<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *2016 IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, Canada, July 24-29, 2016*.

Citation for the original published paper:

Bandaru, S., Andersson, M., Ng, A H. (2016)

Metamodel-based prediction of performance metrics for bilevel parameter tuning in MOEAs.

In: *2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1909-1916). New York:

IEEE

<https://doi.org/10.1109/CEC.2016.7744021>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-13285>

<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *2016 IEEE Congress on Evolutionary Computation, CEC 2016, 24 – 29 July 2016, Vancouver, Canada*.

Citation for the original published paper:

Bandaru, S., Andersson, M., Ng, A H. (2016)

Metamodel-based prediction of performance metrics for bilevel parameter tuning in MOEAs.

In: *2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1909-1916). IEEE Press

<https://doi.org/10.1109/CEC.2016.7744021>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-13285>

Metamodel-Based Prediction of Performance Metrics for Bilevel Parameter Tuning in MOEAs

Sunith Bandaru, Martin Andersson and Amos H.C. Ng
School of Engineering Science, University of Skövde, Skövde, Sweden
Email: firstname.lastname@his.se

Abstract—We consider a bilevel parameter tuning problem where the goal is to maximize the performance of a given multi-objective evolutionary optimizer on a given problem. The search for optimal algorithmic parameters requires the assessment of several sets of parameters, through multiple optimization runs, in order to mitigate the effect of noise that is inherent to evolutionary algorithms. This task is computationally expensive and therefore, in this paper, we propose to use sampling and metamodeling to approximate the performance of the optimizer as a function of its parameters. While such an approach is not unheard of, the choice of the metamodel to be used still remains unclear. The aim of this paper is to empirically compare 11 different metamodeling techniques with respect to their accuracy and training times in predicting two popular multi-objective performance metrics, namely, the hypervolume and the inverted generational distance. For the experiments in this pilot study, NSGA-II is used as the multi-objective optimizer for solving ZDT problems, 1 through 4.

I. INTRODUCTION

Numerical optimization techniques often use one or more algorithmic parameters to control various aspects of search. It is generally known and empirically established in several studies [1] that the choice of these parameters can drastically affect the search process. This is especially true for evolutionary algorithms, which involve more user-defined parameters than classical methods. Moreover, due to the inherent stochasticity of evolutionary methods, it is also difficult to assess the effect of various parameters on the performance of the algorithm. Therefore, the choice of parameters and the process of setting them have a greater significance in the field of evolutionary computation than in that of classical optimization.

According to Eiben and Smit [2], the parameter setting problem can be approached in two different ways, (i) parameter tuning, where the algorithmic parameters are optimized prior to but not changed during runtime, and (ii) parameter control, where the algorithmic parameters are adapted continuously during runtime to optimize performance. Both approaches have received considerable attention in the literature. In this paper, we deal with parameter tuning of multi-objective evolutionary algorithms (MOEAs). For completeness, here we define a standard multi-objective optimization problem as follows:

$$\begin{aligned} &\text{Minimize} && \mathbf{F}(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})\} \\ &\text{Subject to} && \mathbf{x} \in X \end{aligned} \quad (1)$$

where $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ are m (> 2) conflicting objectives that have to be simultaneously minimized and the variable vector

$\mathbf{x} = [x_1, x_2, \dots, x_n]$ belongs to the non-empty feasible region $X \subset \mathbb{R}^n$. The feasible region is formed by the constraints of the problem. A variable vector $\mathbf{x}_1$ is said to *dominate* $\mathbf{x}_2$ and is denoted as $\mathbf{x}_1 \prec \mathbf{x}_2$ if and only if the following conditions are satisfied:

- 1) $f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \forall i \in \{1, 2, \dots, m\}$,
- 2) and $\exists j \in \{1, 2, \dots, m\}$ such that $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2)$.

If only the first of these conditions is satisfied, then $\mathbf{x}_1$ is said to *weakly dominate* $\mathbf{x}_2$ and is denoted as $\mathbf{x}_1 \preceq \mathbf{x}_2$. If neither $\mathbf{x}_1 \preceq \mathbf{x}_2$ nor $\mathbf{x}_2 \preceq \mathbf{x}_1$, then $\mathbf{x}_1$ and $\mathbf{x}_2$ are said to be non-dominated with respect to each other and denoted as $\mathbf{x}_1 \parallel \mathbf{x}_2$. A vector $\mathbf{x}^* \in X$ is said to be Pareto-optimal, if there does not exist any $\mathbf{x} \in X$ such that $\mathbf{x} \prec \mathbf{x}^*$. The set of all such $\mathbf{x}^*$ (which are non-dominated with respect to each other) is referred to as the Pareto-optimal set. The projection of the Pareto-optimal set in the objective space, $\mathbf{F}(\mathbf{x}^*) \forall \mathbf{x}^*$, is called the Pareto-optimal front.

Evolutionary algorithms (EAs) have slightly different goals in single-objective and multi-objective optimization problems. In the former, there exists a total order between the candidate solutions, specified by the values of the objective function. Thus, as long as solutions with better objective values can be found, convergence to the optimum solution is the only goal and can be achieved by simply selecting better solutions. However, in case of multiple conflicting objectives, only a partial order exists among the candidate solutions. In order to best represent the true Pareto-optimal front, an MOEA must promote diversity of solutions in addition to convergence. Most MOEAs achieve this by employing a secondary selection criterion that provides total order between non-dominated solutions. Due to these differences, the best parameters of a single-objective EA may not work well for a multi-objective EA where the right balance between convergence and diversity is important.

A. Performance Metrics for MOEAs

The primary purpose of performance metrics is to compare the quality of solutions obtained from different optimization algorithms. In single-objective EAs, the most suitable measure is undoubtedly the objective function. On the other hand, performance metrics for MOEAs must assess both convergence and diversity of the non-dominated solutions. Many quality indicators have been proposed in the literature [3]. Some of them, like generational distance and ϵ -indicator measure the convergence, while others, like generalized spread, measure

the diversity [4]. However, indicators that can capture both convergence and diversity as a single metric are generally preferable. Two such indicators are inverted generational distance (I_{IGD}) [5] and hypervolume (I_H) [6].

- 1) The inverted generational distance of a non-dominated solution set S obtained from an MOEA is defined as,

$$I_{IGD}(S, P) = \frac{1}{|P|} \sum_{\mathbf{x}^* \in P} \min_{\mathbf{x} \in S} \|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}^*)\|, \quad (2)$$

where, P is a reference set of Pareto-optimal solutions that represent the desired density and distribution of points on the Pareto front, $\|\cdot\|$ denotes the Euclidean distance, and $|\cdot|$ denotes the size of a set. Solution sets with lower I_{IGD} values are preferred since they closely resemble the reference set P . Thus, the best possible value for I_{IGD} is 0.

- 2) The hypervolume of a non-dominated solution set S obtained from an MOEA is defined as,

$$I_H(S, \mathbf{r}) = \lambda \left(\bigcup_{\mathbf{x} \in S} H(\mathbf{F}(\mathbf{x}), \mathbf{r}) \right), \quad (3)$$

where, $\mathbf{r}$ is a reference point that is dominated by all solutions in S , $\lambda(\cdot)$ is the Lebesgue measure, and $H(\mathbf{a}, \mathbf{b})$ represents the hypercube with body-diagonal $\mathbf{ab}$. Solution sets with higher I_H values are better because they dominate a larger region of the objective space. Hypervolume is the only known unary quality indicator that is Pareto-complaint [3], i.e., for any two solution sets S_1 and S_2 ($\neq S_1$), $S_1 \preceq S_2 \Rightarrow I_H(S_1) > I_H(S_2)$.

There are a few fundamental differences between I_{IGD} and I_H that are relevant to this study. Firstly, from the definitions above, it is clear that the calculation of I_{IGD} requires a set P of uniformly distributed true Pareto-optimal points. Alternatively, I_H only requires the specification of a reference point $\mathbf{r}$ in the objective space, making it a suitable metric in problems with unknown Pareto fronts. Secondly, in terms of computational complexity, the calculation of I_{IGD} requires $\mathcal{O}(m|S| \cdot |P|)$ operations, while that of hypervolume, being $\mathcal{O}(|S|^{m-1})$, is much more expensive [4]. Finally, it is to be noted that unlike hypervolume, I_{IGD} is not Pareto-complaint [7], i.e., $I_{IGD}(S_1) < I_{IGD}(S_2) \nRightarrow S_1 \preceq S_2$.

Despite these differences, both I_H and I_{IGD} are equally popular and are among the most commonly used MOEA performance metrics. In this paper, we use bilevel parameter tuning to individually optimize each of these measures, as explained in the following section.

II. THE BILEVEL PARAMETER TUNING PROBLEM

The parameter tuning problem for EAs has received considerable attention in the literature, owing to studies such as [1], [8], [9] that showed significant benefits for moderate tuning efforts. A review of several parameter tuning methods developed over the years can be found in [2], wherein they are also classified into four categories: (i) sampling methods,

which use design of experiments to pick good parameter sets either as starting points for other methods or for further refinement, (ii) model-based methods, which use surrogates or metamodels to approximate the performance landscape, (iii) screening methods, which use *racing* to identify the best parameter set from a number of candidate sets, and (iv) meta-EAs, which use EAs to find the optimal parameter set(s) that maximize the performance of the algorithm being tuned.

The meta-EA approach to parameter tuning resembles a bilevel optimization formulation. A bilevel optimization problem involves a lower level optimization problem nested within the upper level optimization problem, such that the feasible solutions of the upper level problem are the optimal solutions obtained from the lower level problem. The mathematical formulation for the single-objective bilevel parameter tuning problem was provided in [10]. We extend it to include a multi-objective problem at the lower level and generalize it for any given performance function $\mathcal{P}$ (I_H and I_{IGD} in this paper).

$$\begin{aligned} &\text{Maximize} && E[\mathcal{P}|\mathbf{p}] \\ &\text{Subject to} && \mathbf{p}^{(L)} \leq \mathbf{p} \leq \mathbf{p}^{(U)}, \\ &&& \mathcal{P}(\mathbf{p}) = \begin{cases} I_H(S, \mathbf{r}), & \text{if } I_H \text{ metric is used} \\ -I_{IGD}(S, P), & \text{if } I_{IGD} \text{ metric is used} \end{cases}, \\ &\text{where} && S \text{ is the non-dominated set obtained by solving,} \\ &&& \text{Minimize } \mathbf{F}(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})\} \\ &&& \text{Subject to } \mathbf{x} \in X \\ &&& \text{using an MOEA with parameter set } \mathbf{p}. \end{aligned} \quad (4)$$

Note that $\mathcal{P}$ is primarily a function of the upper level variables $\mathbf{p}$, which denotes the parameter set for the MOEA at the lower level. $\mathcal{P}$ is also an indirect function of the lower level variables $\mathbf{x}$, through the non-dominated set S obtained by solving the lower level problem using the MOEA under consideration with a function evaluation budget of FE_l .

In general, meta-EAs outperform other tuning methods at finding the best parameter sets [8]. However, due to the stochastic nature of MOEAs, a single value of $\mathcal{P}(\mathbf{p})$ is not a reliable indicator of the goodness of the parameter set $\mathbf{p}$. Therefore, instead of using a single run, the upper level problem maximizes the expected value of $\mathcal{P}$ over R_l uniquely seeded replications of the lower level optimization. If the upper level optimization has a budget of FE_u , the major contributors to the overall complexity are as follows:

- 1) Overall evaluations of $\mathbf{F}(\mathbf{x})$, $FE = FE_u \times R_l \times FE_l$
- 2) Overall I_H complexity, $\mathcal{O}(FE_u \times R_l \times |S|^{m-1})$
- 3) Overall I_{IGD} complexity, $\mathcal{O}(FE_u \times R_l \times m|S||P|)$

In this paper, we propose to reduce the computational effort by a factor of $1/(R_l \times FE_l)$ by combining the Latin-hypercube sampling method with a model-based method to approximately predict the performance $\mathcal{P}$ of the MOEA at the lower level in Eq. (4). To identify the best model-based method to be used, we evaluate 11 different metamodels with respect to their accuracy and training time. Next, the chosen metamodel replaces the lower level problem and the best parameter sets are obtained by optimizing it. The following section briefly describes the metamodels used in this study.

Section IV discusses the complete experimental methodology. Results are presented and discussed in Section V.

III. SUMMARY OF METAMODELS

A number of meta-modeling methods can be found in literature [11]. We briefly describe the 11 metamodeling models used in the current study, along with their parameters. For all metamodels, the recommended parameter settings were used without any additional tuning.

A. Multivariate Adaptive Regression Splines (MARS)

MARS is a non-parametric regression technique proposed by [12] that can automatically model non-linearities and multi-variable interactions in the training data. The method is non-parametric in the sense that it does not assume a pre-defined form for the fitting function. Instead, the model is derived from the data along with the estimates for coefficients.

We use a Matlab implementation of MARS known as ARESLab [13], short for Adaptive Regression Splines toolbox for Matlab/Octave. The most important parameters for ARESLab are as follows:

- 1) *maxFuncs*: Maximum number of basis functions to be included in the model in the forward phase. We use *maxFuncs* = 21 in this paper.
- 2) *c*: GCV cost per basis function. It acts as a smoothing parameter. Larger values will lead to simpler models. We use *c* = 3
- 3) *cubic*: When a model with continuous first derivatives is desired, the piecewise linear basis functions retained after the backward phase are replaced with piecewise cubic splines. The parameter can either be `true` or `false`. We set *cubic* = `true`.
- 4) *selfInteractions*: Maximum degree of self interactions for any input variable. We set it to 1.
- 5) *maxInteractions*: Maximum degree of interactions between input variables. We set it to 2.
- 6) *threshold*: Another stopping criterion for the forward phase. Larger values of the threshold generate simpler models. We use *threshold* = 10^{-4} in this paper.

B. Kriging or Gaussian Process Regression (DACE)

Kriging was developed by French mathematician Georges Matheron and named after mining engineer Danie G. Krige who provided the empirical work in the form of geostatistics, a set of methods that enable the use of limited available data to estimate values at unsampled locations. Kriging models can be thought of as a combination of a global deterministic model and a local stochastic model. The former approximates the target function $f(\mathbf{x})$, similar to a regression method, while the latter allows the kriging model to interpolate the n_s training samples by introducing local deviations.

The kriging methodology has been implemented in Matlab as the DACE (Design and Analysis of Computer Experiments) toolbox [14]. The parameters to be set by the user are:

- 1) *regr*: Three regression models have been implemented, representing zero, first and second order polynomial

choices for $\mathcal{F}(\beta, \mathbf{x})$. We use the second order polynomial.

- 2) *corr*: Correlation model. Six correlation models have been implemented. The popular Gaussian correlation model is used in this paper. $R_{corrGauss}(\theta, \mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \prod_{k=1}^n e^{-\theta_k (x_k^{(i)} - x_k^{(j)})^2}$.
- 3) θ_0 : Initial value for θ . We use $\theta_0 = [2 \ 2 \ \dots \ 2]_n$.
- 4) θ_L, θ_U : Lower and upper bounds for θ . We use $[0.1 \ 0.1 \ \dots \ 0.1]_n$ and $[20 \ 20 \ \dots \ 20]_n$ respectively.

C. k Nearest Neighbors Regression (kNN)

The k nearest neighbors regression method is one of simplest meta-models to implement. Since there is no model training involved, it is usually among the fastest supervised algorithms. Let $\mathcal{N}_k(\mathbf{x}^{(u)})$ be the set of k nearest neighbors of an unsampled point $\mathbf{x}^{(u)}$. The predicted response is usually the mean of the response values at the points in $\mathcal{N}_k(\mathbf{x}^{(u)})$. In this paper an inverse distance weighted response is used with $k = 10$. It is given by,

$$\hat{f}(\mathbf{x}^{(u)}) = \frac{\sum_{i \in \mathcal{N}_k(\mathbf{x}^{(u)})} w_i f(\mathbf{x}^{(i)})}{\sum_{i \in \mathcal{N}_k(\mathbf{x}^{(u)})} w_i}, w_i = \frac{1}{\|\mathbf{x}^{(u)} - \mathbf{x}^{(i)}\|}. \quad (5)$$

D. Artificial Neural Networks (NN)

A feedforward neural network consists of input, hidden and the output layers with nodes. The number of nodes in the input and the output layers correspond to the number of variables and responses, while the number of nodes in the hidden layers are up to the user to choose. Each node in a layer is connected to all nodes of the next layer with a certain weight. All nodes (except the input nodes) process their input h (a weighted sum of outputs from the previous layer) using an *activation function* to generate an output v . A neural network is trained by first passing all training instances through the network and calculating the mean squared error. This error is *backpropagated* through the network to update the weights of the node connections. We use Matlab's `fitnet` function with the sigmoidal activation function, $v = (1 + e^{-h})^{-1}$, and Levenberg-Marquardt optimization for obtaining the weights. A single hidden layer with 10 nodes is used. A validation set of 20% of training samples is used to terminate the training for generalization.

E. Regression Tree (RT)

We use Matlab's `classregtree` function for growing a regression tree. We set *NVarToSample* = `all`. At any node, the splitting stops if, (i) the number of observations in it is less than *MinParent*, or (ii) factor of reduction in mean squared error is below a threshold *QEtoler*, or (iii) the number of observations in either of the child nodes is less than *MinLeaf*. These three parameters are set to 10, 10^{-6} and 1, respectively. Fully grown regression trees tend to overfit the training data and exhibit poor predictive performance (low bias high variance). Hence, they are often pruned by either penalizing the number of nodes in the cost function or limiting the number of levels. We set *prune* to `true`.

F. Support Vector Regression (SVR)

Support vector machine (SVM) based binary classification aims to find the decision boundary that maximizes the margin between two classes in the feature space. Training samples that lie on the margin boundaries are called support vectors, and the classification model is governed only by these training samples. A similar strategy can be employed in a regression model where only training samples with deviations above a threshold ϵ act as the support vectors. This version of SVM based regression is called ϵ -Support Vector Regression. We use the popular LibSVM [15] implementation of ϵ -SVR. The required algorithmic parameters are ϵ , penalty C and the kernel function. In this paper, we set them to 0.1, 1 and RBF kernel with $\sigma = 1/n$ respectively. SVR performs best when all variables are scaled in $[0, 1]$.

G. Radial Basis Function Network (RBN)

Radial basis function networks are a variation of feedforward neural networks which use a single hidden layer with the number of neurons typically equal to the number of training samples. The main difference however, is the choice of the activation function. Radial basis function networks use Gaussian activation functions centered at points $\mathbf{c}_i$.

$$\phi_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{2\sigma^2}\right), \text{ with } \sigma = \frac{\sum_{i,j=1}^{n_s} \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|}{n_s^2}.$$

When $\mathbf{c}_i$ correspond to the n_s training samples, the problem of finding optimal weights reduces to the inversion of a non-singular matrix. We use Matlab's `newrbe` function in this paper.

H. Polynomial Regression (PR)

Polynomial regression is probably the oldest, yet one of the most common meta-modeling methods. They fall under the category of polynomial regression, but mostly use a quadratic model as shown below:

$$\hat{f}(\mathbf{x}) = \beta_0 + \sum_{i=1}^n \beta_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^n \beta_{ij} x_i x_j + \sum_{i=1}^n \beta_{ii} x_i^2. \quad (6)$$

Other common variants of polynomial regression are the linear model, the interactions model and the pure quadratic model. In all cases, the coefficients are estimated using the ordinary least squares method. First, a *model matrix* or *design matrix* $\mathbf{X}$ of size $n_s \times p$ is formed using the regression terms of the model. For a quadratic model $p = (n+1)(n+2)/2$. The coefficients are estimated using the ordinary least squares method. Eq.(6) can now be written as $\hat{f}(\mathbf{x}) = \mathbf{X}\beta$. According to Gauss-Markov theorem, the best linear unbiased estimator (BLUE) of β obtained by minimizing mean squared error is given by

$$\hat{\beta} = [(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T] \mathbf{Y} = \mathbf{X}^+ \mathbf{Y}, \quad (7)$$

where $\mathbf{Y}$ is the vector of true responses $[f(\mathbf{x}^{(1)}) \ f(\mathbf{x}^{(2)}) \ \dots \ f(\mathbf{x}^{(n_s)})]^T$ and $\mathbf{X}^+$ is called the Moore-Penrose pseudoinverse.

I. Regularized Regression or Elastic Nets (EN)

Elastic nets [16] combine ridge and LASSO (Least Absolute Shrinkage and Selection Operator) regression through a parameter $\alpha \in [0, 1]$. Instead of minimizing the mean squared error, the elastic net regularized regression minimizes

$$\|\mathbf{Y} - \mathbf{X}\beta\|^2 + \lambda \left[\frac{(1-\alpha)}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right], \quad (8)$$

where $\|\beta\|_2^2 = \sum_{j=1}^p \beta_j^2$ and $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$. Here, $\mathbf{X}$ is the model matrix discussed above. With $\lambda = 0$, the above reduces to ordinary least squares estimation. $\alpha = 0$ corresponds to ridge regression and $\alpha = 1$ gives LASSO.

We use the popular GLMNET Matlab toolbox [17] in this paper. The toolbox uses 'warm starts' to calculate the regularization path for a sequence of λ values in an extremely efficient way so that the user can choose one of them. The most common choices are λ_{min} , which corresponds to the minimum cross-validation error model, and λ_{1SE} which is the largest value of λ whose cross-validation error is within one standard error of the minimum. The number of folds (*nfolds*) for cross-validation and the number of lambda values to generate *nlambdas* are user-defined parameters. They are set to 10 and 100 respectively. A quadratic model for $\hat{f}$ is used throughout this paper.

J. Bagged Tree Ensemble and Random Forests (RF)

The bagged tree ensemble uses a user-defined number of *unpruned* regression trees to promote de-correlation among them. De-correlation is necessary for bagging to be effective. In general, bagging improves prediction accuracy at the cost of interpretability.

Random forests provide an improvement over bagged trees by additionally performing *feature bagging*. A random sample of $NVarToSample < n$ variables are considered at each split instead of considering all n variables. Typically, $NVarToSample = \lfloor \sqrt{n} \rfloor$ for classification and $NVarToSample = \lfloor n/3 \rfloor$ for regression. Like the bagged tree ensemble, random forests also use a user-defined number of *unpruned* regression trees, *ntrees*. Feature bagging further de-correlates the trees. We use Matlab's `TreeBagger` function in this paper with *ntrees* = 100, *NVarToSample* = $\lfloor n/3 \rfloor$, *MinLeaf* = 5 and *prune* = false.

K. Boosted Tree Ensemble (BOOST)

While bagging uses multiple independent complex low-bias models and reduces variance, boosting uses multiple nested weak high-variance models and iteratively reduces bias. Each weak model regresses the residuals (difference between the actual function values and the predicted aggregate responses) from previous learners. The process is known as least squares boosting. Typically, the weak models are shallow regression trees. We use Matlab's `fitensemble` function with `LSBoost` option and *ntrees* = 100. The learning rate η is set to the default value of 1.

IV. EXPERIMENTAL METHODOLOGY

In order to focus the scope of this pilot study, we restrict ourselves to NSGA-II [18] as the algorithm to be tuned. The parameters considered for tuning and their bounds are as follows:

- 1) Population size, $pop \in [2, 100]$,
- 2) Child population as a fraction of pop , $C_{pop} \in [0, 1]$,
- 3) SBX probability, $p_c \in [0, 1]$,
- 4) SBX distribution index, $\eta_c \in [0, 100]$,
- 5) Polynomial mutation probability, $p_m \in [0, 1]$, and
- 6) Polynomial mutation distribution index, $\eta_m \in [0, 100]$

The tuning is specific to each of the first four ZDT problems [19] with the default number of variables, i.e. $n = 30$ for ZDT1-3 and $n = 10$ for ZDT4. For any upper-level parameter set $\mathbf{p}$, the function evaluation budget for NSGA-II at the lower-level is set to $FE_l = 5,000$ for all four test problems. At the end of FE_l evaluations, the non-dominated set S is obtained and both $I_{IGD}(S, P)$ and $I_H(S, \mathbf{r})$ are calculated as follows.

- 1) For $I_{IGD}(S, P)$, a reference set P containing $|P| = 101$ Pareto-optimal points is created by uniformly dividing the first objective and calculating the second objective using the known shape of the Pareto front for each problem.
- 2) For $I_H(S, \mathbf{r})$, the reference point $\mathbf{r} = [11, 11]^T$ is used for ZDT1-3 and $\mathbf{r} = [11, 400]^T$ for ZDT4 in order to avoid $I_H = 0$ for poorly performing parameter sets.

As mentioned before, two sets of experiments are conducted in this study. The goal of the first set of experiments is to compare the 11 metamodells described in Section III and possibly identify the best metamodel(s) in terms of accuracy and training time. The pseudocode for the experimental methodology adopted in this paper is shown in Algorithm 1. In order to simultaneously study the effect of number of samples, the Latin-hypercube sampling (LHS) method uses five different training sample sizes, $n_s = \{100, 200, 300, 400, 500\}$. The number of test samples is fixed at $n_t = 100$ for all metamodells (line 5) in all test instances (line 4). Note that a separate metamodel is to be trained for each performance metric $\mathcal{P}$. The trained metamodells are evaluated with respect to the Normalized Root Mean Squared Error, $NRMS_{\mathcal{P}}$, and the training time, $TT_{\mathcal{P}}$. $NRMS$ for a performance metric $\mathcal{P}$ is given by,

$$NRMS_{\mathcal{P}} = \frac{1}{\sqrt{n_t}} \sqrt{\frac{\sum_{i=1}^{n_t} (E[\mathcal{P}|\mathbf{p}_i] - \hat{\mathcal{P}}(\mathbf{p}_i))^2}{(E[\mathcal{P}|\mathbf{p}]_{max} - E[\mathcal{P}|\mathbf{p}]_{min})^2}} \quad (9)$$

where

$$E[\mathcal{P}|\mathbf{p}]_{max} = \max(\{E[\mathcal{P}|\mathbf{p}_i] : i = 1, \dots, n_s + n_t\})$$

$$E[\mathcal{P}|\mathbf{p}]_{min} = \min(\{E[\mathcal{P}|\mathbf{p}_i] : i = 1, \dots, n_s + n_t\})$$

In all experiments, there are three sources of statistical variation, (i) LHS sampling (line 12), (ii) MOEA stochasticity (line 15), and (iii) cross-validation (CV) of metamodells (line 20). Ten uniquely seeded LHS sampling instances are

Algorithm 1 Pseudocode for experimental methodology

```

1:  $n_t \leftarrow 100$ 
2:  $R_l \leftarrow 50$ 
3:  $FE_l \leftarrow 5000$ 
4: for each problem  $Z \in \{ZDT1, ZDT2, ZDT3, ZDT4\}$  do
5:   for each metamodel  $M \in \{1, 2, \dots, 11\}$  do
6:     for each  $n_s \in \{100, 200, 300, 400, 500\}$  do
7:       for each LHS instance  $l \in \{1, 2, \dots, 10\}$  do
8:          $ME_H(l) \leftarrow 0.0$ 
9:          $ME_{IGD}(l) \leftarrow 0.0$ 
10:         $MT_H(l) \leftarrow 0.0$ 
11:         $MT_{IGD}(l) \leftarrow 0.0$ 
12:         $L \leftarrow \text{Generate } (n_s + n_t) \text{ LHS samples}$ 
13:        for each sample  $\mathbf{p}_i \in L$  do
14:          Reset RNG
15:          Perform  $R_l$  replications of MOEA with  $\mathbf{p}_i$ 
16:          Calculate  $E[I_H|\mathbf{p}_i]$  from  $R_l$  replications
17:          Calculate  $E[I_{IGD}|\mathbf{p}_i]$  from  $R_l$  replications
18:        end for
19:        Randomize RNG
20:        for each CV replication  $j \in \{1, 2, \dots, 10\}$  do
21:          Randomly select  $n_s$  samples from  $L$ 
22:          Train  $M$  on  $n_s$  samples for predicting  $I_H$ 
23:          Test  $M$  on remaining  $n_t$  samples
24:          Calculate  $NRMS_H$  and  $TT_H$ 
25:          Train  $M$  on  $n_s$  samples for predicting  $I_{IGD}$ 
26:          Test  $M$  on remaining  $n_t$  samples
27:          Calculate  $NRMS_{IGD}$  and  $TT_{IGD}$ 
28:           $ME_H(l) \leftarrow ME_H(l) + NRMS_H/10$ 
29:           $ME_{IGD}(l) \leftarrow ME_{IGD}(l) + NRMS_{IGD}/10$ 
30:           $MT_H(l) \leftarrow MT_H(l) + TT_H/10$ 
31:           $MT_{IGD}(l) \leftarrow MT_{IGD}(l) + TT_{IGD}/10$ 
32:        end for
33:      end for
34:    end for
35:  end for
36: end for

```

used for each sample size as shown in line 7 of the pseudocode. For each parameter set $\mathbf{p}_i$ of a sampling instance, $R_l = 50$ uniquely seeded MOEA runs are performed at the lower level (line 15) to obtain $E[I_H|\mathbf{p}_i]$ and $E[I_{IGD}|\mathbf{p}_i]$ for training the metamodells. However, to remove any bias due to different initial populations, these seeds remain exactly the same for all samples in all sampling instances. This can be achieved by resetting the Random Number Generator (RNG) at line 14. When evaluating the metamodells, the sampling instance is randomly divided into training and test sets. Ten CV replications are used to reduce any bias induced by random chance. The $NRMS$ and TT values, averaged over the 10 CV replications, are recorded in ME (Mean Errors) and MT (Mean Times) respectively (lines 28-31).

In the second set of experiments, the lower-level MOEA is completely replaced by this pre-trained metamodel. The goal here is to obtain the best parameter sets by solely optimizing the approximation $\hat{\mathcal{P}}(\mathbf{p})$ of the MOEA's performance function.

V. RESULTS AND DISCUSSION

The first set of experiments described in Section IV, result in mean error (ME_H and ME_{IGD}) and mean training time (MT_H and MT_{IGD}) values on 10 LHS instances for each

TABLE I
PERCENTAGE MEDIAN ME_{IGD} VALUES FROM 10 LHS INSTANCES FOR 4 ZDT PROBLEMS WITH 5 SAMPLE SIZES ON ALL METAMODELS.

Z	n_s	MARS	DACE	kNN	NN	RT	SVR	RBN	PR	EN	RF	BOOST
ZDT1	100	8.689	9.268	14.702	11.577	12.666	15.808	55.628	9.653	8.986	11.009	12.457
	200	4.614	5.893	10.843	6.547	8.440	11.373	71.315	6.362	6.250	7.236	8.220
	300	5.415	6.348	13.125	7.453	9.606	13.787	69.475	7.566	7.624	8.680	10.088
	400	3.721	4.193	9.816	4.335	6.813	10.360	43.672	5.320	5.280	5.822	7.211
	500	7.603	8.987	18.925	7.354	12.283	20.186	117.888	10.353	10.280	11.146	13.408
ZDT2	100	9.308	9.357	16.726	12.506	15.475	19.251	65.621	9.705	9.874	13.175	15.199
	200	5.718	6.188	12.394	7.898	9.998	13.937	66.832	7.031	6.875	8.386	9.482
	300	6.911	6.788	15.657	8.015	11.726	17.531	67.962	8.584	8.442	9.992	12.394
	400	4.720	4.936	12.207	5.558	8.106	14.046	49.131	6.099	6.106	7.372	8.979
	500	6.248	5.876	15.176	6.065	9.935	17.613	78.213	7.684	7.668	8.862	11.067
ZDT3	100	7.627	9.148	13.842	10.974	11.859	14.017	61.181	9.455	9.292	10.124	11.431
	200	4.440	5.420	10.084	6.157	7.469	10.313	65.596	6.007	5.947	6.377	7.579
	300	5.070	6.089	12.036	6.558	8.644	12.226	71.294	6.831	6.848	7.290	8.939
	400	3.640	4.310	9.388	4.159	6.743	9.887	46.866	5.494	5.476	5.489	6.836
	500	7.687	8.662	17.818	7.618	11.705	18.364	121.698	10.459	10.435	10.251	13.300
ZDT4	100	3.429	5.998	12.342	6.416	8.298	14.080	20.805	7.678	7.082	9.548	12.267
	200	2.692	3.927	13.031	3.631	5.782	15.513	17.573	7.528	7.370	8.398	12.205
	300	2.259	3.134	12.636	2.952	4.837	14.843	13.198	6.992	6.868	7.544	11.762
	400	1.809	2.446	11.646	2.472	4.178	13.377	11.853	6.322	6.280	6.110	10.138
	500	1.810	2.342	11.475	2.174	3.582	13.579	12.624	6.378	6.326	5.593	10.169

Fig. 1. Correlation between I_H and I_{IGD} for the LHS samples on ZDT1.

Fig. 2. Box plots showing ME_{IGD} of all metamodels for ZDT4 with $n_s = 400$ samples.

Z, n_s, M combination. Tables I and II show the median of these 10 values for ME_{IGD} and ME_{IGD} respectively. Here, the values are expressed in percentage $NRMS$.

Note that for a given test instance, most metamodels have a similar error value for both I_H and I_{IGD} . This is because the two measures are consistent with each other [4], as shown in Figure 1 for ZDT1 by the strong correlation between them on the complete set of LHS samples generated for this study.

To give an indication of the training times (in seconds) involved, we show median MT_{IGD} values in Table III.

The median values shown in these tables can be misleading. For example, on ZDT4 with $n_s = 400$, the median ME_{IGD} of MARS appears to be much better than that of others. However, looking closely at the distributions of ME_{IGD} through the box plots shown in Figure 2, we observe that there is an overlap

of the box plot for MARS with that of other methods. This may indicate that, statistically, the difference between them may not be significant.

Non-parametric statistical test can reveal a more accurate picture. When more than two groups are involved, the Kruskal-Wallis rank-sum test can be used to test the null hypothesis that groups participating in an experiment come from the same population, i.e. there is no significant difference between them. The test is an extension of the Wilcoxon rank-sum test to more than two groups. Rejection of the null hypothesis would imply that at least two of the groups are statistically different. In order to identify such groups, a post-hoc multiple comparisons test can be performed. On applying the Kruskal-Wallis test we find that the medians shown in Table I are

TABLE II
PERCENTAGE MEDIAN ME_H VALUES FROM 10 LHS INSTANCES FOR 4 ZDT PROBLEMS WITH 5 SAMPLE SIZES ON ALL METAMODELS.

Z	n_s	MARS	DACE	kNN	NN	RT	SVR	RBN	PR	EN	RF	BOOST
ZDT1	100	8.202	9.238	15.174	11.345	13.328	16.747	73.355	9.844	9.514	11.467	13.138
	200	4.870	5.904	10.887	6.555	8.921	11.316	83.069	6.322	6.276	7.488	9.011
	300	5.519	6.427	13.357	7.227	9.790	13.782	64.645	7.373	7.324	8.214	10.347
	400	3.985	4.144	9.837	4.781	6.606	10.374	45.234	5.463	5.455	5.896	7.441
	500	7.347	8.008	17.876	7.515	11.801	18.670	101.904	9.932	9.936	10.318	12.950
ZDT2	100	8.906	9.372	16.741	13.049	15.623	19.315	63.353	9.938	9.653	12.871	14.847
	200	5.740	6.095	12.625	7.759	10.217	14.408	69.223	7.049	7.057	8.597	10.171
	300	7.184	6.818	15.602	8.375	11.796	17.888	68.956	8.291	8.323	9.885	12.278
	400	4.779	4.982	12.169	5.387	8.539	14.032	53.705	6.259	6.210	7.192	9.329
	500	6.187	6.235	15.165	6.367	10.343	17.285	54.952	8.148	8.131	9.009	11.108
ZDT3	100	8.186	9.610	15.219	12.119	12.655	16.006	61.396	9.857	9.613	11.174	12.223
	200	4.971	6.276	11.991	7.380	8.818	12.258	73.659	6.842	6.786	7.683	8.643
	300	5.647	6.473	13.616	7.287	9.561	14.036	72.276	7.505	7.513	8.319	10.259
	400	3.940	4.841	10.621	4.352	7.033	10.882	51.591	5.844	5.822	5.932	7.368
	500	7.731	8.095	17.637	7.783	11.548	18.301	103.941	9.673	9.681	9.679	12.640
ZDT4	100	2.969	5.831	11.768	6.338	8.457	13.874	21.891	7.562	7.023	9.142	12.467
	200	2.802	3.841	12.293	3.674	6.084	14.275	18.362	7.213	6.921	7.978	11.718
	300	2.048	3.064	12.715	2.942	4.867	14.577	13.540	6.990	6.866	7.656	11.648
	400	1.723	2.395	11.183	2.148	3.861	13.229	11.833	6.322	6.277	6.208	9.665
	500	1.782	2.263	11.513	2.026	3.716	13.333	14.044	6.353	6.245	5.613	10.125

TABLE III
MEDIAN MT_{IGD} VALUES FROM 10 LHS INSTANCES FOR 4 ZDT PROBLEMS WITH 5 SAMPLE SIZES ON ALL METAMODELS.

Z	n_s	MARS	DACE	kNN	NN	RT	SVR	RBN	PR	EN	RF	BOOST
ZDT1	100	0.334	0.120	0.003	0.261	0.012	0.003	0.042	0.050	0.057	0.297	0.426
	200	0.682	0.386	0.003	0.345	0.018	0.006	0.047	0.048	0.053	0.322	0.426
	300	0.933	0.950	0.003	0.354	0.024	0.011	0.059	0.050	0.055	0.352	0.432
	400	1.256	1.756	0.003	0.442	0.031	0.017	0.079	0.050	0.052	0.383	0.439
	500	1.628	2.737	0.004	0.548	0.036	0.025	0.106	0.051	0.055	0.408	0.443
ZDT2	100	0.332	0.115	0.003	0.265	0.011	0.003	0.040	0.048	0.055	0.294	0.424
	200	0.694	0.384	0.003	0.294	0.017	0.006	0.047	0.049	0.053	0.323	0.427
	300	0.958	0.916	0.003	0.390	0.024	0.011	0.059	0.049	0.054	0.349	0.431
	400	1.276	1.798	0.003	0.460	0.029	0.017	0.080	0.050	0.054	0.376	0.438
	500	1.638	2.878	0.004	0.542	0.037	0.025	0.105	0.050	0.056	0.367	0.442
ZDT3	100	0.394	0.163	0.004	0.324	0.017	0.004	0.054	0.067	0.074	0.374	0.602
	200	0.892	0.532	0.004	0.420	0.027	0.009	0.062	0.068	0.071	0.412	0.615
	300	1.221	1.255	0.004	0.438	0.034	0.014	0.076	0.066	0.071	0.440	0.621
	400	1.323	1.905	0.004	0.546	0.038	0.019	0.089	0.058	0.061	0.406	0.525
	500	1.545	2.822	0.004	0.552	0.040	0.025	0.109	0.053	0.057	0.388	0.482
ZDT4	100	0.353	0.117	0.003	0.314	0.011	0.003	0.042	0.051	0.053	0.284	0.452
	200	0.795	0.344	0.003	0.375	0.017	0.006	0.049	0.051	0.050	0.305	0.453
	300	1.152	0.830	0.004	0.419	0.024	0.011	0.061	0.052	0.049	0.332	0.461
	400	1.599	1.529	0.004	0.574	0.031	0.017	0.083	0.053	0.053	0.361	0.473
	500	1.865	2.455	0.004	0.640	0.039	0.025	0.109	0.055	0.054	0.398	0.483

significantly different in all 20 test instances. Next, we perform the post-hoc Nemenyi test [20] on the means of rank-sums at $\alpha = 0.05$ level of significance. A metamodel will be statistically indistinguishable from the best median metamodel if the post-hoc null hypothesis cannot be rejected. For example, in case of test instance shown in Figure 2, the Nemenyi test reveals that MARS is statistically indistinguishable from DACE, NN, RT and RF. In other words, with respect to ME_{IGD} , the metamodels {MARS, DACE, NN, RT, RF} are statistically better than the rest of the metamodels on ZDT4 with $n_s = 400$. They are therefore shown in **bold text** in Table I. The number of times a metamodel's ME_{IGD} appears in bold is a score of how well it performs over all test instance. The performance scores for all metamodels with respect to ME_{IGD} , ME_H , MT_{IGD} and MT_H are shown in Table IV. The table shows that while MARS, DACE and NN consistently

TABLE IV
PERFORMANCE SCORES (OUT OF 20) OF ALL METAMODELS WITH RESPECT TO ME_{IGD} , ME_H , MT_{IGD} AND MT_H .

M	Metamodel	ME_{IGD}	ME_H	MT_{IGD}	MT_H
1	MARS	20	20	0	0
2	DACE	20	20	0	0
3	kNN	0	0	20	20
4	NN	20	20	0	0
5	RT	8	4	20	20
6	SVR	0	0	20	20
7	RBN	0	0	9	9
8	PR	19	19	19	19
9	EN	19	20	14	16
10	RF	18	18	0	0
11	BOOST	0	0	0	0

provide better predictions, they are significantly slower than all other methods. On the other hand, kNN and SVR despite

TABLE V
PARAMETER VALUES OBTAINED BY OPTIMIZING THE PR AND EN
METAMODELS OF I_{IGD} BUILT USING 500 LHS SAMPLES.

Z	M	pop	p_m	η_m	p_c	η_c	C_{pop}
ZDT1	PR	2.00	0.52	44.77	0.60	0.00	1.00
	EN	2.00	0.55	48.86	0.62	0.00	1.00
ZDT2	PR	100.00	0.39	37.34	1.00	0.00	1.00
	EN	100.00	0.37	31.02	1.00	0.00	1.00
ZDT3	PR	2.00	0.33	13.77	0.54	0.00	1.00
	EN	2.00	0.35	18.55	0.56	0.00	1.00
ZDT4	PR	100.00	0.00	34.34	1.00	61.98	1.00
	EN	100.00	0.00	33.71	1.00	79.87	1.00

TABLE VI
PARAMETER VALUES OBTAINED BY OPTIMIZING THE PR AND EN
METAMODELS OF I_H BUILT USING 500 LHS SAMPLES.

Z	M	pop	p_m	η_m	p_c	η_c	C_{pop}
ZDT1	PR	2.00	0.54	48.59	0.59	0.00	1.00
	EN	2.00	0.57	51.62	0.61	0.00	1.00
ZDT2	PR	100.00	0.33	27.00	1.00	0.00	1.00
	EN	100.00	0.28	18.76	1.00	0.00	1.00
ZDT3	PR	100.00	0.71	100.00	1.00	0.00	1.00
	EN	100.00	0.75	100.00	1.00	0.00	1.00
ZDT4	PR	100.00	0.00	34.36	1.00	74.19	1.00
	EN	100.00	0.00	33.90	1.00	73.76	1.00

being fast have significantly higher error rates. RT is a slight improvement over these methods. PR and EN provide the best compromise between accuracy and speed. BOOST is found to be significantly worse than all other metamodels, followed by RBN.

For the second set of experiments, PR and EN are chosen as the static metamodels for replacing the lower level optimization problem. As a consequence, Eq. (4) is no longer a bilevel problem. Instead, the bilevel parameter tuning problem has reduced to a regular function optimization problem. For all problems, we use the datasets with 500 LHS samples generated earlier for building the metamodels. Optimizing the resultant approximation of the performance landscape, we obtain parameter values as shown in Table V for minimum I_{IGD} and as shown in Table VI for maximum I_H . Matlab's `fmincon()` function is used for optimizing the metamodels within the parameter bounds specified in Section IV.

The tables clearly show that the two metamodels and their optima largely agree with each other for all problems. Few observations regarding the parameter values can also be made. The optimum size of the child population is found to be equal to the population size. The SBX distribution index seems to have little effect on the performance. Mutation probabilities are higher than the recommended value of $1/n$, whereas crossover probabilities are close to recommended values.

VI. CONCLUSIONS

In this paper, we first formulated the parameter tuning problem in MOEAs as a bilevel optimization problem. In order to reduce the computation cost of tuning, we evaluated different metamodeling methods with respect to their accuracy and training time in predicting performance metrics such as the hypervolume and the inverted generational distance

as a function of the parameters of an MOEA, in our case NSGA-II. Statistical hypothesis testing shows that polynomial regression and elastic nets are the best performing metamodels. Next, these metamodels are used as approximations of the performance landscape and optimized to obtain the optimal parameter sets. In the future, we plan to use the approach presented in this paper to tune MOEAs for solving expensive simulation-based multi-objective optimization problems, where the benefits are expected to be much higher.

REFERENCES

- [1] V. Nannen, S. K. Smit, and A. E. Eiben, "Costs and benefits of tuning parameters of evolutionary algorithms," in *Parallel Problem Solving from Nature, PPSN X*. Springer, 2008, pp. 528–538.
- [2] A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, 2011.
- [3] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance Assessment of Multiobjective Optimizers: An Analysis and Review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.
- [4] S. Jiang, Y.-S. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2391–2404, 2014.
- [5] P. A. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 174–188, 2003.
- [6] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
- [7] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified distance calculation in generational distance and inverted generational distance," in *Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 110–125.
- [8] S. K. Smit and A. E. Eiben, "Comparing parameter tuning methods for evolutionary algorithms," in *IEEE Congress on Evolutionary Computation, CEC'09*. IEEE, 2009, pp. 399–406.
- [9] —, "Beating the 'world champion' evolutionary algorithm via revac tuning," in *IEEE Congress on Evolutionary Computation, CEC'10*. IEEE, 2010, pp. 1–8.
- [10] A. Sinha, P. Malo, P. Xu, and K. Deb, "A bilevel optimization approach to automated parameter tuning," in *Proceedings of the 2014 conference on Genetic and evolutionary computation*. ACM, 2014, pp. 847–854.
- [11] T. W. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen, "Meta-models for computer-based engineering design: Survey and recommendations," *Engineering with Computers*, vol. 17, no. 2, pp. 129–150, 2001.
- [12] J. H. Friedman, "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.
- [13] G. Jakobs, "ARESLab: Adaptive Regression Splines Toolbox for MATLAB," 2009.
- [14] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard, "Aspects of the Matlab Toolbox DACE," Technical University of Denmark, Lyngby, Denmark, Tech. Rep., 2002.
- [15] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.
- [16] H. Zou and T. Hastie, "Regularization and Variable Selection Via the Elastic-net," *Journal of the Royal Statistical Society*, vol. 67, no. 2, pp. 301–320, 2005.
- [17] J. H. Friedman, T. Hastie, and R. Tibshirani, "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.
- [18] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan, "A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [19] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–95, Jan. 2000.
- [20] P. Nemenyi, "Distribution-Free Multiple Comparisons," Ph.D. dissertation, Princeton University, 1963.