilon}}\right)\right),$$

where $\Delta_i^{\epsilon_0,\epsilon} = \min\Big\{\max\{\Delta_i,\epsilon_0\}, \frac{1}{4}\max\big\{\min_{j\in P\setminus\{i\}} d(i,j),\ \epsilon\big\}\Big\}$,

$d(i,j) = \min\{|M(i,j)-\frac{\epsilon_0}{2}+\epsilon|,\ |M(j,i)-\frac{\epsilon_0}{2}+\epsilon|\}$, *and P is the set output by Algorithm 1.*

Proof. That Success Condition 2 is satisfied follows from Lemmas 19 and 21 and the results about Algorithm 1. For any operating point i, its sample complexity in Algorithm 1 is determined by $\max\{\Delta_i,\epsilon_0\}$ according to Theorem 4. For the additional samples taken by Algorithm 2, according to Lemma 22 any fixed $i \in Q$ will be at most sampled until $d(i,j) \ge 2(\beta_i+\beta_j)$ for all j. Note that if at some time $\beta_i \ge \beta_j$ for some j in Q, then point i will not be sampled more often than to reach $4\beta_i < d(i,j)$. On the other hand, if for all j always $\beta_i < \beta_j$ then by definition Algorithm 2 (that only samples points with maximal β_j) point i is not sampled at all. Finally, recall that the algorithm terminates when all $\beta_i < \frac{\epsilon}{4}$ and note that as soon as some $\beta_i < \frac{\epsilon}{4}$, it will not be sampled anymore before all remaining points j in Q reach $\beta_j < \frac{\epsilon}{4}$ as well. The result follows by Lemma 6, summing over all points i and combining the bound with Theorem 4. □

8 EXPERIMENTS

To illustrate the performance of our Algorithm 1, we run it on the SW-LLVM data set and compare it to the PAL algorithm of Zuluaga et al. (2013), noting that the comparison is not completely fair as the two algorithms have different goals and make different assumptions on the data.

The PAL algorithm uses Gaussian process regression to estimate the objective values of operating points. This allows PAL to generalize over operating points such that not all operating points need to be sampled. The confidence intervals from the Gaussian process regression are used to identify promising points that are possibly close to the Pareto front. The goal of PAL is to estimate the Pareto front up to a given accuracy, but not to return all Pareto optimal points.

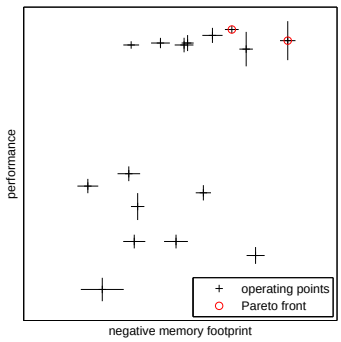

Figure 1: Operating points of the modified SW-LLVM data set.

In contrast, our algorithm is required to return all Pareto optimal points, and to return only points that are close to the Pareto front. In this sense our algorithm takes a more conservative approach to estimating the Pareto front. Furthermore, our algorithm does not generalize over operating points and has to sample each operating point.

The data set. We use the SW-LLVM data set[3] from (Zuluaga et al., 2013) consisting of 1024 operating points characterized by 10 binary values. Each operating point corresponds to a compiler setting, and the objectives are performance and memory footprint of some software compiled with these settings.
To obtain quasi-stochastic data for our algorithm, we use the four most informative binary features (features 7–10) to define 16 operating points for our experiments. Ignoring the other features, this gives 64 samples for each of the 16 operating points. As the "true" objective values for each of these operating points we use the means of the 64 samples for the respective operating point. These operating points with their standard deviations are depicted in Figure 1.[4] There are two Pareto optimal points. The data set with only 4 features is also used in the experiments with PAL.[5]

Modifications of Algorithm 1. To make our theoretically motivated Algorithm 1 practical, we made small modifications: (i) We scale the objective values

[3] An implementation of PAL and the data set are available at `http://www.spiral.net/software/pal.html`.

[4] Since the memory footprint is to be minimized, we plot and maximize the negative memory footprint.

[5] The chosen four features are highly predictive. The squared error of a linear regression model decreases only from 18.6% to 16.5% of the sample variance, if all features are used. This puts PAL at some advantage since employing regression is still useful for these data.

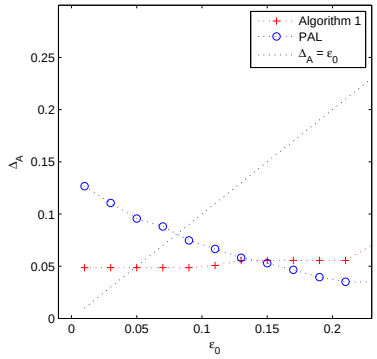

Figure 2: Δ_A averaged over 1000 runs.

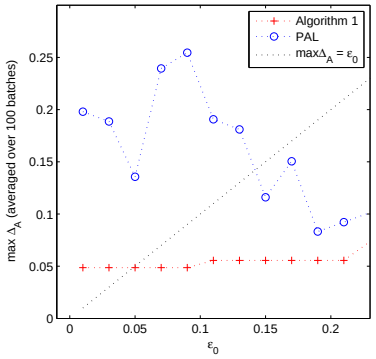

Figure 3: Worst Δ_A in batches of 10 runs.

in the same way as PAL to allow for a general range of the objective values. (ii) Instead of the overly conservative (3), we use individual and tighter confidence values $\beta_i^d = \sqrt{\frac{2\tilde{V}_i^d \log(DKn_i/\delta)}{n_i}}$ for each objective. Here $\tilde{V}_i^d = \hat{V}_i^d + \bar{V}^d\sqrt{4\log(DKn_i/\delta)/n_i}$ is an upper confidence bound on the variance, where $\hat{V}_i^d$ is the empirical variance and $\bar{V}^d$ is a given upper bound. This upper confidence bound is justified by a tail bound on the χ^2-distribution (Lemma 1 of Laurent and Massart, 2000), assuming that the data are sufficiently close to a Gaussian. (iii) In Steps 2–4 of the algorithm instead of $\beta_i + \beta_j$ we use the tighter $\sqrt{\beta_i^2 + \beta_j^2}$.[6]

Results. In all experiments with Algorithm 1 and PAL we use confidence parameter $\delta = 0.1$. To accommodate the different goal of PAL, we define the *estimation distance*

$$\Delta_A = \max\Big\{\max_{i\in P}\Delta_i^*, \max_{i\in P^*}\min_{j\in P} M(i,j)\Big\},$$

considering the maximal distance of the predicted points to the Pareto front, as in Success Condition (1.b), and the maximum distance of an optimal point to the closest predicted point.

Figure 2 reports the estimation distance of the output set P from the Pareto front P^* for different choices of ϵ_0, each averaged over 1000 runs. Algorithm 1 achieves the desired precision unless the 64 available samples per operating point are not sufficient (this happens for small ϵ_0). For small ϵ_0, the estimation distance of PAL is larger, because PAL does not penalize predicted points that are far from the Pareto front. For large ϵ_0, PAL shows smaller estimation distance at the expense of possibly missing Pareto optimal points, cf. Figures 3 and 4. Figure 3 shows the worst estimation distance in batches of 10 runs, averaged over 100 batches. Figure 4 shows the fraction of 1000 runs

[6]This is theoretically justified. In the theoretical analysis with already loose constants we use $\beta_i+\beta_j$ for simplicity of notation and argument.

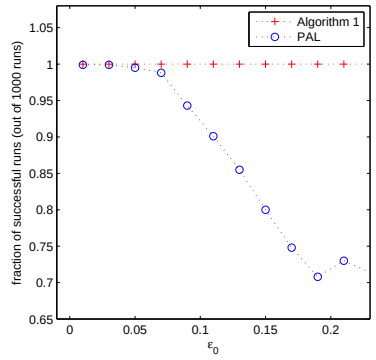

Figure 4: Runs that find all optimal points.

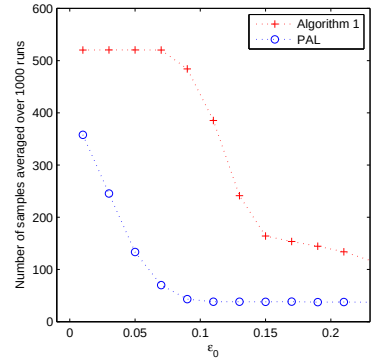

Figure 5: Average number of samples.

in which the algorithms returned all Pareto optimal points. We see that PAL frequently returns points far from the Pareto front for small ϵ_0, and misses Pareto optimal points for large ϵ_0. In contrast, Algorithm 1 always returns all Pareto optimal points and has very stable estimation distance. This conservative behavior comes at the price of larger numbers of samples, see Figure 5.

Acknowledgements This research was funded by the Austrian Science Fund (FWF): P 26219-N15.

References

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT 2010 - The 23rd Conference on Learning Theory*, pages 41–53. Omnipress, 2010.

Sébastien Bubeck, Tengyao Wang, and Nitin Viswanathan. Multiple identifications in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, volume 28 of *JMLR Proceedings*, pages 258–265, 2013.

Madalina M. Drugan, Ann Nowé, and Bernard Manderick. Pareto upper confidence bounds algorithms: An empirical study. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2014*, pages 1–8, 2014.

Victor Gabillon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Sébastien Bubeck. Multi-bandit best arm identification. In *Advances in Neural Information Processing Systems 24, NIPS 2011*, pages 2222–2230, 2011.

Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems 25, NIPS 2012*, pages 3221–3229, 2012.

Yaochu Jin, editor. *Multi-objective machine learning*. Springer, 2006.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, to appear.

Béatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, 28(5):1302–1338, 2000.

Daniel J. Lizotte, Michael H. Bowling, and Susan A. Murphy:. Efficient reinforcement learning with multiple reward functions for randomized clinical trial analysis. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 695–702, 2010.

Shie Mannor and John N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648, 2004.

Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *Journal of Machine Learning Research*, 15:3483–3512, 2014.

Marcela Zuluaga, Guillaume Sergent, Andreas Krause, and Markus Püschel. Active learning for multi-objective optimization. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, volume 28 of *JMLR Proceedings*, pages 462–470, 2013.