Use of Particle Swarm to accelerate convergence in a Surrogate-based algorithm to solve Multi-objective Optimization Problems

Luis Vicente Santana-Quintero
CINVESTAV-IPN
Computer Science Department
Av. IPN No. 2508
Col. San Pedro Zacatenco
México D.F. 07300, MEXICO
Email: lvspenny@hotmail.com

Carlos Coello Coello
CINVESTAV-IPN
Computer Science Department
Av. IPN No. 2508
Col. San Pedro Zacatenco
México D.F. 07300, MEXICO
Email: ccoello@cs.cinvestav.mx

Alfredo G. Hernández-Díaz
Pablo de Olavide University
Department of Quantitative Methods
Seville, SPAIN
Email: agarher@upo.es

Jesús Moisés Osorio Velázquez
Instituto Tecnológico de Culiacán
Systems and Informatics Department
Culiacán, Sinaloa, MEXICO
Email: wcoder.mx@gmail.com

***Abstract*— This work presents a new algorithm that approximate the real function evaluation using supervised learning with a surrogate method which is called support vector machine (SVM). We perform a comparative study among the different variants to select the leader in the PSO algorithm, and found out that the selection based in Random and Random Generator are suitable to deal with the type of problems that we are interested, multimodal problems with high dimensionality (up to 30 decision variables). However, the results show that the spread of solutions achieved by the SVM-based MOEA is poor. So, we decided to add a second phase approach based on a mathematical approach called Rough sets in order to improve the spread of solutions along the true Pareto front. In this case, the rough sets act as a local search procedure which is able to generate solutions in the neighborhood of the nondominated solutions previously generated by the surrogate-based algorithm. We show the performance of the hybrid algorithm with only 2,000 fitness function evaluations comparing the results with an algorithm that is representative of the state-of-the-art.**

I. Introduction

In recent years, the statement and solution of multi-objective optimization problems has become very common in a wide variety of disciplines in which computational efficiency is a critical issue.

Problems with multiple objectives are present in most disciplines, the complexity of their solution is not trivial in most cases, causing a great interest of many researchers to find a suitable solution of these problems in which computational efficiency is a critical issue. From the many techniques adopted to solve such multi-objective optimization problems, evolutionary algorithms are among the most popular mainly because of the population-based nature. However, dealing with a large population and a large number of generations cause the multi-objective evolutionary algorithms (MOEAs) to deal with a large number of objective function evaluations that is unaffordable in certain applications even when parallelism is adopted. Such computational efficiency is precisely the focus of this paper, in which we propose a hybrid scheme that aims to minimize the total number of objective function evaluations performed. Our proposed approach combines a MOPSO with surrogates in order to produce a quick (i.e., with a low number of fitness function evaluations) approximation of the Pareto front. Then, rough sets are used to diversify the neighborhood surrounding each of the nondominated solutions produced before, such that the rest of the Pareto front is reconstructed.

The remainder of this paper is organized as follows. Section II provides some basic background on surrogates. In Section III, we provide a brief introduction to particle swarm and rough sets theory. A review of the previous related work is provided in Section IV. Section V describes our proposed hybrid scheme. The comparison of results is provided in Section VI. Finally, in Section VII we present our conclusions as well as some possible paths to continue with this research.

II. Surrogates

Surrogate models can perform a number of tasks in support of a computational analysis. Through interpolation, extrapolation and/or integration, these models can be used to address complex problems involving experimental design, system analysis and prediction. In a single-objective optimization context, surrogate models have been successful in dealing with highly demanding problems where the cost of evaluating the real fitness function is very expensive (computationally speaking). The accuracy of the surrogate model relies on the number of samples provided in the search space, as well as on the selection of the appropriate model to represent the objective

Carlos A. Coello Coello acknowledges support from project CONACyT number: 45683-Y.

functions. There exist a variety of techniques for constructing surrogate models (see for example [17]). A surrogate model is built when the objective functions are to be estimated. This local model is built using a set of data points that lie on the local neighborhood of the design. Since surrogate models will probably be built thousands of times during the search, computational efficiency is the main objective. This motivates the use of different approaches such as: artificial neural networks, radial basis functions, and support vector machines. All of them can be applied to approximate multiple data. Each of these approaches will be briefly discussed next.

A. Support Vector Machines

The Support Vector Machine learning algorithm is among the best (and many believe is indeed the best) supervised learning algorithms. In a SVM regression, our goal is to find a function $f(x)$ that has at most an ϵ deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. Let's suppose we are given training data $\chi = (x_t, y_t)_{t=1}^{N}$ where $y_t \in \mathbb{R}$. Then, the $f(x)$ is given by:

$$f(x) = \langle w, x \rangle + b \; with \; w \in \mathbb{R}^d, x \in \mathbb{R}^d, b \in \mathbb{R}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product in χ. A small w means that the regression is flat. One way to ensure this is to minimize the norm, $||w||^2 = \langle w, w \rangle$. The problem can be written as a convex optimization problem:

$$\begin{array}{ll} minimize & \frac{1}{2}||w||^2 \\ subject\ to & \left\{ \begin{array}{ll} y_i - \langle w, x_i \rangle - b & \leq \epsilon \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon \end{array} \right. \end{array} \quad (1)$$

And one can introduce slack variables ξ_i, ξ_i^*, for positive and negative deviations:

$$\begin{array}{ll} minimize & \frac{1}{2}||w||^2 + C \sum_{i=1}^{l} (\xi_i + \xi_i^*) \\ subject\ to & \left\{ \begin{array}{ll} y_i - \langle w, x_i \rangle - b & \leq \epsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{array} \right. \end{array} \quad (2)$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated. The ϵ-insensitive loss function (see eq. 3) means that we tolerate errors up to ϵ and also that errors beyond that value have a linear effect and not quadratic. This error function is therefore more tolerant to noise and is thus more robust.

$$|\xi|_\epsilon = \left\{ \begin{array}{ll} 0, & \text{if } |\xi| \leq \epsilon; \\ |\xi| - \epsilon, & \text{otherwise.} \end{array} \right. \quad (3)$$

Figure 1, shows a graphic of the ϵ-insensitive loss function. Note that only the points outside the shaded region contribute to the cost of the function.

In most cases, the optimization problem defined by equation (2) can be solved more easily in its dual formulation.

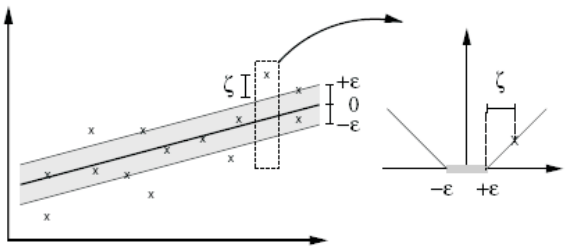

Fig. 1. ϵ-insensitive loss function for SVM

The dual formulation also provides the capability for extending SVM to nonlinear functions using a standard dualization method utilizing Lagrange multipliers, as described by Fletcher [6]. Further details on the use of SVM for regression can be found in [16].

III. Algorithmic Background

A. Particle Swarm Optimization (PSO)

Kennedy & Eberhart [11] proposed an algorithm called "Particle Swarm Optimization" (PSO) which was inspired on the choreography of a bird flock. Each solution is represented by a particle and the evolution of the swarm to the optimal solutions corresponds to the velocity equation. This equation is composed by three elements: a velocity inertia, a cognitive component *pbest* and a social component *gbest*. Each particle is affected by either the best local and the best global particle.

First, in the PSO algorithm the particles are initially randomly through the search space and the *pbest* is also initialize. Next, the fittest particle from all the swarm is selected and assigned to the *gbest* solution. Then, the particle swarm flies the search space. This flight function is determined by the equation 4, which updates the position and fitness of the particle (equation 5). The new fitness is compared with respect to particle's *pbest* position and if its better, then replaces it in the *pbest* position. This procedure is repeated until the swarm is updated and the termination criteria is reached.

$$v_{i,d} = w \cdot v_{i,d} + c_1 \cdot U(0,1)(pbest_{i,d} - x_{i,d}) + c_2 \cdot U(0,1)(gbest_d - x_{i,d}) \quad (4)$$

$$x_{i,d} = x_{i,d} + v_{i,d} \quad (5)$$

where c_1 & c_2 are constants that indicates the attraction form the *pbest* or *gbest* position respectively; w refers to the velocity inertia of the previous movement; $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ represents the i-th particle.

B. Rough Sets Theory

Rough Sets theory is a mathematical approach to imperfect knowledge originally proposed by Pawlak [15]. Let's assume that we are given a set of objects U called the *universe* and an indiscernibility relation $R \subseteq U \times U$, representing our lack of knowledge about elements of U (in our case, R is simply an equivalence relation based on a grid over the feasible set; this is, just a division of the feasible set in (hyper)-rectangles). Let

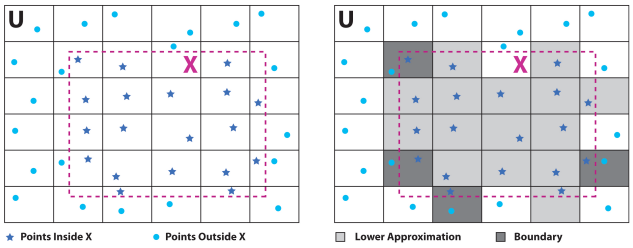

Fig. 2. Rough sets approximation

X be a subset of U. We want to characterize the set X with respect to R. The way rough sets theory expresses vagueness is employing a boundary region of the set X built once we know points both inside X and outside X. If the boundary region of a set is empty it means that the set is $crisp$; otherwise, the set is $rough$ (inexact). A nonempty boundary region of a set means that our knowledge about the set is not enough to define the set precisely (see Figure 2).

Then, each element in U is classified as *certainly* inside X if it belongs to the lower approximation or *partially* (*probably*) inside X if it belongs to the upper approximation (see Figure 2). The *boundary* is the difference of these two sets, and the bigger the boundary the worse the knowledge we have of set X. On the other hand, the more precise is the grid implicity used to define the indiscernibility relation R, the smaller the boundary regions are. But, the more precise is the grid, the bigger the number of elements in U, and then, the more complex the problem becomes. Then, the less elements in U the better to manage the grid, but the more elements in U the better precision we obtain. Consequently, the goal is obtaining "small" grids with the maximum precision possible. These two aspects are called **Density** and **Quality** of the grid. If q is the number of criteria (in our case, the number of objectives), Q_i is the i-th criteria, b_j^i is the j-th value of the i-th criteria (we assume these value are ordered increasingly), then:

$$Density(G) = \sum_{i=1}^{q} \sum_{j=1}^{|Q_i|} x_j^i \; ; \quad Quality(G) = \frac{|Low(X)|}{|X|}$$

where x_j^i is 1 if b_j^i is active in the grid and $|Low(X)|$ is the cardinality of the lower approximation of X.

IV. Previous Related Work

Currently, there exist several evolutionary algorithms that use a meta-model to approximate the real fitness function and reduce the total number of fitness evaluations without degrading the quality of the results obtained. However, most of these approaches only deal with single-objective optimization problems (see for example [9]). Evidently, in the discussion presented next, we will only focus on the work that deals with multi-objective evolutionary algorithms.

Ong et al. [14] used surrogate models (RBFs) to solve computationally expensive design problems with constraints. The authors used a combination of a parallel evolutionary algorithm coupled with sequential quadratic programming in order to find optimal solutions of an aircraft wing design problem. In this case, the authors construct a local surrogate model based on radial basis functions in order to approximate the objectives and the constraint functions of the problem.

Karakasis et al. [10] used surrogate models based on RBFs in order to deal with computationally expensive problems. A method called Inexact Pre-Evaluation (IPE) is applied into a MOEA's selection mechanism. Such method helps to choose the individuals that are to be evaluated using the real objective function, right after a meta-model approximation has been obtained by the surrogate. The results are compared against a conventional MOEA in two test problems, one from a benchmark and one from the turbomachinery field.

Voutchkov & Keane [19] studied several surrogate models (RSM, RBFs and Kriging) in the context of multi-objective optimization using the NSGA-II [5] as the MOEA that optimized the meta-model function given by the surrogate. The surrogate model is trained with 20 initial points and the NSGA-II is run on the surrogate model. Then, the 20 best resultant points given by the optimization are added to the existing data pool of real function evaluations and the surrogate is re-trained with these new solutions. A comparison of results is made in 4 test functions (from 2 to 10 variables), performing 400 real fitness function evaluations in each of them.

Knowles [12] proposed "ParEGO", which consists of a hybrid algorithm based on a single optimization model (EGO) and a Gaussian process, which is updated after each function evaluation, coupled to an evolutionary algorithm. EGO is a single-objective optimization algorithm that uses Kriging to model the search landscape from the solutions visited during the search and learns a model based on Gaussian processes (called DACE). This approach is used to solve multi-objective optimization problems of low dimensionality (up to 6 decision variables) with only 100 and 250 fitness function evaluations.

A. Multi-Objective PSO

Some multi-objective algorithms that are based in PSO are briefly described next:

- MOPSO by Benitez in 2005 [1] The authors propose three methods based on Pareto dominance for selecting leaders from an (external) archive. One technique that explicitly promotes diversity, one technique that explicitly promotes convergence and finally one technique that is a weighted probabilistic method and forms a compromise between diversity and convergence. This approach uses a turbulence factor that is added to the position of the particles with certain probability.

- DOPS by Bartz-Beielstein [2] This approach starts from the idea of introducing elitism into PSO. Different methods for selecting and deleting particles from the archive are analyzed to generate a satisfactory approximation of the Pareto front. Deleting methods are either inversely related to the selection fitness value or based on the previous success of each particle.

- MOPSO by Coello & Lechuga [3] This proposal is based on the idea of having an external archive in which

every particle will deposit its flight experiences after each flight cycle. Additionally, the updates to the repository are performed considering a geographically-based system defined in terms of the objective function values of each individual; this repository is used by the particles to identify a leader that will guide the search. This approach also uses a mutation operator that acts both on the particles of the swarm, and on the range of each design variable of the problem to be solved.

V. Our Proposed Approach

Our proposed approach is an hybrid composed by two different phases, the first one (Surrogate-based) which uses the PSO to optimize the approximation obtained by the surrogate models and the main purpose is to produce a reasonably good approximation of the true Pareto front. The second phase uses Rough Sets as a local search engine in order to improve the solutions produced in the previous phase. Each of these two phases is described next.

A. Phase 1: Surrogate-based MOEA

The surrogate model adopted in this work is shown in Figure 3. A multi-objective particle swarm optimizer (MOPSO) is adopted to optimize the approximate model generated by the surrogate (using SVMs). Our MOPSO maintains two populations: the main one (which is used to select the parents), and a second population that retains the global nondominated solutions.

First, we generate P individuals using Latin-Hypercubes [13], which guarantees a good distribution of the initial population in a multidimensional space. Our approximation model requires a good distribution of the sample points provided in order to build a good approximation of the real functions. A Latin cube is a selection of one point from each row and column of a square matrix. Then, we evaluate these P individuals with the real functions, and train the meta-model using the surrogate model.

Our MOEA is based on the PSO algorithm [11], which uses a leader selection based on the G_{best} model, in which we choose the leader particle from the nondominated set. We also add a turbulence operator in order to jump into search regions that the PSO flight equation is not able to reach. Replacing the comparison operator (to determine whether a solution is betther than other solution b is a natural modification to a PSO algorithm aimed handle multiple objectives.

The analogy of PSO with EAs makes evident the notion that using a Pareto ranking scheme [7] could be the straightforward way to extend the approach to handle multiple objectives. However, if we merge a Pareto ranking scheme with the PSO algorithm, a set of nondominated solutions will be produced (by definition, all nondominated solutions are equally good). Having several nondominated solutions implies the inclusion into the algorithm of both: an additional criteria to decide whether a new nondominated solution is P_{best} or G_{best} and a strategy to select the guide particles (P_{best} and G_{best}).

However, the selection of an "appropiate" leader becomes a difficult task, since there can be more than one leader in the G_{best} set. Therefore, an additional strategy to select one of the multiple G_{best} to use in the PSO's velocity equation is still necessary. Some possible leader selection strategies are: (1) a Randomly Dominator (the leader is randomly selected and preferably if this leader dominates the particle, if not then a random leader is selected), (2) randomly (a leader is randomly selected - no constraints are imposed on what sort of leader can a particle choose-), (3) the closest (a particle picks as a leader the geographical closest leader) and (4) the fartest (a particle picks as a leader the geographical fartest solution in the set)

All the nondominated solutions found by the MOPSO, are evaluated with the real function and added to the main population. Once all the points are in the main population, they are used to re-train the meta-model and get another approximation of the real objectives. As it is shown in Figure 3, this procedure is repeated until the $MaxGen$ number of generations is reached. At the end of the first phase procedure, we split the main population in two: 1) One which contains the nondominated solutions and 2) a second one, which contains the best dominated solutions that are needed for the second phase. Both populations use the $pa\epsilon$-dominance grid proposed in [8] to maintain diversity.

B. Phase 2: Rough Sets in Multi-Objective Optimization

For the problems of our interest, we will try to approximate the Pareto front using a Rough Sets grid. In order to do this, we will use an initial approximation of the Pareto front (provided by the surrogate algorithm) and will implement a grid in order to get more information about the front that will let us improve this initial approximation. To this aim, we must decide which elements of U (that we will call $atoms$ and will be just rectangular portions of decision variable space) are inside the Pareto optimal set and which are not. Once we have the *efficient atoms*, we can easily intensify the search over in decision variable space. To create this grid, we will have as inputs N feasible points divided in two sets: the nondominated points (ES) and the dominated ones (DS).

We must note the importance of the DS set as in a rough sets method the information comes from the description of the boundary of the two sets. Then, the more efficient points provided the better. However, it is also required to provide some dominated points, since we need to estimate the boundary between being dominated and being nondominated. Algorithm 1 describes a Rough Sets iteration.

VI. Analysis of Results

Our main goal is to reduce the number of fitness function evaluations. Thus, our experimental design considers that only a few function evaluations are performed in several multi-dimensional test problems from the **ZDT**, which are bi-objective, unconstrained and have between 10 and 30 decision variables each. The detailed description of these test functions was omitted due to space restrictions (see [20] for further information). Three performance measures were adopted in order to allow a quantitative assessment of our results: (1)

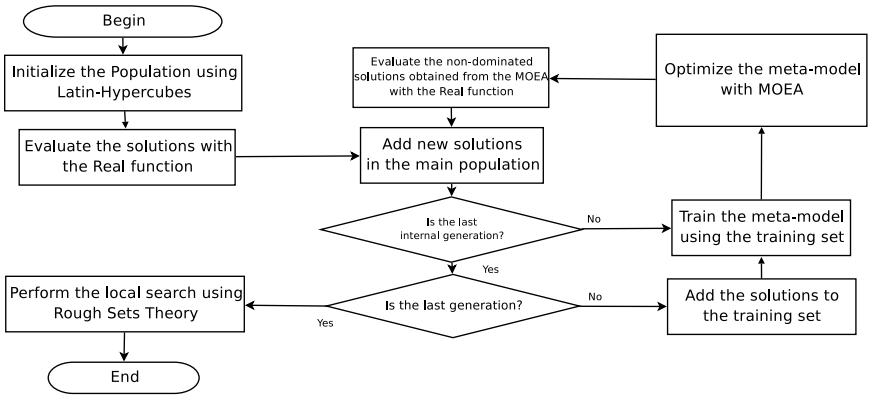

Fig. 3. Surrogate model adopted in this paper.

Algorithm 1 Rough Sets Iteration

1: **Input** nondominated points from the first phase ES.
2: **Input** dominated points from the first phase DS.
3: **Output** nondominated solutions found by the RS.
4: Choose $NumEff$ unexplored points of ES.
5: Choose $NumDom$ unexplored points of DS.
6: Generate $NumEff$ efficient atoms.
7: **for** $i = 0$ to $NumEff$ **do**
8: **for** $j = 0$ to $Offspring$ **do**
9: Generate (randomly) a point new in atom i and send it to ES
10: **if** new is efficient **then**
11: Include it in ES
12: **end if**
13: **if** A point old in ES is dominated by new **then**
14: Send old to DS
15: **end if**
16: **if** new is dominated by a point in ES **then**
17: Remove new
18: **end if**
19: **end for**
20: **end for**

Inverted Generational Distance (**IGD**), which is a variation of a metric proposed by Van Veldhuizen [18] in which the distance from the true Pareto to the approximation produced is measured; (2) Two Set Coverage (**SC**), proposed by Zitzler et al. [20], which performs a relative coverage comparison of two sets; and (3) Spread (**S**), proposed by Deb et al. [4], which measures both progress towards the Pareto optimal front and the extent of spread. For each test problem, 10 independent runs were performed.

This section is divided in two parts: in the first one, we run a comparative analysis with only 600 real function evaluations, comparing the leader selection in the PSO: (1) Randomly Dominator, (2) Randomly, (3) Closest and (4) Fartest. In the second part, the Rough Sets Theory algorithm is applied for another 1,400 real function evaluations and the results are compared with respect to the NSGA-II [5] performing 2,000 fitness function evaluations in total.

A. PSO Phase Analysis

The first phase of our approach uses several parameters: main population size $P = 100$, maximum number of evaluations = 600, MOPSO's internal population size ($P_{mopso} = 100$), maximum number of generations ($G_{mopso} = 100$), PSO flight equation (W= 0.1, $C_1 = 1.4$ and C_2 = 0.1), turbulence_rate = $1/n$ (n = number of decision variables).

The results reported in Table I correspond to the performance metrics adopted (IGD, S and SC). We show in boldface the best mean values per test function of 10 independently runs by all the leader selection models compared. We show the plot of all the nondominated solutions generated by a single run of the different algorithms in Figure 4. In all cases, we generated PF_{true}, so we could make a graphical comparison of the quality of the solutions produced by our approach. The summary of our results is the following:

MOPSO-1 (Randomly Dominator): When we select the leader that usually dominates the Particle, the algorithm shows a good performance in general, obtaining the best results for ZDT1, ZDT2 and ZDT3 in IGD and Spread. Graphically, it can be seen that it obtained a good approximation to the true Pareto front in ZDT1, ZDT2 and ZDT3.

MOPSO-2 (Randomly): When we use the randomly selection from the nondominated set, the performance of the algorithm is acceptable, showing the best performance in ZDT4 and ZDT6 with respect the others approaches. Graphically, it can be seen that is not able to produce better results than other approaches and that this approach stays far away from the true Pareto front.

MOPSO-3 (Closest): When we select the closest particle as a leader from the G_{best} set, the algorithm shows a poor performance in general except for ZDT4, outperforming the other schemes only in this test function.

MOPSO-4 (Fartest): When we select the fartest particle as the leader from the G_{best} set, we intend to help the algorithm to maintain diversity in the population, but apparently cause that the algorithm couldn't approximate quickly to the Pareto front instead of accelerate the convergence. The performance of this scheme was very poor and only in ZDT6 obtain good results.

From the first phase analysis, we can see that in ZDT4 the performance was very poor in general, getting trapped in one of the multiple local optima that this problem contains, but in the others approaches the result is not bad at all if we consider

that only 600 real evaluations are performed. So if we need to choose only one scheme to move forward in the algorithm, the best scheme is MOPSO-1 (randomly dominator), so we choose this scheme and decided to hybridize it with the Rough Sets to extent the solutions generated by the first phase and then compared it against the NSGA-II, an algorithm representative of the state of the art.

B. Second Phase Analysis

After our first experiments with the surrogate methods, it became clear that despite the fact that a good convergence was achieved, this was obtained at the expense of a poor distribution of the results. Thus, we decided to use rough sets as a local search engine, in order to find the solutions that are missing in the approximations obtained during the first phase of our approach. The second phase uses three more parameters: number of points randomly generated inside each atom ($Offspring$), number of atoms per generations ($NumEff$) and the number of dominated points considered to generate the atoms ($NumDom$). $Offspring = 1$, $NumEff = 2$ and $NumDom = 10$. The parameters used by the NSGA-II are the following: crossover rate = 0.9, mutation rate = $1/n$, η_c = 15, $\eta_m = 20$, population size = 52 and maximum number of generations = 39. These parameters make the NSGA-II to perform 2,000 fitness function evaluations in total.

The results reported in Table II are the mean values for each of the three performance measures adopted (SC, IGD and S) and the standard deviation of the 10 runs performed with 2000 evaluations in total of both approaches (SVM+RS and NSGA-II). The best mean values in each case are shown in boldface in Table II. The plot of all the nondominated solutions generated by a single run of both algorithms are in Figure 5.

It can be observed that in the ZDT test problems, our approach produced the best results with respect to the SC performance measure in all cases except for ZDT4. The same applies for the IGD performance measure. Also, our approach outperformed the NSGA-II with respect to the spread performance measure in three cases (ZDT2, ZDT3 and ZDT4). Graphically, it can be seen that our approach gets closer than the NSGA-II to the true Pareto front in ZDT1, ZDT2, ZDT3 and ZDT6, but not in ZDT4. The poor performance of all the approaches in ZDT4 might be attributed to the bad scalability presented by both approaches.

Our results indicate that the NSGA-II, despite being a highly competitive MOEA, is not able to converge to the true Pareto front in most of the test problems adopted when performing only 2,000 fitness function evaluations. It is worth noting, however, that if more evaluations are allowed, the NSGA-II is able to generate a very good (and well-distributed) approximation. Nevertheless, our aim was to propose a scheme for situations in which the allowable number of fitness function evaluations is low (no more than 2,000 in our case).

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT1	IGD	MOPSO-1	**0.0122**	0.0039	0.0074	0.0199
		MOPSO-2	0.0178	0.0034	0.0134	0.0222
		MOPSO-3	0.0201	0.0013	0.0176	0.0227
		MOPSO-4	0.0163	0.0037	0.0117	0.0219
	S	MOPSO-1	**0.6044**	0.0709	0.5160	0.7291
		MOPSO-2	0.6730	0.0677	0.5402	0.7744
		MOPSO-3	0.7287	0.0373	0.6782	0.7936
		MOPSO-4	0.6598	0.0502	0.5312	0.7179

SC	MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
MOPSO-1	–	0.0164	0.0817	0.1000	**0.066**
MOPSO-2	0.9417	–	0.3462	0.5343	0.6074
MOPSO-3	0.8829	0.5385	–	0.6225	0.6613
MOPSO-4	0.6598	0.3290	0.2646	–	0.4178

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT2	IGD	MOPSO-1	**0.0346**	0.0048	0.0200	0.0365
		MOPSO-2	0.0357	0.0003	0.0354	0.0362
		MOPSO-3	0.0366	0.0024	0.0353	0.0437
		MOPSO-4	0.0356	0.0003	0.0348	0.0361
	S	MOPSO-1	**0.1832**	0.3679	0.0000	0.9892
		MOPSO-2	0.2960	0.4522	0.0000	0.9981
		MOPSO-3	0.2993	0.4572	0.0000	1.000
		MOPSO-4	0.3985	0.4881	0.0000	0.9995

SC	MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
MOPSO-1	–	0.6300	0.6200	0.9200	0.7266
MOPSO-2	0.2750	–	0.2000	0.6000	0.3583
MOPSO-3	0.2500	0.7500	–	0.9000	0.6333
MOPSO-4	0.0000	0.3500	0.1000	–	**0.150**

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT3	IGD	MOPSO-1	**0.0287**	0.0099	0.0170	0.0462
		MOPSO-2	0.0351	0.0087	0.0221	0.0489
		MOPSO-3	0.0439	0.0109	0.0329	0.0742
		MOPSO-4	0.0314	0.0062	0.0197	0.0397
	S	MOPSO-1	**0.7178**	0.0718	0.5638	0.8179
		MOPSO-2	0.7544	0.0263	0.7081	0.8016
		MOPSO-3	0.7810	0.0222	0.7307	0.8127
		MOPSO-4	0.7582	0.0797	0.6055	0.8405

SC	MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
MOPSO-1	–	0.2381	0.2025	0.3489	**0.2631**
MOPSO-2	0.6286	–	0.1711	0.4912	0.4303
MOPSO-3	0.7344	0.6835	–	0.7011	0.7063
MOPSO-4	0.5153	0.2799	0.1446	–	0.3132

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT4	IGD	MOPSO-1	1.2384	0.2204	0.8144	1.6364
		MOPSO-2	1.1121	0.2442	0.8715	1.6318
		MOPSO-3	**1.0162**	0.1538	0.7503	1.2818
		MOPSO-4	1.2157	0.2108	0.8910	1.6635
	S	MOPSO-1	0.9806	0.0279	0.9234	1.0338
		MOPSO-2	0.9802	0.0135	0.9614	0.9981
		MOPSO-3	**0.8560**	0.2890	0.0000	1.0516
		MOPSO-4	0.8786	0.2932	0.0000	0.9968

SC	MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
MOPSO-1	–	0.7333	0.6583	0.6333	0.6749
MOPSO-2	0.1821	–	0.3815	0.4125	**0.3253**
MOPSO-3	0.2850	0.6217	–	0.3000	0.4022
MOPSO-4	0.2857	0.5000	0.6000	–	0.4619

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT6	IGD	MOPSO-1	0.0351	0.0028	0.0285	0.0405
		MOPSO-2	**0.0327**	0.0028	0.0286	0.0371
		MOPSO-3	0.0339	0.0034	0.0286	0.0377
		MOPSO-4	0.0330	0.0026	0.0271	0.0361
	S	MOPSO-1	0.9494	0.1284	0.7218	1.1933
		MOPSO-2	0.8850	0.1057	0.6669	1.0961
		MOPSO-3	0.8998	0.0944	0.7791	1.1026
		MOPSO-4	**0.8715**	0.0508	0.7921	0.9764

SC	MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
MOPSO-1	–	0.4494	0.4333	0.5494	0.4773
MOPSO-2	0.3560	–	0.3176	0.5501	0.4079
MOPSO-3	0.3526	0.3536	–	1.0000	0.5687
MOPSO-4	0.3286	0.2087	0.3306	–	**0.2893**

TABLE I

RESULTS OF INVERSE GENERATIONAL DISTANCE (IGD), SPREAD (S) AND SET COVERAGE (SC) FOR THE ZDT TEST PROBLEMS.

VII. CONCLUSIONS AND FUTURE WORK

In this work, we use the surrogate models in combination of a local search procedure based on Rough Sets to solve multi-

	Set Coverage				IGD				Spread			
Function	**S**VM+RS		**N**SGA-II		**S**VM+RS		**N**SGA-II		**S**VM+RS		**N**SGA-II	
	Mean	σ	**M**ean	σ	**M**ean	σ	**M**ean	σ	**M**ean	σ	**M**ean	σ
ZDT1	**0.0725**	0.1676	0.8755	0.1785	**0.0061**	0.0046	0.0168	0.0021	0.8308	0.1987	**0.8194**	0.0324
ZDT2	**0.0000**	0.0000	0.9690	0.0505	**0.0026**	0.0027	0.0382	0.0062	**0.5684**	0.2338	0.9711	0.0696
ZDT3	**0.0372**	0.0668	0.6453	0.3343	**0.0145**	0.0071	0.1190	0.0113	**0.8684**	0.0943	0.9475	0.0296
ZDT4	0.7937	0.2512	**0.0167**	0.0500	0.7397	0.1554	**0.1511**	0.0405	**0.9572**	0.0411	1.0795	0.1026
ZDT6	**0.0000**	0.0000	0.9208	0.1886	**0.0135**	0.0025	0.0548	0.0092	1.1000	0.1297	**0.9588**	0.0274

TABLE II

COMPARISON OF RESULTS BETWEEN SVM+RS ALGORITHM AND THE NSGA-II FOR THE FIVE TEST PROBLEMS ADOPTED (2000 EVALUATIONS).

objective problems with moderate dimensionality (between 10 and 30 decision variables). We used four different schemes to select a leader in the PSO flight equation in combination of the surrogate model to approximate the functions using supervised learning. From the initial comparison, we concluded that the PSO scheme based on selecting a leader that dominates the particle were the most appropriate model to use as the search engine, because it provides a good (but not enough) approximation of the Pareto front. The local search procedure intensify the search around those solutions that the surrogate model find in the first phase. This hybrid algorithm provides competitive results in most of the test problems adopted. We believe that our approach can be used in real-world problems in which each evaluation of the fitness function is very expensive.

As part of our future work, we are interested in refining the interaction mechanism between the surrogate method and the MOPSO, such that the interleaving of these two approaches maximizes performance. We are also interested in including another scheme to retain solutions in the training set, helping the surrogate models to get a better approximation of the real functions.

REFERENCES

[1] J. E. Alvarez-Benitez, R. M. Everson, and J. E. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In C. A. C. Coello, A. H. Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 459–473, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[2] T. Bartz-Beielstein, P. Limbourg, K. E. Parsopoulos, M. N. Vrahatis, J. Mehnen, and K. Schmitt. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1780–1787, Canberra, Australia, December 2003. IEEE Press.

[3] C. A. Coello Coello and M. Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.

[4] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[6] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, New York, 1989.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[8] A. G. Hernández-Díaz, L. V. Santana-Quintero, C. A. Coello Coello, and J. Molina. Pareto-adaptive ϵ-dominance. *Evolutionary Computation*, 15(4):493–517, Winter 2007.

[9] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[10] M. K. Karakasis and K. C. Giannakoglou. Metamodel-Assisted Multi-Objective Evolutionary Optimization. In R. Schilling, W. Haase, J. Periaux, H. Baier, and G. Bugeda, editors, *EUROGEN 2005. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, Munich, Germany, 2005.

[11] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

[12] J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, January 2006.

[13] M. McKay, R. Beckman, and W. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[14] Y. Ong, P. Nair, and A. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, 2003.

[15] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(1):341–356, Summer 1982.

[16] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical report, NeuroCOLT, September 2003.

[17] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[18] D. A. V. Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[19] I. Voutchkov and A. Keane. Multiobjective Optimization using Surrogates. In I. Parmee, editor, *Adaptive Computing in Design and Manufacture. Proceedings of the Seventh International Conference*, pages 167–175, Bristol, UK, April 2006. The Institute for People-centered Computation (IP-CC).

[20] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

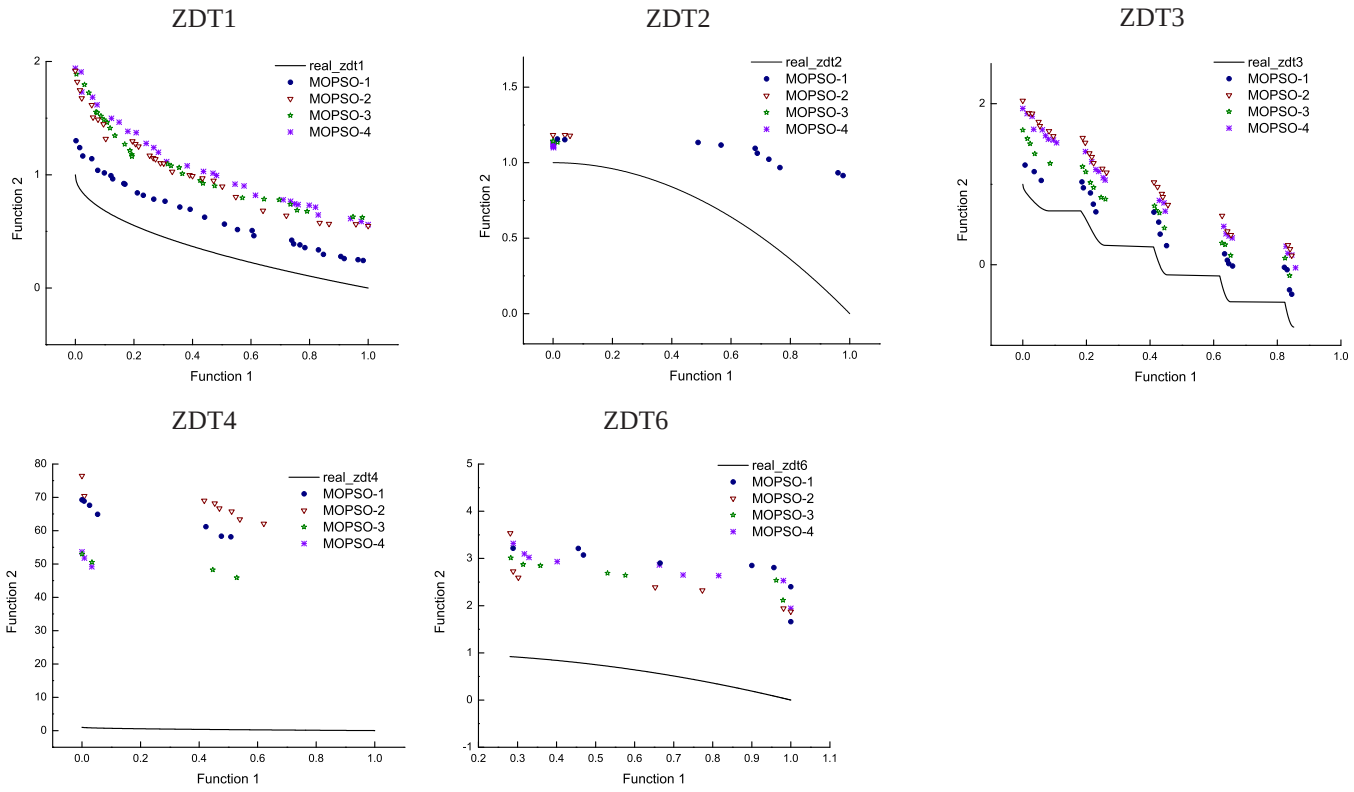

Fig. 4. Pareto fronts generated by the Surrogates methods for the ZDT test problems.

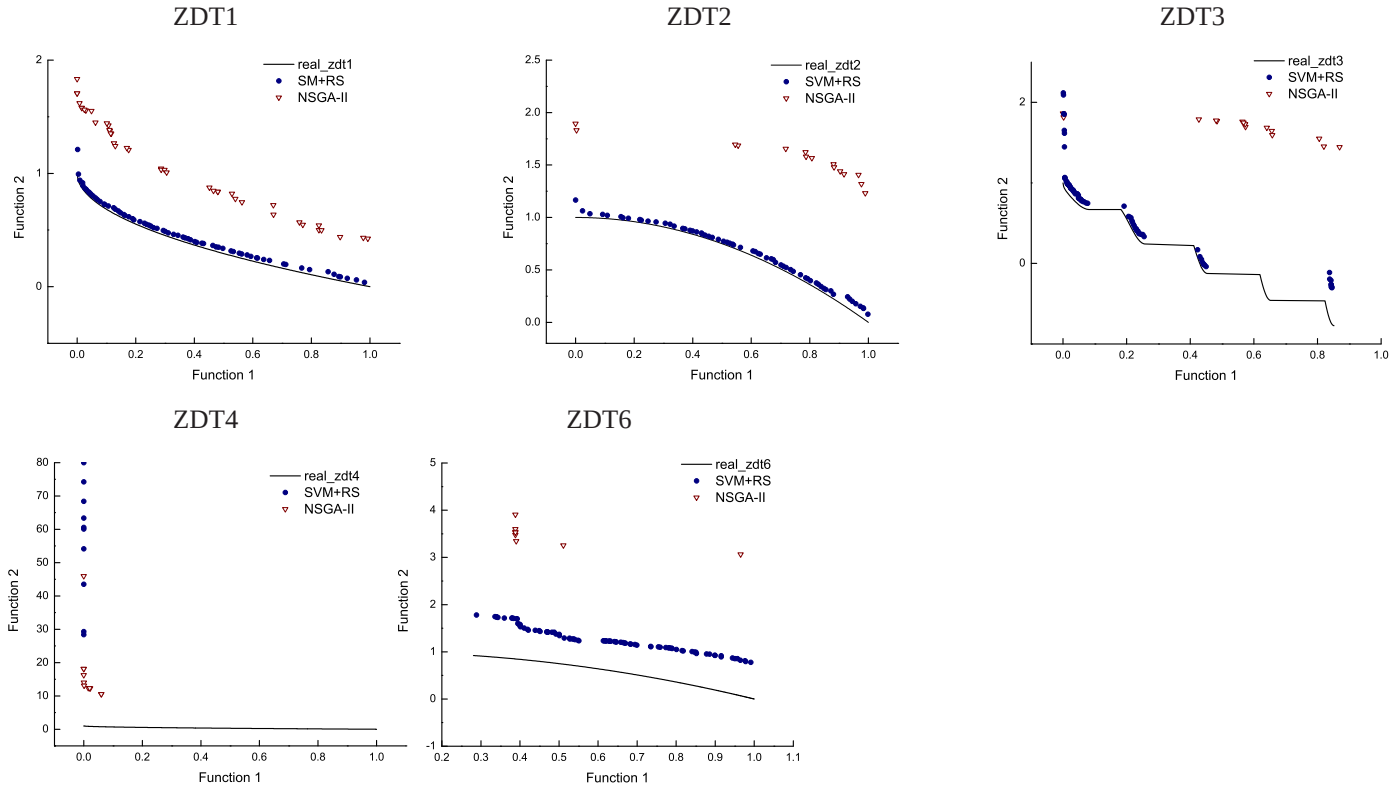

Fig. 5. Pareto fronts generated by SVM+RS and NSGA-II for the ZDT test problems.