Journal of Machine Learning Research 17 (2016) 1-32 Submitted 2/15; Revised 4/16; Published 8/16

ϵ-PAL: An Active Learning Approach to the Multi-Objective Optimization Problem

Marcela Zuluaga ZULUAGA@INF.ETHZ.CH
Department of Computer Science
ETH Zurich
Zurich, Switzerland

Andreas Krause ANDREAS.KRAUSE@INF.ETHZ.CH
Department of Computer Science
ETH Zurich
Zurich, Switzerland

Markus Püschel PUESCHEL@INF.ETHZ.CH
Department of Computer Science
ETH Zurich
Zurich, Switzerland

Editor: Kevin Murphy

Abstract

In many fields one encounters the challenge of identifying out of a pool of possible designs those that simultaneously optimize multiple objectives. In many applications an exhaustive search for the Pareto-optimal set is infeasible. To address this challenge, we propose the ϵ-*Pareto Active Learning* (ϵ-PAL) algorithm which adaptively samples the design space to predict a set of Pareto-optimal solutions that cover the true Pareto front of the design space with some granularity regulated by a parameter ϵ. Key features of ϵ-PAL include (1) modeling the objectives as draws from a Gaussian process distribution to capture structure and accommodate noisy evaluation; (2) a method to carefully choose the next design to evaluate to maximize progress; and (3) the ability to control prediction accuracy and sampling cost. We provide theoretical bounds on ϵ-PAL's sampling cost required to achieve a desired accuracy. Further, we perform an experimental evaluation on three real-world data sets that demonstrate ϵ-PAL's effectiveness; in comparison to the state-of-the-art active learning algorithm PAL, ϵ-PAL reduces the amount of computations and the number of samples from the design space required to meet the user's desired level of accuracy. In addition, we show that ϵ-PAL improves significantly over a state-of-the-art multi-objective optimization method, saving in most cases 30% to 70% evaluations to achieve the same accuracy.

Keywords: multi-objective optimization, active learning, pareto optimality, Bayesian optimization, design space exploration

1. Introduction

A fundamental challenge in many problems in engineering and other domains is to find the right balance amongst several objectives. As a concrete example, in hardware design, one often has to choose between different candidate designs that trade multiple objectives

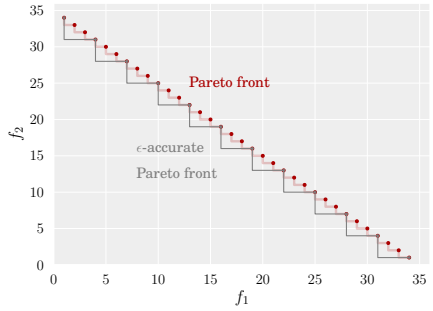

Figure 1: Assume that two objective functions are to be maximized simultaneously. This figure shows an example of the true Pareto front of a design space contrasted with an $\boldsymbol{\epsilon}$-accurate Pareto front, with $\boldsymbol{\epsilon} = (3, 3)$.

such as energy consumption, throughput, or chip area. Usually there is not a single design that excels in all objectives, and therefore one may be interested in identifying all (Pareto-)optimal designs. Furthermore, often in these domains, evaluating the objective functions is expensive and noisy. In hardware design, for example, synthesis of only one design can take hours or even days. The fundamental problem addressed in this paper is how to predict a Pareto-optimal set at low cost, i.e., by evaluating as few designs as possible.

In this paper we propose a solution for finite design spaces that we call the $\boldsymbol{\epsilon}$-*Pareto Active Learning* ($\boldsymbol{\epsilon}$-PAL) algorithm. The parameter $\boldsymbol{\epsilon}$ allows users to control the accuracy of the prediction produced by the algorithm. This accuracy is defined in terms of the density or granularity of the estimated Pareto front that is returned. Decreasing granularity means the algorithm can discard more points along its execution. Fewer Pareto points are returned, but spread to offer a wide range of trade-offs in the objective space.

More specifically, the granularity is controlled by a vector $\boldsymbol{\epsilon}$, containing one value per objective, which is given in the same units as the corresponding objective function. It specifies that for the user a difference of $\boldsymbol{\epsilon}$ in the objective space is negligible. Accordingly, an $\boldsymbol{\epsilon}$-accurate Pareto front might have fewer points than the true Pareto front. Fig. 1 visualizes this idea for two objective functions that are to be maximized simultaneously. The true Pareto-front is shown in red, and an $\boldsymbol{\epsilon}$-accurate Pareto front, which contains fewer points, is shown in grey. While the true Pareto front of a design space is unique, there can be many $\boldsymbol{\epsilon}$-accurate Pareto fronts; finding one of them efficiently is our goal.

$\boldsymbol{\epsilon}$-PAL has several key features. In the spirit of Bayesian optimization, it captures domain knowledge about the regularity of the design space by using Gaussian process (GP) models to predict objective values for designs that have not been evaluated yet. Further, it uses the predictive uncertainty associated with these nonparametric models in order to guide the iterative sampling. Specifically, $\boldsymbol{\epsilon}$-PAL's sampling strategy aims to maximize progress on designs that are likely to be Pareto-optimal. $\boldsymbol{\epsilon}$-PAL iteratively discards points that are either redundant or suboptimal, and it terminates when no more points can be removed in order to guarantee that, with high probability, the remaining points define an $\boldsymbol{\epsilon}$-accurate Pareto set of the given design space.

A main contribution of this paper is the theoretical performance analysis of ϵ-PAL, that provides bounds on the sampling cost required to achieve a desired accuracy. These bounds depend on the parameter ϵ and on the characteristics of the covariance function used for the GP models. Finally, we carry out an extensive empirical evaluation, where we demonstrate ϵ-PAL's effectiveness on several real-world multi-objective optimization problems. Two cases are from different applications in the domain of hardware design, in which it is very expensive to run low level synthesis to obtain the exact cost and performance of a single design (Zuluaga et al., 2012b; Almer et al., 2011). The third application is from software optimization, where different compilation settings are evaluated for performance and memory footprint size (Siegmund et al., 2012).

We compare the performance of ϵ-PAL against PAL, a predecessor of ϵ-PAL proposed by Zuluaga et al. (2013), and with another state-of-the-art multi-objective optimization method called ParEGO, proposed by Knowles (2006). Across all data sets and almost all desired accuracies, ϵ-PAL outperforms ParEGO, requiring in most cases 30% to 70% less function evaluations. In comparison to PAL, our experiments show that ϵ-PAL reduces the runtime by one to two orders of magnitude, while also reducing the number of evaluations and offering more flexibility in the desired error.

1.1 Main Contributions

Our main contributions are summarized as follows.

- We propose ϵ-PAL, an active learning approach towards identifying a set of optimal solutions in a given multi-objective optimization problem in which function evaluations are expensive. This includes sampling strategy and stopping criteria. ϵ-PAL allows users to specify the desired level of granularity through the parameter ϵ, and guarantees that with high probability an ϵ-accurate Pareto front is returned.

- We theoretically analyze ϵ-PAL when the design space is a finite set, and the objective functions satisfy regularity assumptions as specified via a positive definite kernel. We provide bounds on the number of iterations required by the algorithm to achieve a desired target accuracy.

- In comparison to the state-of-the-art algorithm PAL, ϵ-PAL reduces the asymptotic complexity of the number of computations performed in each iteration, making it more suitable for larger design spaces.

- We perform an extensive experimental evaluation to demonstrate ϵ-PAL's effectiveness and superiority over PAL and ParEGO on three real-world multi-objective optimization problems.

1.2 Organization

In Section 2, we introduce background concepts such as multi-objective optimization, Pareto optimality, ϵ-accurate Pareto optimality, and Gaussian processes. In addition, we formally define the problem tackled in this paper. Section 3 presents the proposed algorithm ϵ-PAL, explains the stages that take place during its execution, and analyzes run time. In

Section 4, we present the theoretical analysis of the algorithm and provide upper bounds for the number of samples needed before ϵ-PAL terminates, when the target functions have bounded RKHS norm. Section 5 discusses some implementation issues that might arise in practice when using our algorithm. Section 6 reviews related work in the areas of multi-objective optimization, Bayesian optimization and evolutionary algorithms. Section 7 shows the effectiveness of ϵ-PAL in comparison with state-of-the-art alternative algorithms, using three data sets obtained from real applications. In the Appendix, we prove our main Theorem, which is stated in Section 4.

2. Background and Problem Statement

In this section, we review multi-objective optimization and Pareto optimality, and introduce the terminology and notation used in the rest of the paper. At the end of the section, we formally define the problem addresses by ϵ-PAL.

2.1 Multi-Objective Optimization

We consider a multi-objective optimization problem over a finite set E (called the *design space*). This means that we wish to simultaneously optimize m objective functions $f_1, \ldots, f_m : E \to \mathbb{R}$. In our analysis, we assume that $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ are to be maximized, however our results straightforwardly generalize to a minimization or a combined minimization/maximization problem. We use the notation $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ to refer to the vector of all objectives evaluated on the input $\boldsymbol{x}$. The *objective space* is the image $\boldsymbol{f}(E) \subset \mathbb{R}^m$.[1]

2.2 Pareto Optimality

The goal in multi-objective optimization is to identify[2] the *Pareto set* of E. Formally, we consider the canonical partial order in $\mathbb{R}^m$: $\boldsymbol{y} \succeq \boldsymbol{y}'$ iff $y_i \geq y'_i$, $1 \leq i \leq m$, and define the induced relation on E: $\boldsymbol{x} \succeq \boldsymbol{x}'$ iff $\boldsymbol{f}(\boldsymbol{x}) \succeq \boldsymbol{f}(\boldsymbol{x}')$. We say that $\boldsymbol{x}$ dominates $\boldsymbol{x}'$ in this case. Note that $\succeq$ on E is not a partial order, but only a preorder, since it lacks antisymmetry (i.e., two different designs can have the same objectives).

Definition 1 (Pareto set) *The Pareto set $\Pi(\boldsymbol{f}(E)))$ in the objective space $\boldsymbol{f}(E) \subset \mathbb{R}^m$ is, as usual, the set of maximal points. Further, we call any set $\Pi(E) \subseteq E$ a Pareto set of E if it satisfies*

$$\boldsymbol{f}(\Pi(E)) = \Pi(\boldsymbol{f}(E))).$$

In words, $\Pi(E)$ is any set of designs that yields all optimal objectives. It is not unique since $\succeq$ on E is not antisymmetric.

1. Scalars and functions that return scalars are written unbolded; tuples and vectors are boldfaced.
2. Note that one may also approach multi-objective optimization via scalarization, see, e.g., Roijers et al. (2013). In our approach, we focus on retrieving an approximate Pareto-frontier, so that we do not have to commit to a particular family of scalarization functions.

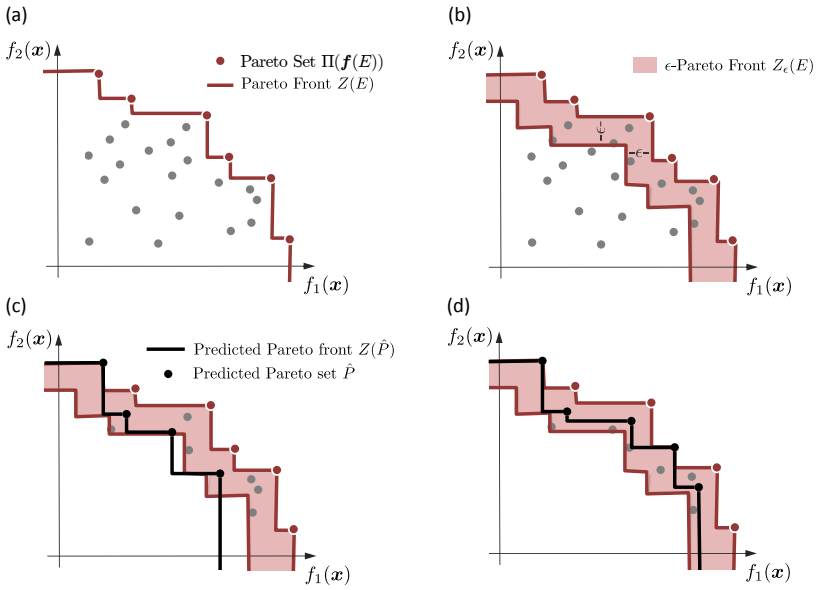

Figure 2: (a) Example of Pareto set and Pareto front for $m = 2$. (b) Example of an $\boldsymbol{\epsilon}$-Pareto front for $m = 2$. (c) Example of a predicted Pareto set that is not ϵ-accurate. (d) Example of a predicted Pareto set that is ϵ-accurate.

Definition 2 (Pareto front) *We define the Pareto front $Z(E)$ as the set of points in $\mathbb{R}^m$ that constitutes the surface of the space dominated by the Pareto set $\Pi(\boldsymbol{f}(E))$. Formally,*

$$Z(E) = \partial\{\boldsymbol{y} \in \mathbb{R}^m : \text{ there is an } \boldsymbol{x} \in E \text{ with } \boldsymbol{f}(\boldsymbol{x}) \succeq \boldsymbol{y})\}. \tag{1}$$

Hereby, the operator ∂ applied to a set Y denotes the boundary of Y.

Fig. 2(a) visualizes the concepts of Pareto set and Pareto front for $m = 2$.

2.3 $\boldsymbol{\epsilon}$-Pareto Optimality

We now relax the relation $\succeq$ in $\mathbb{R}^m$ and E for the purpose of our algorithm by adding a small tolerance. Consider a vector $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_d)$, with $\epsilon_i \geq 0$, $1 \leq i \leq m$. We say that a point $\boldsymbol{y} \in \mathbb{R}^m$ $\boldsymbol{\epsilon}$-dominates $\boldsymbol{y}'$, written as $\boldsymbol{y} \succeq_\epsilon \boldsymbol{y}'$, if $\boldsymbol{y} + \boldsymbol{\epsilon} \succeq \boldsymbol{y}'$. As before, we pull the relation to E: $\boldsymbol{x} \succeq_\epsilon \boldsymbol{x}'$ iff $\boldsymbol{f}(\boldsymbol{x}) \succeq_\epsilon \boldsymbol{f}(\boldsymbol{x}')$. Note that this relation is neither a partial order nor a preorder: antiysmmetry and transitivity do not hold.

We use this relation next to define an appropriate notion of $\boldsymbol{\epsilon}$-accurate Pareto set. To do so we first need two auxiliary definitions.

Definition 3 ($\boldsymbol{\epsilon}$-Pareto front) *We define the $\boldsymbol{\epsilon}$-Pareto front $Z_\epsilon(E)$ of E as the set of points between $Z(E)$ and $Z(E) - \boldsymbol{\epsilon}$, including the boundaries. Formally,*

$$Z_\epsilon(E) = \{\boldsymbol{y} \in \mathbb{R}^m : \boldsymbol{y}' \succeq \boldsymbol{y} \text{ for some } \boldsymbol{y}' \in Z(E) \text{ and } \boldsymbol{y} \succeq_\epsilon \boldsymbol{y}'' \text{ for some } \boldsymbol{y}'' \in Z(E)\}$$

Fig. 2(b) shows the $\boldsymbol{\epsilon}$-Pareto front associated with Fig. 2(a).

Definition 4 (ϵ-Pareto front covering) *We say that a nonempty set $C \subseteq Z_\epsilon(E)$ covers $Z_\epsilon(E)$, if for every point $\boldsymbol{y} \in Z_\epsilon(E)$ there is at least one point $\boldsymbol{y}' \in C$ s.t. $\boldsymbol{y}' \succeq_\epsilon \boldsymbol{y}$.*

In words, Definition 4 requires that the Pareto front spanned by $\Pi(C)$ is contained in $Z_\epsilon(E)$.

Definition 5 (ϵ-Accurate Pareto set) *We call a set of points $\Pi_\epsilon(E) \subseteq E$ an ϵ-accurate Pareto set of E, if $\boldsymbol{f}(\Pi_\epsilon(E))$ covers $Z_\epsilon(E)$.*

In words, for an ϵ-accurate Pareto set, the associated front is contained in the ϵ-Pareto front $Z_\epsilon(E)$ and it contains no suboptimal designs. As examples, the set with objectives $\hat{P}$ in Fig. 2(c) does not satisfy this property, but in Fig. 2(d) it does. Note that the ϵ-accurate Pareto set may have fewer points than the actual Pareto set.

An ϵ-accurate Pareto set $\Pi_\epsilon(E)$ is a natural approximate substitute of the original set E. Having access to it, one can rest assured that for any point on the Pareto front $Z(E)$ associated with E, there is some element $\boldsymbol{x} \in \Pi_\epsilon(E)$ which is at most ϵ worse according to all of the objectives. The high level goal of our algorithm ϵ-PAL is to find, with few evaluations $\boldsymbol{f}(\boldsymbol{x})$, a small ϵ-accurate Pareto set of E.

2.4 Gaussian Processes (GP)

ϵ-PAL models $\boldsymbol{f}$ as a draw from an m-variate Gaussian process (GP) distribution. A GP distribution over a real function $f(\boldsymbol{x})$ is fully specified by its mean function $\mu(\boldsymbol{x})$ and its covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ (Rasmussen and Williams, 2006). The *kernel* or covariance function k captures regularity in the form of the correlation of the marginal distributions $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$.

In our multi-objective setting, we model each objective function $f_i(\boldsymbol{x})$ as a draw from an independent[3] GP distribution.

On every iteration t in our algorithm we choose a design $\boldsymbol{x}_t$ to evaluate, which yields a noisy sample[4] $y_{t,i} = f_i(\boldsymbol{x}_t) + \nu_{t,i}$; after T iterations we have a vector $\boldsymbol{y}_{T,i} = (y_{1,i}, \ldots, y_{T,i})$. Assuming $\nu_{t,i} \sim N(0, \sigma^2)$ (i.i.d. Gaussian noise), the posterior distribution of f_i is a Gaussian process with mean $\mu_{T,i}(\boldsymbol{x})$, covariance $k_{T,i}(\boldsymbol{x}, \boldsymbol{x}')$, and variance $\sigma^2_{T,i}(\boldsymbol{x})$:

$$\mu_{T,i}(\boldsymbol{x}) = \boldsymbol{k}_{T,i}(\boldsymbol{x})^T (\boldsymbol{K}_{T,i} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}_{T,i}, \tag{2}$$

$$k_{T,i}(\boldsymbol{x}, \boldsymbol{x}') = k_i(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}_{T,i}(\boldsymbol{x})^T (\boldsymbol{K}_{T,i} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_{T,i}(\boldsymbol{x}'), \tag{3}$$

$$\sigma^2_{T,i}(\boldsymbol{x}) = k_{T,i}(\boldsymbol{x}, \boldsymbol{x}), \tag{4}$$

where $\boldsymbol{x}, \boldsymbol{x}' \in E$, $\boldsymbol{k}_{T,i}(\boldsymbol{x}) = (k_i(\boldsymbol{x}, \boldsymbol{x}_t))_{1 \leq t \leq T}$ and $\boldsymbol{K}_{T,i} = (k_i(\boldsymbol{x}_j, \boldsymbol{x}_\ell))_{1 \leq j, \ell \leq T}$. Note that this posterior distribution captures our uncertainty about $\boldsymbol{f}(\boldsymbol{x})$ for all points $\boldsymbol{x} \in E$.

2.5 Reproducing Kernel Hilbert Spaces (RKHS)

Using Gaussian processes to model the target functions f_i assumes that we know the prior from which they have been generated. This is rarely the case in practice. Therefore, for

3. Note that dependence amongst the outputs could be captured as well; e.g., Bonilla et al. (2008).
4. We use the term "sampling" to refer to evaluating a design via a noisy measurement.

our theoretical analysis we take a more agnostic approach in which we assume that f_i are arbitrary functions from the RKHS associated with kernel k.

The RKHS $\mathcal{H}_k(E)$ is a Hilbert space consisting of functions f on the domain E, endowed with an inner product $\langle \cdot, \cdot \rangle_k$ that satisfies the following properties with respect to a positive definite function k:

- For every $\boldsymbol{x} \in E$, $k(\boldsymbol{x}, \boldsymbol{x}')$ as a function of $\boldsymbol{x}'$ belongs to $\mathcal{H}_k(E)$.
- The reproducing property holds for k, i.e., $\langle f, k(\boldsymbol{x}, \cdot) \rangle_k = f(\boldsymbol{x})$.

The smoothness of the functions $f \in \mathcal{H}_k(E)$ with respect to k is encoded by the norm $||f||_k = \sqrt{\langle f, f \rangle_k}$. Functions with low norm are usually relatively smooth. In our case, as E is a finite set, any f is guaranteed to have bounded norm, i.e., $||f||_k < \infty$, as long as the kernel is universal (such as the Gaussian kernel).

2.6 Problem Statement

Let E be a finite set with a positive definite kernel. We wish to simultaneously optimize m objective functions $f_1, \ldots, f_m : E \to \mathbb{R}$, considering that evaluating $\boldsymbol{f}(\boldsymbol{x})$ for any $\boldsymbol{x} \in E$ is expensive. We wish to identify an $\boldsymbol{\epsilon}$-accurate Pareto set $\Pi_\epsilon(E) \subseteq E$ while minimizing the number of evaluations $\boldsymbol{f}(\boldsymbol{x})$.

In the following, we develop an active learning algorithm that iteratively and adaptively selects a sequence of designs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ to be evaluated, and that uses these evaluations along with the model's predictive uncertainty to predict an $\boldsymbol{\epsilon}$-accurate Pareto set of E. This iterative algorithm terminates when, with high probability, an $\boldsymbol{\epsilon}$-accurate Pareto set of E has been found, and therefore no more evaluations are needed. In addition, we theoretically analyze our algorithm and provide a bound on the number of evaluations required by the algorithm to generate an $\boldsymbol{\epsilon}$-accurate prediction.

3. $\boldsymbol{\epsilon}$-PAL Algorithm

In this section we describe our algorithm: $\boldsymbol{\epsilon}$-*Pareto Active Learning* ($\boldsymbol{\epsilon}$-PAL).

3.1 Overview

Our approach to predicting an $\boldsymbol{\epsilon}$-accurate Pareto set of E trains GP models on a small subset of E. The models predict the objective functions f_i, $1 \leq i \leq m$, allowing us to make statistical inferences about the Pareto-optimality of every point in E. The true value of $\boldsymbol{f}(\boldsymbol{x})$ is approximated by the models as $\hat{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{\mu}(\boldsymbol{x}) = (\mu_i(\boldsymbol{x}))_{1 \leqslant i \leqslant m}$. Additionally $\boldsymbol{\sigma}(\boldsymbol{x}) = (\sigma_i(\boldsymbol{x}))_{1 \leqslant i \leqslant m}$ is interpreted as the uncertainty of this prediction. We capture this uncertainty through the hyper-rectangle[5]

$$Q_{\boldsymbol{\mu},\boldsymbol{\sigma},\beta}(\boldsymbol{x}) = \{\boldsymbol{y} : \boldsymbol{\mu}(\boldsymbol{x}) - \beta^{1/2}\boldsymbol{\sigma}(\boldsymbol{x}) \preceq \boldsymbol{y} \preceq \boldsymbol{\mu}(\boldsymbol{x}) + \beta^{1/2}\boldsymbol{\sigma}(\boldsymbol{x})\}, \tag{5}$$

where β is a scaling parameter to be chosen later.

The goal of the algorithm is to return a set $\hat{P} \subseteq E$ such that, with high probability, $\hat{P}$ is an $\boldsymbol{\epsilon}$-accurate Pareto set of E. $\hat{P}$ may contain points that are not sampled yet, but

5. conservatively bounding the ellipsoid with radii σ_i.

probabilistically, it is guaranteed to cover $Z_\epsilon(E)$, i.e., with high probability, all points in E are $\boldsymbol{\epsilon}$-dominated by a point in $\hat{P}$.

$\boldsymbol{\epsilon}$-PAL iterates over four stages: modeling, discarding, $\boldsymbol{\epsilon}$-Pareto front covering, and sampling. It stops when all remaining points have been either sampled, discarded or determined to belong to $Z_\epsilon(E)$ with high probability. The following sections explain each of these stages, and Alg. 1 presents the corresponding pseudocode.

The algorithm maintains two working sets, indexed by the iteration number t, starting at 0:

- U_t: undecided points, $U_0 = E$,
- P_t: points predicted to be members of an $\boldsymbol{\epsilon}$-accurate Pareto set of E, $P_0 = \emptyset$.

On the first iteration ($t = 0$), all points in E are copied to a set U_0. Subsequently, on any iteration t, a point to be sampled is chosen from U_t or P_t. Points in U_t that with high probability are $\boldsymbol{\epsilon}$-dominated by another point are discarded, i.e., removed from U_t, and points that are determined to belong to $Z_\epsilon(E)$ are moved from U_t to set P_t. The algorithm terminates at some iteration T when U_T is empty. Then $\hat{P} = P_T$ is returned. $\boldsymbol{\epsilon}$-PAL guarantees that with high probability $\hat{P}$ is an $\boldsymbol{\epsilon}$-accurate Pareto set of E.

3.2 Modeling

$\boldsymbol{\epsilon}$-PAL uses Gaussian process inference to predict the mean vector $\boldsymbol{\mu}_t(\boldsymbol{x})$ and the standard deviation vector $\boldsymbol{\sigma}_t(\boldsymbol{x})$ of any point $\boldsymbol{x} \in E$ that has not been discarded yet. This prediction is generated based on the noisy samples obtained. Each point $\boldsymbol{x} \in P_t \cup U_t$ is then assigned its *uncertainty region*, which is the hyperrectangle

$$R_t(\boldsymbol{x}) = R_{t-1}(\boldsymbol{x}) \cap Q_{\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t, \beta_{t+1}}(\boldsymbol{x}), \tag{6}$$

where β_{t+1} is a positive value that defines how large this region is in proportion to $\boldsymbol{\sigma}_t$. In Section 4 we will suggest a value for this parameter. The iterative intersection ensures that all uncertainty regions are non-increasing with t. Intuitively, for a point $\boldsymbol{x} \in E$ that has not been evaluated yet, we can say that with high probability $\boldsymbol{f}(\boldsymbol{x}) \in R_t(\boldsymbol{x})$.

Within $R_t(\boldsymbol{x})$, the pessimistic and optimistic outcomes are $\min(R_t(\boldsymbol{x}))$ and $\max(R_t(\boldsymbol{x}))$, respectively, both taken in the partial order $\preceq$ and unique. Fig. 3(a) shows this with an example. Notice that points that have been evaluated also have an uncertainty region, since we only have access to noisy samples on those points.

3.3 Discarding

The goal of this stage is to discard points in U_t that with high probability are $\boldsymbol{\epsilon}$-dominated by another point in E, while making sure that at least one of such dominating points ends up in P_T.

Under uncertainty, a point $\boldsymbol{x}$ is $\boldsymbol{\epsilon}$-dominated by another point $\boldsymbol{x}'$, with high probability, if the pessimistic outcome of $\boldsymbol{x}'$ ($\min(R_t(\boldsymbol{x}'))$) $\boldsymbol{\epsilon}$-dominates the optimistic outcome of $\boldsymbol{x}$ ($\max(R_t(\boldsymbol{x}))$):

$$\max(R_t(\boldsymbol{x})) \preceq_\epsilon \min(R_t(\boldsymbol{x}')).$$

In this case, $\boldsymbol{x}$ can be discarded, i.e., removed from U_t.

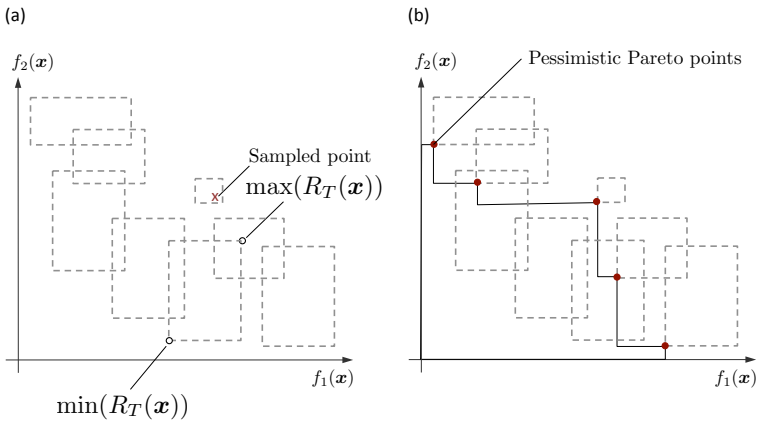

Figure 3: (a) Example of $\min(R_t(\boldsymbol{x}))$ and $\max(R_t(\boldsymbol{x}))$ for $m = 2$. (b) Example of a set of pessimistic Pareto points for $m = 2$.

If this relationship does not hold, either the uncertainty regions are still too large to draw any conclusions, or $\boldsymbol{x}$ and $\boldsymbol{x}'$ are not comparable.

To check if a point is $\boldsymbol{\epsilon}$-dominated by another one in E, $\boldsymbol{\epsilon}$-PAL compares its relationship with all *pessimistic Pareto points*.

Definition 6 (Pessimistic Pareto set) *For a subset $D \subseteq E$, we define $p_{pess}(D)$, or the* pessimistic Pareto set *of D, as the set of points $\boldsymbol{x} \in D$ for which there is no other point $\boldsymbol{x}' \in D$ such that*

$$\min(R_t(\boldsymbol{x})) \preceq \min(R_t(\boldsymbol{x}')).$$

Fig. 3(b) illustrates this situation.

We discard any point $\boldsymbol{x} \in U_t \setminus p_{\text{pess}}(P_t \cup U_t)$ if $\boldsymbol{x} \preceq_\epsilon \boldsymbol{x}'$ for some $\boldsymbol{x}' \in p_{\text{pess}}(P_t \cup U_t)$. This ensures that no element in $p_{\text{pess}}(S_t \cup P_t \cup U_t)$ is discarded and therefore, for every point that is discarded there will always be a point in the working sets that $\boldsymbol{\epsilon}$-dominates it.

A point in $p_{\text{pess}}(P_t \cup U_t)$ is only discarded if it is $\boldsymbol{\epsilon}$-dominated by any point in P_t, since all points in P_t are already guaranteed to belong to the returned set $\hat{P}$. Points in P_t are never discarded.

3.4 $\boldsymbol{\epsilon}$-Pareto Front Covering

The goal of $\boldsymbol{\epsilon}$-PAL is to empty U_t as fast as possible, i.e., with the least amount of points sampled and with the least amount of points in P_t needed to cover the $\boldsymbol{\epsilon}$-Pareto front of E. As previously stated, a point is moved to P_t if it can be determined that with high probability it belongs to $Z_\epsilon(E)$.

Therefore, care must be taken not to sample when all points in U_t can be either discarded or moved to P_t. If there is at least one point in U_t that cannot be discarded or moved to P_t, we proceed with sampling.

Definition 7 ($\boldsymbol{x}$ belongs to $\boldsymbol{Z_\epsilon(E)}$ with high probability) *We can determine that with high probability $\boldsymbol{x} \in E$ belongs to $Z_\epsilon(E)$ if there is no other point $\boldsymbol{x'} \in P_t \cup U_t$ such that*

$$\max(R_t(\boldsymbol{x'})) \succeq_\epsilon \min(R_t(\boldsymbol{x})).$$

In this stage we attempt to update the set P_t, to make progress in covering the $\boldsymbol{\epsilon}$-Pareto front of E, until we find a point in U_t that cannot be moved to P_t. After a point is moved to P_t, we check what points in U_t are $\boldsymbol{\epsilon}$-dominated by it, and therefore can be discarded. Details on this procedure can be found in Alg. 1.

3.5 Sampling

As long as $U_t \neq \emptyset$, on each iteration, a new point $\boldsymbol{x}_t$ is selected for sampling with the following selection rule. Each point $\boldsymbol{x} \in U_t \cup P_t$ is assigned a value

$$w_t(\boldsymbol{x}) = \max_{\boldsymbol{y},\boldsymbol{y'} \in R_t(\boldsymbol{x})} ||\boldsymbol{y} - \boldsymbol{y'}||_2,$$

which is the diameter of its uncertainty region $R_t(\boldsymbol{x})$. Amongst these points, the one with the largest $w_t(\boldsymbol{x})$ (i.e., most uncertain) is chosen as the next sample $\boldsymbol{x}_t$ to be evaluated. We refer to $w_t(\boldsymbol{x}_t)$ as $\overline{w}_t$. Note that our approach does not simply pick the most uncertain point over the whole space E, but only over the set $U_t \cup P_t$, i.e., the points that (in a statistically plausible manner) might be useful in constructing an $\boldsymbol{\epsilon}$-accurate Pareto set. This is similar to exploration–exploitation tradeoffs commonly encountered in Bayesian optimization.

3.6 Stopping Criteria

The iterations terminate at $t = T$ when $U_T = \emptyset$, i.e., all points have been either moved to P_T or have been discarded. The predicted set $\hat{P} = P_T$ is guaranteed to be an $\boldsymbol{\epsilon}$-accurate Pareto set of E. Termination occurs at the latest when $\overline{w}_t \leq \epsilon$. In this case, dominated points can be discarded in one pass.

3.7 Analysis of Execution Time

In this section, we review the most time consuming subroutines of $\boldsymbol{\epsilon}$-PAL and analyze their computational cost, which depends on the number of points in the working sets: $n_t = |U_t| + |P_t|$. While $n_0 = |E|$ in the first iteration, discarded points are removed from the working sets and thus the number of computations is reduced as t increases. As it is not possible to make conclusions in how n_t decreases with t, we assume the upper bound $n = |E|$. In practice, the number of computations is much lower for most iterations of the algorithm. On the other hand, we assume that the number m of objective functions is a small constant.

In the modeling stage, $\boldsymbol{\epsilon}$-PAL updates Gaussian process models and evaluates the prediction μ and the uncertainty σ for every point. Let s be the number of points that have been sampled, the complexity of this step is $O(s^3 + ns^2)$ (Rasmussen and Williams, 2006). Note that typically $s \ll n$.

In the discarding stage, $\boldsymbol{\epsilon}$-PAL first finds the Pareto pessimistic points of a set. The problem of efficiently determining the Pareto points (or maxima) of a set of vectors is

Algorithm 1 The $\boldsymbol{\epsilon}$-PAL algorithm

Input: design space E ($|E| = n$); GP prior $\mu_{0,i}, \sigma_0, k_i$ for all $1 \leq i \leq n$; $\boldsymbol{\epsilon}$; β_t for $t \in \mathbb{N}$
Output: predicted Pareto set $\hat{P}$

$P_0 = \emptyset$ {predicted set}, $U_0 = E$ {undecided set}
$R_0(\boldsymbol{x}) = \mathbb{R}^n$ for all $\boldsymbol{x} \in E$, $t = 1$
repeat
  $P_t = P_{t-1},\ U_t = U_{t-1}$
  ——— *Modeling* ———
  Obtain $\boldsymbol{\mu}_t(\boldsymbol{x})$ and $\boldsymbol{\sigma}_t(\boldsymbol{x})$ for all $\boldsymbol{x} \in P_t \cup U_t$
  $R_t(\boldsymbol{x}) = R_{t-1}(\boldsymbol{x}) \cap Q_{\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t, \beta_{t+1}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in P_t \cup U_t$
  ——— *Discard* ———
  $p_{\mathrm{pess}}(P_t)$ = Pareto pessimistic set of P_t
  discard points $\boldsymbol{x}$ in U_t that are $\boldsymbol{\epsilon}$-dominated by some point $\boldsymbol{x}'$ in $p_{\mathrm{pess}}(P_t)$, i.e., where $\max(R_t(\boldsymbol{x})) \preceq_\epsilon \min(R_t(\boldsymbol{x}'))$ (See Alg. 2 for $m = 2$)
  $p_{\mathrm{pess}}(P_t \cup U_t)$ = Pareto pessimistic set of P_t and U_t
  discard points $\boldsymbol{x}$ in $U_t \setminus p_{\mathrm{pess}}(P_t \cup U_t)$ that are $\boldsymbol{\epsilon}$-dominated by some point $\boldsymbol{x}'$ in $p_{\mathrm{pess}}(P_t \cup U_t)$, i.e., where $\max(R_t(\boldsymbol{x})) \preceq_\epsilon \min(R_t(\boldsymbol{x}'))$. (See Alg. 2 for $m = 2$)
  ——— $\boldsymbol{\epsilon}$*-Pareto Front Covering* ———
  repeat
    Choose $\boldsymbol{x}' = \arg\max_{x \in U_t} \{w_t(\boldsymbol{x})\}$
    if for all $\boldsymbol{x} \in P_t \cup U_t \setminus \{\boldsymbol{x}'\}$ it holds that $\max(R_t(\boldsymbol{x})) \preceq_\epsilon \min(R_t(\boldsymbol{x}'))$ **then**
      $P_t = P_t \cup \{\boldsymbol{x}'\}$, $U_t = U_t \setminus \{\boldsymbol{x}'\}$
    else
      break
    end if
  until $U_t = \emptyset$
  ——— *Sampling* ———
  Choose $\boldsymbol{x}_t = \arg\max_{x \in U_t \cup P_t} \{w_t(\boldsymbol{x})\}$
  Sample $\boldsymbol{y}_t(\boldsymbol{x}_t) = \boldsymbol{f}(\boldsymbol{x}_t) + \boldsymbol{\nu}_t$
  $t = t + 1$
until $U_t = \emptyset$
return $\hat{P} = P_t$

Algorithm 2 Discard points in U that are $\boldsymbol{\epsilon}$-dominated by p_{pess} for $m = 2$.

Input: set U; Pareto pessimistic set p_{pess}; $\boldsymbol{\epsilon}$
Output: updated set U

$U' = \{(\boldsymbol{x}, \min(R_t(\boldsymbol{x})) + \boldsymbol{\epsilon}) : \boldsymbol{x} \in p_{\mathrm{pess}}\} \cup \{(\boldsymbol{x}, \max(R_t(\boldsymbol{x}))) : \boldsymbol{x} \in U \setminus p_{\mathrm{pess}}\}$ {U' is a set of $(\boldsymbol{x}, (\hat{f}_1, \hat{f}_2))$ pairs}
$sortU$ = sort U' with respect to $\hat{f}_1$ in ascending order
$currentMax = -\infty$
for $(\boldsymbol{x}, (\hat{f}_1, \hat{f}_2))$ in $sortU$ **do**
  if $\boldsymbol{x} \in p_{\mathrm{pess}}$ **then**
    $currentMax = \hat{f}_2$
  else
    if $\hat{f}_2 \leq currentMax$ **then**
      $U = U \setminus \{\boldsymbol{x}\}$ {discard $\boldsymbol{x}$}
    end if
  end if
end for

addressed by Kung et al. (1975). For $m = 2$, their algorithm starts by ordering the list of vectors in the first dimension f_1. Then, the list is traversed in ascending order while discarding the vectors for which the second dimension f_2 is smaller than the greatest value

of f_2 seen so far. This procedure exhibits computational complexity of $O(n \log n)$ for $m = 2$. The same computational complexity is achieved by a similar implementation for $m = 3$. For $m > 3$, Kung et al. (1975) presents a divide and conquer algorithm with worst case asymptotic complexity of $O(n(\log n)^{m-2}) + O(n \log n)$.

In the discarding stage, $\boldsymbol{\epsilon}$-PAL also discards points in a set U that are $\boldsymbol{\epsilon}$-dominated by a set p_{pess}. This means that, if done naively, every point $\max(\boldsymbol{x})$ with $\boldsymbol{x} \in U$ must be compared with every $\min(\boldsymbol{x}') + \boldsymbol{\epsilon}$ with $\boldsymbol{x}' \in p_{\text{pess}}$. However, this can be done by adapting the above mentioned implementations to compute the Pareto points of a set of vectors to substantially reduce the runtime of the algorithm. Alg. 2 shows the pseudocode for this adaptation. A joined set U' is created from points $\max(\boldsymbol{x})$ with $\boldsymbol{x} \in U$ and $\min(\boldsymbol{x}') + \boldsymbol{\epsilon}$ with $\boldsymbol{x}' \in p_{\text{pess}}$. This joint list is ordered with respect to the first dimension f_1. Then, the list is traversed in ascending order, but the maximum value of f_2 is only updated by a point that belongs to p_{pess}, and only points that belong to U can be discarded. For $m \geq 3$ the adaptation of the algorithm by Kung et al. (1975) is analogous.

In the $\boldsymbol{\epsilon}$-Pareto front covering stage, $\boldsymbol{\epsilon}$-PAL checks if a point $\boldsymbol{x}$ can be guaranteed to belong to $Z_\epsilon(E)$ according to Definition 7. This is done by comparing $\boldsymbol{x}$ with the rest of the points $\boldsymbol{x}'$ in the working sets, until a case in which $\min(R_t(\boldsymbol{x}')) + \boldsymbol{\epsilon} \preceq \max(R_t(\boldsymbol{x}))$ is found. The number of computations for this step is $O(n)$; in practice, however, it is much smaller than n for most iterations. This operation might be done several times per iteration, until no more points in U_t can be moved to P_t or $U_t = \emptyset$. The number of times is assumed to be much smaller than n.

The computational complexity of an iteration is therefore dependent on m and s. For $m = 2$ and $m = 3$ it is $O(ns^2 + s^3 + n \log n)$, and for $m > 3$ it is $O(n(\log n)^{m-2}) + O(ns^2 + s^3 + n \log n)$.

4. Correctness and Sample Complexity

So far we have left the parameter β_t undefined. In this section, we will choose a β_t that guarantees the correctness of the algorithm, and the number of iterations required to meet our bounds.

Of key importance in the convergence analysis is the effect of the regularity imposed by the kernel function k. In our analysis, this effect is quantified by the *maximum information gain* associated with the GP prior. Formally, we consider the information gain

$$I(\boldsymbol{y}_1 \dots \boldsymbol{y}_T; \boldsymbol{f}) = H(\boldsymbol{f}) - H(\boldsymbol{f} \mid \boldsymbol{y}_1 \dots \boldsymbol{y}_T),$$

i.e, the reduction of uncertainty on $\boldsymbol{f}$ caused by (noisy) observations of $\boldsymbol{f}$ on the T first sampled points. The crucial quantity governing the convergence rate is

$$\gamma_T = \max_{\boldsymbol{y}_1 \cdots \boldsymbol{y}_T} I(\boldsymbol{y}_1 \dots \boldsymbol{y}_T; \boldsymbol{f}),$$

i.e., the maximal reduction of uncertainty achievable by sampling T points. Intuitively, if the kernel k imposes strong regularity (smoothness) on $\boldsymbol{f}$, few samples suffice to gather much information about $\boldsymbol{f}$, and as a consequence γ_T grows sub-linearly (exhibits a strong diminishing returns effect). In contrast, if k imposes little regularity (e.g., is close to diagonal), γ_T grows almost linearly with T. Srinivas et al. (2010, 2012) established γ_T as

key quantity in bounding the regret in single-objective GP optimization. Here, we show that this quantity more broadly governs convergence in the much more general problem of predicting the Pareto-optimal set in multi-criterion optimization.

We obtain bounds on the number of iterations required when assuming that the target functions f_i, $0 \leq i \leq m$, lie in the RKHS $\mathcal{H}_k(E)$ corresponding to kernel $k(\boldsymbol{x}, \boldsymbol{x}')$, and that the noise ν_t is an arbitrary martingale difference sequence which has zero mean and is uniformly bounded by σ. Furthermore, in order to obtain the bounds, it is necessary to specify an upper bound[6] B on all $||f_i||_k$.

The following theorem constitutes our main theoretical result.

Theorem 1 *Assume that the true functions f_i lie in the RKHS $\mathcal{H}_k(E)$ corresponding to kernel $k(\boldsymbol{x}, \boldsymbol{x}')$, and that the noise ν_t has zero mean conditioned on the history and is bounded by σ almost surely. Let $\delta \in (0,1)$ and $||f_i||_k^2 \leq B$. Running $\boldsymbol{\epsilon}$-PAL with $\beta_t = 2B^2 + 300\gamma_t \log^3(m|E|t/\delta)$, prior $GP(0, k(\boldsymbol{x}, \boldsymbol{x}'))$ and noise $N(0, \sigma^2)$, the following holds with probability $1 - \delta$.*

An $\boldsymbol{\epsilon}$-accurate Pareto set can be obtained after at most T iterations, where T is the smallest number satisfying

$$\sqrt{\frac{C_1 \beta_T \gamma_T}{T}} \geq \epsilon. \tag{7}$$

Here, $C_1 = 8/\log(1 + \sigma^{-2})$, $\epsilon = ||\boldsymbol{\epsilon}||_\infty$, and γ_T depends on the type of kernel used.

This means that, with high probability, $\overline{w}_t$ is bounded by $O^*((B\sqrt{\gamma_T} + \gamma_T)/\sqrt{T})$. Note that in practice one might want to, in addition to *identifying* an $\boldsymbol{\epsilon}$-accurate Pareto front, seek to be sufficiently confident such that for all predicted Pareto points (i.e., points $\mathbf{x}$ in P_T in Algorithm 1) it holds that $w_T(\mathbf{x}) \leq \epsilon$. This requirement can be added to the stopping condition in Line 26 of Algorithm 1, and the same sample complexity of Theorem 1 holds.

4.1 Proof Outline

The above theorem implies that by specifying δ and a target accuracy $\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon}$-PAL automatically stops when the target error is achieved with confidence $1-\delta$. Additionally, the theorem bounds the number of iterations T required to obtain this result.

Our strategy for the proof consists of four parts. First, Lemma 1 shows that $|f_i(\boldsymbol{x}) - \mu_{t-1,i}(x)| \leq \beta_t^{1/2} \sigma_{t-1,i}(\boldsymbol{x})$ for $1 \leq i \leq m$, $t \geq 1$, and for all $\boldsymbol{x} \in E$. Then, we analyze how $\overline{w}_t$ decreases with t. Subsequently, we relate $\boldsymbol{\epsilon}$ and $\overline{w}_t$ to ensure the termination of the algorithm. The last two parts are analyzed in Section 4.2. Finally, Lemma 8 supports the accuracy of the $\boldsymbol{\epsilon}$-PAL, given that the proper β_t value is used on every iteration t.

4.2 Reduction in Uncertainty

The first step of the proof is to show that with probability at least $1 - \delta$, $\boldsymbol{f}(\boldsymbol{x})$ lies for all $\boldsymbol{x} \in E$ within the uncertainty region (see (5) and (6)):

$$R_t(\boldsymbol{x}) = Q_{\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0, \beta_1}(\boldsymbol{x}) \cap \cdots \cap Q_{\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t, \beta_{t+1}}(\boldsymbol{x}),$$

6. While in practice this might not be possible, a standard guess-and-doubling approach can be applied.

which is achieved by choosing $\beta_t = 2B^2 + 300\gamma_t \log^3(m|E|t/\delta)$.

Srinivas et al. (2010) showed that information gain can be expressed in terms of the predicted variances. Similarly, we show how the cumulative $\sum_{k=1}^{t} \overline{w}_k$ can also be expressed in terms of maximum information gain γ_t. Since the sampling rules used by $\boldsymbol{\epsilon}$-PAL guarantee that $\overline{w}_t$ decreases with t, we get the following bound for $\overline{w}_t$. With probability $\geq 1-\delta$,

$$\overline{w}_t \leq \sqrt{\frac{C_1 \beta_t \gamma_t}{t}} \text{ for all } t \geq 1, \tag{8}$$

where $C_1 = 8/\log(1+\sigma^{-2})$ and β_t is as before.

The proof is supported by Lemmas 1 to 5 found in the appendix of this paper. Key challenges and differences in comparison to Srinivas et al. (2010) include (1) dealing with multiple objectives; (2) the use of a different sampling criterion; and (3) incorporating the monotonic classification scheme.

We also show that, with probability $1-\delta$, the algorithm terminates with no further sampling at iteration T if

$$\overline{w}_T \leq \epsilon, \tag{9}$$

where $\epsilon = ||\boldsymbol{\epsilon}||_\infty = \max(\epsilon_i)_{1 \leq i \leq m}$. This is proved in Lemma 7.

4.3 Explicit Bounds for the Squared Exponential Kernel

Theorem 1 holds for general covariance functions $k(\boldsymbol{x}, \boldsymbol{x}')$. Srinivas et al. (2010) derived bounds for γ_T depending on the choice of kernel. These can be used to specialize Theorem 1.

We illustrate this using the squared exponential kernel as example, i.e., $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(l^{-2}\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2\right)$ for some $l > 0$.

Let $E \subset \mathbb{R}^d$ be compact and convex, $d \in \mathbb{N}$. According to Srinivas et al. (2010), for $m = 1$, there exists a constant K such that

$$\gamma_t \leq K \log^{d+1} t \text{ for all } t > 1.$$

For $m > 1$, since we assume i.i.d. GPs, we thus get

$$\gamma_t \leq Km \log^{d+1} t$$

and hence the following corollary to Theorem 1.

Corollary 1 *Let k_i be the squared exponential kernel used by $\boldsymbol{\epsilon}$-PAL, and assume f_i lies in the RKHS with its norm bounded by $||f_i||_k^2 \leq B$, for all $1 \leq i \leq m$. When choosing $\delta \in (0,1)$, a target accuracy specified by $\boldsymbol{\epsilon}$, the following holds with probability $1-\delta$. $\boldsymbol{\epsilon}$-PAL terminates after at most T iterations, where T is the smallest number satisfying*

$$\frac{\sqrt{16B^2K^2m\log^{d+1} T + 2400K^2m^2\log^{2(d+1)} T \log^3(m|E|T/\delta)}}{\sqrt{T\log(1+\sigma^{-2})}} \geq \epsilon.$$

These results suggests that, under both scenarios, ϵ increases as T decreases in the following manner: Asymptotically, for any $\rho > 0$, as well as fixed m, n and d, we have $T = O(\frac{1}{\epsilon^{2+\rho}})$.

5. Discussion and Implementation Details

We now discuss some of the limitations, choices made, and other aspects that arose when implementing and using our $\boldsymbol{\epsilon}$-PAL algorithm.

5.1 Sampling Strategy

After clearly non-competitive designs are removed in the discarding phase, our sampling strategy (Section 3.5) chooses the one with the highest uncertainty. Other choices would be possible, e.g, chosing the one that maximizes the expected gain in hypervolume. Our choice aims at reducing time to termination, which is achieved by reducing uncertainty in the model as fast as possible.

5.2 Parameterization

For practical usage, two parameters, namely $\boldsymbol{\epsilon}$ and δ, need to be specified. These parameters relate to the desired level of accuracy of the prediction. Our theoretical bounds are likely to be loose in practice; thus, it may be useful to choose more "aggressive" values than recommended by the theory. The choice of δ impacts the value of β_t and therefore the convergence rate of the algorithm, since the latter scales the uncertainty regions $R_t(\boldsymbol{x})$. Since the analysis is conservative, scaling down β_t, possibly to be constant, is a viable option.

In contrast, the choice of $\boldsymbol{\epsilon}$ should pose no problem. One only may consider scaling the objective functions so that all ϵ_i components of $\boldsymbol{\epsilon}$ have comparable values.

5.3 Kernel Hyper-Parameters

So far, we have assumed that the kernel function is given. Usually, its parameters need to be chosen. Therefore, prior to running $\boldsymbol{\epsilon}$-PAL, it may be practical to randomly sample a small fraction of the design space and to optimize the parameters (e.g., by maximizing the marginal likelihood). One may also consider maintaining uncertainty in the hyper-parameters by placing and updating priors on them. These can then be marginalized to obtain suitable hyper-rectangles to capture the uncertainty about the designs. This extension poses new challenges in computational efficiency (i.e., using approximate Bayesian inference), and we defer it to future work.

5.4 Scalability

The computational complexity of our algorithm depends on our ability to perform efficient inference in Gaussian processes. In general, this complexity grows cubically with the number of samples (function evaluations), which can pose a challenge in large design spaces. This is a general issue in Bayesian optimization, and not specific to our approach. Fortunately, there are many techniques available for scaling up Gaussian process inference that our approach can immediately benefit from (see, e.g., Rasmussen and Williams (2006)).

5.5 Continuous vs. discrete domains

In our approach, we have focused on finite, discrete domains E, which often naturally arise in design-space exploration problems as those considered in our experiments. For Lipschitz-continuous objectives (which are commonly assumed in Bayesian optimization), it is be possible to handle continuous compact domains E via standard covering arguments. Given the allowed tolerances $\boldsymbol{\epsilon}$ and Lipschitz constants, one can construct a discretization $\hat{E}$ such that for any point $\boldsymbol{x} \in E$ in the domain, there is one in the discretization $\boldsymbol{x}' \in \hat{E}$ such that $\boldsymbol{f}(\boldsymbol{x}') \succeq_\epsilon \boldsymbol{f}(\boldsymbol{x})$. It can be seen that any $\boldsymbol{\epsilon}$-accurate Pareto set of $\hat{E}$ is a $2\boldsymbol{\epsilon}$-accurate Pareto set of E. This discretization-based approach becomes impractical with higher dimensions. Developing a variant of our approach that does not require an explicit discretization is an interesting direction for future work.

5.6 Current Implementation

For our experimental evaluation, we implemented a prototype that is restricted to $m = 2$ objectives. The code is available at Zuluaga et al. (2015).

6. Related Work

We now discuss different lines of related work.

6.1 Evolutionary Algorithms

One class of approaches uses evolutionary algorithms to approximate the Pareto frontier via a population of evaluated designs that is iteratively evolved (Künzli et al., 2005; Coello et al., 2006; Zitzler et al., 2002). Most of these approaches do not use models for the objectives, and consequently cannot make predictions about unevaluated designs. As a consequence, a large number of evaluations are typically needed for convergence with reasonable accuracy. To overcome this challenge, model-based (or "response surface") approaches approximate the objectives by models, which are fast to evaluate. A concrete example is the algorithm presented by Emmerich et al. (2006), which uses Gaussian random fields metamodels to predict the objective functions. These models are used to screen each generation of the population and to extract promising individuals. On the other hand, Laumanns and Ocenasek (2002) and Buche et al. (2005) investigate the use of Gaussian models to maintain and improve the diversity of the population during the offspring-generation phase.

We will compare against ParEGO (Knowles, 2006) (explained in Section 6.6 below), which improves on the above work by combining the evolutionary approach with optimization to reduce the number of evaluations needed.

6.2 Scalarization to the Single-Objective Setting

An alternative approach to multi-objective optimization problems is the reduction to a single-objective problem (for which a wealth of methods are available). This is commonly done via scalarization, for example by considering convex combinations of the objective functions (Boyd and Vandenberghe, 2004). Some evolutionary algorithms use this approach to tackle multiple objectives. For example, the approach presented by Zhang and Li (2007)

optimizes simultaneously several single-objective subproblems that are created using different scalarization criteria. The diversity amongst the subproblems will lead to diversity in the approximated Pareto front.

A number of global optimization algorithms are also conditioned to the multi-objective setting by using scalarization. As a concrete example, Zhang et al. (2010) and Knowles (2006) extend the single-objective efficient global optimization (EGO) approach of Jones et al. (1998) by decomposing the optimization problem into several single-objective subproblems. A GP model is built for every subproblem, and sample candidates are selected based on their expected improvement.

A major disadvantage of the scalarization approach is that without further assumptions (e.g., convexity) on the objectives, not all Pareto-optimal solutions can be recovered (Boyd and Vandenberghe, 2004). Therefore, we avoid scalarization in our approach.

6.3 Heuristics-based Methods

Instead of weighted combinations, numerous domain-specific heuristics have been proposed that aim at identifying Pareto-optimal solutions. These approaches typically combine search algorithms to suit the nature of the problem (Deng et al., 2008; Palermo et al., 2009; Zuluaga et al., 2012a) and defy theoretical analysis to provide bounds on the sampling cost. With this work we aim at creating a method that generalizes across a large range of applications and target scenarios and that is analyzable, i.e., comes with theoretical guarantees.

6.4 Single-Objective Bayesian Optimization and Active Learning

In the single-objective setting, there has been much work on active learning, in particular for classification (see, e.g., Settles (2010) for an overview). For optimization, model-based approaches are used to address settings where the objective is noisy and expensive to evaluate. In particular in Bayesian optimization (see the survey by Brochu et al. (2010)), the objective is modeled as a draw from a stochastic process (often a Gaussian process), as originally proposed by Mockus et al. (1978). The advantage of this approach is the flexibility in encoding prior assumptions (e.g., via choice of the kernel and likelihood functions), as well as the ability to guide sampling: several different (usually greedy) heuristic criteria have been proposed to pick the next sample based on the predictive uncertainty of the Bayesian model. A common example is the EGO approach of Jones et al. (1998), which uses the expected improvement. In recent years, there have been considerable theoretical advances in Bayesian optimization. Several analyses focus on the case of deterministic (i.e., noise-free) observations. Vazquez and Bect (2010) prove that under some conditions the EGO approach produces a dense sequence of samples in the domain, i.e., asymptotically getting arbitrarily close to the optimum. Bull (2011) go further in providing convergence rates for this setting. Srinivas et al. (2010) analyzed the GP-UCB criterion (Cox and John, 1997), and proved global convergence guarantees and rates for Bayesian optimization allowing noisy observations. We build on their results to establish guarantees about our ϵ-PAL algorithm in the multi-objective setting. de Freitas et al. (2012) improve the results of Srinivas et al. (2010) using a UCB-like algorithm under deterministic observations to obtain exponential regret bounds.

6.5 Multi-Objective Bayesian Optimization and Active Learning

The algorithm PAL by Zuluaga et al. (2013), is an earlier version of ϵ-PAL. It also requires a parameter ϵ that allows the user to set different levels of prediction accuracy. PAL attempts to classify points as Pareto optimal or not Pareto-optimal until all points have been classified; it uses ϵ to ease this classification. However, it fails to generate an ϵ-accurate Pareto set. It only uses ϵ to stop the algorithm in different stages of training, and as a result, less accuracy is achieved when the value of ϵ is increased.

There are two main advantages of ϵ-PAL over PAL. The more effective use of ϵ is one of them. This not only allows it to generate an ϵ-accurate Pareto set, but it also reduces the runtime of the algorithm, as it attempts to remove redundancy and discards points more efficiently along the execution of the algorithm. In addition, the convergence bounds are also defined in terms of ϵ, which is intuitive. In contrast, PAL uses the hypervolume error, a concept that is more difficult to reason about.

The other advantage is the asymptotic improvement in the number of computations required per iteration, which is independent to the improvements just mentioned above. As analyzed in Section 3.7, for problems with two or three objective functions, ϵ-PAL requires $O(n \log n)$ computations on every iteration, whereas PAL requires $O(n^2 \log n)$. For problems with more than two objective functions, ϵ-PAL requires $O(n(\log n)^{m-2}) + O(n \log n)$ computations on every iteration, whereas PAL requires $O(n^2(\log n)^{m-2}) + O(n \log n)$. Here, n is the number of elements in the design space. This is very important since it allows ϵ-PAL to handle larger design spaces.

There have been recent approaches that simplify to some extent the PAL algorithm. Campigotto et al. (2014) proposes an active learning algorithm that also uses Gaussian process modeling to predict the objective function. It iteratively samples the most uncertain point in the design space until a threshold in information gain is reached. On the other hand, Steponavice et al. (2014) propose an algorithm that probabilistically classifies points as Pareto-optimal or not. This probability is used to provide flexibility on the accuracy of the prediction. This algorithm does not use Gaussian process modeling with the goal of improving runtime. Neither of these two approaches provides an approximation to the Pareto front with some granularity as ϵ-PAL does. Also, they do not provide any theoretical support on the convergence of the algorithm.

Multi-objective Bayesian optimization is also related to several other variants of Bayesian optimization studied in the literature. For example, there has been work on contextual (Krause and Ong, 2011), multi-task (Swersky et al., 2013) and collaborative (Bardenet et al., 2013) Bayesian optimization. These approaches can be viewed as carrying out Bayesian optimization for multiple related objectives, exploiting dependencies between them in order to share statistical strength. While also considering multiple objectives, the goal is to find separate optimizers for each of them, rather than identifying a Pareto frontier. Lastly Multi-objective Bayesian optimization is also related to Bayesian optimization under unknown constraints (Gelbart et al., 2014), since one way to obtain Pareto-optimal solutions is to optimize one objective under constraints on the others. The approach of (Gelbart et al., 2014) however only aims to identify a (single) optimal feasible solution as opposed to an entire Pareto frontier.

6.6 Multi-Objective Global Optimization and Evolutionary Algorithms

Global optimization algorithms have been paired with evolutionary algorithms to find the solution that maximizes an acquisition function, for example expected improvement. The best amongst these appears to be ParEGO (Knowles, 2006), which also uses GP models of the objective functions. On every iteration, the problem is scalarized using a different weight vector, and the solution that maximizes the expected improvement, based on the current single-objective function, is chosen for evaluation. An evolutionary algorithm performs the search using the GP models to asses the expected improvement of each solution. A similar approach, presented by Zhang et al. (2010), generates several populations per iteration to take advantage of a parallel implementation. As a result, several points are sampled on every iteration. We will compare our approach against ParEGO, as it also generates serialized function evaluations; this method might be preferred when the objective functions are expensive to evaluate.

7. Experiments

In this section we evaluate $\boldsymbol{\epsilon}$-PAL on three real world data sets obtained from different applications in computer science and engineering. We assess the prediction error versus the number of evaluations required to obtain it, for different settings of the accuracy parameter $\boldsymbol{\epsilon}$. Further, we compare with PAL (Zuluaga et al. (2013)) and ParEGO (Knowles, 2006), a state-of-the-art multi-objective optimization method based on evolutionary algorithms, using an implementation provided by the authors and adapted to run with our data sets. ParEGO also uses GP modeling to aid convergence.

Before presenting the results we introduce the data sets, and explain the experimental setup.

7.1 Data Sets

The first data set, called SNW, is taken from Zuluaga et al. (2012b). The design space consists of 206 different hardware implementations of a sorting network for 256 inputs. Each design is characterized by $d = 3$ parameters. The objectives are area and throughput when synthesized for a field-programmable gate array (FPGA) platform. This synthesis is very costly and can take up to many hours for large designs.

The second data set, called NoC, is taken from Almer et al. (2011). The design space consists of 259 different implementations of a tree-based network-on-chip, targeting application-specific circuits (ASICs) and multi-processor system-on-chip designs. Each design is defined by $d = 4$ parameters. The objectives are energy and runtime for the synthesized designs run on the Coremark benchmark workload. Again, the evaluation of each design is very costly.

The third data set, called SW-LLVM, is taken from Siegmund et al. (2012). The design space consists of 1023 different compiler settings for the LLVM compiler framework. Each setting is specified by $d = 11$ binary flags. The objectives are performance and memory footprint for a given suite of software programs when compiled with these settings. Note that the Pareto-optimal set consists of two points only.

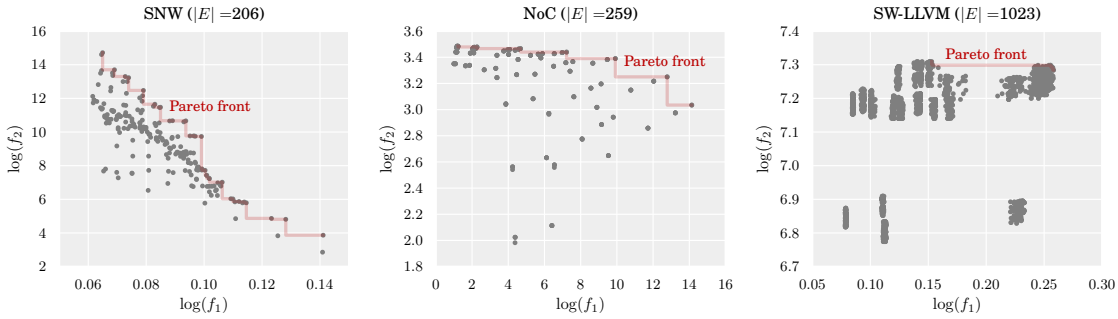

Figure 4: Objective space of the input sets use in our experiments.

Data set	d	m	$\lvert E\rvert$
SNW	3	2	206
NoC	4	2	259
SW-LLVM	11	2	1023

Table 1: Data sets used in our experiments.

The main characteristics of the data sets are summarized in Table 1; note that in all cases $m = 2$. To obtain the ground truth, we completely evaluated all data sets to determine P in each case. This was very costly, most notably, it took 20 days for NoC alone. The evaluations are plotted in log scale in Fig. 4; the Pareto fronts are emphasized. All the data is normalized such that all objectives are to be maximized.

7.2 Experimental Setup

Our implementation of $\boldsymbol{\epsilon}$-PAL uses the Gaussian Process Regression and Classification Toolbox for Matlab (Rasmussen and Nickisch, 2013). In our experiments we used the squared exponential covariance function with automatic relevance determination. The standard deviation of the noise ν is fixed to 0.1. We use

$$\beta_t^{1/2} = \frac{1}{3}\sqrt{2\log(m|E|\pi^2 t^2/(6\delta))},$$

which is the value suggested in Zuluaga et al. (2013) scaled by 1/3. This factor is empirical and used because the theoretical analysis is too conservative. In Srinivas et al. (2010) 1/5 is used.

All of the experiments were repeated 200 times and the median of the outcomes is shown in the plots. Additionally, several values of $\boldsymbol{\epsilon} = (\epsilon_i)_{1\leq i\leq 2}$ were evaluated, where ϵ_i is proportional to the range $r_i = \max_{\boldsymbol{x}\in E}\{f_i(\boldsymbol{x})\} - \min_{\boldsymbol{x}\in E}\{f_i(\boldsymbol{x})\}$. We start with $\epsilon_i = 0.01\times r_i$ and increase it up to $\epsilon_i = 0.3\times r_i$. When we say $\epsilon = 1\%$, we mean that ϵ_i is 1% of the range r_i.

Prior to running $\boldsymbol{\epsilon}$-PAL, a subset of the design space was evaluated in order to initialize the training set and optimize the kernel parameters used by the model. For SNW and NoC,

15 points were chosen uniformly at random. For LLVM-SW, as the dimensionality of the design space is much larger, 30 points were taken.

All of the experiments were run on a machine equipped with two 6-Core Intel Xeon X5680 (3.06GHz) and 144GB (1333MHz) of memory.

7.3 Prediction Error

After running $\boldsymbol{\epsilon}$-PAL with any of the data sets, a prediction $\hat{P} \subset E$ of the true Pareto set is returned. Some of the points in $\hat{P}$ are then already evaluated, i.e., $\boldsymbol{f}$ is already known; for the others the exact value of $\boldsymbol{f}$ was not necessary to generate the prediction. For such points, the objective functions are evaluated after running $\boldsymbol{\epsilon}$-PAL to determine their values and thus all of $\boldsymbol{f}(\hat{P})$. We compare the prediction $\hat{P}$ with the true Pareto set P, where $\boldsymbol{f}(P)$ is obtained through exhaustive evaluation. The comparison uses a percentage prediction error $e(P, \hat{P})$ based on the concept of $\boldsymbol{\epsilon}$-dominance. It is the average maximum distance found between a point in $\boldsymbol{f}(P)$ and the closest point in $\boldsymbol{f}(\hat{P})$.

When considering two objective functions, i.e., $m = 2$, $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$. In this case, the prediction error is calculated as

$$e(P, \hat{P}) = \text{avg}_{x \in P} \text{min}_{x' \in \hat{P}} \text{max}_{1 \leq i \leq 2} \frac{(f_i(\boldsymbol{x}) - f_i(\boldsymbol{x}')) \cdot 100}{r_i}.$$

The distance is expressed as a percentage of the range r_i, in order to compare the results between the two dimensions and amongst the different data sets.

7.4 Quality of Pareto Front Prediction

The first row of Fig. 5 shows the $\boldsymbol{\epsilon}$-accurate Pareto front $Z(\hat{P})$ obtained with $\boldsymbol{\epsilon}$-PAL for three different configurations of $\boldsymbol{\epsilon}$, using the first data set. The red shaded line is the true Pareto front found by exhaustive evaluation. The gray points are the ones that have been either sampled during the execution of $\boldsymbol{\epsilon}$-PAL or selected to be part of the $\boldsymbol{\epsilon}$-Pareto front. We can see that, as intended, with larger values of $\boldsymbol{\epsilon}$, a less accurate Pareto front is obtained, but also fewer points are evaluated. The rest of the points are discarded by $\boldsymbol{\epsilon}$-PAL during its execution.

The second row of Fig. 5 shows equivalent plots obtained with PAL. PAL attempts to classify points as Pareto optimal or not Pareto-optimal until all points have been classified; it uses $\boldsymbol{\epsilon}$ to ease this classification. Therefore, as $\boldsymbol{\epsilon}$ increases, more points are selected as Pareto optimal. The gray points in the plots correspond to solutions that are either evaluated during the execution of PAL or classified as Pareto optimal. It is clear that PAL does not generate an $\boldsymbol{\epsilon}$-accurate Pareto set, but simply uses a parameter $\boldsymbol{\epsilon}$ to make the algorithm terminate at different stages. The prediction can be as precise as the models for f_i generated with the samples gathered by the termination of the algorithm. For small values of $\boldsymbol{\epsilon}$, the resulting Pareto front is close to the true Pareto front because more samples have been taken and a more accurate classification can be made. On the other hand, for large values of $\boldsymbol{\epsilon}$, this happens because most points are classified as Pareto optimal, even though they are far from the true Pareto front. In all cases, the returned Pareto front attempts to be as close as possible to the true Pareto front.

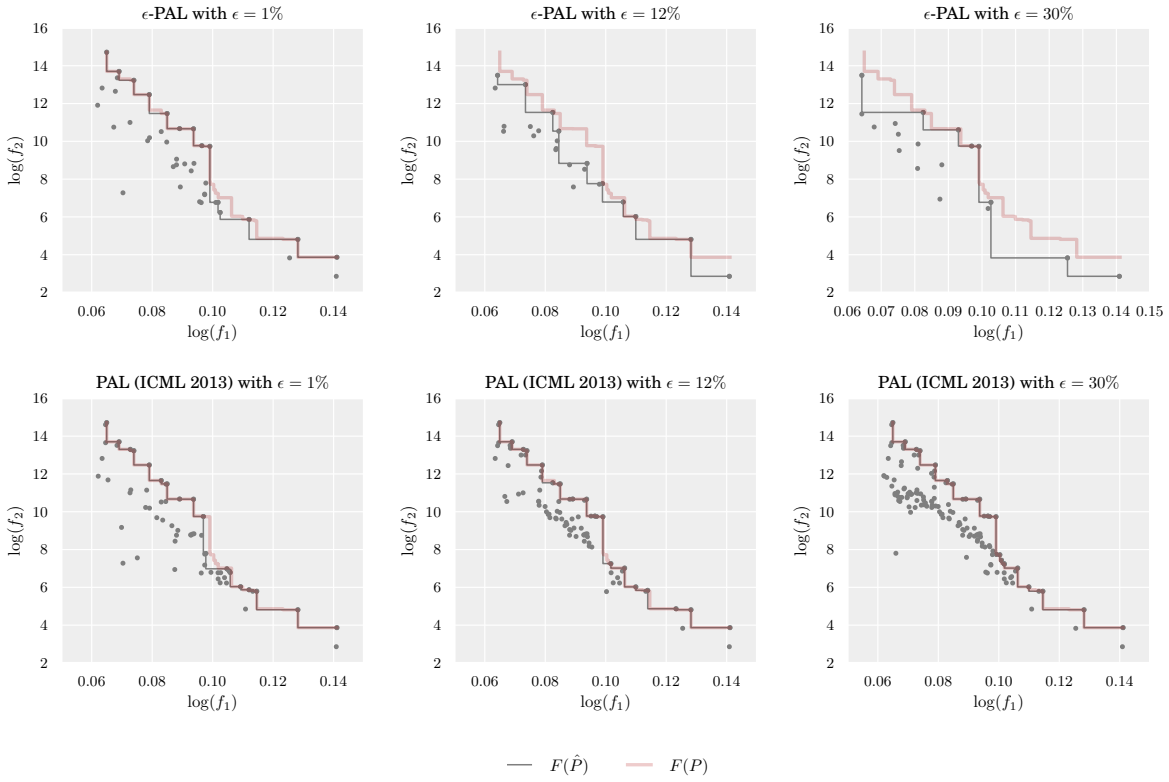

Figure 5: The first row shows the $\boldsymbol{\epsilon}$-accurate Pareto fronts (black line) of the SNW design space obtained with $\boldsymbol{\epsilon}$-PAL for $\epsilon \in \{1\%, 12\%, 30\%\}$. The second row shows equivalent plots for PAL in which the prediction $\hat{P}$ is composed of the sampled points and the points classified as Pareto optimal at the termination of the algorithm. The red shaded line in each plots is the true Pareto front of the design space.

Fig. 6 shows a set of experiments in which we do both, exploring the effect of choosing $\boldsymbol{\epsilon}$ and comparing against PAL and ParEGO (Knowles, 2006). Every plot in Fig. 6 corresponds to one data set in Table 1. In each case, the x-axis shows the number of evaluations (sampling cost) of $\boldsymbol{f}$. For $\boldsymbol{\epsilon}$-PAL and PAL this is T (the total number of iterations) plus the evaluations of designs in $\hat{P}$ that have not been evaluated yet while running the algorithm. On the y-axis, we show the average percentage error $e(P, \hat{P})$ as defined above. We measure the error at each iteration, and plot $(t, e(P, \hat{P}))$. As expected, the error in all cases decreases with increasing numbers of evaluations.

$\boldsymbol{\epsilon}$-PAL provides a wide range of accuracy-cost trade-offs as an effect of choosing different $\boldsymbol{\epsilon}$ configurations. When $\epsilon = 30\%$, only a few iterations are made by the algorithm, returning an error of less than 7% with less than 30 function evaluations in all cases. On the other hand, for $\epsilon = 1\%$, $\boldsymbol{\epsilon}$-PAL yields an error of less than 0.7% with less than 50 function evaluations in all cases.

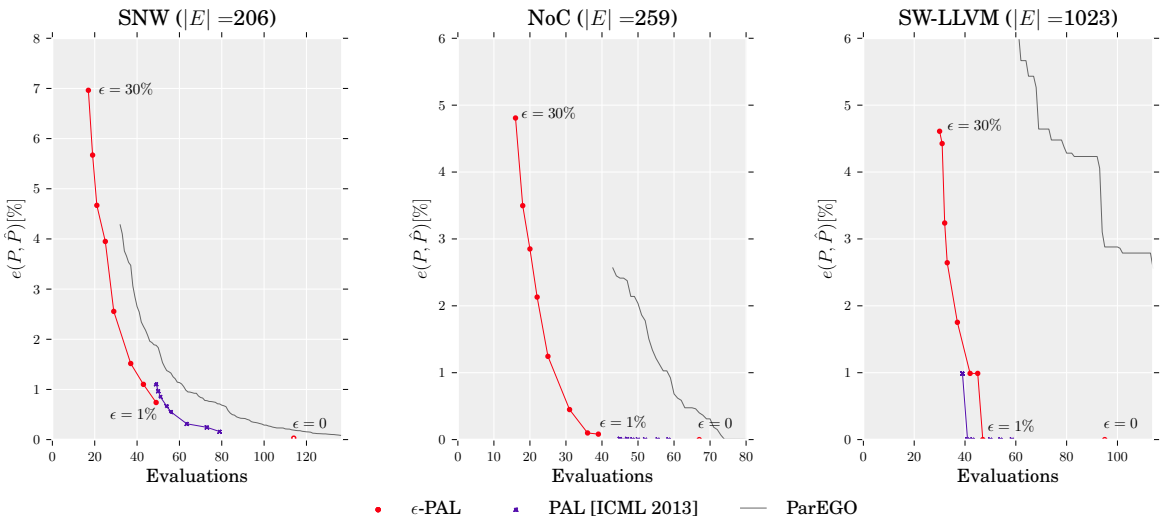

Figure 6: Prediction error $e(P, \hat{P})$ vs. number of evaluations. For $\boldsymbol{\epsilon}$-PAL and PAL the values $\epsilon \in \{30\%, 20\%, 16\%, 12\%, 8\%, 4\%, 2\%, 1\%\}$ are used. The corresponding points are consecutive in the lines for $\boldsymbol{\epsilon}$-PAL and PAL.

The measured error $e(P, \hat{P})$ is comparable with ϵ as they both measure how far the predicted Pareto front is from the true Pareto front. ϵ measures the desired accuracy, and $e(P, \hat{P})$ measures what is obtained. We can then say that $\boldsymbol{\epsilon}$-PAL meets the expectations on the prediction for all $\boldsymbol{\epsilon}$ configurations and in all cases.

The plots in Fig. 6 also show a point in which $\boldsymbol{\epsilon}$-PAL achieves an error of 0. This is obtained with $\epsilon = 0$, $\delta = 0.05$ and β_t having the exact value suggested by Theorem 1. Under these configurations, $\boldsymbol{\epsilon}$-PAL yields a perfect prediction with less than 115 evaluations in all cases.

PAL (purple line) generates a more constrained set of trade-offs using the same $\boldsymbol{\epsilon}$ configurations that were used for $\boldsymbol{\epsilon}$-PAL. The actual $\hat{P}$ reported in (Zuluaga et al., 2013) corresponds to the Pareto-optimal points amongst all pairs $(\mu_1(\boldsymbol{x}), \mu_2(\boldsymbol{x}))$ for $\boldsymbol{x} \in E$. On the other hand, ParEGO (gray line) does not provide a methodology to stop the algorithm in different stages of accuracy and cost. We then plot a continuous line that shows the error obtained by the set S_t of sampled points on every iteration of the algorithm t, i.e., we plot points $(|S_t|, e(P, S_t))$. ParEGO uses a heuristic to find the number of samples of the starting population depending on the characteristics of the design space. Hence the line always starts with a certain minimum number of evaluations.

We observe that $\boldsymbol{\epsilon}$-PAL and PAL in all cases significantly improve over ParEGO. In particular, the gains on SW-LLVM were considerable. Overall, the number of evaluations was reduced by 1/3 to 2/3 in all cases when comparing against ParEGO. $\boldsymbol{\epsilon}$-PAL overall compares favorably to PAL except for SW-LLVM, where PAL wins for a small range in the error/performance tradeoff space.

As explained above, ϵ-PAL returns an ϵ-accurate Pareto set and PAL returns a more dense Pareto set, therefore, the prediction error in PAL is always fairly small and in most of the cases it requires more function evaluations. In the case of SW-LLVM, there are some trade-offs obtained with ϵ-PAL that yield slightly higher errors than those obtained with PAL. This is because ϵ-PAL eliminates solutions along its execution and as the corresponding design space has only one Pareto point, when this is removed, the error obtained increases considerably. However the error is within the expected range, i.e., below that indicated by ϵ. Additionally, if the design space is not too large, after the termination of the algorithm, all points that were discarded can be again predicted by the model to find a more accurate Pareto set similar to what PAL returns.

In conclusion, ϵ-PAL in most cases produces better trade-offs than PAL and ParEGO. In comparison to PAL, it uses more effectively the fact that the users, by setting ϵ, are expressing their desire to generate only ϵ-accurate Pareto fronts in exchange for a smaller number of function evaluations. This means that it can offer a wider range of trade-offs between accuracy and number of evaluations. Moreover, the resulting Pareto fronts contain a variety of trade-offs between the underlying multi-objective design space, that are spread evenly along the range of possibilities.

7.5 Execution Time

A drawback of PAL is the high number of computations required per iteration, which constrains it to small design spaces. As explained in Section 6, ϵ-PAL reduces the number of computations per iteration by a factor of n while achieving better results than PAL. In this section, we compare the runtime of executing PAL and ϵ-PAL with our three data sets. Another advantage of ϵ-PAL that helps improving its computational efficiency is that it uses ϵ to discard more points from the design space as its value increases. In PAL on the other hand, as ϵ increases, the number of points classified as Pareto optimal increases, and thus those points require computations on every iteration until the algorithm terminates.

The first plot in Fig. 7 shows runtime speedups, obtained with the three data sets and all the values of ϵ considered before. We measured for each value of ϵ the runtime for ϵ-PAL and PAL and divide them to obtain the speedups. Here we assume that the cost of evaluating $\boldsymbol{f}$ is 0. The plot shows that, as expected, the speedups improve as n increases. Therefore, LLVM-SW, which has the largest design space of all data sets, obtains the largest speedups, from 45× with $\epsilon = 1\%$ up to 420× with $\epsilon = 30\%$. Better speedups are obtained with larger values of ϵ since more points can be discarded during the first iterations.

The second plot in Fig. 7 shows equivalent results but this time assuming that evaluating $\boldsymbol{f}$ takes 30 minutes. In this case, the speedups are smaller, going up to almost 3× for $\epsilon = 30\%$. This is because the data sets are relatively small and thus the gains in runtime are offset by the high cost of evaluating $\boldsymbol{f}$. In other words, the speedups that we observed here are speedups on function evaluations and are related to the improvements that can be visualized in Fig. 6. These improvements come from the fact that ϵ-PAL requires less function evaluations than PAL to achieve an error equivalent to the requested ϵ. As a result, the denser the true Pareto front of the design space, the better the speedups obtained by using ϵ-PAL. For this reason, LLVM-SW does not generate important speedups on function evaluations, since the true Pareto front is composed of only one point. The best speedups

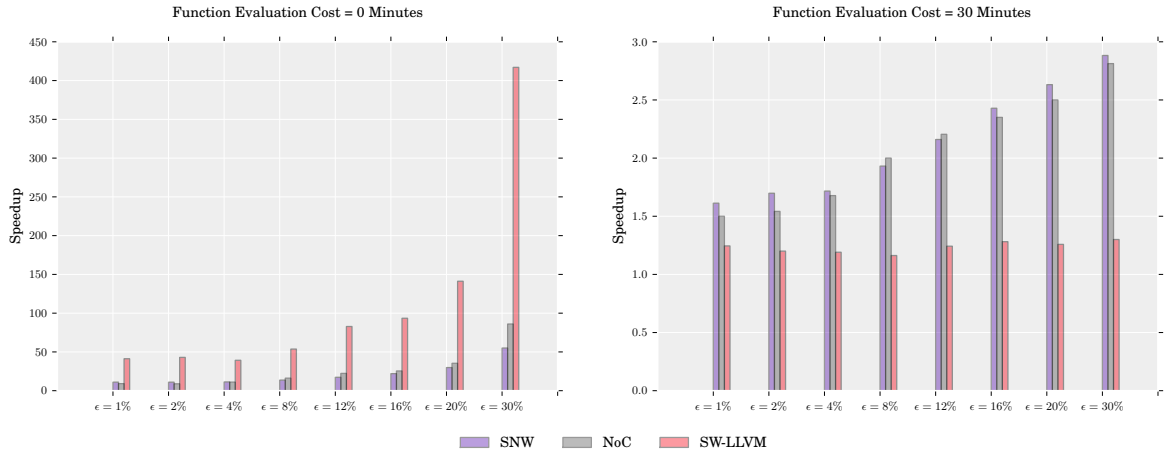

Figure 7: Speedups obtained by using ϵ-PAL against using PAL. The plot of the left shows the speedups on the runtime of ϵ-PAL, assuming that function evaluations are free. The plot on the right shows speedups assuming that one function evaluation takes 30 minutes.

are obtained with SNW, followed by NoC, tha same order as the densities of their true Pareto curves.

8. Conclusions

In this paper, we proposed a novel algorithm for efficiently localizing an ϵ-accurate Pareto-frontier in multi-objective optimization. In the spirit of Bayesian optimization, it uses Gaussian processes to exploit prior information about the objective functions' regularity to predict the objective functions, to guide the sampling process, and to make inferences about the optimality of designs without directly evaluating them. We presented an extensive theoretical analysis including bounds for the number of samples required to achieve the desired target accuracy.

The algorithm described in this paper presents major improvements from its predecessor PAL. First, it returns an ϵ-accurate Pareto front instead of a dense approximation of the true Pareto front. It is often the case that a small difference in the objective function, conveyed to the algorithm with the parameter ϵ, does not affect the quality of a design significantly. Thus, it is preferable to reduce the number of evaluations made and to obtain a set of close-to-optimal solutions well distributed in the objective space.

A second improvement over PAL is that ϵ-PAL reduces the asymptotic runtime, which facilitates the exploration of larger design spaces, in which perhaps the function evaluation is less expensive.

Finally, we demonstrated the effectiveness of our approach on three case studies obtained from real engineering applications. Our results show that we offer better cost-performance trade-offs in comparison to ParEGO and PAL. Across all data sets and almost all desired accuracies, ϵ-PAL outperforms ParEGO, requiring in most cases 30% to 70% less function

evaluations. Moreover, we showed that our parameterization strategy provides a wide range of cost-performance trade-offs. In comparison to PAL, our experiments show that ϵ-PAL reduces the amount of computations by up to 420×, and the number of samples from the design space required to meet the user's desired level of accuracy by up to 2.9×.

Acknowledgments. We would like to thank the authors of (Almer et al., 2011) and (Siegmund et al., 2012) for providing the data for our experiments. This research was supported in part by SNSF grants 200021_137971, 200021_137528, DARPA MSEE FA8650-11-1-7156 and ERC StG 307036.

Appendix A. Theoretical Analysis and Asymptotic Bounds

This section provides the proofs of Theorem 1, which follows from Lemmas 1 to 8.

Lemma 1 *Assume that the target functions f_i, $0 \leq i \leq m$, are bounded on their RKHS norm $||f_i||_k$. Given $\delta \in (0,1)$ and $\beta_t = 2||f_i||_k^2 + 300\gamma_t \log^3(m|E|t/\delta)$, the following holds with probability $\geq 1-\delta$:*

$$\begin{aligned} |f_i(\boldsymbol{x}) - \mu_{t-1,i}(x)| &\leq \beta_t^{1/2}\sigma_{t-1,i}(\boldsymbol{x}) \\ \text{for all } 1 \leq i \leq m,\ \boldsymbol{x} \in E,\ &\text{for all } t \geq 1. \end{aligned} \tag{10}$$

In other words, with probability $\geq 1-\delta$:

$$\boldsymbol{f}(\boldsymbol{x}) \in R_t(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \in E,\ \text{for all } t \geq 1$$

Proof According to Theorem 6 in (Srinivas et al., 2012), the following inequality holds:

$$Pr\left\{|f_i(\boldsymbol{x}) - \mu_{t-1,i}(\boldsymbol{x})| > \beta_t^{1/2}\sigma_{t-1,i}(\boldsymbol{x})\right\} \leq \delta',$$

with $\beta_t = 2||f_i||_k^2 + 300\gamma_t \log^3(t/\delta')$. Applying the union bound for $i, t \in \mathbb{N}$, we obtain that the following holds with probability $\geq 1 - m|E|\delta'$:

$$\begin{aligned} |f_i(\boldsymbol{x}) - \mu_{t-1,i}(\boldsymbol{x})| &\leq \beta_t^{1/2}\sigma_{t-1,i}(\boldsymbol{x}) \\ \text{for all } 1 \leq i \leq m,\ &\text{for all } \boldsymbol{x} \in E, \end{aligned} \tag{11}$$

with $\beta_t = 2||f_i||_k^2 + 300\gamma_t \log^3(t/\delta')$. The lemma holds by choosing $\delta = m|E|\delta'$. ■

Lemma 2 *If $m = 1$ and $\boldsymbol{f}_T = (\boldsymbol{f}(\boldsymbol{x}_t))_{1 \leq t \leq T}$, then*

$$I(\boldsymbol{y}_T; \boldsymbol{f}_T) = \frac{1}{2}\sum_{t=1}^{T} \log(1 + \sigma^{-2}\sigma_{t-1}^2(\boldsymbol{x}_t)).$$

This is directly taken from Lemma 5.3 in (Srinivas et al., 2010). Hereby $I(\boldsymbol{y}_T; \boldsymbol{f}_T) = H(\boldsymbol{f}_T) - H(\boldsymbol{f}_T \mid (\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_T, \boldsymbol{y}_T))$ is the mutual information between the function values $\boldsymbol{f}_T$ and the noisy observations $\boldsymbol{y}_T = \boldsymbol{f}_T + \nu_T$ where where $\nu_T \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I})$. This relation holds no matter in what sequence the samples $\boldsymbol{x}_t$ are picked by $\boldsymbol{\epsilon}$-PAL, and no matter what values $\boldsymbol{y}_t$ are being observed. Latter statement is true since for Gaussian processes the posterior variance over $\boldsymbol{f}$ only depends on where it is evaluated, i.e., $\boldsymbol{x}_T$, not on the actual values $\boldsymbol{y}_T$ observed (Srinivas et al., 2010).

Lemma 3 *Given $\delta \in (0,1)$ and $\beta_t = 2\log(m|E|\pi_t/\delta)$, the following holds:*

$$\sum_{t=1}^{T} \overline{w}_t^2 \leq \beta_T C_1 I(\boldsymbol{y}_T; \boldsymbol{f}_T) \leq C_1 \beta_T \gamma_T \text{ for all } T \geq 1,$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

Proof One of the rectangles of which $R_t(\boldsymbol{x}_t)$ is the intersection has a diagonal length of $2\beta_t^{1/2}\|\boldsymbol{\sigma}_{t-1}(\boldsymbol{x}_t)\|_2$: as a consequence,

$$\overline{w}_t^2 \leq 4\beta_t \|\boldsymbol{\sigma}_{t-1}(\boldsymbol{x}_t)\|_2^2.$$

As β_t is increasing, we have that

$$\begin{aligned}\overline{w}_t^2 &\leq 4\beta_T\sigma^2 \sum_{i=1}^{m} \sigma^{-2}\sigma_{t-1,i}^2(\boldsymbol{x}_t) \\ &\leq 4\beta_T\sigma^2 C_2 \sum_{i=1}^{m} \log(1+\sigma^{-2}\sigma_{t-1,i}^2(\boldsymbol{x}_t))\end{aligned}$$

with $C_2 = \sigma^{-2}/\log(1+\sigma^{-2}) \geq 1$, since $s^2 \leq C_2\log(1+s^2)$ for $0 \leq s \leq \sigma^{-2}$, and $\sigma^{-2}\sigma_{t-1,i}^2(\boldsymbol{x}_t) \leq \sigma^{-2}k_i(\boldsymbol{x}_t,\boldsymbol{x}_t) \leq \sigma^{-2}$.

Using $C_1 = 8\sigma^2 C_2$ and Lemma 2 we have that

$$\begin{aligned}\sum_{t=1}^{T} \overline{w}_t^2 &\leq \beta_T C_1 \sum_{i=1}^{m} I(\boldsymbol{y}_T; f_{T,i}) \\ &\leq \beta_T C_1 I(\boldsymbol{y}_T; \boldsymbol{f}_T)\end{aligned}$$

■

Lemma 4 *Given $\delta \in (0,1)$ and $\beta_t = 2\log(m|E|\pi_t/\delta)$, the following holds with probability $\geq 1-\delta$:*

$$\sum_{t=1}^{T} \overline{w}_t \leq \sqrt{C_1 T \beta_T \gamma_T} \text{ for all } T \geq 1$$

Proof This follows from Lemma 3, since $(\sum_t^T \overline{w}_t)^2 \leq T\sum_{t=1}^T \overline{w}_t^2$ by the Cauchy-Schwarz inequality. ■

Lemma 5 *Running $\boldsymbol{\epsilon}$-PAL with a monotonic classification, it holds that $\overline{w}_t$ decreases with t.*

Proof As a direct consequence of the sample selection rule, $w_{t-1}(\boldsymbol{x}_t) \leq \overline{w}_{t-1}$. On the other hand, $w_t(\boldsymbol{x}) \leq w_{t-1}(\boldsymbol{x})$ and thus, $\overline{w}_t \leq w_{t-1}(\boldsymbol{x}_t)$. The lemma follows. ■

Lemma 6 *Running $\boldsymbol{\epsilon}$-PAL with $\delta \in (0,1)$ and $\beta_t = 2\log(m|E|\pi_t/\delta)$, the following holds:*

$$Pr\left\{\overline{w}_T \leq \sqrt{\frac{C_1\beta_T\gamma_T}{T}} \text{ for all } T \geq 1\right\} \geq 1-\delta, \tag{12}$$

where $C_1 = 8/\log(1+\sigma^{-2})$ and $\pi_t = \pi^2 t^2/6$.

Proof This is derived from Lemmas 4 and 5, since $\sum_{t=1}^{T} \overline{w}_t/T \geq \overline{w}_T$.

■

Lemma 7 *If when running $\boldsymbol{\epsilon}$-PAL, $\overline{w}_t \leq ||\boldsymbol{\epsilon}||_\infty$ holds at iteration t, then the algorithm terminates without further sampling.*

Proof Since $\overline{w}_t$ is un upper bound of $||\max(R_t(\boldsymbol{x})) - \min(R_t(\boldsymbol{x}))||_2$, it is enough to show that the statement is true if $\boldsymbol{w} = \max(R_t(\boldsymbol{x}_i)) - \min(R_t(\boldsymbol{x}_i)) = (\overline{w}_t^{(1)}, \ldots, \overline{w}_t^{(m)})$ for all $\boldsymbol{x}_i \in U_t \cup P_t$. Therefore, $\overline{w}_t \leq ||\boldsymbol{\epsilon}||_\infty$ implies $\boldsymbol{w} \preceq \boldsymbol{\epsilon}$.

We show that if $\boldsymbol{w} \preceq \boldsymbol{\epsilon}$, in the same iteration of $\boldsymbol{\epsilon}$-PAL all points in U_t will be either moved to P_t or discarded.

If a point $\boldsymbol{x}$ does not belong to $Z_\epsilon(E)$, then there is a point $\boldsymbol{x}'$ such that

$$\max(R_t(\boldsymbol{x}')) \succeq \min(R_t(\boldsymbol{x})) + \boldsymbol{\epsilon}.$$

Using $\max(R_t(\boldsymbol{x}')) = \min(R_t(\boldsymbol{x}')) + \boldsymbol{w}$, we can rewrite

$$\min(R_t(\boldsymbol{x})) + \boldsymbol{\epsilon} \preceq \min(R_t(\boldsymbol{x}')) + \boldsymbol{w}.$$

Since $\boldsymbol{w} \preceq \boldsymbol{\epsilon}$, this is equivalent to

$$\min(R_t(\boldsymbol{x})) + \boldsymbol{w} \preceq \min(R_t(\boldsymbol{x}')) + \boldsymbol{\epsilon}.$$

It follows that

$$\max(R_t(\boldsymbol{x})) \preceq \min(R_t(\boldsymbol{x}')) + \boldsymbol{\epsilon},$$

which is the condition that ensures that $\boldsymbol{x} \preceq_\epsilon \boldsymbol{x}'$. This means that there is a point $\boldsymbol{x}' \in U_t \cup P_t$ that dominates $\boldsymbol{x}$, and therefore $\boldsymbol{x}$ will be discarded.

■

Lemma 8 *If $\boldsymbol{\epsilon}$-PAL is run until its termination with some $\boldsymbol{\epsilon}$ and confidence parameter δ, then with probability at least $1 - \delta$ it holds that $\Pi(\hat{P})$ is an $\boldsymbol{\epsilon}$-accurate Pareto set of E.*

Proof According to Definition 5, an $\boldsymbol{\epsilon}$-accurate Pareto set of E requires every point of E to have an $\boldsymbol{\epsilon}$-dominator in $\hat{P}$. In the discarding stage of $\boldsymbol{\epsilon}$-PAL, points in U_t are discarded if they are $\boldsymbol{\epsilon}$-dominated by a pessimistic Pareto point. If a point is $\boldsymbol{\epsilon}$-dominated in E, it is $\boldsymbol{\epsilon}$-dominated by a point in the pessimistic Pareto set of E. Therefore, the discarding stage of $\boldsymbol{\epsilon}$-PAL ensures that there will be always be a point in the remaining sets that $\boldsymbol{\epsilon}$-dominates the points that are discarded. Pessimistic Pareto points are only discarded if they are $\boldsymbol{\epsilon}$-dominated by any point in P_t. Hence, all discarding decisions are "safe", in the sense that for every discarded point, there is always at least one $\boldsymbol{\epsilon}$-dominating point remaining. They are also "complete", i.e., only points that belong to $Z_\epsilon(E)$ are moved to P_t. Hence, at termination, since U_t is empty, all remaining points P_t form an $\boldsymbol{\epsilon}$-accurate Pareto set of E.

■

References

O. Almer, N. Topham, and B. Franke. A learning-based approach to the automated design of MPSoC networks. *Architecture of Computing Systems (ARCS)*, pages 243–258, 2011.

R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. Collaborative hyperparameter tuning. In *Intl. Conference on Machine Learning (ICML)*, pages 199–207, 2013.

E. Bonilla, K. M. A. Chai, and C. K. I. Williams. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 153–160, 2008.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Arxiv preprint arXiv:1012.2599*, 2010.

D. Buche, N. N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with Gaussian process fitness function models. *IEEE Trans. on Systems, Man, and Cybernetics*, 35:183–194, 2005.

A. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.

P. Campigotto, A. Passerini, and R Battiti. Active learning of Pareto fronts. *IEEE Trans. on Neural Networks and Learning Systems*, 25:506–519, 2014.

C. Coello, G. B. Lamont, and D. Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2006.

D. D. Cox and S. John. SDO: A statistical method for global optimization. *Multidisciplinary Design Optimization: State of the Art*, pages 315–329, 1997.

N. de Freitas, A. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *Intl. Conference on Machine Learning (ICML)*, pages 1743–1750, 2012.

L. Deng, K. Sobti, and C. Chakrabarti. Accurate models for estimating area and power of FPGA implementations. In *Intl. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1417–1420, 2008.

M. T. M. Emmerich, K. C. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Trans. on Evolutionary Computation*, 10(4):421–439, 2006.

M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian optimization with unknown constraints. In *Uncertainty In Artificial Intelligence (UAI)*, 2014.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Trans. on Evolutionary Computation*, 10(1):50 – 66, 2006.

A. Krause and C. S. Ong. Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2447–2455, 2011.

H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of the ACM*, 22:469–476, 1975.

S. Künzli, L. Thiele, and E. Zitzler. Modular design space exploration framework for embedded systems. *Computers & Digital Techniques*, 152(2):183–192, 2005.

M. Laumanns and J. Ocenasek. Bayesian optimization algorithms for multi-objective optimization. In *Conference on Parallel Problem Solving from Nature*, pages 298–307. Springer, 2002.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

G. Palermo, C. Silvano, and V. Zaccaria. ReSPIR: A response surface-based Pareto iterative refinement for application-specific design space exploration. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 28:1816 –1829, 2009.

C. E. Rasmussen and H. Nickisch. Gaussian process regression and classification toolbox Version 3.2 for Matlab 7.x, 2013.

C. E Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

Burr Settles. Active Learning Literature Survey. Technical Report 1648, University of Wisconsin-Madison, 2010.

N. Siegmund, S.S. Kolesnikov, C. Kastner, S. Apel, D. Batory, M. Rosenmuller, and G. Saake. Predicting performance via automated feature-interaction detection. In *Intl. Conference on Software Engineering (ICSE)*, pages 167–177, 2012.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Intl. Conference on Machine Learning (ICML)*, pages 1015–1022, 2010.

N. Srinivas, A. Krause, S.M. Kakade, and M. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Trans. on Information Theory*, 58(5):3250–3265, 2012.

I. Steponavice, R. J. Hyndman, Smith-Miles K., and L. Villanova. Efficient identification of the Pareto optimal set. In *Learning and Intelligent Optimization Conference (LION)*, volume 8426 of *Lecture Note of Computer Science*, pages 341–352, 2014.

K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 2004–2012, 2013.

E. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and inference*, 140(11):3088–3095, 2010.

Q. Zhang and H. Li. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. on Evolutionary Computations*, 11:712–731, 2007.

Q. Zhang, L. Wudong, E. Tsang, and B. Virginas. Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Trans. on Evolutionary Computation*, 14(3):456 –474, 2010.

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*, pages 95–100, 2002.

M. Zuluaga, A. Krause, Milder P. A., and M. Püschel. "Smart" design space sampling to predict Pareto-optimal solutions. In *Languages, Compilers, Tools and Theory for Embedded Systems (LCTES)*, pages 119–128, 2012a.

M. Zuluaga, P. Milder, and M. Püschel. Computer generation of streaming sorting networks. In *Design Automation Conference (DAC)*, pages 1245–1253, 2012b.

M. Zuluaga, A. Krause, G. Sergent, and M. Püschel. Active learning for multi-objective optimization. In *Intl. Conference on Machine Learning (ICML)*, pages 462–470, 2013.

M. Zuluaga, A. Krause, and M. Püschel. e-PAL source code, 2015. URL `http://www.spiral.net/software/pal.html`.