A Hybrid Surrogate-Based Approach for Evolutionary Multi-Objective Optimization

Alejandro Rosales-Pérez*, Carlos A. Coello Coello†, Jesus A. Gonzalez*, Carlos A. Reyes-Garcia*
and Hugo Jair Escalante*
* Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
Computer Science Department
Santa Ma. Tonantzintla, Puebla, Mexico
Email: {arosales, jagonzalez, kargaxxi, hugojair}@inaoep.mx
† CINVESTAV-IPN
Computer Science Department
Evolutionary Computation Group (EVOCINV)
Mexico City, Mexico
Email: ccoello@cs.cinvestav.mx

***Abstract*—Evolutionary algorithms have gained popularity as an alternative for dealing with multi-objective optimization problems. However, these algorithms require to perform a relatively high number of fitness function evaluations in order to generate a reasonably good approximation of the Pareto front. This can be a shortcoming when fitness evaluations are computationally expensive. In this paper, we propose an approach that combines an evolutionary algorithm with an ensemble of surrogate models based on support vector machines (SVM), which are used to approximate the fitness functions of a problem. The proposed approach performs a model selection process for determining the appropriate hyperparameters values for each SVM in the ensemble. The ensemble is constructed in an incremental fashion, such that the models are updated with the knowledge gained during the evolutionary process, but the information from previous evaluated regions is also preserved. A criterion based on surrogate fidelity is also proposed for determining when should the surrogates be updated. We evaluate the performance of our proposal using a benchmark of test problems widely used in the literature and we compare our results with respect to those obtained by the NSGA-II. Our proposed approach is able to significantly reduce the number of fitness function evaluations performed, while producing solutions which are close to the true Pareto front.**

I. Introduction

Many real world problems (such as aeronautical/aerodynamic design, circuit design, etc.) require optimizing several conflicting objectives simultaneously. These problems are known as *multi-objective optimization problems* (MOPs), and their solution implies to find a set of points that satisfy a trade-off among the objectives. This set of points is known as the *Pareto optimal set*.

Over the last 25 years, evolutionary algorithms (EAs) have become popular techniques for solving MOPs, mainly because they are able to generate several elements of the Pareto optimal set in a single run and because they are less susceptible than mathematical programming techniques to the shape and continuity of the Pareto front [1], [2]. However, most of them require performing several objective functions evaluations in order to obtain a good approximation of the Pareto front. This could be a shortcoming when using them in real-world problems, where the objective functions evaluations could be computationally expensive.

A promising approach for dealing with computationally expensive problems is the use of surrogate-assisted evolutionary computation methods. A surrogate model is a cheaper approximation model of the objective function. There has been an increasing interest for using surrogate models in optimization with the aim of reducing the number of objective functions evaluations while maintaining a good quality of the final solutions. There exists a wide diversity of techniques that allow constructing surrogate models, such as: support vector machines (SVMs), artificial neural networks (ANNs), radial basis functions (RBFs), polynomial regression (PR), etc. The fidelity of the surrogate model depends on the appropriate selection of the model and its hyperparameters.

In this work, we propose a surrogate-based approach for evolutionary multi-objective optimization (EMO). Our approach is based on the combination of a multi-objective evolutionary algorithm (MOEA) with an incremental ensemble of surrogate models, which is used to approximate the objective functions. Since surrogate models can be constructed by different techniques, in this study we adopted the use of SVMs, due to their high performance and scalability over different problems [3], [4]. In order to determine the appropriate hyperparameters (i.e., the kernel type and kernel parameters) for the surrogate, a grid search procedure is performed. We evaluated our proposal on benchmark problems commonly adopted in the specialized literature. The obtained results show a significant reduction of the number of evaluations, while producing reasonably good approximations.

The remainder of this paper is organized as follows. First, some basic concepts related to multi-objective optimization are presented in section II. In section III, we provide a brief description of support vector machines. The most relevant previous related work is described in section IV. In section V, we describe our proposed ap-

proach, and our experiments and results are presented in section VI. Finally, in section VII, we present the main conclusions as well as possible future research directions.

II. Multi-Objective Optimization

A multi-objective optimization problem (MOP) is defined as the problem of finding a set of solutions that satisfy a set of equality and inequality constraints, and optimize two (or more) (possibly conflicting) objectives simultaneously. Assuming minimization, a MOP is stated as follows:

$$\begin{array}{ll} \text{minimize} & \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_l(\mathbf{x})]^T \\ \text{subject to} & g_j(\mathbf{x}) \leq 0 \; j = 1, \ldots, p \\ & h_k(\mathbf{x}) = 0 \; k = 1, \ldots, q \end{array}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ is a vector of decision variables, $f_i(\mathbf{x})$, $i = 1, \ldots, l$, are the objective functions, $g_j(\mathbf{x})$, $j = 1, \ldots, p$ are the inequality constraints and $h_j(\mathbf{x})$, $k = 1, \ldots, q$, are the equality constrains.

Most modern MOEAs use the concept of **Pareto dominance** to determine if a solution is better than another, in their population. Formally, the Pareto dominance concept is defined as follows:

Definition 1: A solution $\mathbf{x^{(1)}}$ dominates a solution $\mathbf{x^{(2)}}$ ($\mathbf{x^{(1)}} \preceq \mathbf{x^{(2)}}$) if and only if $\mathbf{x^{(1)}}$ is better than $\mathbf{x^{(2)}}$ at least in one objective and it is not worse in the rest, i.e. $\forall i \in \{1, \ldots, l\}$, $f_i\left(\mathbf{x^{(1)}}\right) \leq f_i\left(\mathbf{x^{(2)}}\right) \wedge \exists i \in \{1, \ldots, l\}, f_i\left(\mathbf{x^{(1)}}\right) < f_i\left(\mathbf{x^{(2)}}\right)$.

The notion of optimum in MOP refers to obtaining the best possible trade-off among the objectives. In order to establish these trade-offs, the most accepted notion of optimum is the so-called *Pareto optimality*. Pareto optimality is formally defined as:

Definition 2: A solution $\mathbf{x}^* \in \Omega$, where Ω is the feasible region, is a Pareto Optimal if there does not exists another solution $\mathbf{x}' \in \Omega$ such that $\mathbf{x}' \preceq \mathbf{x}^*$.

This definition produces not one, but a set of trade-off solutions among the different objectives. The set of trade-off solutions (in decision variable space) is known as **Pareto optimal set**.

Definition 3: The Pareto optimal set (PS) is defined as:

$$PS = \{\mathbf{x} \in \Omega \mid \mathbf{x} \, \texttt{is a Pareto optimal solution}\}$$

The objective function values corresponding to the elements of the Pareto optimal set constitute the so-called **Pareto front**. Formally,

Definition 4: The Pareto front (PF) is defined as:

$$PF = \{\mathbf{F}(\mathbf{x}) \mid \mathbf{x} \in PS\}$$

III. Support Vector Machines

Support Vector Machines (SVMs) [5] are supervised learning algorithms based on statistical learning theory. SVMs are able to construct classification and regression models from a training data set. In a regression task, a SVM aims to learn a function, $f(\mathbf{x})$, that has at most an ε deviation from the targets for the training data.

Let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be a training data set, where $\mathbf{x}_i \in \mathbb{R}^n$ is an input vector and $y_i \in \mathbb{R}$ corresponds to the desired output. The support vector regression constructs a linear function:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{1}$$

where $f(\mathbf{x})$ is the estimated function, $\mathbf{w}$ is a coefficient vector, b is the bias. These parameters can be obtained by solving the following constrained optimization problem:

$$\begin{array}{ll} \min & \frac{1}{2} \parallel \mathbf{w} \parallel^2 \\ s.t. & \mathbf{w} \cdot \mathbf{x} + b - y_i \leq \varepsilon, \\ & y_i - \mathbf{w} \cdot \mathbf{x} - b \leq \varepsilon, \\ & i = 1, \ldots, m \end{array} \tag{2}$$

where $\varepsilon > 0$ is a constant that controls the tolerable error.

This equation assumes that the convex optimization problem is always *feasible*, but this may not be the case. Furthermore, sometimes it is desirable to allow for some errors. For that reason, one can introduce slack variables ξ_i, ξ_i^* in equation (2), re-expressing the optimization problem as follows:

$$\begin{array}{ll} \min & \frac{1}{2} \parallel \mathbf{w} \parallel^2 + \frac{C}{m} \sum_{i=1}^{m} (\xi_i + \xi_i^*) \\ s.t. & \mathbf{w} \cdot \mathbf{x} + b - y_i \leq \varepsilon + \xi_i, \\ & y_i - \mathbf{w} \cdot \mathbf{x} - b \leq \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, \; i = 1, \ldots, m \end{array} \tag{3}$$

where C is a parameter that controls the trade-off between the complexity of the model parameterization and the margin-based error.

Equation (3) can be solved using quadratic programming techniques. The interested reader is referred to [6] for more details about support vector regression.

IV. Previous Related Work

In recent years, the use of surrogate models in optimization has received great attention, as an alternative to approximate the objective function and to reduce the number of required (objective function) evaluations without degrading, in a significant manner, the quality of the obtained solutions. Surrogate models have been used both in single-objective optimization and multi-objective optimization. In multi-objective optimization, a number of strategies have been adopted. Arias-Montaño et al. [7] proposed a surrogate-based approach for multi-objective optimization based on differential evolution. Their proposal is based on multiple surrogate models. Instead of combining the surrogates in an ensemble, the authors proposed that the multiple surrogates are used in parallel.

For each model, a solution is chosen to be evaluated with the real fitness function and it is also used to update the models.

In [8], Zhang et al. proposed to couple the MOEA/D [9] algorithm with a meta-model called Gaussian Process (also known as Kriging method). Their proposed approach is called MOEA/D-EGO, which uses the Gaussian process meta-model for modeling each one of the objective functions and to derive models for the decomposed problems. In each generation, a number of k points are selected to be evaluated with the expensive fitness functions, and these are added to an external population. MOEA/D-EGO was evaluated using a suit of benchmarks problems with up to 8 decision variables, and with 200 and 300 fitness functions evaluations.

Lim et al. [10] proposed the use of surrogate models within local search in a memetic algorithm. The authors used two types of surrogates: an ensemble and a low-order polynomial. The surrogates were trained to approximate a weighted sum of the objectives. A single-objective algorithm coupled with the surrogates was run to find an optimal solution. A selection strategy was proposed for deciding which solutions had to be added to the population.

Knowles [11] proposed ParEGO, a hybrid approach that combines Gaussian processes with an extension to the EGO algorithm for multi-objective problems. ParEGO applies the EGO algorithm to a randomly selected weighting vector to scalarize the objective functions, and it is used to find a point to be evaluated with the expensive fitness functions. In order to avoid the computational overhead, this algorithm limits the maximum number of samples used for training the surrogate, such that information from previous evaluated points can be lost, causing that these points could be evaluated again. This approach is used to solve multi-objective problems with low-to-medium dimensionality (in fact, up to 8 decision variables) with only 100 and 250 fitness functions evaluations.

Zhou et al. [12] used multiple surrogate models, such as polynomial regression and radial basis functions, with a memetic algorithm. An evolutionary algorithm was coupled with a local search engine, which used multiple surrogates. These surrogates were combined in an ensemble. This ensemble was used to evaluate the solutions during the local search.

It has also been of interest the issue of choosing an appropriate surrogate model for using in evolutionary optimization. In this direction, a comparative study has been performed by Montemayor-Garcia and Toscano-Pulido [4] and Diaz-Manriquez et al. [3]. In both works, four different surrogate models (polynomial regression (PR), kriging, radial basis functions (RBFs) and support vector regression (SVR)) are compared in different aspects of the models during the evolutionary process, such as accuracy, robustness, efficiency, and scalability. In [4], the authors concluded that for low dimensional problems, kriging methods and PR are the most suitable approaches, while for high dimensional problems, RBF and SVR are preferred. In contrast, in [3], the authors concluded that for low dimensional problems, the best approach can be kriging or even SVR, while for high dimensional problems, RBF can be the most suitable.

Santana-Quintero et al. [13] also perform a comparative study between support vector machines, artificial neural networks, and radial basis functions. From that comparative study, the authors concluded that the SVM provides better approximations. Then, an SVM is coupled with a multi-objective particle swarm optimizer (MOPSO) and Rough Sets to solve multi-objective optimization problems.

Pilat and Neruda [14] proposed a framework to choose a supervised learning algorithm, to be used to construct the surrogate. The constructed surrogate is used in EMO. The surrogate is used in two ways: to pre-select individuals and in a local search engine. As their surrogate models, the authors used linear regression, 3 types of support vector regression, 3 types of artificial neural networks, and 2 types of RBF networks and Gaussian processes. In that framework, the selection for hyperparameters values for each kind of surrogate is not performed; instead, the default hyperparameters values were used.

Based on the previous literature review, we found out that there exists an interest in using different surrogate models to improve fitness estimation, as well as in determining the most suitable surrogate model to be used in an optimization problem. Notwithstanding, in these works the automated model selection step is not included in the MOEA. An exception of this is Pilat and Nerudas's work, in which a learning algorithm selection stage is incorporated into the framework. They do not consider to choose the appropriated set of hyper-parameters for the selected learning algorithm, but from the machine learning field it is well known that the performance of a model highly depends on the hyper-parameters configuration. For that reason, in this work we propose an algorithm that automatically chooses the most suitable hyper-parameters for the SVMs, which are combined into an ensemble method to improve the fitness estimation. We chose the SVM algorithm because it has a high performance over different problems, included those used in our study.

V. Our Proposed Approach

The proposed approach combines a MOEA with an ensemble of surrogate models for approximating the objective functions. The ensemble is constructed in an incremental fashion. As our surrogate model, we used SVMs. Since different SVMs models can be constructed using different hyperparameters values, this raises the issue of model selection, i.e., determining an appropriate combination for the hyperparameters values that provides the most accurate model that is possible. The proposed approach is presented in Algorithm 1, and it is described in detail in the following sections.

A. Initialization

In order to ensure a well-distributed initial population, this is created using the Latin-hypercube sampling technique [15]. The initial population is evaluated using the real functions. Our proposal uses two external archives: the first one (external archive A) stores the non-dominated

Algorithm 1 SAMOEA/SVM

Require: **f**: objective functions vector,
N: population size,
G: maximum number of generations,
m: size of external archive B,
n: number of models in the ensemble.
Ensure: A set of non-dominated solutions (external archive A)
1: Create an initial population, P_0, using latin-hypercubes sampling
2: Evaluate initial population's members using the real fitness functions
3: Store the non-dominated solutions in external archive A
4: Store the population's members in external archive B
5: Construct surrogate model.
6: **for** $g = 1$ **to** G **do**
7: Apply evolutionary operators (selection, crossover and mutation) over the parents and archive A populations to create an offspring population
8: Evaluate offspring's members using the surrogate models
9: Determinate the non-dominated solutions from offspring population
10: Evaluate the non-dominated solutions using the real fitness functions
11: Update external archives.
12: Update surrogate models.
13: **end for**

solutions found during the evolutionary process, while the second one (external archive B) stores the evaluated solutions with the real functions and it is used as training samples for the surrogates. Therefore, the external archive A initially stores the non-dominated solutions from the initial population, and the external archive B stores the solutions from the initial population. It must be noted that both external files are limited to store a fixed number of samples (N samples for external archive A, and m samples for external archive B).

B. Constructing Models

After the initial population has been created and evaluated with the real objective functions, the next step is to construct the surrogates. The solutions stored in external archive B are used as training samples for constructing a surrogate. As it was previously indicated, we used an SVM as a surrogate model. In order to improve the surrogate's reliability, a model selection procedure is first performed. Therefore, for each SVM kernel type (polynomial, Gaussian, and sigmoid), a grid search is performed for determining the most appropriate hyperparameters values among a set of them. The set of hyperparameters values that gives the lowest expected generalization error is chosen. For determining the expected generalization error, we used the k-fold cross validation sampling technique. After determining the combination of hyperparameters, the surrogate is trained and used for the fitness value estimation. This process is summarized in Algorithm 2.

C. Choosing Solutions for being Evaluated with the Real Fitness Functions

Once the surrogate has been constructed and an offspring population has been created (applying the evolutionary operators), we proceed to determine the solutions to be evaluated with the real fitness functions. At this stage, the offspring's members are evaluated using the surrogate. Then, we determine which are the non-dominated solutions from the offspring population. The non-dominated solutions are then chosen for being evaluated with the real fitness functions.

Algorithm 2 construct_Models

Require: External archive B
Ensure: A set of trained models
1: Let $\mathbf{X}$ be a set of samples points which were evaluated with the fitness functions
2: Let $\mathbf{Y}$ be a vector with the responses of each point to each objective function
3: **for** $g = 1$ **to** l, where l is the number of objectives **do**
4: Perform a grid search for determining a set of hyper-parameters for an SVM appropriated for $\mathbf{X}$ with the corresponding responses $\mathbf{Y}^{(g)}$ (see Algorithm 3)
5: Train a meta-model using the determined hyper-parameters and $\mathbf{X}$ and $\mathbf{Y}^{(g)}$
6: Add the trained meta-model to the set of models.
7: **end for**

Algorithm 3 grid_Search

Require: d: number of grids for each dimension
Ensure: A set of hyper-parameter
1: Let $\mathbf{X}$ be a set of samples points which were evaluated with the fitness functions
2: Let $\mathbf{Y}$ be a vector with the responses of each point to an objective function
3: **for** each kernel type $\in$ {linear, polynomial, Gaussian, sigmoid} **do**
4: **for** each i $\in \{1, \ldots, d\}$ **do**
5: Use k-fold cross validation to estimate the expected generalization error of a meta-model when it is trained with a particular set of hyper-parameters and the samples $\mathbf{X}$ and $\mathbf{Y}$.
6: **end for**
7: **end for**
8: Choose the set of hyper-parameters with the lowest expected generalization error

D. Updating External Archives

For storage and processing reasons, the size of the external archives is limited. External archive A is limited to store N solutions, where N is equal to the population size. This archive stores the non-dominated solutions found during the search process. For adding a new solution, this new solution should be non-dominated with respect to the solutions stored in archive A. If the added solution dominates some solutions stored in archive A, such solutions are removed from the archive. Once archive A has N samples, it is allowed to add new non-dominated solutions, but the archive is pruned to N samples. For pruning, a clustering technique is used, where the solutions are grouped in N clusters, and the nearest solutions to each cluster center are chosen.

External archive B stores all solutions evaluated with the real fitness functions. Once this archive has stored its maximum allowable number of samples, adding new

solutions is allowed, but the external archive B is pruned for that sake. Pruning is based on Pareto ranking. First, the solutions are sorted according to their non-dominance level. The solutions from the first non-dominated front are added, followed by the second front, and so on, until B has at least the maximum allowable size. The fronts that were not added are deleted. If the size of the external archive B is larger than its allowable value, a clustering technique is used to prune the last added front.

E. Updating Models

The surrogate models are used to approximate the solutions' fitness values. Nonetheless, the new knowledge acquired during the evolutionary process should be included to update the information behavior of the fitness functions. Therefore, the surrogate models will probably be constructed several times along the search process. To avoid increasing the computational cost to train the model in each generation of the evolutionary algorithm, we propose a criterion based on surrogate fidelity for determining when a model should be trained. Since the suggestions for solutions to be evaluated with the real fitness functions are based on the fitness estimation provided by the surrogate, when these suggestions are worse than a random one, then the surrogates must be trained (performing the corresponding model selection), otherwise, the current models are still used.

Furthermore, the use of ensembles could be beneficial for improving the approximations of the fitness values, due to the diversity in the behavior of the ensemble members. For that reason, we explore the use of an ensemble of SVMs. The ensemble is constructed incrementally. First, the ensemble has only one member, which is the model constructed with the solutions in the initial population. After that, when a model is trained (according to the criterion that was previously explained), this is added to the ensemble. In this way, new information is added at the time the information from previous explored regions is preserved. The size of the ensemble is limited to a fixed number of models. Finally, when the maximum number of models is reached, in order to add a new model, the oldest member of the ensemble is removed.

VI. Experiments and Results

For our experiments, we used the Zitzler-Deb-Thiele (ZDT) test suite [16]. This benchmark consists of 6 bi-objective optimization problems, with between 10 and 30 decision variables. However, here, we did not consider ZDT5, because it is a binary problem. The considered problems have diverse characteristics, such as convex, non-convex, discontinuous, and multi-frontal problems. The interested reader is referred to [16] for a detailed description of these problems.

We used the following parameters for our proposal in our experiments:

TABLE I: Number of solutions evaluated with the real fitness function and trained models averaged over the 30 replications.

Function	Evaluations	No. trained models
ZDT1	2033.762 ± 420.701	31.267 ± 09.116
ZDT2	1608.350 ± 719.080	15.700 ± 09.917
ZDT3	2371.683 ± 192.815	66.667 ± 07.087
ZDT4	2313.894 ± 463.189	30.267 ± 09.989
ZDT6	2186.435 ± 198.136	24.367 ± 19.200

Population size	100
Generations	100
Crossover rate	0.9
Mutation rate	$\frac{1}{n}$, where n is the number of decision variables
Distrib. index for crossover	20
Distrib. index for mutation	20
max. num. of models	5
External archive size	100

As we indicated before, the purpose of using surrogate models is to reduce the number of fitness functions evaluations. Therefore, it is important to assess this reduction in the first place. In Table I, we present the number of real fitness function evaluations performed by our proposed approach, as well as the number of trained models used during the search process. The reported results are the averages and standard deviations from 30 independent runs. From that table, it can be noted that the test problem that required the highest number of fitness evaluations was ZDT3, with an average of 2,371.683 fitness function evaluations. This represents a reduction of 76.28%, on average, of the number of fitness evaluations performed by the MOEA, when not using surrogates (i.e., the MOEA without surrogates performs 100 individuals $\times$ 100 generations $= 10,000$ evaluations). This would evidently transform in a significant reduction in CPU time if our proposed approach is used in computationally expensive problems. It is also noted that the number of trained models ranges from 15.70 to 66.67, which also represents computational savings, since it was not necessary to update the model at each generation.

In order to quantitatively assess the performance of our proposed approach, we adopted the following performance measures: Averaged Hausdorff Distance (Δ_p) [17], Spread (Δ) [2], and Set Coverage (SC) [18]. We performed 30 independent runs for each test problem. Our obtained results are compared with respect to those obtained by the NSGA-II [19]. In order to make the comparison as fair as possible, the parameters of the NSGA-II were fixed so that it performed 3,000 fitness functions evaluations. Therefore, the population size for the NSGA-II was fixed to 60, the maximum number of generations was set to 50, the crossover probability was set to 0.9, the mutation probability was set to $\frac{1}{n}$ (where n is the number of decision variables), and the distribution index for crossover and mutation was set to 20.

Table II shows the obtained results with both approaches: NSGA-II and our proposal (SAMOEA/SVM), for each performance measure. The reported results are the

TABLE II: Results obtained by the NSGA-II and our proposed approach. The reported results are the average and standard deviation for the 30 independent runs performed for each test problem. The best result is shown in **boldface**.

Test Problem	Δ_p		Δ		SC	
	SAMOEA/SVM	NSGA-II	SAMOEA/SVM	NSGA-II	SAMOEA/SVM	NSGA-II
ZDT1	$\mathbf{1.41e-2 \pm 4.20e-3}$	$2.28e-1 \pm 7.39e-2$	$\mathbf{0.590 \pm 0.120}$	0.730 ± 0.049	$\mathbf{0.000 \pm 0.000}$	0.996 ± 0.012
ZDT2	$\mathbf{1.75e-2 \pm 1.51e-2}$	$6.43e-1 \pm 2.06e-1$	$\mathbf{0.435 \pm 0.248}$	0.892 ± 0.073	$\mathbf{0.000 \pm 0.000}$	0.876 ± 0.300
ZDT3	$\mathbf{2.06e-2 \pm 1.70e-2}$	$2.40e-1 \pm 5.10e-2$	$\mathbf{0.671 \pm 0.085}$	0.750 ± 0.044	$\mathbf{0.000 \pm 0.000}$	0.997 ± 0.011
ZDT4	$\mathbf{1.30e+1 \pm 0.70e+1}$	$1.49e+1 \pm 1.17e+1$	$\mathbf{0.973 \pm 0.055}$	1.028 ± 0.119	0.814 ± 0.255	$\mathbf{0.245 \pm 0.275}$
ZDT6	$\mathbf{3.02e-1 \pm 3.15e-1}$	$1.71e+0 \pm 2.41e-1$	$\mathbf{0.872 \pm 0.408}$	0.926 ± 0.059	$\mathbf{0.010 \pm 0.011}$	0.686 ± 0.176

TABLE III: Obtained p-values from the Z-test. Cases with a p-value less than a 0.05 are statistically significant with a 95% of confidence and these are indicated with an asterisk (*).

Test Problem	p-value		
	Δ_p	Δ	SC
ZDT1	$< 0.001^{(*)}$	$< 0.001^{(*)}$	$< 0.001^{(*)}$
ZDT2	$< 0.001^{(*)}$	$< 0.001^{(*)}$	$< 0.001^{(*)}$
ZDT3	$< 0.001^{(*)}$	$< 0.001^{(*)}$	$< 0.001^{(*)}$
ZDT4	0.451	$0.022^{(*)}$	$< 0.001^{(*)}$
ZDT6	$< 0.001^{(*)}$	0.660	$< 0.001^{(*)}$

average and standard deviation for the 30 runs performed for each test problem. The best result obtained for each test problem is shown in **boldface**. From this table, it can be observed that our proposal had a better performance for most of the test problems considered (ZDT1, ZDT2, ZDT3, and ZDT6) with respect to the performance measures adopted: Δ_p, Δ, and SC. The only exception is ZDT4 in the SC metric. We performed the Z statistical significance test in order to determine if the differences between the two algorithms was statistically significant. The Z-score can be computed as follows:

$$Z = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{4}$$

where $\overline{x_1}$ and $\overline{x_2}$ are the means of the performance for a pair of algorithms, σ_1 and σ_2 represent the corresponding standard deviations, and n_1 and n_2 are the corresponding numbers of replications.

The Z-score can be converted to a significant level through a normal cumulative distribution function, such that a Z-value of 1.96 corresponds to a 95% significance level (i.e. this Z-value corresponds to an alpha level of 0.05). We applied this statistical test to the obtained results for both approaches for each metric, and the obtained p-values are presented in Table III. From that table, and according to the statistical significance test, our proposal statistically outperformed NSGA-II in the ZDT1, ZDT2, ZDT3, and ZDT6 test functions when the Averaged Hausdorff Distance (Δ_p) and SC metrics were considered. But, in the case of SC metric, it was statistically outperformed in the ZDT4 test function. With respect to spread (Δ), the statistical test showed that our proposed approach statistically outperformed NSGA-II in ZDT1 to ZDT4, but the difference with the ZDT6 function was not statistically significant.

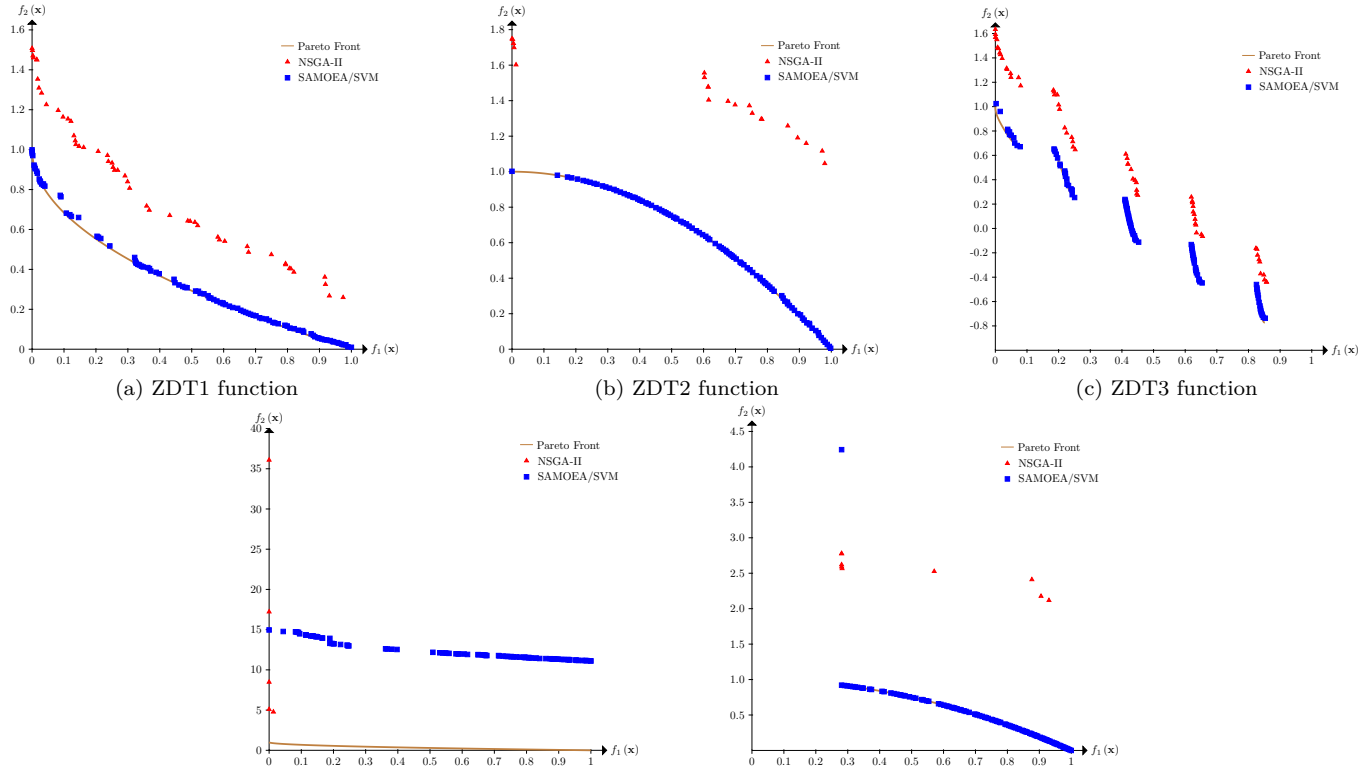
Figure 1 shows the Pareto fronts generated by both our proposed approach and the NSGA-II, for each test problem. This figure allows us to graphically observe that, for most of the test problems adopted, our proposal gets points which are closer to the true Pareto front than those obtained with the NSGA-II. This figure also allows us to observe that the solutions generated by the NSGA-II are far from the true Pareto front. Furthermore, for ZDT2, ZDT4, and ZDT6, the number of non-dominated points generated by the NSGA-II is scarce after performing 3,000 fitness functions evaluations. In contrast, our proposed approach, besides generating points closer to the true Pareto front, it generates points that are better distributed along it. Note that for ZDT4, the approximations generated by both approaches are far from the true Pareto front. This is caused by the evident difficulty that both approaches have to deal with multi-frontal problems.

VII. Conclusions and Future Work

In this paper, we proposed an approach that uses an ensemble of surrogate models for solving multi-objective optimization problems, coupled with a model selection strategy for determining the hyperparameters values for an SVM. Since choosing these hyperparameters adds an extra cost to the proposed approach, a criterion based on the surrogate fidelity allows determining when a surrogate should be trained. In this way, we avoid training a surrogate at each generation of the evolutionary process. This allows saving computational time.

Our experimental results showed the advantages of our proposed approach when compared to the NSGA-II. In fact, the NSGA-II was not able to converge to the true Pareto front when performing 3,000 fitness functions evaluations, while our proposed approach generated better approximations with fewer evaluations of the real fitness functions. In spite of the poor performance of our proposed approach on ZDT4, the statistical tests that were conducted, showed that the improvements obtained in ZDT1, ZDT2, ZDT3, and ZDT6 are statistically significant. Based on these results, we assume that our proposed approach can be useful in real-world problems in which the fitness function evaluations are time-consuming.

As part of our future work, we would like to extend our approach to consider more surrogate models types, besides the SVM, so that our approach can be able to choose among a set of surrogate models. We are interested in studying other approaches to perform the model selection step, making this procedure more efficient. We are also interested in coupling our proposal with a local search engine in order to improve its convergence. Finally, we



(a) ZDT1 function (b) ZDT2 function (c) ZDT3 function

(d) ZDT4 function (e) ZDT6 function

Fig. 1: ZDT test problems. The above plots correspond to the Pareto fronts generated by the NSGA-II and our proposed approach (SAMOEA/SVM) for each of the ZDT test problems.

want to test our approach with problems having more than two objectives.

Acknowledgements

The first author is grateful for the support from CONACyT scholarship no. 329013. The second author acknowledges support from CONACyT project no. 103570.

References

[1] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, 2nd ed. Springer, US, 2007.

[2] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[3] A. Diaz-Manriquez, G. Toscano-Pulido, and W. Gomez-Flores, "On the selection of surrogate models in evolutionary optimization algorithms," in *2011 IEEE Congress on Evolutionary Computation (CEC'2011)*, 2011, pp. 2155 –2162.

[4] G. Montemayor-Garcia and G. Toscano-Pulido, "A study of surrogate models for their use in multiobjective evolutionary algorithms," in *2011 8th International Conference on Electrical Engineering Computing Science and Automatic Control (CCE'2011)*, oct. 2011, pp. 1 –6.

[5] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[6] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, pp. 199–222, 2004.

[7] A. Arias-Montaño, C. A. Coello Coello, and E. Mezura-Montes, "Multi-Objective Airfoil Shape Optimization Using a Multiple-Surrogate Approach," in *2012 IEEE Congress on Evolutionary Computation (CEC'2012)*. Brisbane, Australia: IEEE Press, June 10-15 2012, pp. 1188–1195.

[8] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by moea/d with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010.

[9] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 712–731, 2007.

[10] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing Surrogate-Assisted Evolutionary Computation," *IEEE Transactions On Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, June 2010.

[11] J. Knowles, "Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[12] Z. Zhou, Y. S. Ong, M. H. Lim, and B. S. Lee, "Memetic algorithm using multi-surrogates for computationally expensive optimization problems," *Soft Computing*, vol. 11, no. 10, pp. 957–971, 2007.

[13] L. V. Santana-Quintero, C. A. Coello Coello, and A. G. Hernández-Díaz, "Hybridizing Surrogate Techniques, Rough Sets and Evolutionary Algorithms to Efficiently Solve Multi-Objective Optimization Problems," in *2008 Genetic and Evolutionary Computation Conference (GECCO'2008)*. Atlanta, USA: ACM Press, July 2008, pp. 763–764, iSBN 978-1-60558-131-6.

[14] M. Pilat and R. Neruda, "Meta-learning and model selection in multi-objective evolutionary algorithms," in *11^{th} Inter-*

national Conference on Machine Learning and Applications (ICMLA'12), vol. 1, 2012, pp. 433 –438.

[15] M. D. Mckay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[16] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, Summer 2000.

[17] O. Schutze, X. Esquivel, A. Lara, and C. A. Coello Coello, "Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 504–522, 2012.

[18] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.