Active Learning of Pareto Fronts

Paolo Campigotto, Andrea Passerini, and Roberto Battiti, *Fellow, IEEE*

***Abstract*— This paper introduces the active learning of Pareto fronts (ALP) algorithm, a novel approach to recover the Pareto front of a multiobjective optimization problem. ALP casts the identification of the Pareto front into a supervised machine learning task. This approach enables an analytical model of the Pareto front to be built. The computational effort in generating the supervised information is reduced by an active learning strategy. In particular, the model is learned from a set of informative training objective vectors. The training objective vectors are approximated Pareto-optimal vectors obtained by solving different scalarized problem instances. The experimental results show that ALP achieves an accurate Pareto front approximation with a lower computational effort than state-of-the-art estimation of distribution algorithms and widely known genetic techniques.**

***Index Terms*— Active learning, Gaussian process regression, multiobjective optimization, uncertainty sampling.**

I. Introduction

REAL-WORLD optimization tasks usually require the optimization of several conflicting objectives: a solution simultaneously optimizing all of them does not exist. Therefore, the solution to the multiobjective optimization problem (MOP) becomes the quantitative identification of tradeoffs between the multiple objectives. The tradeoffs between the competing objectives are captured by the Pareto-optimal solutions, for which any single objective cannot be improved without compromising at least one of the others.

The set of Pareto-optimal solutions in the decision space (Pareto set or PS) and of the corresponding objective vectors in the objective space (Pareto front or PF) typically have a large or even infinite cardinality, as it is the case for continuous problems. In these cases, the typical solution consists of an approximation of the Pareto set (Pareto front) by a finite number of representative solutions (objective vectors). The better the approximation of the Pareto front, the better is the choice of the favorite compromise between the objectives that is offered to the decision maker.

The traditional approach to obtain a finite set of Pareto-optimal solutions (and the corresponding images in the PF) involves the sequential generation and solution of scalarized instances of the MOP. Scalarization [1] consists of transforming the original MOP into a singleobjective optimization problem (SOP). The generated SOP is a parametric combination of the multiple objectives of the original MOP into a single objective. This combination may involve the generation of additional constraints not included in the original MOP. Different Pareto-optimal solutions can be obtained by appropriately varying the scalarization parameters. The generated SOP can be solved by applying common methods and widely developed theory for singleobjective optimization. However, scalarization-based methods are usually sensitive to the shape or continuity of the Pareto front [2]. For example, nonconvex parts of the Pareto front cannot be recovered by optimizing convex combinations of the objective functions.

Rather than relying on scalarization techniques, current state-of-the-art approaches for MOPs are represented by the evolutionary multiobjective optimization algorithms (EMOAs) [3], which are less susceptible to the shape of the Pareto front, handling, e.g., discontinuous or concave shapes. EMOAs are heuristic techniques generating an approximation of the whole PF in a single run and without using any derivative (i.e., gradient) information. The approximation is obtained by repeatedly improving a set of candidate solutions in the decision space (referred to as "population" within the evolutionary metaphor) until their images in the objective space have converged to the PF. However, neither the convergence to the PF nor, in case of convergence, a uniform distribution of the population over the PF is guaranteed. Furthermore, the performance is typically sensitive to the setting of the algorithm parameters (e.g., population size, number of iterations, genetic operators parameters), whose tuning depends on the specific problem instance being solved.

The active learning of Pareto fronts (ALP) algorithm introduced in this paper is different from existing EMOAs. Because the PF has infinite cardinality in the case of continuous MOPs, ALP generates an analytical representation of the PF, rather than an approximation by a finite and preset number of points. An analytical representation of the entire Pareto front may significantly improve the decision-making process, particularly when the preferences of the decision maker (DM) cannot be stated *a priori*. The compact analytical representation of the PF offers to the DM the possibility of visually inspecting the front and of focusing on the preferred regions and selecting the favorite solution $\hat{\mathbf{z}}$, as the desired compromise between the different objectives. ALP can then identify the solution in the decision space corresponding to $\hat{\mathbf{z}}$. With a large number of objectives, dimensionality reduction techniques [4] may be employed to enable the investigation of the learned analytical PF representation in a three- or two-dimensional visualization.

The ALP algorithm generates the analytical PF representation by learning a model of the PF from a training set of approximated Pareto-optimal vectors, which are obtained by solving different SOP instances. In order to minimize the computational effort (measured as number of evaluations of

Manuscript received June 27, 2012; accepted July 20, 2013. Date of publication September 23, 2013; date of current version February 14, 2014. This work was supported by PRIN under Grant 2009LNP494 from the Italian Ministry of University and Research.

The authors are with the Department of Information Engineering and Computer Science, University of Trento, Trento I-38123, Italy (e-mail: campigotto@disi.unitn.it; passerini@disi.unitn.it; battiti@disi.unitn.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2013.2275918

the MOP objective functions), informative training objective vectors are selected by applying active learning principles (AL). In particular, the learning stage is accomplished by Gaussian process (GP) regression [5], as it provides an explicit measure of predictive uncertainty that can be used to guide the selection of the training examples (active learning by uncertainty sampling [6]). The adoption of the AL paradigm also favors the anytime property: when increasing the number of function evaluations, the accuracy of the analytical PF representation improves. Furthermore, the GP method offers a natural termination criterion for ALP: when the information gain obtained by including any additional training example is negligible, the algorithm stops. The training objective vectors are generated by solving different instances of a scalarized optimization problem. ALP does not require any derivative information and is a generic framework: neither the machine learning method learning the PF model nor the AL principles and the optimization techniques for solving SOPs instances are limited to the ones discussed in this paper.

Let us note that there are two possible connections between machine learning and multiobjective optimization. The first connection arises because machine learning problems contain challenging multiobjective optimization tasks, from general cases, such as the tradeoff between intracluster similarity and intercluster dissimilarity in clustering problems, to specific cases, such as reinforcement learning problems with multiple conflicting reward functions [7], [8]. The second connection, implemented in this paper, goes in the opposite direction: it uses learning techniques from the machine learning community to solve the general MOP.

The next section describes the problem settings assumed in this paper. Details of the ALP algorithm are provided in Section III, while Section IV describes the GP regression technique used to model the Pareto front. Section V reviews different active learning strategies for regression tasks, with particular emphasis on the uncertainty sampling principle applied within the GP regression, as this is the strategy adopted by ALP. Related work is discussed in the following section. An experimental comparison between ALP and state-of-the-art EMOAs is reported in Section VII, while the ability of ALP in learning disconnected Pareto fronts is evaluated in Section VIII. The experiments in Section IX validate the choice of the active learning strategy based on the predictive uncertainty of the GP regression model. Finally, possible generalizations of ALP and interesting directions for future research are discussed in Section X. A more comprehensive explanation of GP regression, an exhaustive discussion of the related work, and a more detailed description of the experiments performed can be found in the preprint technical report of this paper [9].

II. Problem Settings

The ALP algorithm introduces a new strategy to tackle MOPs. In this Section, we give the formal definition of the MOP, specifying the terminology used throughout this paper, and define the properties of the Pareto front which our algorithm relies on.

A. Multiobjective Optimization Problem

An MOP is formulated as follows:

$$\min_{\mathbf{x}} \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\} \quad \text{subject to} \quad \mathbf{x} \in \Omega \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of n decision variables; $\Omega \subset \mathbb{R}^n$ is the feasible region and is typically specified as a set of constraints on the decision variables; and $\mathbf{f} : \Omega \to \Phi \subset \mathbb{R}^m$ is made of m objective functions which need to be jointly minimized. Objective vectors are images of decision vectors and can be written as $\mathbf{z} = \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$. Problem (1) is ill-posed whenever the objective functions are conflicting, a situation which typically occurs in real-world applications. In these cases, an objective vector is considered optimal if none of its components can be improved without worsening at least one of the others. An objective vector $\mathbf{z}$ is said to dominate $\mathbf{z}'$, denoted as $\mathbf{z} \succ \mathbf{z}'$, if $z_k \leq z'_k$, $k = 1, 2, \ldots, m$, and there exists at least one $h \in \{1, 2, \ldots, m\}$ such that $z_h < z'_h$. A point $\mathbf{x}$ is Pareto-optimal if there is no other $\mathbf{x}' \in \Omega$ such that $\mathbf{f}(\mathbf{x}')$ dominates $\mathbf{f}(\mathbf{x})$. The set of Pareto-optimal solutions is called the Pareto set (PS). The corresponding set of Pareto-optimal objective vectors is called Pareto front (PF). The ideal objective vector represents the best value of each objective. It is generally infeasible and is obtained by separately minimizing each objective function in the feasible region, i.e., $z_k^{\mathrm{ID}} = \min_{\mathbf{x}\in\Omega} f_k(\mathbf{x})$. Analogously, the worst objective vector contains the maximum possible value of each objective over the entire search space. Another notable vector in the objective space is the nadir point, whose kth component is the maximum value of the kth objective among all the Pareto-optimal vectors: $z_k^N = \max_{\mathbf{x}\in\Omega^*} f_k(\mathbf{x})$, where $\Omega^* \subset \Omega$ is the set of the Pareto-optimal solutions. The ideal and nadir points define the range of values that the Pareto-optimal objective vectors may attain.

B. Assumptions

Under mild smoothness conditions, the Pareto front of continuous MOPs is an $(m-1)$-dimensional piecewise-continuous manifold [1], [10], with m being the number of objectives. The dominance relation defined in the objective space enables a further characterization of the PF by expressing an arbitrary objective z_d (dependent function variable) as a function of the remaining objectives $\mathbf{z}_I$ (independent function variables): $z_d = g(\mathbf{z}_I)$. Without loss of generality, the mth objective of an m-objective problem is considered the dependent objective, i.e., $z_m = g(\mathbf{z}_I)$, with $\mathbf{z}_I$ including objectives $z_i, i = 1, \ldots, m-1$.

In this paper, we focus on bi-objective optimization problems, where the PF is characterized by the continuous function $z_2 = g(z_1)$. The only assumption of ALP is the knowledge of the domain of g. When the PF is connected, the domain of g is a closed interval of real numbers, and it is thus completely specified by the lower and the upper bound of the interval. While the lower bound can be easily obtained by computing the value z_1^{ID} of the ideal point, the upper bound can be specified by the decision makers themselves, who are often aware of a critical threshold defining the set of

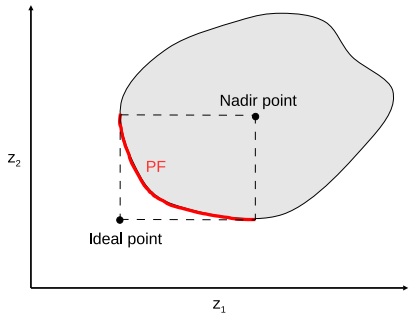

Fig. 1. Bi-objective minimization problem. The gray-shaded area denotes the feasible objective space, and the bold-marked curve depicts the PF. The ideal and nadir points bounding the PF are highlighted.

interesting values for a given objective. When the decision makers cannot provide the desired information, the upper bound defining the domain of g is represented by the value z_1^N of the nadir point, which, in the case of bi-objective problems can be computed exactly (see Fig. 1). In fact, in the case of bi-objective optimization problems, each component of the nadir point is obtained from the Pareto-optimal objective vector containing the ideal value for the other component. Therefore, the component z_1^N can be determined by solving the following program:

$$z_1^N = \min_{\mathbf{x}} \quad f_1(\mathbf{x}) \qquad (2)$$
$$\text{subject to } f_2(\mathbf{x}) = z_2^{\text{ID}}$$
$$\mathbf{x} \in \Omega.$$

For MOPs with more than two objectives or bi-objective MOPs with a disconnected PF, the domain of function g cannot be defined by just a couple of (user supplied) points. In Section VIII, our approach is generalized to bi-objective MOPs with disconnected Pareto fronts, where the *a priori* knowledge of the domain of g is an impractical assumption. Finally, Section X outlines a possible generalization of ALP to tackle MOPs with the number of objectives larger than two.

III. ALP Algorithm

The dominance relation enables a functional formulation of the PF by expressing an arbitrary objective z_d as a continuous function g of the remaining objectives $\mathbf{z}_I$. Without loss of generality, in this paper the objective z_m is expressed as a function of the remaining objectives. The ALP algorithm recovers an analytical model of the PF by solving a supervised regression task with input feature vector $\mathbf{z}_I$ and output z_m. The supervised information consists of approximated Pareto-optimal vectors, providing training examples in the form $(\mathbf{z}_I, z_m)$. This section first motivates the choice of modeling the PF and then presents the ALP framework in detail.

A. Motivation for Our Approach

Continuous MOPs have an infinite number of Pareto-optimal vectors, each one representing a compromise among the conflicting objectives, and the PF is usually is a continuous manifold. The evolutionary techniques approximate the PF by identifying a finite and discrete number of Pareto-optimal vectors (discrete PF representation). On the contrary, the ALP algorithm learns an analytical representation of the PF. This approach provides to the DM the ability of evaluating and selecting any possible Pareto-optimal vector, thus improving the decision-making process, particularly when the DM preferences cannot be clearly stated *a priori*.

Learning an analytical model of the PF is a more affordable task than recovering an analytical approximation of the PS. As a matter of fact, a functional formulation of the PS is not possible. Under mild smoothness conditions, the PS is a manifold [11], but its dimensionality, larger or equal to $m-1$, is in general unknown. Furthermore the PS manifold is embedded in the decision space, and the dimensionality of the decision space is usually much larger than that of the objective space ($n \gg m$). Learning an analytical model of the arbitrarily complicated PS shape is therefore computationally more expensive, and, in general, even not necessary. Once an analytical model of the PF is provided to the DM, he/she can search for any preferred compromise z^{DM} among the objectives. The feasible solution in the decision space associated with the selected compromise vector can be identified by minimizing with respect to $\mathbf{x} \in \Omega$ the distance between $\mathbf{f}(\mathbf{x})$ and z^{DM}.

B. Framework

Let us assume that a supervisor, which provides the value z_m for each training input $\mathbf{z}_I$, is available. Given an initial set T of training examples in the form $(\mathbf{z}_I, z_m)$ (initialization phase), ALP iteratively learns a model $\tilde{g}$ (refinement phase), providing the analytical representation of the PF. Each refinement iteration consists of training the model $\tilde{g}$ on the current training set T and, based on the learned model, selecting a new training vector $\hat{\mathbf{z}}_I$. The new training example $(\hat{\mathbf{z}}_I, \hat{z}_m)$, where $\hat{z}_m$ is the supervised information, is included in T. The update of the training set T completes the refinement iteration.

The pseudocode of ALP is given in Fig. 2. In the following, we provide the details of the algorithm.

1) Initialization Phase: The initial training set T is generated by selecting v vectors uniformly at random within the input space S of the regression task, spanned by the $m-1$ independent objectives $\mathbf{z}_I$. Supervised information for each selected vector is generated.

2) Refinement Phase (Active Learning): Training examples generation is expensive, as computing the supervised information involves the evaluation of the objective functions $\mathbf{f}$ of the MOP. In order to minimize the number of training examples without affecting the accuracy of the learned model, training inputs are selected by the uncertainty sampling principle [6]. At each refinement iteration, the new training input is the feature vector $\hat{\mathbf{z}}_I$ in S on which the prediction of the current model is most uncertain. The couple $(\hat{\mathbf{z}}_I, \hat{z}_m)$, with $\hat{z}_m$ being the supervised regression output, is considered the most informative training data for the current model.

Given the training set T, the model $\tilde{g}$ approximating the PF is learned by applying Gaussian process regression (GPR).

```
1.  procedure ALP
2.      input: multi-objective optimization problem
3.      output: analytical PF representation
4.      Let S the input space of the regression task
5.      /* Initialization phase */
6.      Select v training inputs z_I uniformly at random in S
7.      Generate regression output z_m for each training input by solving v instances of problem (3) /* Supervision */
8.      Initialize training set T by the v instances (z_I, z_m)
9.      Remove dominated training examples from T /* Dominance-based filtering */
10.     /* Refinement phase */
11.     do
12.         Train GP regression model on set T /* Modeling */
13.         Select most informative training input ẑ_I /* Active learning */
14.         Generate regression output ẑ_m for ẑ_I by solving problem (3) /* Supervision */
15.         Include training example (ẑ_I, ẑ_m) in T
16.         Remove dominated training examples from T /* Dominance-based filtering */
17.     until (termination_criterion)
18.     return learnt GP model /* Analytical PF representation */
```

Fig. 2. Pseudocode for the ALP algorithm generating the analytical PF representation by iteratively refining the learned GP model.

This choice is motivated by the ability of GP learners in quantifying prediction uncertainties, which enables a suitable application of the uncertainty sampling principle. GP learners provide a Gaussian distribution $\mathcal{N}(\mu(\bar{\mathbf{z}}_I), \sigma(\bar{\mathbf{z}}_I))$ of the values predicted for any single test input $\bar{\mathbf{z}}_I$. The mean $\mu(\bar{\mathbf{z}}_I)$ of the predictive distribution can be interpreted as the prediction $\tilde{z}_m$ of the GPR model $\tilde{g}$ when applied on test input $\bar{\mathbf{z}}_I$, while the variance of the distribution quantifies the confidence of the model about the prediction. Large variance means that the test sample is not represented by the model learned on the current training examples. Therefore, the point $\hat{\mathbf{z}}_I$ in S maximizing $\sigma(\mathbf{z}_I)$ is used to generate an informative training example for the current model $\tilde{g}$. A description of GPR including all aspects relevant to our algorithm is reported in Section IV.

3) Supervised Information Generation: Supervised information consists of the regression output, i.e., the value of the dependent objective $\hat{z}_m$, for a given input feature vector $\hat{\mathbf{z}}_I$. The couple $(\hat{\mathbf{z}}_I, \hat{z}_m)$ constitutes a training example. The value $\hat{z}_m$ is obtained by solving the following mathematical problem:

$$\begin{aligned} \min_{\mathbf{x},\epsilon} \quad & f_m(\mathbf{x}) \qquad (3) \\ \text{subject to} \quad & \\ & f_j(\mathbf{x}) = \hat{z}_j + \epsilon_j,\ j = 1, \ldots, m-1, \\ & |\epsilon_j| \leq 10^{-2} \\ & \mathbf{x} \in \Omega \end{aligned}$$

where $\Omega \subseteq \mathbb{R}^n$ identifies the feasible region of the MOP. Let $\hat{\mathbf{x}}$ be the solution of the problem (3). Then, $\hat{z}_m = f_m(\hat{\mathbf{x}})$. The objective vector $\hat{\mathbf{z}} = \{\hat{\mathbf{z}}_I, \hat{z}_m\}$ is Pareto-optimal. Slack variables ϵ_j in problem (3) relax the equality constraints $f_j(\mathbf{x}) = \hat{z}_j$. Equality constraints relaxation is introduced to solve instances of problem (3) suffering from the presence of local minima. When $|\epsilon_j| > 0$ for at least one value of index j, the new training input is a point $\tilde{\mathbf{z}}_I$ in the neighborhood of $\hat{\mathbf{z}}_I$. As position $\tilde{\mathbf{z}}_I$ is located in the region where the predictive uncertainty is higher, it represents an informative query point. The tolerance value of 10^{-2} in problem (3) is in general related to the input range and to the objective functions, the value of 10^{-2} is adequate for all benchmark problems (and the experimental results show that this choice is a robust setting).

4) Dominance-Based Filtering: When the solution of problem (3) is an approximation $\tilde{\mathbf{x}}$ of the Pareto-optimal solution $\hat{\mathbf{x}}$, a noisy training example $(\tilde{\mathbf{z}}_I, \tilde{z}_m)$, with $\tilde{z}_m = f_m(\tilde{\mathbf{x}}) > \hat{z}_m$, may be generated. Therefore, when new training examples are included in the training set T, a "dominance check" is performed to detect and remove training examples which are dominated by the new training examples.

5) Termination Criterion: The ALP approach is a general framework to solve MOPs, enabling different termination criteria. A simple one is provided by the AL paradigm. When the information gain obtained from any additional training example is negligible, the accuracy of the learned PF model is not expected to improve in a significant manner, and the algorithm stops. The information gain is estimated by the uncertainty of the model about its predictions for the candidate training examples. Alternative termination criteria for the iterative learning process include a limit on the number of refinement iterations or an upper bound on the number of evaluations of the MOP objectives (limit on the computational effort).

6) Computational Complexity: The computational complexity of ALP is dominated by the GP training, which takes time $O(|T|^3)$ (see Section IV). However, because of the limited number of training examples that can be used by ALP to efficiently recover the PF, the GP training is completed in a negligible amount of time. Furthermore, in real-world MOPs, the runtime bottleneck is typically represented by the evaluation of the objective functions. The fitness computation may involve, for example, time-consuming experiments or simulations. In some real-world cases, the analytical formulation of the fitness surface may even not exist, e.g., when one objective is the optimization of quantitative judgments provided by the decision maker.

Training the GP does not require the evaluation of the objective functions, which is instead needed by the generation of the training outputs [see (3)]. Therefore, the computational cost of ALP corresponds to the effort spent in generating the supervised information.

IV. Gaussian Process Regression

The ALP algorithm learns an analytical approximating of the PF by applying the GPR [5], [12]. This choice is motivated by the ability of GP learners to provide an explicit uncertainty model for the individual predictions, thereby enabling a suitable application of the uncertainty sampling principle [6]: an input on which the current learner has maximum uncertainty is selected as new training example.

In this section, we describe the regression based on GPs. First, we show how the traditional regression task is tackled by using GPs. Section IV-B elucidates how to make predictions from a trained GP. The training (i.e., learning) of a GP model is explained in Section IV-C.

A. Gaussian Process Model

The traditional regression task consists of estimating a latent function $g(\mathbf{x})$ from a noisy training dataset $T = \{(\mathbf{x}_i, y_i), i = 1, \ldots, |T|\}$. A GP is a collection of random variables, any finite subset of which have consistent joint Gaussian distributions. The consistency requirement means that the joint distribution of any finite subset of random variables is obtained from the joint distribution of any arbitrary superset of the original subset by marginalizing out the additional variables.

When a GP is used to model the regression task, the random variables represent the values of the latent function $g(\mathbf{x})$ at different locations $\mathbf{x}$. Therefore, the random (function) variables can be indexed by the continuous function $g(\mathbf{x})$. In this paper, g_i denotes the random function variable associated with the input $\mathbf{x}_i$ which characterizes the possible values for $g(\mathbf{x}_i)$. By the definition of GP, n arbitrary random function variables $\mathbf{g} = \{g_1, g_2, \ldots, g_n\}$ have a joint Gaussian distribution

$$\mathrm{p}(\mathbf{g}|X) = \mathcal{N}(\bar{\mathbf{g}}, K) \tag{4}$$

where X is the set of corresponding inputs (indexes), $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, and $\mathcal{N}(\bar{\mathbf{g}}, K)$ denotes a multivariate Gaussian distribution with mean vector $\bar{\mathbf{g}}$ and covariance matrix K.

A GP is a conditional probabilistic model. The distribution $\mathrm{p}(\mathbf{x})$ on the inputs is not specified, and only the conditional distribution $\mathrm{p}(\mathbf{g}|X)$ is actually modeled. For notational simplicity, in the rest of this paper the explicit conditioning on the input is omitted, with the convention that $\mathrm{p}(\mathbf{g})$ stands for $\mathrm{p}(\mathbf{g}|X)$, where X represents the appropriate inputs where the function variables $\mathbf{g}$ are conditioned on.

A GP is completely specified by the mean $\mu(\mathbf{x})$ and the covariance $c(\mathbf{x}_i, \mathbf{x}_j)$ functions. The mean function is usually set to zero (bias and offsets can be subtracted from the data without losing generality); therefore the definition of the GP reduces to the choice of the suitable covariance function, which measures the similarity between inputs $\mathbf{x}$ by computing the covariance among the function variables g associated with the inputs. The covariance matrix K in (4) is calculated from the covariance function $K_{i,j} = \mathrm{cov}(g_i, g_j) = c(\mathbf{x}_i, \mathbf{x}_j)$.

B. Gaussian Process Prediction

Gaussian process regression (GPR) assumes that the observations $y_i = g(\mathbf{x}_i) + \epsilon$ are affected by white noise ϵ, following an independent and identically distributed Gaussian distribution with zero mean and variance σ_n^2. Let $\mathbf{y} = \{y_1, y_2, \ldots, y_{|T|}\}$. Analogous to the symbol $\mathrm{p}(\mathbf{g})$, in this paper $\mathrm{p}(\mathbf{y})$ is used as short notation for $\mathrm{p}(\mathbf{y}|X)$, where X is the set of training inputs corresponding to the observations $\mathbf{y}$.

GPR is a Bayesian probabilistic method. A GP is used to express the prior belief about the function $g(\mathbf{x})$ to be modeled, before considering any training observation. This GP is thus referred to as prior GP. By using a prior GP, the joint distribution of the latent function variables $\mathbf{g}$ given the training inputs X is a multivariate Gaussian $\mathrm{p}(\mathbf{g}) = \mathcal{N}(\bar{\mathbf{g}}, K)$. This probability distribution defines the prior for Bayesian inference. The components of $\bar{\mathbf{g}}$ are usually set to zero, thus the Gaussian can be rewritten as $\mathrm{p}(\mathbf{g}) = \mathcal{N}(\mathbf{0}, K)$. Under the assumption of Gaussian white noise affecting the observations $\mathbf{y}$, a suitable noise model (or likelihood) is $\mathrm{p}(\mathbf{y}|\mathbf{g}) = \mathcal{N}(\mathbf{g}, \sigma_n^2 I)$, where I is the identity matrix. By integrating over the unobserved function variables $\mathbf{g}$, the marginal likelihood (or evidence) is obtained as

$$\mathrm{p}(\mathbf{y}) = \int \mathrm{p}(\mathbf{y}|\mathbf{g})\mathrm{p}(\mathbf{g})\mathrm{d}\mathbf{g} = \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I). \tag{5}$$

The term marginal refers to the marginalization over $\mathbf{g}$. With the prior, the likelihood and the evidence (5) components, Bayesian inference can be applied to make a prediction on a test input $\mathbf{x}_*$. In particular, GPR estimates the probability distribution $\mathrm{p}(g_*|\mathbf{y})$ over the candidate values for $g(\mathbf{x}_*)$ at the test input $\mathbf{x}_*$, rather than providing a single prediction (estimated best guess) for $g(\mathbf{x}_*)$. The predictive distribution $\mathrm{p}(g_*|\mathbf{y})$ is a Gaussian $\mathcal{N}(\bar{g}_*, \sigma^2(g_*))$, with mean and variance given by

$$\bar{g}_* = \mu(\mathbf{x}_*) = \mathbf{k}_*'(K + \sigma_n^2 I)^{-1}\mathbf{y} \tag{6}$$

$$\sigma^2(g_*) = k_* - \mathbf{k}_*'(K + \sigma_n^2 I)^{-1}\mathbf{k}_*. \tag{7}$$

The value $\mu(\mathbf{x}_*)$ is used as the estimation for g_*, while the variance $\sigma^2(g_*)$ defines how much the GP model is confident with the estimation (larger variance means smaller confidence).

The Cholesky decomposition of matrix $K + \sigma_n^2 I$ enables a fast and numerically stable implementation of (6) and (7). However, the complexity of inverting the matrix $K + \sigma_n^2 I$ is $O(|T|^3)$, where $|T|$ is the number of training observations. Therefore, the application of GPR with more than few thousands of training examples is prohibitive. However, within our problem settings, much smaller datasets are used by the ALP algorithm, as the generation of training examples is expensive.

C. Gaussian Process Training (Model Selection)

The covariance function of a GP model typically has a set of free hyperparameters that control the shape of the Gaussian distributions. Training a GP model consists of tuning the covariance function hyperparameters, together with the Gaussian noise variance, to fit the current data. In the rest of this paper, the hyperparameters of the GP model, consisting of the covariance function parameters and the noise variance σ_n^2, will be collectively referred to by the vector θ.

The optimal value of θ can be inferred from the observations $\mathbf{y}$ by maximizing with respect to θ the marginal likelihood

defined in (5). For numerical reasons, the marginal log-likelihood is used

$$-\frac{|T|}{2}\log 2\pi - \frac{1}{2}\log |K + \sigma_n^2 I| - \frac{1}{2}\mathbf{y}'(K + \sigma_n^2 I)^{-1}\mathbf{y}. \quad (8)$$

The above quantity represents the log-evidence for the specific GP model defined by the hyperparameter settings θ.

In general, maximizing the log-likelihood is a nonconvex optimization task. Standard gradient-based optimization algorithms, e.g., conjugate gradient techniques or quasi-Newton methods, are typically adopted to search for locally optimal points. The cost of computing the log-marginal likelihood and its gradient is dominated by the inversion of the covariance matrix $K + \sigma_n^2 I$.

In addition to the optimization of the covariance function hyperparameters, the model selection procedure may also include the discrete choice between different functional forms (i.e., models) for the covariance function. In particular, the functional form can be selected by simply comparing the maximum likelihood values computed for each candidate model. Finally, rather than resorting to a zero-mean prior GP, a parametric formulation for the mean function can also be specified. Its hyperparameters are included in the vector θ and optimized together with the noise model and the covariance function hyperparameters.

V. Active Learning for Regression Tasks

This section provides an overview of different strategies for the active learning of real-valued functions, starting from the uncertainty sampling principle adopted by ALP within the GPR framework. In particular, we show how GPR enables a sound application of the uncertainty sampling principle.

Acquiring training examples for supervised learning tasks is typically an expensive and time-consuming process. Active learning approaches attempt to reduce this cost by actively suggesting inputs for which supervision should be collected, in an iterative process alternating learning and feedback elicitation. At each iteration, active learning methods select the inputs maximizing an informativeness measure, which estimates how much an input would improve the current model if it was added to the current training set. The informativeness of the query inputs can be defined in different ways, and several active learning techniques exist (see [13] for a recent review).

The uncertainty sampling (US) principle [13] considers the input with highest predictive uncertainty as the most informative training example for the current model. The US paradigm therefore requires a learning model that can quantify its predictive uncertainty. The ability of GPR in estimating the confidence for individual predictions enables a suitable application of the uncertainty sampling principle. A large variance $\sigma^2(g_*)$ of the predictive distribution $\mathcal{N}(\bar{g}_*, \sigma^2(g_*))$ for a single test input $\mathbf{x}_*$ means that the test sample is not represented by the GP model learned from the training data. The predictive variance quantifies the predictive uncertainty of the GP model, and therefore the input maximizing the predictive variance is selected by the uncertainty sampling principle. With GPR, the predictive uncertainty grows in regions away from training inputs.

Active learning strategies more sophisticated than the US principle exist, but they usually demand more expensive computation. The query-by-committee strategy [13] maintains a committee of models, trained on the current set of examples and representing competing hypotheses. Each committee member votes on the output value of the candidate inputs, and the input considered most informative is the one on which they disagree most. The expected model change principle [13] considers as the most informative query the input that, when added to the training set would yield the greatest change in the current model (i.e., that has the greatest influence on the model parameters). The expected error reduction criterion [13] selects the input that is expected to yield the greatest reduction in the generalization error of the current model. Computing the expected generalization error is, however, computationally expensive, and, in general, it even cannot be expressed analytically. To overcome this limitation, the variance reduction approach [13] queries the input that minimize the model variance. The generalization error can in fact be decomposed into three components, refereed to as the noise, the bias, and the model variance. The noise component defines the variance of the output distribution given the input and is independent of the model and the training set. The bias error is introduced by the model class (e.g., a linear model class adopted to approximate a nonlinear function). The model variance estimates how much the model predictions vary when changing the training set. Because the model parameters can influence neither the noise nor the bias, the only way to reduce the future generalization error consists of minimizing the model variance. An effective application of the variance reduction approach is possible only under the assumption that the bias error is negligible.

VI. Related Work

Estimation of distribution algorithms (EDAs) are evolutionary techniques that generate a probability distribution of promising solutions based on statistical information extracted from the current population. The probability distribution models the pattern characterizing the set of promising solutions. The pattern is generated by dependencies among the decision variables, often referred to as variable linkages [3]. At each generation of an EDA, the probability distribution is sampled to obtain new candidate solutions which are used to refine the current population. The motivation for EDAs is given by the limitation of common genetic operators: when two parents are in the PS, their offspring may not be close to the PS.

Among the evolutionary methods, EDAs are closest to our approach, sharing with ALP the extraction of global statistical information from the current data. However, our technique is not developed within the evolutionary framework. EDAs typically generate a probability distribution of promising solutions in the decision space. A finite PF approximation is provided by the population of the EDA. On the contrary, the model learned by ALP is an analytical PF approximation and the training examples are Pareto-optimal vectors generated by solving different SOPs. Furthermore, the SOPs are not fixed at the initialization phase, but they are dynamically generated based on the predictive uncertainty of the current PF model.

Selecting the new training inputs based on the predictive model uncertainty provides two advantages. First, it enables a computationally cheaper and accurate PF approximation. Second, because the predictive uncertainty increases when moving away from the current training inputs, a diversified set of Pareto-optimal solutions, corresponding to objective vectors distributed over the whole PF, is provided to the decision maker. A more comprehensive discussion of the related works can be found in the preprint technical report of this paper [9], including a description of the state-of-the-art MMEA algorithm [14], which is the ALP benchmark technique in our experimental studies.

VII. Experimental Results

ALP is tested over the RM-MEDA benchmark problems introduced in [11] to evaluate the ability of EDAs in recovering the Pareto front. As the best results over the RM-MEDA benchmark problems has been achieved by the MMEA algorithm [14], the latter represents the touchstone in our experimental studies. In particular, the optimal setting of MMEA parameters is taken from [14].

The experimental studies are as follows. First, the RM-MEDA benchmark is outlined. The setting of the ALP algorithm is then detailed and a single ALP run is shown, in order to provide a paradigmatic case of the PF approximation refinement observed during the learning iterations. The comparison with the MMEA algorithm is given in Section VII-E, following the description of the comparison metric adopted. In the preprint technical report of this paper [9], ALP is also tested over the welded beam design MOP [15], a widely used benchmark introduced in the spirit of real-world optimization tasks.

A. RM-MEDA Benchmark Problems

For a comparison with ALP, we consider the bi-objective problems of the RM-MEDA benchmark [11]. By using the numbering adopted in [14], the benchmark set is thus formed by the instances ZZJ08-*F1*, ZZJ08-*F2*, ZZJ08-*F3*, ZZJ08-*F5*, ZZJ08-*F6*, and ZZJ08-*F7*. These instances are derived from the widely known Zitzler–Deb–Thiele (ZDT) test problems [16]. The PS of the bi-objective ZDT problems is parallel to the coordinate axes because of the lack of dependencies among the decision variables. This deficiency introduces a bias in favor of the commonly used crossover operator: if two solutions are Pareto-optimal, their offspring is likely to be Pareto-optimal. Furthermore, variable linkages exist in many applications and their inclusion into test suites has been suggested by different works [17], proposing variable transformations for introducing dependencies among the decision variables. The RM-MEDA instances F1, F2, F3 introduce linear linkages in the formulation of the ZDT1, ZDT2, and ZDT6 test problems [16]. Nonlinear dependencies among the decision variables have been added to the definition of the same ZDT instances to generate the RM-MEDA problems F5, F6, F7. Their Pareto sets consist of bounded continuous curves.

B. ALP Setting

In order to avoid any bias favoring our method over the competitor, the PF manifold is arbitrarily expressed by considering z_2 as a function of z_1: $z_2 = g(z_1)$. To apply the ALP algorithm, the ideal and nadir points $\mathbf{z}^{\mathrm{ID}}$ and $\mathbf{z}^{N}$ of the above MOPs are computed offline. In particular, the domain $[z_1^{\mathrm{ID}}, z_1^{N}]$ of the regression task is equal to [0, 1] for all the considered MOPs, with the exception of problems *F3* and *F7* where the domain is the interval [0.281, 1]. ALP is implemented in MATLAB R2008a. In particular, the optimization problem (3) generating supervised information is solved by using the continuous local search algorithms for constrained optimization provided by the MATLAB Optimization Toolbox library. In detail, a sequential quadratic programming algorithm [18], [19] is used, except for problems *F3* and *F7*, where an interior-point method [20] is adopted. The choice of the optimization strategy is based on the observed convergence rate to the (local) optima of the function f_m. Both strategies are implemented by the fmincon MATLAB routine. A single run of fmincon algorithm is performed, initializing at random the starting point and limiting the number of scalarized function evaluations to 3000. For the considered MOPs, within the value of 3000, the algorithm usually converges to a (local) minimum of the function f_m. In general, ALP is a flexible framework enabling the usage of alternative constrained optimization algorithms (e.g., derivative-free approaches) rather than the fmincon routine to solve (3).

ALP training set is initialized by selecting two training examples uniformly at random within the input space. In the performed experiments, the mean function of the prior GP is the line $a * x + b$, with $\{a, b\}$ being the mean function hyperparameters. Three candidate forms for the covariance function are considered: the squared exponential, the neural network, and the Matérn covariance functions (see book [5] for their formulation). In combination with the mean and covariance functions, we make use of a white Gaussian noise model with variance ρ_n^2. The hyperparameters vector θ thus includes ρ_n^2, the mean function hyperparameters a and b and the covariance function hyperparameters. The model selection phase consists of two tasks: 1) the choice of the functional form for the covariance function and 2) the optimization of the hyperparameters θ. Both tasks are accomplished in one step by evidence maximization [see (8)]: for each candidate covariance function, the vector θ is optimized. The setting with largest likelihood value is then selected. In particular, the optimization task is solved by applying a conjugate gradient algorithm. Ten runs of the algorithm are performed, each one consisting of 50 conjugate gradient steps. A different starting point θ_{init} is used for each run. A suitable selection of the starting point, based on the prior knowledge about the desired PF model, drives the optimization algorithm toward (local) minima which provide plausible interpretations of the training data. In particular, the slope a of the mean function in vector θ_{init} is initialized to the value -1. In fact, any connected PF of a bi-objective minimization problem can be modeled by a strictly decreasing function. The gain from the inclusion of prior knowledge in the starting point design is more significant

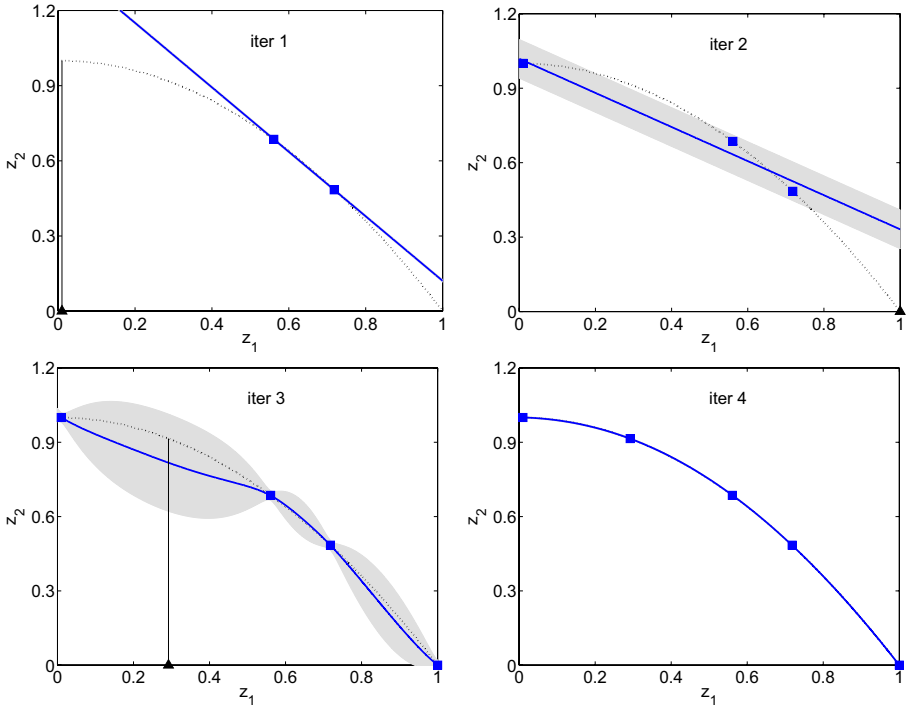

Fig. 3. Generation of most informative training examples while solving MOP *F2*. Each figure depicts the PF (dotted line) of the learned model (solid line). The figures refer to the first, second, third, and fourth refinement iterations, respectively. The shaded area denotes the 95% confidence interval for the predictive GP model. On the x-axis, the most informative training example, selected by active learning, is showed by a triangular marker. At the fourth iteration (second row, right figure) the PF is correctly learned.

at the initial iterations of the ALP algorithm, when the number of training examples is limited.

C. Single ALP Run in Detail

A single run of the ALP algorithm over MOP ZZJ08-*F2* is shown in Fig. 3. The predictive GP model is depicted by the solid line, while the unknown Pareto front $z_2 = g(z_1)$ is represented by the dotted line. The shaded area represents the 95% confidence interval for the predictive GP model. At the first iteration, with a couple of training examples only, the learned PF model is a line, while the order of magnitude of the predictive variance is 10^{-6} and thus the shaded area cannot be visualized. The inclusion of the new training example located at input $z_1 = 0.01$ (triangular marker over the x-axis) changes the slope of the line, while constant predictive uncertainty over the input space is observed. In this case, the ALP algorithm places the new query point at the input $\mathbf{z}_1$ maximizing the minimum distance from the training examples. At the third iteration, the accuracy of the learned model improves. Predictive uncertainty increases when moving away from training examples and the shaded area entails the Pareto front to be modeled. The uncertainty sampling method selects the input $z_1 = 0.29$ as the new training example. At the fourth iteration, with 1536 evaluations of the MOP objective functions $\mathbf{f}(\mathbf{x})$, the PF is successfully recovered. Additional refinement iterations slightly improve the accuracy of the learned model (however the improvements cannot be visually observed), while the predictive uncertainty becomes null. In general, the adoption of the active learning to generate the training examples favors the anytime property: when increasing the number of functions evaluations, the PF approximation improves.

D. Comparison Metric

The performance of the ALP and MMEA algorithms is evaluated by measuring the quality of the recovered PF approximation with respect to the number of objective function evaluations. The quality of the PF approximation is estimated by the inverted generational distance (IGD) metric. It evaluates the quality of the PF approximation recovered by an evolutionary multiobjective algorithm (EMOA) by measuring both the convergence and the diversity of its population. The adopted metric is calculated according to the procedure described in [14].

A test set P^* of Pareto-optimal objective vectors uniformly spread over the PF is generated. With P denoting the population of the EMOA, the IGD metric is defined as follows:

$$\mathrm{IGD}(P^*, P) = \frac{\sum_{\mathbf{z}^* \in P^*} d(\mathbf{z}^*, P)}{|P^*|} \tag{9}$$

where $d(\mathbf{z}^*, P)$ is the minimum Euclidean distance from the Pareto-optimal vector $\mathbf{z}^*$ to any objective vector in P, while $|P^*|$ indicates the cardinality of P^*. According to the notation used in [14], the IGD metric is denoted as IGDF because it measures the quality of the PF approximation (i.e., P is a set of objective vectors and $d(\mathbf{z}^*, P)$ is the Euclidean distance in the objective space). The lower the value $\mathrm{IGDF}(P^*, P)$, the better is the quality of the population P approximating the PF.

The IGDF metric measures the quality of a finite set of points (population) approximating the PF. In the case of ALP,

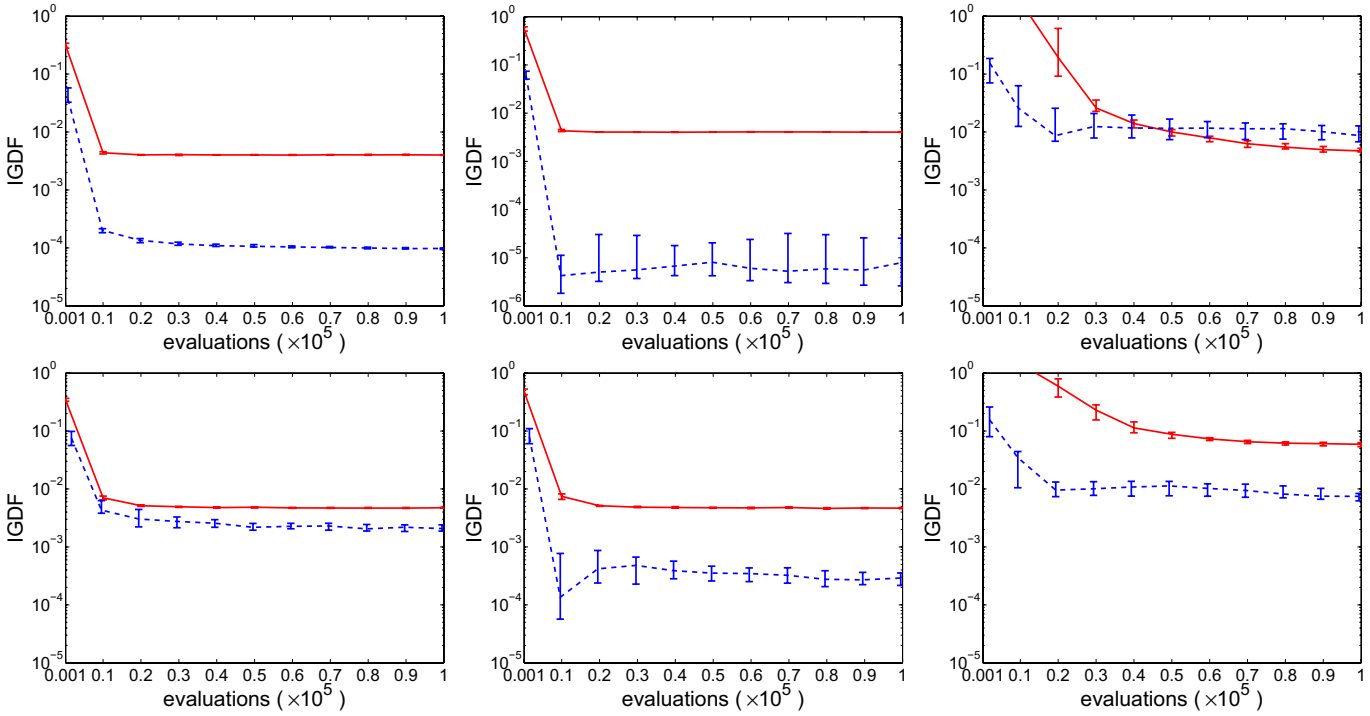

Fig. 4. Performance metric (median values) evolution over the number of function evaluations. Error bars represent the IQR of data distribution. Dashed and solid lines plot the ALP and MMEA results, respectively. Figures in row 1 refer to the MOPs *F1* (left), *F2* (middle), and *F3* (right). The second row contains the results for the MOPs *F5* (left), *F6* (middle), and *F7* (right).

which recovers an analytical PF model $\tilde{g}$ rather than a discrete PF representation, the population P is computed by sampling the analytical PF model. In particular, $|P^*|$ sample vectors of the form $\tilde{\mathbf{z}} = \{\mathbf{z}_I^*, \tilde{g}(\mathbf{z}_I^*)\}$ are generated by evaluating $\tilde{g}(\mathbf{z}_I^*)$ for each $\mathbf{z}^* \in P^*$. Any dominated objective vector among the $|P^*|$ samples generated is discarded. The IGDF metric can be used to evaluate analytical PF models which are (partially) infeasible (this behavior has been occasionally observed during the initial refinement iterations of ALP), as it is based on a distance measure.

E. Detailed Comparative Results

Fig. 4 compares the performance of the ALP and MMEA algorithms. The curves depict the evolution of the IGDF metric over the number of evaluations of the MOP objective vector $\mathbf{f}(\mathbf{x})$. The dashed and solid lines correspond to the ALP and MMEA algorithms, respectively. Each point is the median value over 20 runs, executed with different seeds. Error bars denote the interquartile range (IQR) of data distribution.

With the exception of MOPs *F3* and *F5*, the ALP algorithm dominates the MMEA technique, achieving on average a PF approximation with sensibly better quality. For example, over MOP *F1*, within 10 000 function evaluations, a more accurate Pareto front is recovered by the ALP algorithm, and after 10 000 evaluations, the quality of ALP solution is at least 12 times better than the quality of MMEA solution.

A stable behavior is observed for both algorithms. The large size of the ALP error bars in the case of MOP *F2* is due to the logarithmic scale. In fact, the order of magnitude of ALP IQR values for MOP *F2* is actually lower than that of MMEA ones. For example, when 70 000 function evaluations are performed, the orders of magnitude of the IQR values for ALP and MMEA results are 10^{-5} and 10^{-3}, respectively.

There is no qualitative difference in terms of the IGDF metric among ALP and MMEA results for MOP *F5*. When more than 30 000 function evaluations are used to solve MOP ZZJ08-*F3*, the performance of ALP is not better than that of MMEA. However, within 20 000 function evaluations, the quality of ALP solution is better by at least a factor 10 than the PF recovered by MMEA. Reasonable approximations obtained within few function evaluations are in many cases preferred to more accurate solutions which require a heavier computational effort. In fact, as noted by Lovison in [10], "even a roughly sketched global picture of the whole situation can give crucial information on the problem at hand, suggesting correctly the location of paramount zones."

Because of space limitations, the results observed when using the hypervolume difference metric [14] rather than the IGDF metric are not included here. They can be found in the preprint technical report of this paper [9]. In general, they are consistent with the performance measured by the IGDF metric, with the exception of MOP *F5*, where ALP achieves a sensibly better performance compared to MMEA in terms of the hypervolume difference metric.

VIII. Handling Disconnected Pareto Fronts

The ALP framework described in Section III assumes knowledge of the domain of the function $z_d = g(\mathbf{z}_I)$ characterizing the PF. In the case of bi-objective MOPs with connected PF characterized by $z_2 = g(z_1)$, this prior information can be

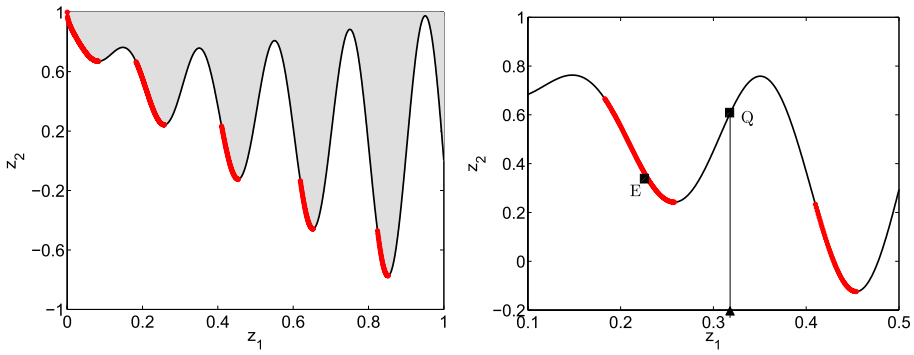

Fig. 5. Left: ZDT3 problem. The boundary (black thin curve) of the feasible region (gray-shaded area) contains both convex and concave parts. The disconnected PF is formed by the bold-marked curve segments. Right: When the query point (triangular marker on the z_1-axis) does not belong to the domain of g, the exact solution of the scalarized program providing supervised information generates an objective vector Q that may be dominated by the current training examples (point E in the graph).

easily obtained by computing the components z_1^{ID} and z_1^N of the ideal and the nadir point, respectively, or by asking the decision maker for the interval of the interesting values of the objective z_1.

When the PF is disconnected, the function g is discontinuous and its domain cannot be defined by a couple of z_1 values only. Therefore, assuming *a priori* knowledge of the domain of g is impractical. However, a disconnected PF can be identified by learning a regression function $z_2 = h(z_1)$ approximating the entire lower boundary of the feasible objective region, including the dominated portions of the boundary (e.g., the concave parts). As a matter of fact, the PF is a subset of the objective boundary and thus a model of the disconnected PF can be extracted straightforwardly by sampling the function h and keeping the nondominated samples only (e.g., the points lying in the concavities of the function h are discarded). For example, Fig. 5 (left) shows the feasible objective region of the ZDT3 problem, taken from the ZDT benchmark suite [16]. The boundary of the feasible region (gray-shaded area) is the thin black curve, consisting of both convex and concave parts. The disconnected PF is depicted by the bold-marked portions of the black curve.

When the ALP algorithm is used to learn the function h, both dominated and nondominated examples are needed. The former are in fact used to model the concave portions of the boundary (to be discarded when presenting the PF). In order to allow this, the dominance-based filtering discarding false Pareto-optimal training examples has to be switched off. Otherwise, the learning phase may get stuck when trying to model the concavities of the boundary, as other dominating training examples will likely be present in this case [see Fig. 5 (right)].

In order to model entire feasible region boundaries, an additional functional form has been included in the set of candidate covariance functions considered by the GP model selection procedure. This consists of a combined function, obtained summing up a periodic covariance function c_p with a squared exponential covariance function c_s (see [5] for their formulation). The rationale for the choice of function c_p is the modeling of the concavities characterizing the boundary of the feasible area. The combination with function c_s enables a decay away from exact periodicity, as the boundary shape is not expected to be exactly periodic. Furthermore, the squared exponential covariance function models the smooth trend characterizing disconnected Pareto fronts, which, in the case of bi-objective minimization problems, are monotonically decreasing piecewise continuous functions.

Let us note that this combined covariance function is not designed for a specific MOP but is a general choice driven by the typical form of feasible region boundaries. Furthermore, we are not imposing this covariance function in the prior GP, but only suggesting it to the model selection procedure as a candidate covariance function. The identification of the covariance function (and its parameters setting) that best fits (i.e., explains) the training data is performed by the model selection procedure.

No further modifications are required to adapt the ALP algorithm presented in Section III to model disconnected PFs.

A. Case Studies With Disconnected PF

The ability of ALP in learning disconnected Pareto fronts is evaluated over four well-known MOPs: the ZDT3 [16], D99 [2], [21], TNK [22], and KUR [23] problems. The selected MOPs differ from each other in several features. The preprint technical report of this paper [9] provides the problem formulation and compares their features.

The MMEA algorithm, which was used as touchstone in the experiments of Section VII, has been designed for connected Pareto fronts and cannot represent a fair ALP competitor over the four selected problems. ALP is therefore compared with both the popular state-of-the-art genetic algorithm NSGA-II [24] and with the state-of-the-art estimation of distribution algorithm RM-MEDA [11]. We used the original NSGA-II code provided by the authors (version 1.1.6) and, for each MOP, the specific setting of the NSGA-II parameters available with the code (the parameter values are detailed in the preprint technical report of this paper [9]). For RM-MEDA, the original implementation is used, together with the default robust parameter setting specified in the original paper [11] (the alternative parameter settings tested in our investigation did not yield any sensible performance improvement).

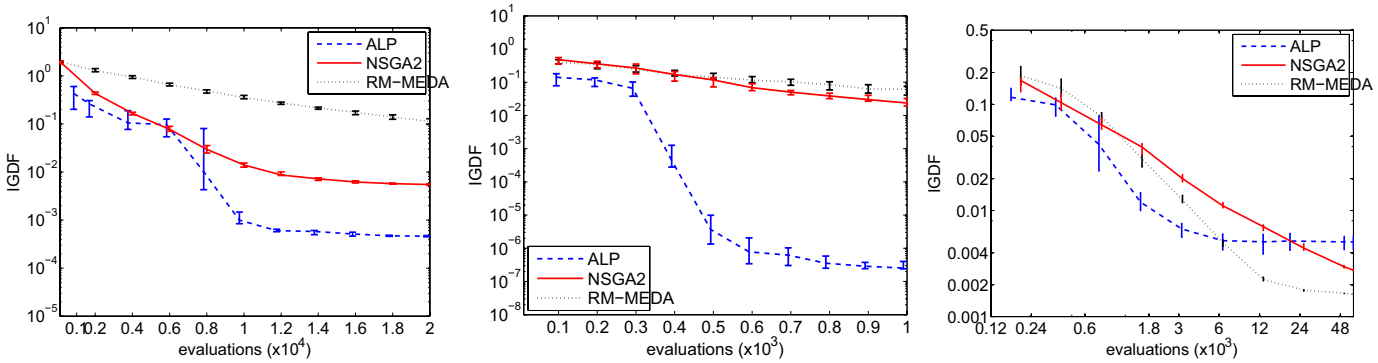

Fig. 6. Performance metric (median values) evolution over the number of function evaluations when solving the ZDT3 (left graph), D99 (middle graph), and TNK (right graph) MOPs. Error bars represent the IQR of data distribution. The blue dashed line plots the ALP results, while the red solid line shows the NSGA-II performance and the black dotted line depicts the observed RM-MEDA values. The right graph adopts the logarithmic x-scale to represent both the early ALP convergence within 6000 evaluations, and the two crossover points of the performance, located at 6000 and at about 20 000 function evaluations.

The ALP algorithm outperforms NSGA-II and RM-MEDA over the ZDT3, D99, and TNK MOPs, while it fails in learning an accurate model of the KUR PF. Fig. 6 reports the convergence of ALP, NSGA-II, and RM-MEDA to the Pareto front of the ZDT3 (left graph), D99 (middle graph), and TNK (right graph) MOPs. The blue dashed, red solid, and black dotted curves plot the average IGDF performance (median value over 20 runs) of the ALP, NSGA-II, and RM-MEDA algorithms. The error bars plot the interquartile range (IQR) of the data distribution.

A detailed description of the experimental results is provided by the preprint technical report of this paper [9], including also sample PF approximations, showing that the difference among the IGDF values observed in Fig. 6 for ALP and for the benchmark algorithms is statistically significant and substantial. The rest of this section summarizes the main results obtained for each MOP.

1) ZDT3 MOP: With 6000 function evaluations (left graph in Fig. 6), the performance of ALP is comparable with that of NSGA-II and about one order of magnitude better than that of RM-MEDA. When progressively increasing the number of function evaluations from 6000 up to 20 000, the quality of the ALP model improves more rapidly than that of NSGA-II and RM-MEDA. Eventually, the ALP performance converges to an IGDF value of less than 10^{-3}, about one and three orders of magnitude better than that of NSGA-II and RM-MEDA, respectively.

2) D99 MOP: The superior performance of ALP with respect to NSGA-II and RM-MEDA observed for ZDT3 is confirmed. The quality of the PF approximation returned by ALP rapidly improves after 300 evaluations (middle graph in Fig. 6), converging to a value smaller than 10^{-6} after 600 evaluations. On the other hand, after 1000 objective function evaluations, the performance of both NSGA-II and RM-MEDA is still five orders of magnitude worse than the ALP performance.

3) TNK MOP: The quality of the PF approximation returned by ALP converges to the IGDF value of $4e^{-3}$ within 6000 function evaluations (right graph in Fig. 6). When considering a bounded computational budget of 600–3000 function evaluations, the results by NSGA-II and RM-MEDA are two or three times worse than ALP performance. However, with 6000 function evaluations, the rank of the algorithms changes, with RM-MEDA asymptotically dominating ALP. As a matter of fact, ALP learns a reasonable approximation within fewer function evaluations than NSGA-II and RM-MEDA, but it misses a small portion of the PF (see [9] for details). This weakness is due to the poor quality of the training examples generated in the interval considered. On the other hand, NSGA-II and RM-MEDA reach a uniform spread of the individuals of the population over the whole PF, but at a computational cost larger (considerably larger, in the case of NSGA-II) than the number of function evaluations performed by ALP at convergence.

4) KUR MOP: For this problem, the sequential quadratic programming algorithm [18], [19] used by ALP to generate training examples [see (3)] returns extremely poor quality approximations of the Pareto-optimal solutions. Based on this very noisy training information, an accurate PF model cannot be recovered. However, ALP can easily realize the poor quality of the learned model by measuring its predictive uncertainty, which is exceptionally large also in regions densely populated by training examples. Possible countermeasures for this suboptimal behavior can be found in [9].

IX. Efficiency of Uncertainty Sampling

In order to learn a model of the PF, ALP generates training examples in the regression domain. However, generating supervised information is expensive, as it involves the evaluation of the objective functions of the MOP. In order to decrease the computational cost of the learning process, the next query instance selected by ALP is the point $\hat{z}_1$ in the regression domain maximizing the predictive uncertainty of the learned PF model (uncertainty sampling). The adoption of the uncertainty sampling principle motivates the choice of the GPR method, which can quantify the uncertainty of the predictive model.

The following experiments test the efficiency of the uncertainty sampling (UNC) principle, by comparing it with an alternative query selection strategy. The competing method consists of picking the query points uniformly at random within the regression domain. The random selection principle

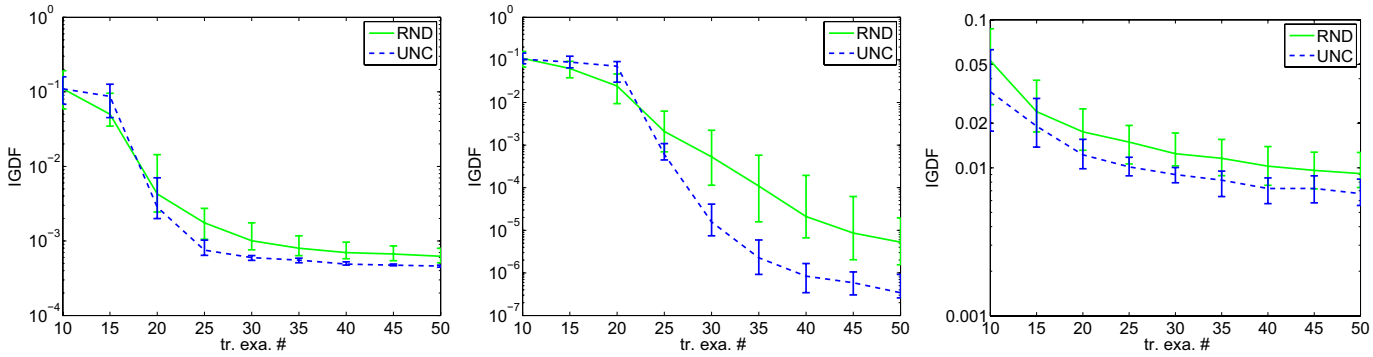

Fig. 7. Comparison between active selection of training examples based on model uncertainty (UNC) and passive selection based on random sampling (RND). The dashed lines plots the uncertainty sampling results (i.e., the performance of the original ALP algorithm), while the solid lines show the performance of the ALP variant with uncertainty sampling replaced by random sampling. Median IGDF values over 100 runs are plotted for an increasing number of training examples, with error bars representing the IQR of data distribution. The left, middle, and right graphs refer to the ZDT3, D99, and TNK MOPs, respectively.

(RND) is less informed than the UNC method, as it does not exploit the learned PF model. However, the RND principle provides a robust approach with respect to the general shape of the PF, thus representing a nontrivial competitor in our settings.

Fig. 7 reports the performance of both the UNC and RND sampling principles when increasing the number of query points to solve ZDT3 (left graph), D99 (middle graph), and TNK (right graph). The dashed and solid lines refer to the UNC and the RND strategies, respectively. For both strategies, 100 runs of ALP are performed, with different random seeds. Median values of the IGDF metric over the different runs are plotted, together with error bars denoting the 25th and the 75th percentiles.

For ZDT3 (left graph), with 10 (or less) training examples, the UNC and RND strategies are statistically equivalent under a Kolmogorov–Smirnov (KS) test with a Bonferroni-corrected confidence level of 99.5%. With 15 query points, there is statistical evidence for better results by the RND method (the p-value is 0.003). However, when increasing the number of training examples, the UNC strategy keeps improving faster than the RND strategy and significantly outperforms it if 25 or more query points are generated. With more than 25 examples, the order of magnitude of the observed p-values is lower than $1e^{-14}$, providing strong evidence against the null hypothesis of equivalent performance.

The observed behavior is consistent with intuition. The limited amount of supervised information generated by few training examples, either selected passively (i.e., at random) or actively, does not enable the learning of the PF. However, when a larger number of training instances is considered, the UNC strategy yields better results than the less informed RND method. Finally, the UNC results are more stable than the RND results, as clearly showed by the error bars.

The middle graph refers to the D99 MOP. The previously observed behavior is confirmed. A superior asymptotic performance of the UNC method over the RND strategy is observed with large statistical confidence: the order of magnitude of the p-values is lower than or equal to $1e^{-20}$ when the number of query points is larger than 30 (the p-value $1e^{-8}$ is observed for 25 examples). The asymptotic difference among the performance of the RND and UNC strategies is even more pronounced that that observed for ZDT3. The more complicated shape of the D99 PF, which consists of six nonconvex disconnected curves compared to the five convex components of the ZDT3 PF, is a likely explanation for this behavior.

The experiments over TNK confirm the better performance of the UNC strategy, whose curve dominates the curve corresponding to the RND technique. As for D99, with 10 and 15 training examples, there is no significant difference between the two strategies. However, with more than 15 training examples, UNC significantly outperforms RND: the p-values lower than $1e^{-6}$ provide strong support for the superiority of the UNC sampling strategy.

X. Discussion

This paper introduced the ALP algorithm. While current state-of-the-art algorithms for MOPs are developed within the evolutionary framework, ALP adopts a different strategy. Pareto-optimal objective vectors were generated by combining the active learning paradigm with the solution of a scalarized optimization problem. The Pareto-optimal objective vectors recovered were used as training examples to learn a model of the PF. The model was iteratively refined until the information gain obtained by the new candidate training examples became negligible. The information gain was estimated by the maximum predictive uncertainty of the learned model in the regression domain.

ALP enables an analytical representation for the PF, which simplifies the decision making process. When the analytical PF representation is available, the DM is free to select and compare any Pareto-optimal solution, in particular within his/her preferred region. Once the favorite Pareto-optimal vector is selected, an associated Pareto-optimal solution is generated by solving a single instance of the scalarized optimization problem (3). Furthermore, depending on the optimization algorithm adopted for (3), ALP may not require any derivative information. The experimental results on the RM-MEDA benchmark

showed that ALP on average generates the analytical PF representation within a lower number of function evaluations than required by the state-of-the-art MMEA algorithm to provide a discrete PF approximation. On the ZDT3 and D99 MOPs with disconnected PF, better results were obtained when applying ALP rather than the well-known NSGA-II algorithm (e.g., ALP IGDF value was asymptotically one and five orders of magnitude better than NSGA-II performance on the above-mentioned problems). For the TNK MOP, NSGA-II needed more than 12 000 function evaluations to outperform ALP. With a lower number of function evaluations, ALP performance was $2-5$ times better. ALP cannot learn an accurate model of the KUR PF, due to the high noise affecting the training set. However, ALP can detect the poor quality of the learned model by simply observing the predictive variance, in this case exceptionally high, also in regions densely populated by training examples.

We plan to extend this paper along different directions. In the case of MOPs with more than two objectives, the regression domain in the subspace of the independent objectives $\mathbf{z}_I$ cannot be specified by a couple of (user-supplied) vectors only. We therefore plan to introduce a preliminary phase to learn the regression domain. The investigation of strategies to reduce the computational effort, in terms of MOP objectives evaluation, required to solve (3) is another interesting direction for future research. Approaches tackling expensive optimization by the generation of surrogate models are suitable for this purpose. Consider, for example, GPs designed for global optimization of unknown functions [25]. By equipping ALP with a mechanism for learning the DM preferences, the search may focus on the most relevant areas of the PF, guided by the user feedback [26], [27]. The incorporation of the DM preferences reduces the wastage of computational resources occurring when modeling PF regions that are not of any interest for the user. Finally, possible extensions of ALP to recover the PF under changing conditions (dynamic multiobjective optimization [28], [29]) will be explored.

REFERENCES

[1] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 12. Boston, MA, USA: Kluwer Academic Publishers, 1999,

[2] C. C. Coello, G. Lamont, and D. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed., New York, NY, USA: Springer-Verlag, 2007.

[3] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.

[4] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski, "Information retrieval perspective to nonlinear dimensionality reduction for data visualization," *J. Mach. Learn. Res.*, vol. 11, pp. 451–490, Feb. 2010.

[5] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[6] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 1994.

[7] D. J. Lizotte, M. H. Bowling, and S. A. Murphy, "Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis," in *Proc. 27th ICML*, Jun. 2010, pp. 695–702.

[8] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Mach. Learn.*, vol. 84, nos. 1–2, pp. 51–80, Jul. 2011.

[9] P. Campigotto, A. Passerini, and R. Battiti, "Active learning of Pareto fronts," Dept. Inf. Eng. Comput. Sci., Univ. Trento, Italy, Tech. Rep. DISI-13-001, 2012.

[10] A. Lovison, "Singular continuation: Generating piecewise linear approximations to pareto sets via global analysis," *SIAM J. Optim.*, vol. 21, no. 2, pp. 463–490, 2011.

[11] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.

[12] E. Snelson, "Flexible and efficient Gaussian process models for machine learning," Ph.D. dissertation, Gatsby Comput. Neurosci. Unit, Univ. College London, London, U.K., 2007.

[13] B. Settles, "Active learning literature survey," Dept. Comput. Sci. Tech., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1648, 2009.

[14] A. Zhou, Q. Zhang, and Y. Jin, "Approximating the set of pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1167–1189, Oct. 2009.

[15] K. Deb, "Multi-objective evolutionary optimization: Past, present and future," in *Evolutionary Design and Manufacture*, I. C. Parmee, Ed. New York, NY, USA: Springer-Verlag, 2000, pp. 225–236.

[16] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000.

[17] K. Deb, A. Sinha, and S. Kukkonen, "Multi-objective test problems, linkages, and evolutionary methodologies," in *Proc. 8th Conf. Genet. Evol. Comput.*, 2006, pp. 1141–1148.

[18] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. New York, NY, USA: Wiley, 1987.

[19] M. Powell, "Variable metric methods for constrained optimization," in *Mathematical Programming: The State of the Art*, A. Bachem, M. Grotschel, and B. Korte, Eds. New York, NY, USA: Springer-Verlag, 1983, pp. 288–311.

[20] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Math. Program.*, vol. 107, no. 3, pp. 391–408, 2006.

[21] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, no. 3, pp. 205–230, 1999.

[22] M. Tanaka, "GA-based decision support system for multi-criteria optimization," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, vol. 2. Oct. 1995, pp. 1556–1561.

[23] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Proc. 1st Workshop Parallel Problem Solving Nature*, vol. 496. 1991, pp. 193–197.

[24] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[25] M. A. Osborne, R. Garnett, and S. J. Roberts, "Gaussian processes for global optimization," in *Proc. 3rd Int. Conf. Learn. Intell. Optim.*, 2009, pp. 1–15.

[26] R. Battiti and A. Passerini, "Brain-computer evolutionary multi-objective optimization (BC-EMO): A genetic algorithm adapting to the decision maker," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 671–687, Feb. 2010.

[27] P. Campigotto, A. Passerini, and R. Battiti, "Handling concept drift in preference learning for interactive decision making," in *Proc. Int. Workshop Handling Concept Drift Adapt. Inf. Syst.*, 2010, pp. 1–12.

[28] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: Test cases, approximations, and applications," *IEEE Trans. Evol. Comput.*, vol. 8, no. 5, pp. 425–442, Oct. 2004.

[29] E. Tantar, A.-A. Tantar, and P. Bouvry, "On dynamic multi-objective optimization, classification and performance measures," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2011, pp. 2759–2766.

Paolo Campigotto received the M.S. degree in computer science and the Ph.D. degree in information and communication technologies from the University of Trento, Trento, Italy, in 2007 and 2010, respectively.

He is currently a Post-Doctoral Fellow with the Department of Information Engineering and Computer Science, University of Trento. His current work focuses on developing techniques for optimization under uncertainty, where the problem to be optimized cannot be completely defined before the solution process. His current research interests include optimization, machine learning and interactive decision making.

Andrea Passerini received the M.S. and Ph.D. degrees in computer science from the University of Florence, Florence, Italy, in 2000 and 2004, respectively.

He is currently an Assistant Professor with the Department of Information Engineering and Computer Science, University of Trento, Trento, Italy. He has developed techniques aimed at combining statistical and symbolic approaches to learning, via the integration of inductive logic programming and kernel machines. He is pursuing a deeper integration of machine learning approaches and complex optimization techniques. He has co-authored more than 60 scientific publications. His current research interests include machine learning and computational biology.

Roberto Battiti (F'09) received the B.S. and M.S. degrees in physics from the University of Trento, Trento, Italy, in 1985, and the Ph.D. degree in computation and neural systems from the California Institute of Technology, Pasadena, CA, USA, in 1990.

He started his research in the area of parallel computing and neural networks. He is currently a Professor of computer science with the Department of Information Engineering and Computer Science, University of Trento, where he is the Director of the Machine Learning and Intelligent Optimization Laboratory. He has made contributions to machine learning techniques and for intelligent optimization and neural networks. His current research interests include heuristic algorithms for optimization problems, in particular reactive search optimization algorithms for discrete optimization problems.