Active Learning for Multi-Objective Optimization

Marcela Zuluaga ZULUAGA@INF.ETHZ.CH
ETH Zurich, Clausiusstrasse 59, 8092 Zurich, Switzerland

Andreas Krause KRAUSEA@ETHZ.CH
ETH Zurich, Universitätstrasse 6, 8092 Zurich, Switzerland

Guillaume Sergent GUILLAUME.SERGENT@ENS-LYON.FR
ENS de Lyon, 46 Allée d'Italie, 69007 Lyon, France

Markus Püschel PUESCHEL@INF.ETHZ.CH
ETH Zurich, Clausiusstrasse 59, 8092 Zurich, Switzerland

Abstract

In many fields one encounters the challenge of identifying, out of a pool of possible designs, those that simultaneously optimize multiple objectives. This means that usually there is not one optimal design but an entire set of Pareto-optimal ones with optimal trade-offs in the objectives. In many applications, evaluating one design is expensive; thus, an exhaustive search for the Pareto-optimal set is unfeasible. To address this challenge, we propose the *Pareto Active Learning* (PAL) algorithm which intelligently samples the design space to predict the Pareto-optimal set. Key features of PAL include (1) modeling the objectives as samples from a Gaussian process distribution to capture structure and accomodate 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 PAL's sampling cost required to achieve a desired accuracy. Further, we show an experimental evaluation on three real-world data sets. The results show PAL's effectiveness; in particular it improves significantly over a state-of-the-art multi-objective optimization method, saving in many cases about 33% evaluations to achieve the same accuracy.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

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 such as energy consumption, throughput, or chip area. Usually there is not a single design that excels in all objectives, and therefore one is 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 the Pareto-optimal set at low cost, i.e., by evaluating as few designs as possible.

In this paper we propose a solution that we call the *Pareto Active Learning* (PAL) algorithm. PAL has several key features. It captures domain knowledge about the regularity in 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, PAL's sampling strategy aims to maximize progress on designs that are likely to be Pareto-optimal. A set of classification rules identifies designs that are Pareto-optimal and not Pareto-optimal with high probabilty; PAL terminates when all designs are classified. Finally, PAL is parameterized to enable an intuitive, user-controlled tradeoff between sampling cost and prediction accuracy.

A main contribution of this paper is the theoretical performance analysis of PAL that provides bounds on the sampling cost required to achieve a desired accu-

racy. These bounds involve the use and quantification of the so-called hypervolume error, a metric that is commonly used in multiobjective optimization.

Finally, we carry out an extensive empirical evaluation, where we demonstrate PAL's effectiveness on several real-world multiobjective optimization problems. Two of these problems (Zuluaga et al., 2012b; Almer et al., 2011) 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. The third problem is from software optimization (Siegmund et al., 2012) where different compilation settings are evaluated for performance and memory footprint size. We compare the performance of PAL against a state-of-the-art multi-objective optimization method called ParEGO (Knowles, 2006). Across all data sets and almost all desired accuracies PAL outperforms ParEGO, requiring usually about 33% less evaluations.

1.1. Related Work

We now discuss different lines of related work.

Evolutionary algorithms. One class of approaches uses evolutionary algorithms to approximate the Pareto frontier using 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 is 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. The best among these appears to be ParEGO (Knowles, 2006), which also uses GP models of the objective functions. We will compare against this approach.

Scalarization to the single-obective 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 & Vandenberghe, 2004). As a concrete example, Zhang et al. (2010) proposes a multi-objective evolutionary algorithm framework that decomposes the optimization problem into several single-objective subproblems. A predictive model based on Gaussian processes 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 & Vandenberghe, 2004). Therefore, we avoid scalarization in our approach.

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 (D. 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.

Single-objective active learning and Bayesian optimization. In the single-objective setting, there has been much work on active learning, in particular 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 Brochu et al. (2010)), the objective is modeled as a sample from a stochastic process (often a Gaussian process). 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. Recently, Srinivas et al. (2010) analyzed the GP-UCB criterion, and proved global convergence guarantees and rates for Bayesian optimization. We build on their results to establish guarantees about our PAL algorithm in the multi-objective setting.

1.2. Main Contributions

In summary, our main contributions include:

- the PAL algorithm, which efficiently (i.e., with few evaluations) identifies the set of Pareto-optimal designs in a multi-objective scenario with expensive evaluations, and which allows user control of accuracy and sampling cost;
- the analysis of PAL that provides theoretical bounds on the algorithm's sampling cost to achieve a desired target accuracy;
- an experimental evaluation to demonstrate PAL's effectiveness on three real-world multi-objective optimization problems.

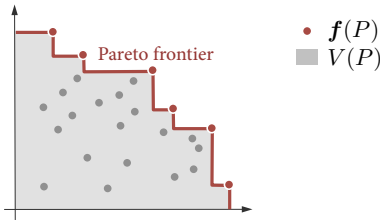

Figure 1. Example of a Pareto frontier in the objective space for $n = 2$ objectives.

2. Background and Problem Statement

We consider a multi-objective optimization problem over a finite[1] subset E (called the *design space*) of $\mathbb{R}^d$ for some $d \in \mathbb{N}$. This means we wish to simultaneously optimize n objective functions $f_1, \ldots, f_n : E \mapsto \mathbb{R}$. We use the notation $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ to refer to the vector of all objectives evaluated on the input $\boldsymbol{x}$. The *objective space* is the image $\boldsymbol{f}(E) \subset \mathbb{R}^n$.[2]

Pareto-optimality. The goal in multi-objective optimization is to identify the *Pareto frontier* of $\boldsymbol{f}$. Formally, we consider the canonical partial order in $\mathbb{R}^n$: $\boldsymbol{y} \preceq \boldsymbol{y}'$ iff $y_i \leq y'_i$, $1 \leq i \leq n$, and define the induced order on E: $\boldsymbol{x} \preceq \boldsymbol{x}'$ iff $\boldsymbol{f}(\boldsymbol{x}) \preceq \boldsymbol{f}(\boldsymbol{x}')$. The set $P \subseteq E$ of maximal (or Pareto-optimal) points in this order determines the Pareto frontier $\boldsymbol{f}(P)$. Figure 1 visualizes these concepts for $n = 2$ objectives.

The interest of finding Pareto-optimal points in the design space is clear: they represent the best compromises amongst the chosen objectives and are the only designs that need to be considered in an application.

Problem statement. In many applications, evaluating the objectives $f_1, \ldots, f_n$ is expensive. Therefore, we wish to identify the Pareto-optimal set $P \subset E$ without evaluating all inputs $\boldsymbol{x} \in E$. Our goal is to 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 this evaluation to classify all designs as Pareto optimal or not. The algorithm terminates when all designs are classified and returns a prediction $\hat{P}$ of the Pareto-optimal set P.

Prediction quality. A fundamental difference between single- and multi-objective optimization is that for the latter it is not obvious which metric to use to evaluate the prediction quality. A very natural proposed solution uses the so-called *hypervolume* (Zitzler et al., 2007), to compute the volumes enclosed by the actual Pareto frontier $\boldsymbol{f}(P)$ and its prediction $\boldsymbol{f}(\hat{P})$. Formally, we first assume that all the f_i are non-negative (a property that can be established by suitable shifting if the minimum is known). The hypervolume $V(P)$ of a Pareto-optimal set $P \subset E$ is the volume of the area $\bigcup_{\boldsymbol{p} \in P} \{\boldsymbol{y} \in \mathbb{R}^n : 0 \preceq \boldsymbol{y} \preceq \boldsymbol{f}(\boldsymbol{p})\}$ enclosed between the origin and $\boldsymbol{f}(P)$. In Fig. 1 this area is shaded gray. The hypervolume of an arbitrary set $S \subset E$ is defined similarly after all dominated points in S have been removed. The quality of a prediction $\hat{P}$ is then given by the hypervolume error

$$\eta = V(P) - V(\hat{P}), \tag{1}$$

which is always positive. The trivial prediction $\hat{P} = E$ has error 0; thus, an important feature of a reasonable algorithm is that $\hat{P}$ contains few dominated points.

It is clear that prediction is only possible under certain assumptions about $\boldsymbol{f}$, which are introduced next.

Gaussian processes. We model $\boldsymbol{f}$ as a sample from an n-variate Gaussian process (GP) distribution. A GP distribution over a univariate 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 & 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 sample 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 $y_{t,i} = f_i(\boldsymbol{x}_t) + \nu_{t,i}$; 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 points $\boldsymbol{x} \in E$ that have not been evaluated yet. We now design an active learning algorithm informed by this uncertainty.

3. PAL Algorithm

In this section we describe our algorithm: *Pareto Active Learning* (PAL). Our approach to predicting Pareto-optimal points in E is to train GP models (see

[1] While our results can be generalized, we focus on the finite setting to simplify exposition.

[2] Scalars and functions that evaluate on scalars are written unbolded; vectors and tuples of functions are boldfaced.

[3] Note that dependence among the outputs could be captured as well; e.g., (Bonilla et al., 2008).

above) with a small subset of E. The models predict the objective functions f_i, $1 \leq i \leq n$, allowing us to identify points in E that are Pareto-optimal with high probability. A point $\boldsymbol{x}$, that has not been sampled, is predicted as $\hat{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{\mu}(\boldsymbol{x}) = (\mu_i(\boldsymbol{x}))_{1 \leqslant i \leqslant n}$. Additionally $\boldsymbol{\sigma}(\boldsymbol{x}) = (\sigma_i(\boldsymbol{x}))_{1 \leqslant i \leqslant n}$ is interpreted as the uncertainty of this prediction. We capture this uncertainty through the hyperrectangle[4]

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

where β is a scaling parameter to be chosen later.

We use this uncertainty information to guide our sampling and to make a probabilistic assumption on the optimality of every point $\boldsymbol{x}$. Since we are only interested in Pareto-optimal points, our algorithm aims at sampling the design space E such that the predictions generated for $\boldsymbol{f}(\boldsymbol{x})$ are more accurate for points $\boldsymbol{x}$ that are likely to be Pareto-optimal.

At every iteration t, the algorithm uses the predictions $\boldsymbol{\mu}_t(\boldsymbol{x})$ and the uncertainties $\boldsymbol{\sigma}_t(\boldsymbol{x})$ to classify a point $\boldsymbol{x}$ as Pareto-optimal, or not Pareto-optimal. However, some points may remain unclassified. Then, the next sample to evaluate in the design space is chosen to further reduce the number of unclassified points. The training process is terminated when all points are classified; the points classified as Pareto-optimal are then returned as the prediction $\hat{P}$ for P.

The pseudocode in Algorithm 1 outlines our approach. After intialization we iterate until a stopping criterion is met; every iteration t consists of three stages: modeling, classification, and sampling. These are discussed next including the stopping criterion.

Modeling. We use 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 $\boldsymbol{x} \in E$ considering the former evaluations. Each point $\boldsymbol{x} \in E$ 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 $\boldsymbol{\sigma}_t$. Hereby $R_{-1}(\boldsymbol{x}) = \mathbb{R}^n$. In Section 4 we will suggest a value for this parameter. The iterative intersection ensures that all uncertainty regions are non-increasing with t.

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.

Classification. Our algorithm maintains three sets that partitions E: in iteration t, P_t is the set of points that are predicted to be Pareto-optimal, N_t the set of points that are predicted to be not Pareto-optimal, and U_t the set of yet unclassified points. In each iteration t, each $\boldsymbol{x}$ is assigned to exactly one of the these classes. Our algorithm is monotonic in that P_t and N_t are non-decreasing sets with respect to t. In other words, as soon as a point $\boldsymbol{x}$ is classified as Pareto-optimal (or not Pareto-optimal), it remains classified as such. Further, our classification is relaxed by a parameter ϵ, which informally means inequalities are considered "up to a small error $\boldsymbol{\epsilon}$", where $\boldsymbol{\epsilon} = (\epsilon, \ldots, \epsilon)$. ϵ is an input to the algorithm.

At iteration t, the points in P_{t-1} and N_{t-1} keep their classification. The only points $\boldsymbol{x}$ to be reclassified are those in U_{t-1}, done as follows:

- If the pessimistic outcome $\min(R_t(\boldsymbol{x}))$ of $\boldsymbol{x}$ is not dominated by the optimistic outcome $\max(R_t(\boldsymbol{x}'))$ of any other point (up to a shift of $\boldsymbol{\epsilon}$ by both),

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

  then $\boldsymbol{x}$ is classified as Pareto-optimal.
- If the optimistic outcome $\max(R_t(\boldsymbol{x}))$ of $\boldsymbol{x}$ is dominated by the pessimistic outcome $\min(R_t(\boldsymbol{x}'))$ of any $\boldsymbol{x}'$ (up to a shift of $\boldsymbol{\epsilon}$ by both),

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

  then $\boldsymbol{x}$ is classified as non-Pareto-optimal.
- All other points remain unclassified.

Algorithm 1 The PAL algorithm.

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

$P_0 = \emptyset, N_0 = \emptyset, U_0 = E$ {classification sets}
$S_0 = \emptyset$ {evaluated set}
$R_0(\boldsymbol{x}) = \mathbb{R}^n$ for all $\boldsymbol{x} \in E$
$t = 0$
repeat
  Modeling
  Obtain $\boldsymbol{\mu}_t(\boldsymbol{x})$ and $\boldsymbol{\sigma}_t(\boldsymbol{x})$ for all $\boldsymbol{x} \in E$
  $\{\boldsymbol{\mu}_t(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x})$ and $\boldsymbol{\sigma}_t(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in S_t\}$
  $R_t(\boldsymbol{x}) = R_{t-1}(\boldsymbol{x}) \cap Q_{\boldsymbol{\mu}_t,\boldsymbol{\sigma}_t,\beta_{t+1}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in E$
  Classification
  $P_t = P_{t-1},\ N_t = N_{t-1},\ U_t = U_{t-1}$
  for all $\boldsymbol{x} \in U_t$ **do**
    if there is no $\boldsymbol{x}' \neq \boldsymbol{x}$ such that $\min(R_t(\boldsymbol{x})) + \boldsymbol{\epsilon} \preceq \max(R_t(\boldsymbol{x}')) - \boldsymbol{\epsilon}$ **then**
      $P_t = P_t \cup \{\boldsymbol{x}\}, U_t = U_t \setminus \{\boldsymbol{x}\}$
    else if there exists $\boldsymbol{x}' \neq \boldsymbol{x}$ such that $\max(R_t(\boldsymbol{x})) - \boldsymbol{\epsilon} \preceq \max(R_t(\boldsymbol{x}')) + \boldsymbol{\epsilon}$ **then**
      $N_t = N_t \cup \{\boldsymbol{x}\}, U_t = U_t \setminus \{\boldsymbol{x}\}$
    end if
  end for
  Sampling
  Find $w_t(\boldsymbol{x})$ for all $\boldsymbol{x} \in (U_t \cup P_t) \setminus S_t$
  Choose $\boldsymbol{x}_{t+1} = \arg\max_{x \in (U_t \cup P_t) \setminus S_t} \{w_t(\boldsymbol{x})\}$
  $t = t + 1$
  Sample $\boldsymbol{y}_t(\boldsymbol{x}_t) = \boldsymbol{f}(\boldsymbol{x}_t) + \boldsymbol{\nu}_t$
until $U_t = \emptyset$
$\hat{P} = P_t$

[4] Pessimistically bounding the ellipsoid with radii σ_i.

Intuitively, the parameter ϵ speeds up classification at the cost of accuracy, since it makes the requirements less strict. Figure 2 shows for $n = 2$ and $\epsilon = 0$ an example of a classification at some iteration t.

Sampling. After the classification is done, a new point $\boldsymbol{x}_t$ is selected for sampling with the following selection rule. Each point $\boldsymbol{x} \in E$ 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 length of the diagonal of its uncertainty region $R_t(\boldsymbol{x})$. Among the points $\boldsymbol{x} \in P_t \cup U_t$, the one with the largest $w_t(\boldsymbol{x})$ is chosen as the next sample $\boldsymbol{x}_t$ to be evaluated. We refer to $w_t(\boldsymbol{x}_t)$ as $\overline{w}_t$.

Intuitively, this rule biases the sampling towards exploring, and thus improving the model for, the points most likely to be Pareto-optimal.

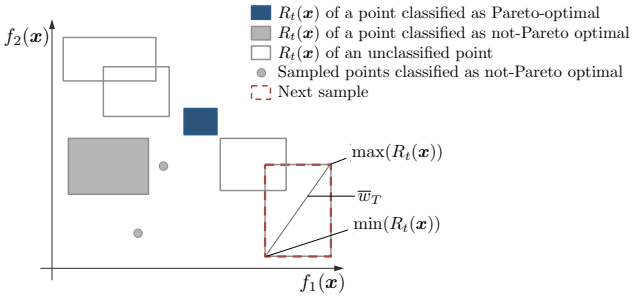

Figure 2. Classification and sampling example for $n = 2$ and $\epsilon = 0$.

Stopping criteria. The training process stops after, say, T iterations when all points in E are classified, i.e., when $U_T = \emptyset$. The prediction returned is $\hat{P} = P_T$. The selection of the parameter $\boldsymbol{\epsilon}$ used in the classification rule impacts both the accuracy and the sampling cost T of the algorithm.

4. Theoretical Analysis

In this section we analyze the sample complexity of PAL. 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 sublinearly (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.

The following theorem is our main theoretical result.

Theorem 1. *Let $\delta \in (0, 1)$. Running* PAL *with $\beta_t = 2\log(n|E|\pi^2 t^2/(6\delta))$, the following holds with probability $1 - \delta$.*

To achieve a maximum hypervolume error of η, it is sufficient to choose

$$\epsilon = \frac{\eta(n-1)!}{2na^{n-1}}, \tag{7}$$

where $a = \max_{\boldsymbol{x} \in E, 1 \le i \le n} \{\sqrt{\beta_1 k_i(\boldsymbol{x}, \boldsymbol{x})}\}$.

In this case, the algorithm terminates after at most T iterations, where T is the smallest number satisfying

$$\sqrt{\frac{T}{C_1 \beta_T \gamma_T}} \ge \frac{na^{n-1}}{\eta(n-1)!}. \tag{8}$$

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

This means that by specifying δ and a target hypervolume error η, PAL can be configured through the parameter ϵ to stop when the target error is achieved with confidence $1-\delta$. Additionally, the theorem bounds the number of iterations T required to obtain this result. Later, in Corollary 2, we will specialize the theorem to the case of a squared exponential kernel.

Our strategy for the proof consists of three parts. In Section 4.1, we first analyze how $\overline{w}_t$ decreases with t. We then relate ϵ and $\overline{w}_t$ to ensure the classification of all points and thus the termination of the algorithm. In Section 4.2, we relate $\overline{w}_t$ to the hypervolume error in the predicted Pareto frontier.

Finally, in Section 4.3 we analyze the scenario in which PAL is run with the squared exponential kernel.

4.1. Reduction in Uncertainty

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

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

which is achieved by choosing $\beta_t = 2\log(n|E|\pi^2 t^2/(6\delta))$.

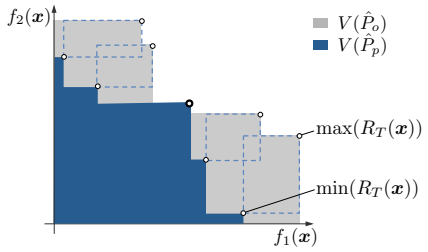

Figure 3. Example situation after termation.

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 and classification rules used by PAL guarantee that $\overline{w}_t$ decreases with t, we get the following bound for $\overline{w}_t$. With probability $\geq 1-\delta$,

$$\overline{w}_t \leq \sqrt{\frac{C_1 \beta_t \gamma_t}{t}} \text{ for all } t \geq 1, \tag{9}$$

where $C_1 = 8/\log(1-\sigma^{-2})$ and β_t is as before.
This means that, with high probability, $\overline{w}_t$ is bounded by $O^*(\sqrt{\gamma_t/t})$, where O^* is a variant of the O notation that hides log factors. The proof is supported by Lemmas 3 to 7 found in the supplementary material 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 $1-\delta$, all points will be classified at iteration T if

$$\frac{\overline{w}_T}{2} \leq \epsilon. \tag{10}$$

This is proved in Lemma 11 found in the supplementary material of this paper.

4.2. Reduction in Hypervolume Error

We derive a bound on the hypervolume error (1) that is achieved once all points have been classified after T iterations and $\hat{P}$ is returned. As example see Fig. 3, where $\hat{P}$ has five elements, one of which has been evaluated (uncertainty due to noise is not shown). With probability $1-\delta$, the points in $\boldsymbol{f}(P)$ lie inside the uncertainty regions $R_T(\boldsymbol{x})$, $\boldsymbol{x} \in \hat{P}$. Hence, in this case $\eta = V(P) - V(\hat{P}) \leq V(\hat{P}_o) - V(\hat{P}_p)$, where $\hat{P}_o$, $\hat{P}_p$ are the sets of the optimistic ($\max(R_T(\boldsymbol{x}))$ and pessimistic ($\min(R_T(\boldsymbol{x}))$ outcomes, respectively, of the points in $\hat{P}$. The difference is colored gray in Fig. 3.
Using $\overline{w}_T$, the length of the largest diagonal of any $R_T(\boldsymbol{x})$, $\boldsymbol{x} \in \hat{P}$, η is bounded by

$$\eta \leq \frac{n a^{n-1} \overline{w}_T}{(n-1)!}, \tag{11}$$

where a is defined as in Theorem 1. The proof is presented in Lemma 12, which is found in the supplementary material of this paper. Combining (11) with (10) and (9) yields the results from Theorem 1.

4.3. Explicit Bounds for the Squared Exponential Kernel

Theorem 1 holds for general GP models for $\boldsymbol{f}(\boldsymbol{x})$, with prior μ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$. Srinivas et al. (2010) derived bounds for γ_T depending on the choice of kernel for the GP. 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 $l > 0$. From (Srinivas et al., 2010), for $n = 1$, there exists a constant K such that

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

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

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

and hence the following corollary to Theorem 1.

Corollary 2. *Let k_i for all $1 \leq i \leq n$ be the squared exponential kernels used by* PAL. *When choosing $\delta \in (0,1)$, a target hypervolume error η, and an ϵ that satisfies* (7)*, the following holds with probability $1-\delta$.* PAL *terminates after at most T iterations, where T is the smallest number satisfying*

$$\frac{\sqrt{T \log(1-\sigma^{-2})}}{4K\sqrt{\log(n|E|\pi^2 T^2/(6\delta)) \log^{d+1} T}} \geq \frac{n a^{n-1}}{\eta (n-1)!}.$$

This result suggests that T increases as η decreases in the following manner: Asymptotically, for any $\rho > 0$, as well as fixed n and d, we have $T = O(\frac{1}{\eta^{2+\rho}})$.

5. Implementation Details

We now discuss some aspects that arise when implementing and using our PAL algorithm.
Parameterization. For practical usage, two parameters, namely ϵ and β_t, need to be specified. These parameters relate to the desired level of accuracy of the prediction. Although we provide theoretical bounds, these may be loose in practice, and it may be useful to choose more "aggressive" values than recommended by the theory. The choice of β_t impacts the convergence rate of the algorithm, since it 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 ϵ should pose no problem. One only may consider, in contrast to our simplifying assumption, to choose in $\boldsymbol{\epsilon}$ components of different magnitude, proportional to the range of the objectives.

Absolute versus relative values. In our exposition, we consider absolute values for $\boldsymbol{f}$ and thus $\eta, w_t, \boldsymbol{\epsilon}$ and others. In particular for the hypervolume error η this poses a problem since, for example, averaging along one objective f_i will underemphasize errors for small values of f_i. In contrast to the single-objective case, the relative average error in this case cannot be retrieved from the absolute average error. This problem can be solved by transforming all objectives with the logarithm and then running PAL in the log domain. We take this approach in our experiments later.

Kernel hyper-parameters. So far, we have assumed, the kernel function is given. Usually, its parameters need to be chosen. Therefore, prior to running PAL, it may be practical to randomly sample a small fraction of the design space, and optimizing the parameters (e.g., by maximizing the marginal likelihood). One may also consider updating the hyperparameters as new points are evaluated.

6. Experiments

We evaluate PAL on three real world data sets obtained from different applications in computer science and engineering. We assess the reduction in hypervolume error versus the number of evaluations required to obtain it for different settings of the accuracy parameter ϵ. Further, we compare with a state-of-the-art multi-objective optimization method based on evolutionary algorithms, ParEGO (Knowles, 2006), 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, discuss the use of log scale, and explain the experimental setup.

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 $d = 4$ 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 input 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 multiprocessor system-on-chip designs. Each design is defined by $d = 4$ configurations. The objectives are energy and runtime for the synthesized designs run on the Coremark benchmark workload. Again, the evaluation 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 one point only. The main characteristics of the data sets are summarized in Table 1; note that in all cases $n = 2$. 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 Fig. 4; the Pareto frontiers are emphasized. All the data is normalized so that all objectives are to be maximized.

Data Set	d	n	$\lvert E\rvert$
SNW	4	2	206
NoC	4	2	259
SW-LLVM	11	2	1023

Table 1. Data sets used in our experiments.

Use of logarithmic scale. As explained in Section 5, we applied the base-e logarithm to every objective function, thus ensuring that relative instead of absolute values are considered. This also allows us to obtain an average percentage error for the prediction $\hat{P}$ with respect to each objective, based on the hypervolume error η_T.

Formally, we define the average percentage error after termination for objective 1 as

$$\mathcal{E}_{T,1}(\hat{P}) = (1 - e^{-\frac{\eta_T}{a_2}}) \cdot 100,$$

where $a_2 = \max_{\boldsymbol{x} \in E}\{f_2(\boldsymbol{x})\}$; $\mathcal{E}_{T,2}$ analogously.

Experimental setup. Our implementation of PAL uses the Gaussian Process Regression and Classification Toolbox for Matlab (Rasmussen & Nickisch, 2010). In our experiments we used the squared exponential covariance function with automatic relevance determination. We fixed the standard deviation of the noise ν to 0.1, and scaled $\beta_t^{1/2}$ down by a factor 5 as suggested by Srinivas et al. (2010). The training set was initialized with $m = \max\{0.02|E|, 15\}$ samples chosen uniformly at random.

All of the experiments were repeated 200 times and the average outcomes are shown in the plots. Additionally, several values of ϵ were evaluated, we used $\boldsymbol{\epsilon} = (\epsilon_i)_{1 \le i \le 2}$, where ϵ_i is proportional to the range of f_i: $\max_{\boldsymbol{x} \in E}\{f_i(\boldsymbol{x})\} - \min_{\boldsymbol{x} \in E}\{f_i(\boldsymbol{x})\}$. We start with $\epsilon = 0.001\%$ of each range and increase it through doubling up to 0.512%.

Quality of Pareto frontier prediction. Fig. 5 shows a set of experiments in which we do both explore the effect of choosing ϵ and compare against the evolutionary algorithm ParEGO (Knowles, 2006). Every plot in Fig. 5 corresponds to one data set in Table 1. In each case, the x-axis shows the number t of evalua-

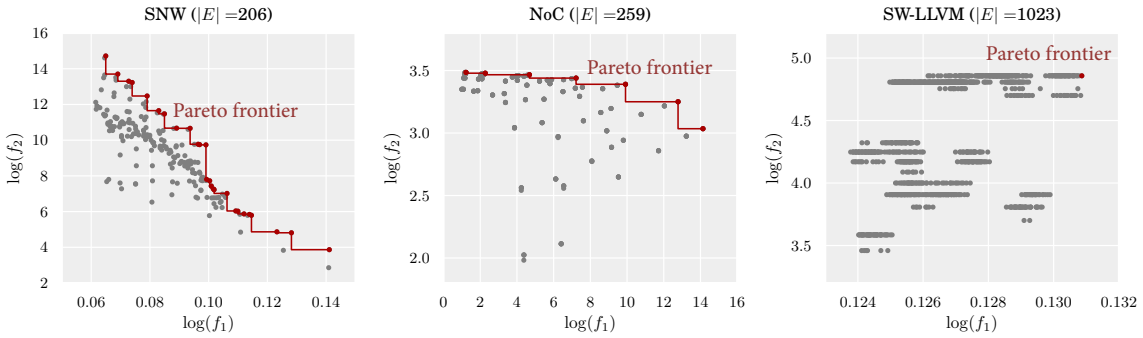

Figure 4. Objective space of the input sets use in our experiments.

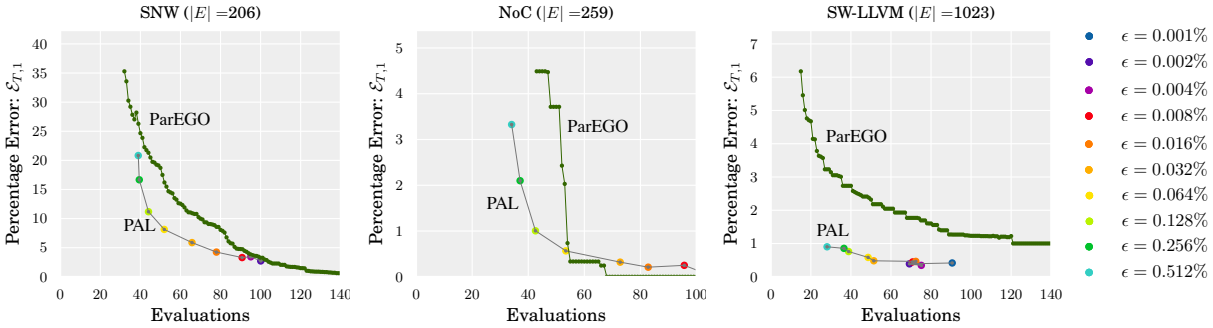

Figure 5. Avg. percentage error in f_1 vs. number of evaluations after termination; for PAL different values for ϵ are used.

tions (sampling cost) of $\boldsymbol{f}$. For 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 PAL. On the y-axis, we show the average percentage error (as defined above) for the first objective. The results corresponding to the second objective function are analogous and can be found in Fig. 7 of the supplementary material of this paper.

ParEGO (green line) uses a heuristic to find the number of samples m 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 measure the error at each iteration $m < t < 150$, and plot $(t, \mathcal{E}_{t,1})$ and $(t, \mathcal{E}_{t,2})$.

As expected, the error in all cases decreases with increasing number of evaluations. We observe that PAL in almost all cases significantly improves over ParEGO. Only for NoC and high accuracy, the peformance of ParEGO is slightly better. At the other extreme the gains on SW-LLVM are considerable. In most cases, for a fixed desired accuracy, PAL requires about 33% less evaluations than ParEGO.

The plots also show the effect of choosing ϵ on the termination of PAL. As expected a larger ϵ causes PAL to stop earlier. Further, the continuous doubling (since the data is in the log domain) of ϵ shows to offer fine grain control over termination.

7. Conclusions

In this paper we addressed the challenging problem of predicting the set of Pareto-optimal solutions in a design space from the evaluations of only a subset of the designs. We use Gaussian processes to predict the objective functions and to guide the sampling process in order to improve the prediction of the Pareto optimal set. PAL can be intuitively parameterized to achieve the desired level of accuracy at the lowest possible evaluation cost. We presented an extensive theoretical analysis including bounds for the required number of evaluations to achieve the target accuracy. 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. In most cases, for a desired accuracy, PAL requires about 33% less evaluations. Moreover, we showed that our parameterization strategy provides a wide range of cost-performance trade-offs.

Acknowledgments. We would like to thank Alkis Gkotovos for detailed comments. This research was supported in part by SNSF grant 200021 137971, DARPA MSEE FA8650-11-1-7156 and ERC StG 307036.

References

Almer, O., Topham, N., and Franke, B. A Learning-Based Approach to the Automated Design of MP-

SoC Networks. *Architecture of Computing Systems (ARCS 2011*, pp. 243–258, 2011.

Bonilla, E., Chai, K.M.A., and Williams, C.K.I. Multi-task Gaussian Process Prediction. In *Conference on Neural Information Processing Systems (NIPS)*, 2008.

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

Brochu, E., Cora, V.M., and de Freitas, N. 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.

Coello, C., Lamont, G. B., and Veldhuizen, D. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

D., Lanping, Sobti, K., and Chakrabarti, C. Accurate Models for Estimating Area and Power of FPGA Implementations. In *Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1417–1420, 2008.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient Global Optimization of Expensive Black-box Functions. *J Glob. Opti.*, 13:455–492, 1998.

Knowles, J. ParEGO: a Hybrid Algorithm with Online Landscape Approximation for Expensive Multiobjective Optimization Problems. *IEEE Trans. on Evolutionary Computation*, 10(1):50 – 66, 2006.

Künzli, S., Thiele, L., and Zitzler, E. Modular Design Space Exploration Framework for Embedded Systems. *Computers & Digital Techniques*, 152(2): 183–192, 2005.

Palermo, G., Silvano, C., and Zaccaria, V. 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.

Rasmussen, C.E. and Nickisch, H. Gaussian Process Regression and Classification Toolbox Version 3.1 for Matlab 7.x, 2010.

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

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

Siegmund, N., Kolesnikov, S.S., Kastner, C., Apel, S., Batory, D., Rosenmuller, M., and Saake, G. Predicting Performance via Automated Feature-Interaction Detection. In *Int'l Conference on Software Engineering (ICSE)*, pp. 167 –177, 2012.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Int'l Conference on Machine Learning (ICML)*, 2010.

Srinivas, N., Krause, A., Kakade, S.M., and Seeger, M. Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting. *IEEE Trans. on Information Theory*, 58(5):3250 – 3265, 2012.

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

Zitzler, E., Laumanns, M., and Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Evolutionary Methods for Design, Optimisation, and Control*, pp. 95–100, 2002.

Zitzler, E., Brockhoff, D., and Thiele, L. The Hypervolume Indicator Revisited: on the Design of Pareto-compliant Indicators via Weighted Integration. In *Int'l Conference on Evolutionary Multi-criterion Optimization (EMO)*, pp. 862–876, 2007.

Zuluaga, M., Krause, A., P.A., Milder, and Püschel, M. "Smart" Design Space Sampling to Predict Pareto-optimal Solutions. In *Languages, Compilers, Tools and Theory for Embedded Systems (LCTES)*, pp. 119–128, 2012a.

Zuluaga, M., Milder, P., and Püschel, M. Computer Generation of Streaming Sorting Networks. In *Design Automation Conference (DAC)*, 2012b.