Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

CLASSIFICATION AIDED DOMAIN REDUCTION FOR HIGH DIMENSIONAL OPTIMIZATION

Prashant Singh
Francesco Ferranti
Dirk Deschrijver
Ivo Couckuyt
Tom Dhaene

Ghent University - iMinds
Dept. of Information Technology (INTEC)
Gaston Crommenlaan 8
9050 Ghent, BELGIUM

ABSTRACT

Engineering design optimization often involves computationally expensive time consuming simulations. Although surrogate-based optimization has been used to alleviate the problem to some extent, surrogate models (like Kriging) struggle as the dimensionality of the problem increases to medium-scale. The enormity of the design space in higher dimensions (above ten) makes the search for optima challenging and time consuming. This paper proposes the use of probabilistic support vector machine classifiers to reduce the search space for optimization. The proposed technique transforms the optimization problem into a binary classification problem to differentiate between feasible (likely containing the optima) and infeasible (not likely containing the optima) regions. A model-driven sampling scheme selects batches of probably-feasible samples while reducing the search space. The result is a reduced subspace within which existing optimization algorithms can be used to find the optima. The technique is validated on analytical benchmark problems.

1 INTRODUCTION

Numerous real-world problems are high-dimensional in nature. The 'curse of dimensionality' limits the applicability of existing optimization techniques and often makes the search for optima computationally infeasible. Problems are compounded when the objective function to be optimized is expensive to evaluate.

Surrogate-based methods are often used to solve such expensive optimization problems (Forrester and Keane 2009). Surrogate based methods aim to minimize the number of evaluations of the expensive objective function by constructing a cheaper *surrogate* model that serves as a full or partial replacement of the expensive objective function or simulator. Although attractive, surrogate based optimization becomes impractical for medium to high scale (above 50 dimensions) problems. The modeling times and memory requirements needed to construct the surrogate model scale exponentially with the dimensionality of the problem (Couckuyt, Forrester, Gorissen, De Turck, and Dhaene 2012) and severely limit the applicability of surrogate based methods.

Reducing the search space is a way to relieve the curse of dimensionality for optimization problems. This can be done by either reducing the number of dimensions, or reducing the domain of each dimension. Kohavi and John (Kohavi and John 1997) discuss feature subset selection techniques for machine learning. Feature selection deals with removing irrelevant dimensions, while feature extraction deals with reducing the dimensionality of the problem by clumping multiple features or dimensions together. Feature selection

has been used in surrogate based optimization (Couckuyt, Forrester, Gorissen, De Turck, and Dhaene 2012) but surrogate based methods are still limited to low or medium-scale optimization problems due to the computational expense of building a model in high dimensional spaces which would also need a lot of samples.

This paper does not concern feature selection techniques but instead focuses on domain reduction. Spaans and Luus (Spaans and Luus 1992) illustrate the importance of domain reduction for derivative-free methods by showing that a relatively small number of function evaluations are sufficient to converge when domain reduction is applied, even if the starting point is far from the optima. Domain reduction has been applied to simulation-based optimization problems in literature (Stander and Craig 2002), (Wan, Pekny, and Reklaitis 2005). Stander and Craig (Stander and Craig 2002) construct linear models in a sub-region of the design space and iteratively contract the size of this subregion. Wan et. al. (Wan, Pekny, and Reklaitis 2005) use domain reduction in a surrogate modeling setting. They find a *promising region* by building a regression tree and focus the sampling on this region alone. The surrogate model is also only built over the promising region.

This work focusses on using classification methods to reduce the search domain for unconstrained single-objective optimization problems. Classification techniques have previously been applied to optimization problems in literature. Handoko et. al. (Handoko, Keong, and Soon 2008), (Handoko, Kwoh, and Ong 2009) proposed classification-assisted memetic algorithms for constrained optimization problems. The classifier is used to approximate the decision boundary in this case and the search for optima is concentrated in regions near this boundary (Handoko, Kwoh, and Ong 2010). Singh et. al. used adaptive classification methods to find multi-class regions in the input space by combining support vector machine (SVM) classifiers and sequential sampling strategies (Singh, Deschrijver, Pissoort, and Dhaene 2013).

The proposed Iterative Volume Reduction Algorithm (IVRA) formulates the optimization problem as a binary classification problem with classes feasible (likely to contain the optimum) and infeasible (not likely to contain the optimum). It uses a probabilistic SVM classifier to drive the modeling/sampling process towards feasible regions. Experimental results show that the algorithm is capable of significantly reducing the volume of the search space in very few sampling iterations depending upon the complexity of the problem. However, being a sampling-based approach, the algorithm can miss global optima if the size of the initial design is too small.

The paper is organized as follows. Support vector machine classifiers are discussed in Section 2. The proposed algorithm is described in Section 3. Its performance is demonstrated in Section 4 and the concluding remarks are presented in Section 5.

2 SUPPORT VECTOR MACHINE CLASSIFIERS

Support Vector Machines or SVMs have been extensively applied to various problems in literature such as cancer diagnosis, signal processing, image recognition, etc. (Wang 2005). The popularity of SVMs stems from their good generalization capability demonstrated on benchmark problems (Van Gestel, Suykens, Baesens, Viaene, Vanthienen, Dedene, De Moor, and Vandewalle 2004), (Liu, Nakashima, Sako, and Fujisawa 2003) as well as practical applications across various fields.

SVM classifiers are *supervised machine learning* classification models proposed by Vapnik (Vapnik 1979) as early as 1979, but only gained popularity after good results were obtained in digit recognition, computer vision, text classification and benchmark problems (Wang 2005) and soft margin classifier was proposed by Cortes and Vapnik (Cortes and Vapnik 1995) in 1995. SVMs have been an active research area since, and availability of implementations like LIBSVM (Chang and Lin 2011) and LS-SVMlab (Pelckmans, Suykens, Van Gestel, De Brabanter, Lukas, Hamers, De Moor, and Vandewalle 2002) has contributed to their increased use over the years.

Standard SVMs output a class label given a data point. However, in certain situations it is more desirable to have a measure of the degree to which a data point belongs to a certain class, i.e., the *probability* of the data point belonging to a certain class. Platt (Platt et al. 1999) proposed a method to obtain posterior

probabilities along with class labels for binary classification problems, and this gave birth to *Probabilistic SVMs*. The method of Wu et. al. (Wu, Lin, and Weng 2004) to obtain posterior probabilities is implemented in the LIBSVM library and is used in this work.

The classification problem can be defined as follows. We define D as a n-dimensional input space spanned by a set of features (or attributes) $\mathbf{A} = \{A_1, ..., A_n\}$. Denoting the domain of feature A_i as $dom(A_i)$, $\forall i$ if $dom(A_i) \subset \mathbb{R}$ then $D \subset \mathbb{R}^n$. $S = \{\mathbf{x_1}, ..., \mathbf{x_l} || \mathbf{y}\} \in D$ is a *training set* of l samples. Each training instance $\mathbf{x_i} \in S$ has a corresponding target value y_i, and the vector $\mathbf{y}$ consists of all target values in S. The target value $y_i \in \{-1, +1\}$ for a binary classification problem and $y_i \in [1..k]$ for a $k-$class classification problem. This work only concerns the binary formulation.

2.1 Soft margin SVMs

An SVM classifier maps the input vectors into a high-dimensional feature space Z using a specified non-linear mapping such that the input vectors are linearly-separable in Z. This linear separating hyper-surface in Z is constructed with the aim of achieving good generalization capability (Cortes and Vapnik 1995). This is done by selecting a decision boundary from candidate decision boundaries which maximizes the *margin*, or the sum of distances between the candidate decision boundary and the closest positive training instance, and between the candidate decision boundary and the closest negative training instance.

Given a *weight* vector $\mathbf{w}$ and a scalar b, the training set S is said to be linearly separable if the inequalities

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x_i} + b \geq 1 \text{ when } y_i = 1, \\ \mathbf{w} \cdot \mathbf{x_i} + b \leq -1 \text{ when } y_i = -1, \forall i \in [1..l] \end{aligned}$$

are valid for all elements of S. The inequalities can be written in the form (Cortes and Vapnik 1995):

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 \tag{1}$$

The optimal hyperplane that separates the training instances is of the form:

$$\mathbf{w_0} \cdot \mathbf{x} + b_0 = 0$$

and is unique as it has the maximal margin. The margin is $2/|\mathbf{w_0}|$ for the hyperplane with arguments ($\mathbf{w_0}$, b_0) which minimizes $\mathbf{w} \cdot \mathbf{w}$ under the constraints specified by Eq. 1.

When S is not linearly-separable, the learning task becomes minimization of:

$$\frac{1}{2}\mathbf{w^2} + CF\left(\sum_{i=1}^{l} \xi_i\right)$$

subject to the following constraints:

$$\begin{aligned} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \\ \xi_i \geq 0, \forall i \in [1..l] \end{aligned}$$

where $\xi_i \geq 0$, i = 1,...,l are non-negative slack variables which measure constraint violations, $F(\cdot)$ is a monotonic convex function and C is a constant error penalty for regularization. C-SVMs are also called *soft margin SVMs*.

The training vectors $\mathbf{x_i} \cdot \mathbf{x_j}$ are mapped onto the feature space $\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$ using a kernel function $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$. The learning now involves maximization of the Lagrangian (Chen, Bourlard, and Thiran 2001):

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i} \cdot \mathbf{x_j})$$

subject to the constraints:

$$\alpha_i \geq 0,$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0.$$

where α_i are the Lagrange multipliers. $W(\alpha)$ can be solved using quadratic programming techniques and upon finding the optimal value of α, the classification task reduces to the evaluation of the function

$$G(\mathbf{x_{test}}) = sign\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x_{test}}, \mathbf{x_i}) + b\right),$$

where $\mathbf{x_{test}}$ is the sample to be classified.

2.2 Probabilistic SVMs

Platt (Platt et al. 1999) used the sigmoid function as a probability model to directly fit $P(y=1|G)$, where G is the decision function of the two-class SVM. The probability model can be defined as:

$$P(y=1|G) = \frac{1}{1+exp(MG+N)}$$

where M and N are scalars fit by maximum likelihood estimation. Lin et. al. (Lin, Lin, and Weng 2007) proposed an improved formulation of this scheme which is implemented in LIBSVM and used for the experiments in this work.

3 ITERATIVE VOLUME REDUCTION ALGORITHM

The motivation behind IVRA is to find a smaller subregion in the input or design space which contains the optima of a high-dimensional function $f(\mathbf{x})$. An existing optimization algorithm often finds it easier to converge when applied to a smaller domain. The flowchart of IVRA can be seen in Fig. 1 and the algorithm is described in Alg. 1. The algorithm can be divided into the following four phases:

Algorithm 1 IVRA($f, s, t_r, \tau, \mathbf{A}, l_c$)

$n \leftarrow NumberOfColumns(\mathbf{A})$ {Dimensionality}
$\mathbf{x} \leftarrow I(s,n)$ {Initial Design} {Generate samples according to an initial design (e.g., Latin Hypercubes)}
$\mathbf{f_e} \leftarrow f(\mathbf{x})$ {Evaluate the function over the points obtained from the initial design}
if $\tau = NULL$ **then**
  $\tau \leftarrow (n+1)^{th}$ order statistic of $\mathbf{f_e}$ {Optima was not known beforehand/user did not specify a value}
end if
$i \leftarrow 0$
while $i \leq \lfloor \frac{t_r - s}{l_c} \rfloor$ **do**
  $\mathbf{y}_\tau \leftarrow PartitionIntoClasses(\mathbf{f_e}, \tau)$ {Values in $\mathbf{f_e} \leq \tau$ are assigned the class label +1, and the rest -1}
  $model \leftarrow PSVM - TRAIN(\mathbf{x}, \mathbf{y}_\tau)$
  $\mathbf{D}_\tau \leftarrow DomainOfFeasibleRegion(\mathbf{x}, \mathbf{y}_\tau)$ {Simply the min and max of positive samples for A_i (See Eq. 4)}
  $\mathbf{x_c} \leftarrow RandomCandidates(l_c, \mathbf{D}_\tau)$ {Generate l_c candidates within the feasible region}
  $\mathbf{p_c} \leftarrow PSVM - PREDICT(model, \mathbf{x_c})$
  $\mathbf{x_{l_c}} \leftarrow SortDecreasing(\mathbf{p_c}, \mathbf{x_c})$ {Choose top l_c points ranked according to decreasing probability}
  $\mathbf{x} \leftarrow \mathbf{x} \cup \mathbf{x_{l_c}}$
  $\mathbf{f_e} \leftarrow \mathbf{f_e} \cup f(\mathbf{x_{l_c}})$
  $\tau \leftarrow (n+1)^{th}$ order statistic of $\mathbf{f_e}$
  $i \leftarrow i+1$
end while

return $\mathbf{D_{red}} \leftarrow DomainOfFeasibleRegion(\mathbf{x}, \mathbf{y}_\tau)$

3.1 Initial Design

The algorithm begins by generating an initial design I (e.g., a Latin Hypercube) of a user-specified size s which aims at capturing as much of the design space as possible. The initial design also serves as IVRA's *exploration* component. The size s should be sufficiently large to cover the entire design space. The authors recommend setting s to at least $10*n$, where n is the dimensionality of the problem.

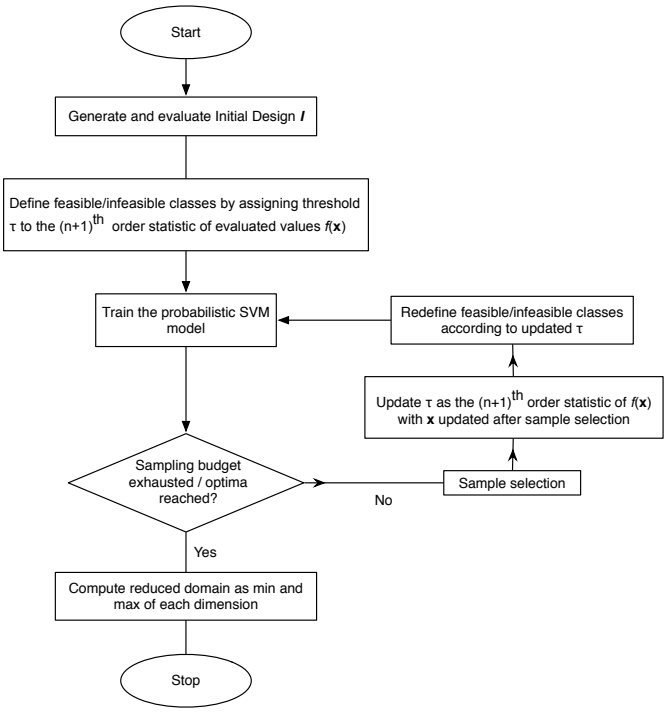

Figure 1: Flowchart of the Iterative Volume Reduction Algorithm.

3.2 Reducing Optimization to Classification

A binary classification problem is now derived from the optimization problem. The idea is to subdivide the input region into a feasible class that contains only positive samples (which correspond to the optima), and an infeasible class that contains the negative samples. Since it is highly unlikely that any of the points landed at locations corresponding to optima, all the points will correspond to the negative class.

In order to have a dataset which contains both positive and negative samples for the classification problem, a threshold value τ is introduced to distinguish between positive and negative samples. The value of τ decides how quickly IVRA reduces the domain in successive iterations. A smaller value of τ would lead to a rapid reduction at the cost of being more prone to be trapped in local minima. A larger value of τ leads to a slower reduction, although with the benefit of being less likely to get stuck in local minima. IVRA sets the threshold τ to the $(n+1)^{th}$ order statistic of values $f(\mathbf{x})$ by default, where n is the dimensionality of the problem. [1] The authors used $\tau = (n+1)^{th}$ order statistic of evaluated function values for the experiments in this paper and find this to be a good choice for a majority of problems[2].

The samples $\mathbf{x} \in I$ are evaluated and the values $f(\mathbf{x})$ are mapped to class labels $\{+1,-1\}$ based on the threshold τ resulting in a binary vector $\mathbf{y}_\tau$ with:

$$y = +1, \text{ if } f(\mathbf{x}) \leq \tau, \text{ (in case of minimization)} \tag{2}$$

$$y = -1, \text{ if } f(\mathbf{x}) > \tau. \tag{3}$$

[1] It is ensured that $min(A_i^\tau) \neq max(A_i^\tau)$, $\forall i \in D_\tau$ so that an n-dimensional hypercube can be specified.

[2] Given an initial design of s samples, intuitively it might be desirable to set $\tau = \frac{s}{2}$ to obtain a balanced training set. However, as the authors have observed, this allows for a slow reduction of the feasible region. Since the number of iterations available to the algorithm for domain reduction is limited, it is desirable to shrink the feasible region as quickly as possible. Allotting $s = (10 * n)$ samples to initial design, the value $n+1$ is roughly $\frac{s}{10}$, which was found to be a good compromise between having sufficient positive samples and achieving fast reduction

This task is performed by the routine *PartitionIntoClasses*. The process results in a training set $S = \{\mathbf{x}||\mathbf{y}_\tau\}$, which is used to build a probabilistic SVM model [3].

3.3 Domain Reduction using Adaptive Sampling

The presence of positive samples (Section 3.2) allows for a definition of a hypercube around the feasible region. This hypercube (which is likely to contain the optima) is defined by the *min* and *max* value of the attributes for each dimension.

A large number of new candidate samples $\mathbf{x_c}$ are generated randomly within this hypercube using the routine *RandomCandidates* and the probabilistic SVM model is used to predict the probability of each candidate sample belonging to the feasible class. The samples with high probability values will lie within or close to the region containing the optima. In each iteration IVRA selects a user-specified number l_c of *best* candidate samples which are the top l_c candidate samples ranked in decreasing order of probability predicted by the SVM model. These new samples are used to augment the training set S and the value of τ is updated. The probabilistic model is rebuilt after redefining the classes according to the updated value of τ. The significance of relaxing τ is that the feasible class corresponds to a region $\mathbf{A}_\tau$ in the design space which likely contains the optimum. In future iterations $\mathbf{A}_\tau$ (and the corresponding hypercube) will be progressively shrunk.

This sample selection process is IVRA's *exploitation* component since it allows the algorithm to rapidly shrink the feasible region. This process continues iteratively till the sampling budget t_r specified by the user has been exhausted. The total number of iterations performed by IVRA are $\lfloor \frac{t_r - s}{l_c} \rfloor$.

3.4 Defining the Reduced Domain

Once the sampling budget is exhausted, the final hypercube corresponding to the reduced search space D_{red} is defined as:

$$D_{red} = \begin{bmatrix} min(A_1^f) & min(A_2^f) & \cdots & min(A_n^f) \\ max(A_1^f) & max(A_2^f) & \cdots & max(A_n^f) \end{bmatrix} \tag{4}$$

where the domain of each attribute or dimension $A_i^f, i \in [1..n]$ is set to the *min* and *max* value per dimension of the samples lying within the feasible region.

D_{red} now serves as a reduced subspace of D within which any optimization algorithm can be used to search for the optimum. For the problems in this paper, optimization algorithms from the NLopt non-linear optimization library [4] were used.

3.5 Limitations

The algorithm suffers from the inherent limitations of sampling-based methods. There is always the danger of missing the region containing the optima, or getting stuck in local minima. Since the algorithm generates candidate samples only within the feasible region, it might miss the true optima in case it is far away in a region where the initial design was unable to land any samples.

Using a *large enough* space-filling initial design circumvents this problem. Since the initial design is the only component responsible for exploration, it is crucial to make sure its size s is large enough as explained in Section 3.1.

---

[3] Handling Data Bias:
As the algorithm progresses and more samples are selected, the total number of samples in the negative class $(t_r - (n+1))$ will be much larger than the number of samples in the positive class $(n+1)$. The imbalance can pose a problem during the training of the SVM model, making the classifier biased. To solve this problem, a penalty is imposed on misclassification of positive samples during the training process. The reader is referred to the section on weighted SVMs in Osuna et. al. (Osuna, Freund, and Girosi 1997).

[4] Steven G. Johnson, The NLopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt

Table 1: Experimental settings for the test problems.

Component	Number of samples
Initial Design	400
Domain Reduction using IVRA	200
Optimizer	400
Total budget	1000
Batch size for sample selection	10

4 EXPERIMENTS

The efficacy of the proposed IVRA algorithm was tested on analytical benchmark optimization problems listed in the Appendix.

The experimental settings were kept the same for all functions and are described in Table 1.The experimental setup for the benchmark problems can be seen in Fig. 2. IVRA was used to reduce the search space and thereafter optimizers were used within this reduced space to find the optima. The optimization algorithms tested were Controlled Random Search (CRS) with local mutation (Kaelo and Ali 2006) and Improved Stochastic Ranking Evolution Strategy (ISRES) (Runarsson and Yao 2005).

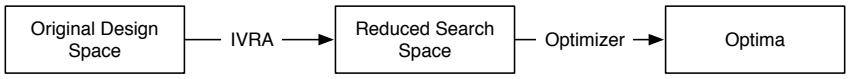

Figure 2: Experimental setup for the test problems.

The results of the experiments can be seen in Table 2. Each value in the table is an average of 20 runs and indicates the mean and standard deviation ($\mu \pm 1.96 * \sigma$). The values for the 'Optimizer Only' case correspond to running the respective optimizer with $t = 1000$ evaluations to match IVRA's experimental settings.

It can be seen that IVRA always helps the optimizer improve the best optima reached even when the function is shifted or asymmetric. The extent of improvement IVRA offers depends upon the nature of the function/simulator. In case of the Ackley function, the benefit of domain reduction was not substantial as the optimum lies in a very narrow valley. This is a perfect example of the limitations of a sampling-based approach. Since the likelihood of landing a sample in the extremely narrow valley is very low, more often than not sampling-based methods will struggle to reach the valley. The probabilistic SVM model did not have any training samples in the valley and hence was unable to assign higher probabilities to candidates lying in the valley.

Contrarily, IVRA performed better and offered substantial reduction in search space for rest of the functions which do not have the optima in a very narrow region. The volume of the feasible region itself was very small compared to the entire design space. As seen in Table 2, the reduction in search space is greater than 99.9% in all the cases.

The nature of two of the test functions can be seen in Fig. 3 and Fig. 4. The non-linearities of Ackley and Griewank functions are very pronounced near the optima. Optimization methods like surrogate-based methods which build a model of the underlying function or simulator will struggle to be accurate in light of these pronounced non-linearities, while IVRA performs well by virtue of negating the non-linearities by only considering classes - using the threshold τ. This simplifies the problem to an extent and yields substantial performance gains as can be seen in the case of the Griewank function.

Table 2 also lists IVRA's running time which hovers around approximately a quarter of a minute for 20-dimensional problems and three quarters of a minute for 50-dimensional problems. The running times of the optimizers themselves are of the order of a few (less than 3) seconds. In comparison to IVRA, surrogate-based methods like Kriging would take several hours for a single run comprising 1000 total

Table 2: IVRA's performance on benchmark analytical functions. Each value is the average of optima reached over 20 runs. The values are indicated in the form $(\mu \pm 1.96 * \sigma)$ where μ is the mean, and σ is the standard deviation over 20 runs.

Function	IVRA	Optimizer Only	$time_{IVRA}$ (s)	Reduction (%)
CRS Optimizer				
Ackley 20D	**7.22 ± 1.03**	16.63 ± 2.62	16.40 ± 1.80	> 99.9
Ellipsoid 20D	**12.49 ± 5.15**	237.79 ± 215.61	19.35 ± 1.97	> 99.9
Griewank 20D	**4.40 ± 1.25**	93.01 ± 74.28	17.45 ± 2.10	> 99.9
Rosenbrock 20D	**59.18 ± 24.97**	518.75 ± 330.51	16.60 ± 1.59	> 99.9
Shifted Griewank 20D	**−177.15 ± 1.08**	−70.73 ± 105.25	15.30 ± 1.53	> 99.9
Shifted Rosenbrock 20D	**4.08e+06 ± 9.78e+06**	2.37e+09 ± 3.68e+09	15.20 ± 1.71	> 99.9
Ackley 50D	**18.40 ± 0.55**	20.57 ± 0.29	48.65 ± 5.78	> 99.9
Ellipsoid 50D	**2244.76 ± 367.32**	6294.53 ± 976.35	46.30 ± 4.99	> 99.9
Griewank 50D	**318.47 ± 55.47**	911.21 ± 151.88	47.75 ± 2.96	> 99.9
Rosenbrock 50D	**2400.25 ± 449.13**	10657.30 ± 1355.38	43.40 ± 1.68	> 99.9
Shifted Griewank 50D	**83.04 ± 26.62**	754.18 ± 90.12	40.00 ± 2.47	> 99.9
Shifted Rosenbrock 50D	**3.89e+09 ± 1.53e+09**	4.47e+10 ± 1.11e+10	42.25 ± 2.63	> 99.9
ISRES Optimizer				
Ackley 20D	**7.17 ± 1.02**	19.26 ± 0.65	16.55 ± 1.57	> 99.9
Ellipsoid 20D	**12.63 ± 6.41**	472.29 ± 112.76	17.95 ± 1.80	> 99.9
Griewank 20D	**5.18 ± 2.02**	189.51 ± 39.02	18.05 ± 2.27	> 99.9
Rosenbrock 20D	**61.87 ± 14.78**	1341.57 ± 646.68	18.20 ± 1.58	> 99.9
Shifted Griewank 20D	**−177.26 ± 1.08**	−1.70 ± 52.39	15.25 ± 1.37	> 99.9
Shifted Rosenbrock 20D	**4.50e+06 ± 1.25e+07**	4.20e+09 ± 2.21e+09	15.40 ± 2.98	> 99.9
Ackley 50D	**18.44 ± 0.49**	20.57 ± 0.27	45.40 ± 2.51	> 99.9
Ellipsoid 50D	**2297.01 ± 384.02**	6313.49 ± 1040.49	46.80 ± 1.82	> 99.9
Griewank 50D	**323.47 ± 52.51**	907.38 ± 138.77	47.80 ± 5.92	> 99.9
Rosenbrock 50D	**2447.37 ± 529.91**	10372.90 ± 1915.99	46.90 ± 7.47	> 99.9
Shifted Griewank 50D	**80.14 ± 31.85**	714.00 ± 109.80	39.75 ± 1.84	> 99.9
Shifted Rosenbrock 50D	**3.55e+09 ± 1.36e+09**	4.48e+10 ± 7.88e+09	39.75 ± 1.72	> 99.9

function evaluations. The main overhead in case of surrogate-based methods (e.g., Kriging) is model-building, often having a complexity cubic in the number of samples, and growing exponentially with the number of dimensions. The complexity of the SVM implementation used is quadratic in the number of samples. The running time becomes even more important when the dimensionality increases and simulations are expensive. The time taken by IVRA to reduce the search space is small compared to the time taken per simulation (which might even be hours in some cases) when the simulations are expensive.

For the purpose of experiments, no parameter optimization was performed for IVRA. Parameters such as the size s of the initial design I, the number of function evaluations given to IVRA and the optimizer can be optimized to obtain better results. Additionally, the choice of the optimizer also affects the speed of convergence. Table 3 lists the results of experiments in which the most appropriate optimizer was used in conjunction with IVRA. The Ellipsoid function is quadratic and hence the BOBYQA (Bound Optimization BY Quadratic Approximation) (Powell 2009) algorithm was chosen for the function as is performs quadratic approximation of the objective function. The Griewank function has quadratic and cosine terms, so the

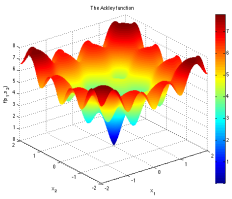

Figure 3: Ackley function in 2D within the range [-2, 2].

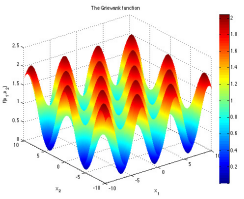

Figure 4: Griewank function in 2D within the range [-10,10].

Table 3: IVRA's performance when the most appropriate optimizer is chosen for each benchmark function. Each value is the average of optima reached over 20 runs.

Function	Best	Worst	Average	Optimizer Used
Ackley 20D	$3.43e-2$	3.79	0.95	DIRECT
Ellipsoid 20D	$2.65e-17$	1.51	0.24	BOBYQA
Griewank 20D	$3.54e-06$	1.67	0.47	BOBYQA
Rosenbrock 20D	16.72	36.31	23.25	DIRECT
Shifted Griewank 20D	-179.39	-177.84	-178.64	BOBYQA
Shifted Rosenbrock 20D	$4.25e+4$	$1.96e+07$	$1.99e+06$	DIRECT
Ackley 50D	4.78	6.04	5.58	DIRECT
Ellipsoid 50D	$3.02e-13$	0.33	0.03	BOBYQA
Griewank 50D	0.02	15.34	1.31	BOBYQA
Rosenbrock 50D	72.62	147.07	106.66	DIRECT
Shifted Griewank 50D	-179.98	-178.98	-179.54	BOBYQA
Shifted Rosenbrock 50D	$4.75e+5$	$1.52e+06$	$9.27e+5$	DIRECT

BOBYQA algorithm was appropriate. The Ackley function is Gaussian while the Rosenbrock function has low function values spanning a vast region in the input domain. Since no algorithm available to the authors was particularly more suitable, the DIRECT (DIviding RECTangles) algorithm (Gablonsky and Kelley 2001) was chosen which is a good general purpose optimizer.

The DIRECT algorithm was not used for comparisons in Table 2 since it divides dimensions in halves in subsequent iterations which would make the comparison unfair, since the benchmark functions have symmetric domains and the optima lies at the origin.

The results show the potential gain offered by IVRA, which can be further enhanced by hyper-parameter optimization and choosing an appropriate optimizer according to the nature of the problem at hand (or a good general purpose optimizer). As a pointer, IVRA is inappropriate when the simulation budget is very limited (e.g., less than $n*10$) since it will not have enough sampling iterations to reduce the search space. Allocating the entire budget to the optimizer is advisable in such cases.

5 CONCLUDING REMARKS

A novel fast algorithm for solution of expensive high-dimensional optimization problems is presented in this work. The algorithm reduces the search space by transforming the optimization problem into a binary classification problem which is modeled using probabilistic support vector machines (PSVMs). Existing optimization algorithms can then be applied to the reduced search space to find the optima quickly. The efficacy of the proposed algorithm is demonstrated on analytical benchmark examples with comparisons to optimization algorithms used with and without the aid of the proposed algorithm.

ACKNOWLEDGMENTS

This research has been funded by the Interuniversity Attraction Poles Programme BESTCOM initiated by the Belgian Science Policy Office, and the Fund for Scientific Research in Flanders (FWO-Vlaanderen). Francesco Ferranti, Dirk Deschrijver, and Ivo Couckuyt are post-doctoral research fellows of FWO-Vlaanderen.

6 APPENDIX

The following benchmark test functions were used for the experiments in this paper.

6.1 Ackley Function

$$\underset{x}{\text{Minimize}} \quad -20 \cdot exp(-0.2 \cdot \sqrt{\frac{1}{d} \cdot \sum_{i=1}^{d} x_i^2}) - exp(\frac{1}{d} \cdot \sum_{i=1}^{d} cos(2\pi \cdot x_i)) + 20 + e$$

for

$$-32.768 \leq x_i \leq 32.768 \ \forall i,$$
$$e = 2.7183,$$
$$d \in \{20, 50\},$$
$$min : x^* = 0.$$

6.2 Ellipsoid Function

$$\underset{x}{\text{Minimize}} \quad \sum_{i=1}^{d} i x_i^2$$

for

$$-5.12 \leq x_i \leq 5.12 \ \forall i,$$
$$d \in \{20, 50\}$$
$$min : x^* = 0.$$

6.3 Griewank Function

$$\underset{x}{\text{Minimize}} \quad 1 + \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right)$$

for

$$-600 \leq x_i \leq 600 \ \forall i,$$
$$d \in \{20, 50\},$$
$$min : x^* = 0.$$

6.4 Rosenbrock Function

$$\underset{x}{\text{Minimize}} \quad \sum_{i=1}^{d} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$$

for

$$-2.048 \leq x_i \leq 2.048 \ \forall i,$$
$$d \in \{20, 50\},$$
$$min : x^* = 0.$$

6.5 Shifted Griewank Function

$$\underset{x}{\text{Minimize}} \quad 1 + \sum_{i=1}^{d} \frac{z_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{z_i}{\sqrt{i}}\right) + f_{bias}$$

with

$$\mathbf{z} = \mathbf{x} - \Delta,$$

for

$$-600 \leq x_i \leq 600 \ \forall i,$$
$$d \in \{20, 50\},$$
$$\Delta = \{5, 5, ..., 5\}_d \text{ (the shifted optimum)}$$
$$f_{bias} = -180,$$
$$min : x^* = f_{bias}.$$

6.6 Shifted Rosenbrock Function

$$\underset{x}{\text{Minimize}} \quad \sum_{i=1}^{d-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$$

with

$$\mathbf{z} = \mathbf{x} - \Delta + 1,$$

for

$$-100 \leq x_i \leq 100 \ \forall i,$$
$$d \in \{20, 50\},$$
$$\Delta = \{5, 5, ..., 5\}_d \text{ (the shifted optimum)}$$
$$f_{bias} = 390,$$
$$min : x^* = f_{bias}.$$

REFERENCES

Chang, C.-C., and C.-J. Lin. 2011. "LIBSVM: A Library for Support Vector Machines". *ACM Transactions on Intelligent Systems and Technology* 2 (3): 27.

Chen, D., H. Bourlard, and J.-P. Thiran. 2001. "Text Identification in Complex Background using SVM". In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 621–626.

Cortes, C., and V. Vapnik. 1995. "Support-vector Networks". *Machine learning* 20 (3): 273–297.

Couckuyt, I., A. Forrester, D. Gorissen, F. De Turck, and T. Dhaene. 2012. "Blind Kriging: Implementation and Performance Analysis". *Advances in Engineering Software* 49:1–13.

Forrester, A. I., and A. J. Keane. 2009. "Recent Advances in Surrogate-based Optimization". *Progress in Aerospace Sciences* 45 (1): 50–79.

Gablonsky, J. M., and C. T. Kelley. 2001. "A Locally-biased Form of the DIRECT Algorithm". *Journal of Global Optimization* 21 (1): 27–37.

Handoko, S., K. C. Keong, and O. Y. Soon. 2008, Oct. "Using Classification for Constrained Memetic Algorithm: A New Paradigm". In *IEEE International Conference on Systems, Man and Cybernetics*, 547–552.

Handoko, S., C. K. Kwoh, and Y.-S. Ong. 2010, Oct. "Feasibility Structure Modeling: An Effective Chaperone for Constrained Memetic Algorithms". *IEEE Transactions on Evolutionary Computation* 14 (5): 740–758.

Handoko, S. D., C. K. Kwoh, and Y. S. Ong. 2009. "Classification-Assisted Memetic Algorithms for Equality-Constrained Optimization Problems". In *AI 2009: Advances in Artificial Intelligence*, edited by A. Nicholson and X. Li, Volume 5866 of *Lecture Notes in Computer Science*, 391–400. Springer Berlin Heidelberg.

Kaelo, P., and M. Ali. 2006. "Some Variants of the Controlled Random Search Algorithm for Global Optimization". *Journal of optimization theory and applications* 130 (2): 253–264.

Kohavi, R., and G. H. John. 1997. "Wrappers for Feature Subset Selection". *Artificial Intelligence* 97 (1): 273–324.

Lin, H.-T., C.-J. Lin, and R. C. Weng. 2007. "A Note on Platt's Probabilistic Outputs for Support Vector Machines". *Machine learning* 68 (3): 267–276.

Liu, C.-L., K. Nakashima, H. Sako, and H. Fujisawa. 2003. "Handwritten Digit Recognition: Benchmarking of State-of-the-art Techniques". *Pattern Recognition* 36 (10): 2271–2285.

Osuna, E., R. Freund, and F. Girosi. 1997. "Support Vector Machines: Training and Applications". *AI Memo 1602*.

Pelckmans, K., J. A. Suykens, T. Van Gestel, J. De Brabanter, L. Lukas, B. Hamers, B. De Moor, and J. Vandewalle. 2002. "LS-SVMlab: A Matlab/c Toolbox for Least Squares Support Vector Machines". *Tutorial. KULeuven-ESAT. Leuven, Belgium*.

Platt, J. et al. 1999. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". *Advances in Large Margin Classifiers* 10 (3): 61–74.

Powell, M. J. 2009. "The BOBYQA Algorithm for Bound Constrained Optimization without Derivatives". *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*.

Runarsson, T., and X. Yao. 2005. "Search Biases in Constrained Evolutionary Optimization". *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 35 (2): 233–243.

Singh, P., D. Deschrijver, D. Pissoort, and T. Dhaene. 2013, November. "Adaptive Classification Algorithm for EMC-compliance Testing of Electronic Devices". *Electronics Letters* 49 (24): 1526–1528.

Spaans, R., and R. Luus. 1992. "Importance of Search-domain Reduction in Random Optimization". *Journal of Optimization Theory and Applications* 75 (3): 635–638.

Stander, N., and K. Craig. 2002. "On the Robustness of a Simple Domain Reduction Scheme for Simulation-based Optimization". *Engineering Computations* 19 (4): 431–450.

Van Gestel, T., J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. 2004. "Benchmarking Least Squares Support Vector Machine Classifiers". *Machine Learning* 54 (1): 5–32.

Vapnik, V. N. 1979. *Estimation of Dependencies Based on Empirical Data*. Moscow, Russia: Nauka. English translation, New York: Springer-Verlag, 1982.

Wan, X., J. F. Pekny, and G. V. Reklaitis. 2005. "Simulation-based Optimization with Surrogate Models—Application to Supply Chain Management". *Computers & Chemical Engineering* 29 (6): 1317–1328.

Wang, L. 2005. *Support Vector Machines: Theory and Applications*. Springer.

Wu, T.-F., C.-J. Lin, and R. C. Weng. 2004. "Probability Estimates for Multi-class Classification by Pairwise Coupling". *The Journal of Machine Learning Research* 5:975–1005.

AUTHOR BIOGRAPHIES

PRASHANT SINGH received his B.Sc. (H) and M.Sc. degrees in Computer Science from the University of Delhi, India in 2009 and 2011 respectively. From July 2011 till September 2012, he worked as a Software Engineer at Nagarro Software. Since October 2012, he is active as a Ph.D. student in the research group Internet Based Communication Networks and Services (IBCN) at Ghent University. His research interests include surrogate modeling, machine learning, scientific computing and high performance computing. His email address is prashant.singh@ugent.be.

FRANCESCO FERRANTI received the B.S. degree (summa cum laude) in electronic engineering from the Università degli Studi di Palermo, Palermo, Italy, the M.S. degree (summa cum laude with honors) in electronic engineering from the Università degli Studi dell'Aquila, L'Aquila, Italy, and the Ph.D. degree in electrical engineering from Ghent University, Ghent, Belgium, in 2005, 2007, and 2011, respectively. He is currently an FWO Post-Doctoral Research Fellow with the Department of Information Technology (INTEC), Ghent University, Ghent. His current research interests include parametric macromodeling, parameterized model order reduction, electromagnetic compatibility, signal integrity, numerical modeling, optimization, stochastic modeling, and system identification. His email address is francesco.ferranti@ugent.be.

DIRK DESCHRIJVER was born in Tielt, Belgium, on September 26, 1981. He received the Masters degree (licentiaat) and the Ph.D. degree in computer science from the University of Antwerp, Antwerp, Belgium, in 2003 and 2007, respectively. He was a Marie Curie Fellow with the Scientific Computing Group, Eindhoven University of Technology, Eindhoven, The Netherlands, from May to October 2005. He is currently an FWO Post Doctoral Research Fellow with the Department of Information Technology, Ghent University, Ghent, Belgium. His current research interests include robust parametric macromodeling, rational least-squares approximation, orthonormal rational function, system identification, and broadband macromodeling techniques. His email address is dirk.deschrijver@ugent.be.

IVO COUCKUYT received his M.Sc. degree in Computer Science from the University of Antwerp (UA) in 2007. In October 2007 he joined the research group Computer Modeling and Simulation (COMS) (now merged with CoMP), supported by a research project of the Fund for Scientific Research Flanders (FWO-Vlaanderen). Since January 2009 he is active as a PhD student in the research group Internet Based Communication Networks and Services (IBCN) at Ghent University, where he obtained the PhD degree in 2013. He is currently working as an FWO post-doctoral research fellow in the IBCN research group of the Department of Information Technology (INTEC) in the Faculty of Engineering at Ghent University, Belgium. His research interests include global and local surrogate modeling (metamodeling), optimization of expensive functions, evolutionary computing (Genetic Algorithms, Evolutionary Strategies,...), etc. His email address is ivo.couckuyt@ugent.be

TOM DHAENE received the Ph.D. degree in electrotechnical engineering from Ghent University, Ghent, Belgium, in 1993. He was a Research Assistant with the Department of Information Technology, Ghent University, from 1989 to 1993, where his research focused on different aspects of full-wave electromagnetic circuit modeling, transient simulation, and timedomain characterization of high-frequency and highspeed interconnections. He joined the EDA Company Alphabit (now part of Agilent) in 1993. He was one of the key developers of the planar EM simulator ADS Momentum, ADS Model Composer, and ADS Broadband SPICE. Since 2007, he has been a Full Professor with the Department of Information Technology, Ghent University - Institute for Broadband Technology. He has authored or coauthored more than 250 peer-reviewed papers and abstracts in international conference proceedings, journals, and books. He holds five U.S. patents. His email address is tom.dhaene@ugent.be.