Improving Global Numerical Optimization using a Search-space Reduction Algorithm

Vinícius V. de Melo
University of São Paulo
São Carlos - SP, Brazil
vmelo@icmc.usp.br

Alexandre C. B. Delbem
University of São Paulo
São Carlos - SP, Brazil
acbd@icmc.usp.br

Dorival L. P. Júnior
University of São Paulo
São Carlos - SP, Brazil
leao@icmc.usp.br

Fernando M. Federson
São Carlos - SP, Brazil
federson@gmail.com

ABSTRACT

We have developed an algorithm for reduction of search-space, called Domain Optimization Algorithm (DOA), applied to global optimization. This approach can efficiently eliminate search-space regions with low probability of containing a global optimum. DOA basically works using simple models for search-space regions to identify and eliminate non-promising regions. The proposed approach has shown relevant results for tests using hard benchmark functions.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, Search-Heuristic methods

General Terms

Algorithms

Keywords

Search-space Reduction, Heuristics, Metaheuristics, Optimization

1. INTRODUCTION

Several real-world problems can be modeled as global optimization problems, which are common in fields such as engineering and science. In general, these problems cannot be efficiently solved by deterministic techniques. As a consequence, probabilistic and non-deterministic algorithms have been largely used for global optimization. Two of the main difficulties in solving these problems are to escape from local optima and to prevent premature convergence of the algorithm. As the problem complexity increases, due to a great number of variables or local optima, the possibility of the algorithm finding a global optimum drastically decreases. Thus, the resolution of these problems requires efficient and robust search techniques.

One frequent search strategy to deal with complex optimization problems have been the exploration of scattered points in the solution space. As there is no information about a global optimum location before solving an optimization problem, algorithms based on such strategy can evenly scan a feasible region of the search-space to determine good solutions (points) for better exploration in subsequent iterations. As the algorithm iterates improving a population (set) of solutions, a subset of solutions in general move closer to a global optimum.

In order to reduce the running time of optimization algorithms, advanced metaheuristics have been developed. For example, these approaches can emphasize promising regions of the solution space obtaining better performance. Metaheuristics using complex probabilistic models to determine promising regions have shown relatively high capacity to find optimum solutions. On the other hand, such strategies are computationally less efficient due to the complexity of the probabilistic models employed.

The main difficulties in using metaheuristic algorithms are: inadequate initialization of the population, bad configuration of the algorithm, poor operators to determine the next population and guide the search, etc. In the Genetic Algorithm (GA)[8], the crossover operator generates new points (children) based on selected points (parents) which were previously sampled from the search-space. Different crossover implementations try to help the creation of better children solutions. The similarity between GAs and Design of Experiments techniques (DoE) [18], first presented by Reeves and Wright [23, 22], guided some researches to incorporate DoE into the crossover operator, resulting in a more robust and statistically sound one, improving the solution quality of a GA.

Lung and Zhang [16, 28] developed a crossover operator using a DoE method called Orthogonal Design. In the Orthogonal Crossover (OC), the values of the parents' chromosome genes are grouped into an orthogonal array. However, instead of considering the full combination of the parents' genes, the orthogonal array stores a small but representative sample of combinations for testing the resulting children. As the optimal combination may not be included in the orthogonal array, it may not be applicable for parametrical problems.

Chan, Aydin and Fogarty tried another DoE technique: the Taguchi method. It is an efficient DoE approach that can be used in simultaneous modeling, evaluation and optimization of operating conditions in complex systems [3]. Using this method, the potential drawback in OC can be overcame because combinations not

GECCO'07 , July 7-11, 2007, London, England, United Kingdom

included in the orthogonal array can be considered. Based on this, they developed the Taguchi Crossover operator (TC) in which the children are formed from the best combinations of the genes with the best main effect (the gene with the greatest influence in the response variable). Experimental results show that TC outperforms, in solution quality, OC and classical crossover operators on a set of global numeric optimization benchmark problems [3].

Estimation of Distribution Algorithms (EDAs) [20], also known as probabilistic model-building evolutionary algorithms, have attracted increasing attention by researchers during the last few years. EDAs incorporate methods for automated learning of correlations among variables of the encoded problem solutions. The process of sampling new individuals from a probabilistic model, that is, the creation of the offspring based on a model obtained by a set of solutions, respects these mutual dependencies such that disruption of important building blocks is avoided, in comparison with classical recombination operators. Using these operators there is no need to specify certain EA parameters. The complex probabilistic model used to guide the search through the offspring sampling is one of the main drawbacks of this method, due to the process of statistical information extraction and manipulation [11].

We propose a search strategy using simple models (linear) [18] to determine non-promising regions. This approach is iteratively applied to eliminate regions where the algorithm guarantees, with high probability, that there is no global optimum. The algorithm finishes if it cannot guarantee that an additional reduction will not lose the optimum solution. Our approach, a Domain Optimization Algorithm (DOA) belongs to a class called search-space reduction algorithms (SRAs) [24, 4]. A SRA is a technique to determine one or more promising regions before the use of an optimization algorithm, instead of detecting them during the optimization process. This way, one can concentrate the population into the limits defined by the SRA, where there is a high probability of finding the global optimum.

As the proposed technique is a pre-optimization process, the objective of this paper is to show how a search-space reduction using simple linear models can help global optimization algorithms. The results obtained with the optimization algorithms tested in this work dont't intend to be better than results obtained by state-of-art algorithms, but to be better than a simple random initialization or, at least, to obtain similar results.

This paper is presented as follows. In section two, the DoE methodology of Multiple Linear Regression is introduced. In section 3, the Domain Optimization Algorithm (DOA) is presented and briefly commented. In section 4, the applicability of our proposed SRA is demonstrated using some global optimization benchmark functions. Next, we finalize this paper with some conclusions and future work.

2. DESIGN OF EXPERIMENTS

The objective of statistical methods is to make this process the most efficient possible and is exactly what optimization algorithms, like GAs, need. There is an interesting relationship between GAs and an area of statistics known as Design of Experiments (DoE) [1, 5, 7, 12, 18, 27]. Reeves and Wright [22, 23] show details about this relationship, and some of the main differences between them are presented in Table 1.

Thus, it is reasonable to think that one ideal algorithm should be capable of automating the decisions that must be taken in a typical DoE to guide the search process (using a GA, for example) to evaluate the next best point, chosen according to the whole information available. This strategy was adopted in the Orthogonal Crossover [16] and Taguchi Crossover [3] operators, with great success.

In this work, we use the information obtained by a DoE to select a limited region of the search-space where the best point (global optimum) can be located, called a promising region. Concentrating the search in this promising region, one can expect a reduction of the number of evaluated points needed to find the global optimum. To select the promising region, our approach uses the Multiple Linear Regression technique, explained in the next subsection.

2.1 Multiple Linear Regression

Multiple Regression is a set of statistical methods used to build mathematical models when one intends to study the behavior of a response variable according to one or more explicative/predictive variables of a determined process. As such model is subtle to errors, it is common to accept models with reasonable precision. The regression analysis is a technique applied in several fields of knowledge [17]. The main objectives of the regression analysis can be summarized in the following stages:

1. **Choice of variables:** in general, the set of variables which affects the response variation *y* is unknown. To highlight this question, initial studies are made with a big number of variables. The regression analysis aids in the process of selection of variables, eliminating (in the final stage of the process) the ones which contribution to the model explanation isn't significant;

2. **Parameters estimation:** from a set of observations (sample of size *n*) and a model related to the response and predictive variables, one applies some process to fit a model (linear or non-linear) to the data by obtaining values (estimates) to the model parameters;

3. **Inference:** the fitting of a regression model allows to make inferences like hypothesis tests of the model reliability and confidence intervals to the prediction;

4. **Prediction:** with the regression analysis, one expects that the response variation *y* be explained by the fitting of a model to the predictive variables *x*, with a reasonable quality. Therefore, one can use this model to predict *y* values corresponding to *x* values not sampled. In general, *x* values that are inside the sampling region are used. Values outside this interval are called extrapolation and the prediction of these values with the fitted model will probably be of low confidence.

The *multiple linear regression* tries to fit a straight line (linear model) and to identify how *k* variables contribute to make the observed phenomenon. The functional part of this method is a linear function

$$y = f(x_1, x_2, ..., x_k) + \varepsilon, \tag{1}$$

such that

$$f(x_1, x_2, ..., x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k. \tag{2}$$

Thus, f is an unknown function which relates the x variables with the y response. β_0 describes the interception of the hyper-plan with the k+1 axis and ε is the error (associated to another unknown function which relates other variables not included in the model, which can or not be known). As the sample set has n observations, one can rewrite the equation 1 as

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_k x_{ki} + \varepsilon_i, \tag{3}$$

for *i=1,...,n*.

Table 1: Some differences between GAs and DoEs

Genetic Algorithms	Design of Experiments
Started with random subsets of the solution universe.	Uses a carefully structured subset.
Need a big set of points.	Try to extract information using a minimum set of points.
Doesn't remember, explicitly, previous results.	Incorporate the discovered information in a cumulative manner.
Can be automatically executed.	Need human interaction and interpretation to take decisions
Make no explict use of modelling concepts.	Has an explicity model with which it attempts to account the observed phenomena.

Table 2: Data input to the multiple linear regression

X				Y
x_1	x_2	$\cdots$	x_k	y
x_{11}	x_{21}	$\cdots$	x_{k1}	y_1
x_{12}	x_{22}	$\cdots$	x_{k2}	y_2
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
x_{1n}	x_{2n}	$\cdots$	x_{kn}	y_n

Table 2 presents the data input to the response variable *y* and its respective explicative variables $(x_1, x_2, ..., x_k)$. Separately, one has an *X* matrix, which corresponds to the explicative variable's observations, and an *Y* vector, which contains the set of the observed values of the sample's response variable.

This model describes a hyper-plan with *k* spatial dimensions, according to the *k* explicative variables. The estimatives $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_k$ of the coefficients $\beta_0, \beta_1, , ..., \beta_k$ can be calculated by the Minimum Square Method [12, 17]. Such method is one of the most used to estimate the regression parameters and it's purpose is to minimize the differences between the observed and predicted values [9].

2.1.1 Minimum Square Method

One assume that the errors (ε_i, *i* = 1, ..., *n*) are independent and identically distributed with zero mean and σ^2 variance. Supposing that the relation among the variables is satisfactory, one can estimate the regression model and solve some related inference problems.

The minimum square method is an efficient strategy to estimate the regression parameters. The sum of the squared errors ε_i in the equation 3 can be minimized, and the $\hat{\beta}s$ values can be estimated using the following equation:

$$\hat{\beta} = (X'X)^{-1}(X'Y), \tag{4}$$

such that *Y* is a vector *n* * 1, *X* is a matrix *n* * $(k + 1)$ (the column zero has only values equal to 1, the intercept) and $\hat{\beta}$ is the vector $(k + 1)$ * 1 of the coefficients estimated by the equation.

2.1.2 Regression Hypothesis Tests

In the regression analysis, it is important to evaluate how much the relation between the response value (*y*) and the explicative variables (*x*) can be considered significant. Hypothesis tests are usable to verify which the significant parameters of the model are.

One suppose that the errors ε_i have a normal distribution and are independent with zero mean and unit variance. The y_i variables also have normal distribution and are independent, but with $\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}$ mean and σ^2 variance.

To achieve good predictions using the model found by the regression, it is interesting that the model contains only significant parameters, that is, the ones which have some influence upon the response. The choice of these parameter is made applying formal tests which determine the significance of each estimated coefficient, as presented in [19].

2.1.3 Significance Tests for Regression

This test verifies if there is any influence of the explicative variables upon the response variable. The hypothesis are:

$$H_0 : \beta_1 = ... = \beta_k = 0$$

$$H_1 : \beta_j \neq 0, \text{ for at least one } j$$

To reject H_0 means that at least one of the independent variables contributes significantly to the model. The null hypothesis can be tested by an analysis of variance (ANOVA) [18]. The test procedure is presented in the equation 5:

$$SS_T = \sum_{i=1}^{n} (y_i - \bar{y})^2 = \underbrace{\sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2}_{SS_R} + \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{SS_E} \tag{5}$$

such that:

$\bar{y}$ is the arithmetic mean of the *n* observations (sample) of *Y*;

y_i is the output value of the observation *i*;

$\hat{y}_i$ is the response predicted by the model for the observation *i*;

SS_R is the amount of the $\hat{y}_i$ variation due to the regression mean (Sum Square of Regression);

SS_E is the Sum Square of Errors, that measures the amount of variation not explained by the regression;

There is yet that SS_R and SS_E are independent and their respective Mean Squared are given by $MS_R = \frac{SS_R}{k}$ and $MS_E = \frac{SS_E}{n-k}$. The quotient $F_0 = \frac{MS_R}{MS_E}$ follows the distribution $F_{(k,n-k-1)}(1 - \alpha)$, being α the test significance level, which represents the probability of rejecting H_0, being H_0 true [18]. Thus, one rejects H_0 if F_0 is bigger than $F_{(k,n-k-1)}(1 - \alpha)$.

Yet, one can calculate the p-value, which is the probability of $F_{(k,n-k-1)}(1 - \alpha) > F_0$. If the p-value is less than α, one rejects H_0. The standard value for α is 5%.

2.1.4 Tests for each Coefficient

The hypothesis to test the significance of the coefficient $\hat{\beta}_j$ are

$$H_0 : \beta_j = 0$$

$$H_1 : \beta_j \neq 0, \; j = 0, ..., k.$$

If H_0 isn't rejected, one has the indication that the variable x_j doesn't need to be included in the model, because the statistics shows that this variable is little significant. The test statistics for this hypothesis is:

$$t_0 = \frac{\hat{\beta}_j}{\sqrt{\sigma^2 C_{jj}}} \sim t_{(n-k-1)}(1 - \alpha/2), \tag{6}$$

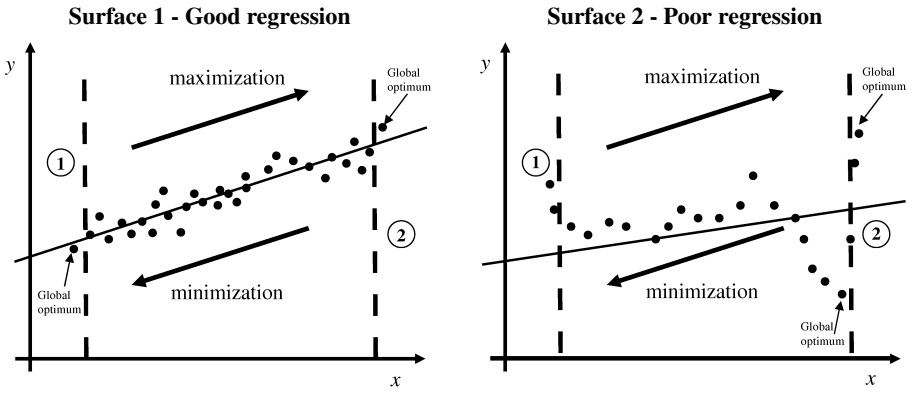

Figure 1: DOA using a linear model

such that C_{jj} is an element of the matrix $C = (X'X)^{-1}$. As σ^2 is unknown, one uses the MS_E value. The null hypothesis H_0 is rejected if $|t_0| > t_{(n-k-1)}(1 - \alpha/2)$. Significant values for t_0 must be bigger than 2.

In the next section, the use of linear regression models to locate promising regions is presented.

3. SEARCH-SPACE REDUCTION

In this work, we present a different approach to use DoE. Instead of elaborating complex probabilistic models to determine promising regions in the search-space, we apply a simple DoE method (Multiple Linear Regression) to, starting from the search-space limits, iteratively eliminate portions of the search-space where the algorithm guarantees, with a high probability, that there is no global optimum. The algorithm ends when it cannot guarantee a further reduction. Our approach, a Domain Optimization Algorithm (DOA) is a Search-space Reduction Algorithm (SRAs) [24, 4, 10]. A SRA is a technique to determine one or more promising regions before the use of an optimization algorithm, instead of detecting one region during the global optimization process. This way, one can, for instance, concentrate the initial population into the limits defined by the SRA, where there is a high probability to find the global optimum. The DOA is presented to follow.

- Domain Optimization Algorithm

Domain Optimization Algorithms (DOAs) have as objective to aid the global optimization algorithms by indicating the initial search-space areas (domain) with larger possibility of finding the global optimum. Such algorithms can be used to reduce the initial search-space, excluding areas considered unfavorable. This way, DOAs try to increase the efficiency of global optimization algorithms, and can be essential to obtain satisfactory results in situations where the execution of a great number of experiments is unviable.

Aiming efficiency and precision, DOAs must be generated in a simple way so that they can be quickly tested. More sophisticated algorithms are generated according to the need of each problem. Box [1] says that "all models are wrong, but some are useful". Following that idea, the DOA proposed in this work uses a model which guarantees, with high probability, that the global optimum *is not* in the discarded area of the search-space. Some stochastic-search techniques generate highly complex models of the solution space, trying to guide the search [14, 15, 21, 26]. The approach proposed in this work uses the opposite strategy, where a simple model (linear) is employed. If that model presents an appropriate fitting, one can take the decision of eliminating the area which appears to be farther from the global optimum. Two examples can be seen in the Figure 1.

Analyzing the sampled points, it is possible to notice that, in spite of being noisy, the Surface 1 can be appropriately explained by a linear model. That means that the values predicted by the model, represented by the straight line, aren't very different from the sampled values (points). This means that the model possesses a relatively low error. This way, the decision of eliminating an area can be taken with a considerable confidence. In a minimization problem, the cut (elimination of an area of the search-space) is accomplished in the superior limit (area 2), because the global optimum tends to be in the inferior part on the left side. Otherwise, the inferior limit is cut (area 1). The *slice size* can be defined by the user or estimated from a small sampling in the area to be eliminated (area 1 or 2). That small sampling tries to give larger safety in the cut, avoiding the elimination of a big area which could contain the global optimum.

On the other hand, the Surface 2 could not be well explained. It is possible to notice that the minimization and maximization global optima are at the same side. A cut to minimize (area 2) unfortunately would lose the optimum. The same would not happen if one wanted to maximize the function. However, in area 1 there is a point which *y* value is very close to that of the maximization global optimum, in other words, this point is a local optimum. Optimization algorithms often get stuck in local optima. After area 1 elimination, this point isn't part of the promising area anymore.

The basic algorithm of our approach is presented in Figure 2. It works basically as follows: First, the algorithm generates some random points (X matrix) inside the user defined search-space and evaluates them using an evaluation function (Y vector). Then, the algorithm uses the linear regression technique to build a linear model using the generated XY matrix. If it is a good model, the algorithm selects the more significant variables of the model, identifies the direction to do the reduction, determine the slice size, and eliminates one small region. If not, it samples new points and try again for a maximum of 10 trials. If it fails, the algorithm increases the STOP criterion and reduces the search-space in both sides of each variable just a little and restart the loop until the STOP criterion is met.

By optimizing the initial search-space, the initialization of the points can be concentrated in a promising region. Thus, it tries to avoid the creation of points distant from the global optimum, and to guarantee a good sampling near the optimum. The population is initialized, for example, in the vertexes and in the central point

of the promising region (and around these points). Figure 3 is a 2D example of this kind of initialization.

The *slice size* in the basic algorithm defines the DOA's flexibility. A small size means that the algorithm will make small reductions and can stop prematurely. Bigger sizes can make fast and extreme reductions and, sometimes, get very close to the global optimum. However, DOA can lose it too. To deal with this problem, a portion of the population can be randomly initialized outside the promising region, to try to put at least one individual near to the global optimum when the optimization loses it.

After the domain optimization, a local search can be started with an heuristic initialization (inside the promising region). To evaluate DOA's efficiency, we tested some benchmark functions and the results are presented in the next section.

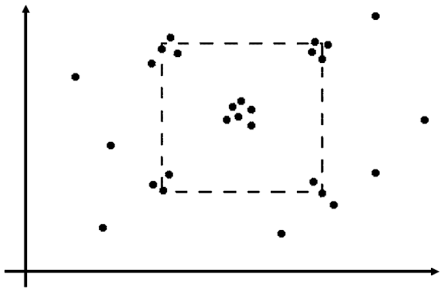

Figure 3: Preferential initialization inside the promising region

4. EXPERIMENTS

The benchmark functions used in the tests with the DOA are part of a set of benchmarks proposed recently in the literature (see [25]) to evaluate new global optimization algorithms. In the preliminary experiments, presented in this paper, the functions from *1* to *12* are used. The benchmark functions proposed in [25] are variations of functions previously cited in the literature. There are evolutionary algorithms which take advantage of certain known properties of the original functions as global optimum located along the axes, global optimum which possess equal values in some variables, among others. For that reason, Liang and colleagues [13] proposed modifications in the benchmark functions, such as displacement, rotation and positioning of the global optimum close to one of the search-space edges, that increases the difficulty of these problems for any optimization algorithm.

Two well known global optimization algorithms are tested in this work: the GA and the PSO (Particle Swarm Optimization) [6, 2] The configuration of the algorithms, presented in Table 3, were determined after experimental analysis.

There are two methods to experiment: RAND (simple GA or PSO with random initialization) and DOA (GA or PSO executed after the search-space reduction). Each one of the methods is executed 50 times, for each function from *f1* to *f12*, with a maximum amount of evaluations equal to 10,000. The number of variables/dimensions (*k*) is 30. One half of the population is generated inside the promising region. The other half is generated anywhere in the search-space. We evaluate the population and replace the worst individual with the best point found by the DOA.

For the RAND method, consider $f^{*(i)}_{RAND} = f(S^{*(i)}_{RAND})$, where $S^{*(i)}_{RAND}$ is the best solution found by the RAND method in the i^{th} repetition, *i*=1,... ,50. This way, f^{*}_{RAND} is a vector of size 50 with the best solutions found by the RAND method in the *i* repetitions.

For the DOA method, consider the variables $f^{*(i)}_{DOA} = f(S^{*(i)}_{DOA})$, where $S^{*(i)}_{DOA}$ is the best solution found by the DOA method in the i^{th} repetition, *i*=1,... ,50. Hence, f^{*}_{DOA} is a vector of size 50 with the best solutions found by the DOA method in the *i* repetitions. In that method, for each repetition a search-space reduction is made and the number of evaluations used by DOA is stored. After that, the optimization algorithm (GA or PSO) is used to minimize the function, limiting the amount of evaluations to 10,000 minus the evaluations made by DOA. Besides the promising region found by DOA, the best point sampled during the process is also passed to the optimization algorithm.

Given the high magnitude of some standard deviations obtained in this experiment, the simple comparison between two means can lead to low confidence conclusions. For that reason, a statistical test was adopted to compare the methods.

Let f^{*}_{RAND} and f^{*}_{DOA} be two independent random samples from a normal distribution with unequal variances, one can test the hypothesis that both possess the same means using a *t-test* [18]. For such, a significance level $\alpha = 0.05$ and the null hypothesis H_0 = equal means were adopted. This way, if the *p-value* returned by the *t-test* is smaller than α, then H_0 is rejected. Otherwise, H_0 is accepted and one concludes that the means are equal. If H_0 is rejected, the best algorithm is the one which presents the smallest mean. Some staticians consider that as the *p-value* gets closer to 1, more similar the means are. Therefore, the closer to 0, more they are different. However, this affirmation isn't fully accepted in the literature.

The results of the experiments using a GA and a PSO, with stop criterion equal to 10 fails (Table 3) are presented in the Tables 4 and 5. The *Min*, *Mean*, *Max* and σ columns are calculated from the f^* vectors. The *p-value* is calculated by the *t-test* with the independent samples f^{*}_{RAND} and f^{*}_{DOA}. The $\surd$ symbol in the *Best* column indicates which of the two initialization methods obtained the best result in the optimization process. That occurs only when H_0 is rejected. If H_0 is accepted, in the *Best* column is indicated that the means are equal.

In the experiment with the GA (Table 4), one can see in the *Mean* column that the RAND algorithm achieved better results in the functions 3, 5, 10, and 11. In the other 8 functions, the GA was benefited by the DOA. However, a statistical analysis presents that many of the differences in the means aren't significant. This way, the p-value obtained by the t-tests presents that the RAND algorithm achieved a better result *only* in the function 10. On the other hand, the DOA helped the GA only in 4 functions (1, 6, 7, and 9), instead of 8. In the other 7 functions, the means are equal.

In very hard problems, as functions 8 and 11 where the means are very close, even the reduction can't help the optimization algorithm. On the other hand, in simple problems, as functions 2 to 5, a simple optimization algorithm can lead to good results. One can conclude about this experiment that, in the majority of the functions, the computational cost of the DOA doesn't worth. The GA algorithm can deal with them. However, in the functions where the DOA initialization lead to a better result, the mean differences are very clear. Thus, in these functions, the DOA is really useful.

The PSO algorithm, which results are presented in Table 4, had the opposing behavior. The RAND algorithm achieved better results in 8 functions: 1 to 7, and 11. On the other hand, the DOA helped the PSO only in 4 functions: 8, 9, 10, and 12. There are clear differences in the means in the functions 2, 6, 9, and 10.

In this experiment, the statistical analysis shows that the RAND algorithm was the better one in the functions 1, 2, 6,7, and 11. On the other hand, the DOA helped the PSO in 3 functions (8, 9 and 10). In the other 4 functions, the means are considered equal.

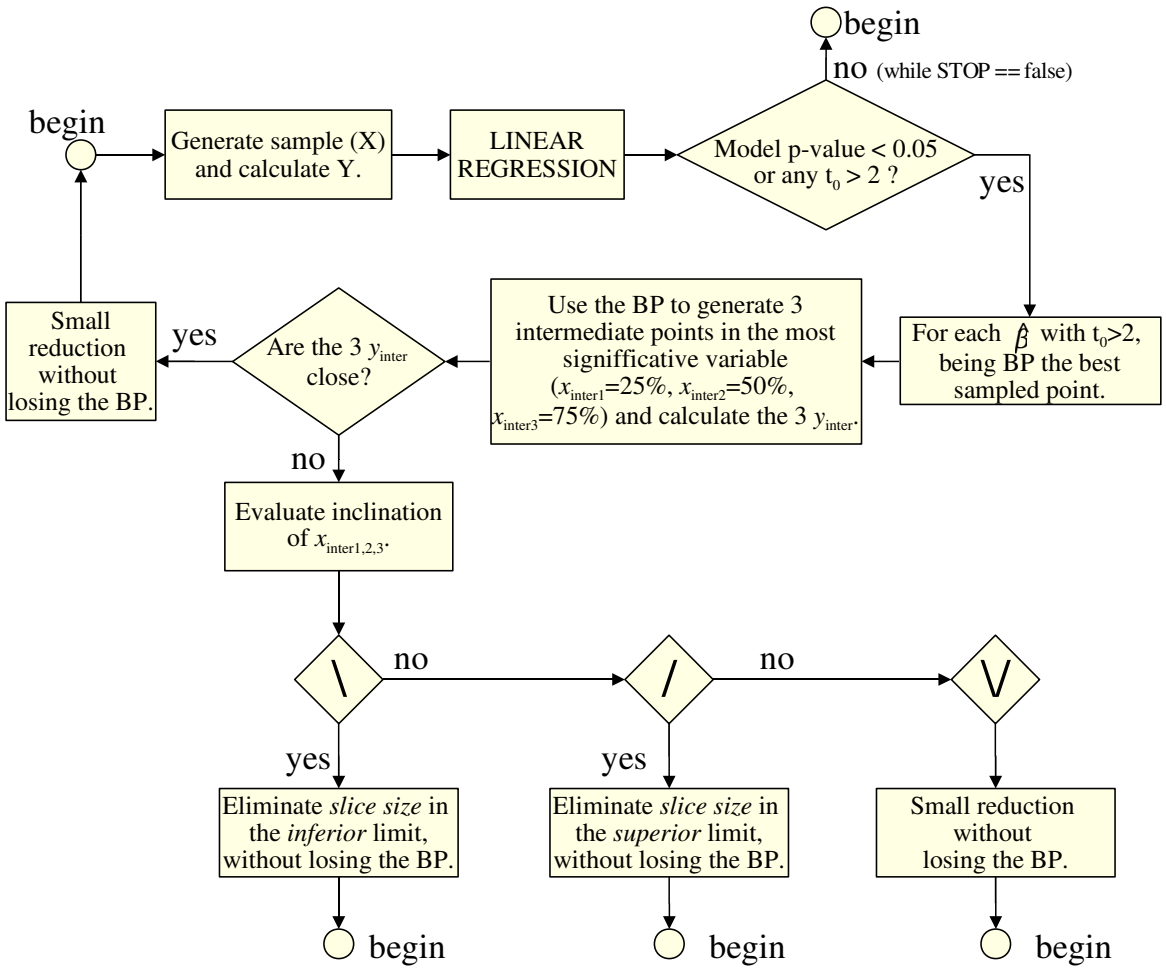

Figure 2: Basic Flowchart of DOA using Linear Regression

In general, we would expect that the DOA would lead to a clear advantage over the simple random initialization in *all* benchmark functions or, at least, be as good as the random initialization. However, as one could see in the presented results, this doesn't happened. In 24 functions (12 GA + 12 PSO) the results were: RAND = won 6 times, DOA = won 7 times, Equal means = 11 times.

Analyzing the promising regions found in the experiments, we detected that the DOA lost the global optimum in some of the 30 variables. When the optimum is located near of an edge in a determined variable, the first reduction can lose it. Thus, further reductions gets even farther from the optimum in that variable. Even initializing 50% of the population outside the promising region, the optimization algorithm needed, sometimes, a higher number of function evaluations to find where the lost optimum is.

When the promising region is very close to the global optimum in 20 of the 30 variables, for example, the GA and PSO had difficult searching outside only for the other 10 variables. This way, another conclusion is that more sophisticated optimization algorithms can use the promising region found by the DOA and achieve much better results than the GA and PSO.

5. CONCLUSIONS

In this work, we presented an approach to use a simple DoE method (Multiple Linear Regression) to, starting from the search-space limits, iteratively eliminate portions of the search-space where the algorithm guarantee, with a high probability, that there is no global optimum. The algorithm ends when it cannot guarantee a further reduction. Our approach, a Domain Optimization Algorithm (DOA) belongs to a class of algorithms called Search-space Reduction Algorithms (SRAs). A SRA is a technique to determine a promising region before the use of an optimization algorithm, instead of detecting one region during the global optimization process.

The DOA proposed in this work uses a simple model that tries to guarantee that the global optimum is inside the optimized area. In general, algorithms as EDAs try to guide the search by modeling the space of solutions through highly complex models. That is exactly the opposing of what we propose in our algorithm.

Analyzing the presented results, one can conclude that the proposed approach definitely shows relevant results for tests using hard benchmark functions. However, we will explore ways to keep the global optimum inside the promising region. Other search-space analysis can be taken in consideration before eliminating a portion of the space, trying to guarantee even more that the global optimum remains inside the promising region.

Another important step is to test our approach with more sophisticated optimization algorithms, which can use information presented by the DOA in a more efficient way.

Table 3: Algorithms Configurations

DOA	GA	PSO
Stop criterion: 10 errors	Population: 60	Population: 20
Slice size: 20%	BLX Crossover (%): 60	Weight: 0.1
Cut on both sides: 3%	Mutation (%): 10	Mode: Asynchronous
Sampling: 60 points	Tournament selection	

Table 4: GA with random and DOA initialization. *G.O.* is the global optimum and k=30

F	Method	G.O.	Min	Mean	Max	σ	p-value	Best
f1	RAND	-450	-416.036	-266.110	282.983	138.230	0.007	
	DOA		-434.750	-347.824	-59.531	85.351	0.007	$\surd$
f2	RAND	-450	15871.700	41634.426	64128.000	11777.791	0.377	=
	DOA		20456.900	39038.016	62851.400	11184.884	0.377	=
f3	RAND	-450	17417300.000	71901009.677	147706000.000	29016433.685	0.758	=
	DOA		25783200.000	74741180.645	172674000.000	41942346.051	0.758	=
f4	RAND	-450	33548.700	53035.910	81493.700	12485.205	0.437	=
	DOA		23011.300	50236.706	91727.800	15518.717	0.437	=
f5	RAND	-310	5169.960	8623.271	14491.600	2272.487	0.835	=
	DOA		4691.000	8738.822	12690.100	2083.124	0.835	=
f6	RAND	390	39309400.000	324228864.516	1379190000.000	351368505.112	0.011	
	DOA		9122110.000	146523417.097	385401000.000	113833448.586	0.011	$\surd$
f7	RAND	-180	-156.087	-66.622	110.808	69.105	0.012	
	DOA		-166.871	-107.231	150.549	53.716	0.012	$\surd$
f8	RAND	-140	-119.058	-118.902	-118.828	0.061	0.714	=
	DOA		-119.013	-118.907	-118.800	0.052	0.714	=
f9	RAND	-330	-214.750	-168.576	-129.405	18.917	0.002	
	DOA		-260.884	-194.958	-115.171	41.172	0.002	$\surd$
f10	RAND	-330	-114.094	-90.767	-60.215	14.369	0.019	$\surd$
	DOA		-127.690	-81.024	-53.162	17.388	0.019	
f11	RAND	90	125.002	132.705	134.661	1.980	0.818	=
	DOA		129.069	132.604	134.927	1.424	0.818	=
f12	RAND	-460	258605.000	338641.531	520407.000	67000.262	0.615	=
	DOA		157757.000	329313.121	568836.000	81160.613	0.615	=

6. REFERENCES

[1] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for experimenters*. John Willey, New York, 1978.

[2] A. Carlisle and G. Dozier. An off-the-shelf pso. In *Proceedings of the Workshop on Particle Swarm Optimization*, pages 1–6, Indianapolis, IN, 2001.

[3] K. Y. Chan, M. E. Aydin, and T. C. Fogarty. A taguchi method-based crossover operator for the parametrical problems. In R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 971–977, Canberra, 8-12 December 2003. IEEE Press.

[4] S. Chen and S. Smith. Improving genetic algorithms by search space reduction (with applications to flow shop scheduling). In *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 1999.

[5] D. R. Cox. *Planning of Experiments*, volume 1. John Wiley and Sons, Inc, 1958.

[6] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micromachine and Human Science.*, pages 39–43, Nagoya, Japan, 1995.

[7] R. A. Fisher. *The Design of Experiments*. Oliver and Boyd, Ltd, Edinburgh and London, 1935.

[8] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[9] M. J. Guerra and D. Donaire. *Estatística indutiva: teoria e aplicacões*. Livraria Ciência e Tecnologia Editora, São Paulo, 2 edition, 2000.

[10] L. Guo and I. Matta. Search space reduction in qos routing. *Comput. Networks*, 41(1):73–88, 2003.

[11] D. Heckerman, D. Geiger, and M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. Technical report, Microsoft Research, Redmond, WA., 1994.

[12] K. Hinkelmann and O. Kempthorne. *Design and Anaysis of Experiments: Introduction to Experimental Design*, volume 1. Wiley Series in Probability and Mathematical Statistics, 1994.

[13] J. J. Liang, P. N. Suganthan, and K. Deb. Novel composition test functions for numerical global optimization. In *IEEE Swarm Intelligence Symposium*, pages 68–75, 2005.

[14] K.-H. Liang, X. Yao, and C. Newton. Combining landscape approximation and local search in global optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2, pages 1514–1520, Piscataway, NJ, 1999. IEEE Press.

[15] K.-H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated n-dimensional landscapes. *International*

Table 5: PSO with random and DOA initialization. *G.O.* is the global optimum and k=30

F	Method	G.O.	Min	Average	Max	σ	p-value	Best
f1	RAND	-450	-447.682	-439.794	-412.590	6.229	0.024	$\surd$
	DOA		-445.621	-437.286	-424.130	4.871	0.024	
f2	RAND	-450	617.424	1544.849	2794.051	445.678	8.9e-5	$\surd$
	DOA		711.902	1950.386	3457.954	559.029	8.9e-5	
f3	RAND	-450	6488343.000	16131098.125	26858816.000	4842266.138	0.336	=
	DOA		7812808.500	17048649.702	28909354.000	4841748.991	0.336	=
f4	RAND	-450	4296.308	12593.333	28682.590	5367.218	0.307	=
	DOA		5456.117	13606.006	26738.850	4665.040	0.307	=
f5	RAND	-310	2121.965	5139.053	10477.521	1707.183	0.807	=
	DOA		2140.629	5211.528	8404.872	1273.425	0.807	=
f6	RAND	390	2905.464	14324.023	38332.352	8504.785	0.001	$\surd$
	DOA		2972.884	21848.110	68429.039	13183.726	0.001	
f7	RAND	-180	-178.349	-177.059	-174.130	1.002	0.025	$\surd$
	DOA		-178.129	-176.579	-172.750	1.136	0.025	
f8	RAND	-140	-119.081	-118.952	-118.826	0.060	8.4e-5	
	DOA		-119.208	-119.012	-118.862	0.087	8.4e-5	$\surd$
f9	RAND	-330	-221.651	-162.135	-106.961	24.742	<2e-16	
	DOA		-288.037	-262.061	-209.642	16.555	<2e-16	$\surd$
f10	RAND	-330	-96.411	11.932	170.920	55.155	1.4e-8	
	DOA		-213.001	-70.873	81.880	78.416	1.4e-8	$\surd$
f11	RAND	90	107.660	117.060	125.471	3.854	7.7e-6	$\surd$
	DOA		112.219	120.537	127.752	3.583	7.7e-6	
f12	RAND	-460	10929.297	44685.193	107209.211	21361.855	0.513	=
	DOA		10311.808	42165.107	98771.375	17627.689	0.513	=

Journal of Knowledge-Based Intelligent Engineering Systems, 4(3):172–183, July 2000.

[16] Y. W. Lung and Y. Wang. An orthogonal genetic algorithm with quantization for global numerical optimization. In *IEEE Transactions on Evolutionary Computation*, volume 5, pages 41–53, 2001.

[17] S. R. S. Magalhaes. A avaliacão de métodos para a comparacão de modelos de regressão por simulacão de dados. Master's thesis, (Mestrado em Estatística e Experimentacão Agropecuária), Universidade Federal de Lavras, Lavras, 2002.

[18] D. C. Montgomery. *Design and Anaysis of Experiments*, volume 1. John Wiley and Sons, Inc, 1997.

[19] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley, New York, second edition, 2002.

[20] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. Technical Report 99018, IlliGAL Report, 1999.

[21] A. Ratle. Kriging as a surrogate fitness landscape in evolutionary optimization. *Artif. Intell. Eng. Des. Anal. Manuf.*, 15(1):37–49, 2001.

[22] C. R. Reeves and C. C. Wright. Epistasis in genetic algorithms: an experimental design perspective. In *6th International Conference on Genetic Algorithms*, San Mateo, CA, 1995. Morgan Kaufmann.

[23] C. R. Reeves and C. C. Wright. An experimental design perspective on genetic algorithms. In *D. Whitley and M. Vosa (Eds.). Foundations of Genetic Algorithms 3*, San Mateo, CA, 1995. Morgan Kaufmann.

[24] M. Srinivas and L. M. Patnaik. Learning neural network weights using genetic algorithms- improving performance by search-space reduction. *1991 IEEE International Joint Conference on Neural Networks*, 3(IEEE Cat. No. 91CH3065-0):2331–2336, 18-21 Nov 1991.

[25] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report KanGAL Report 2005005, Nanyang Technological University, Singapore, 2005.

[26] X. Yao, Y. Liu, K.-H. Liang, and G. Lin. Fast evolutionary algorithms. *Advances in evolutionary computing: theory and applications*, pages 45–94, 2003.

[27] F. Yates. *The Design and Anaysis of Factorial Experiments*. Number 35. Commonwealth Bureau of Soil Society, Tech. Comm., 1935.

[28] Q. Zhang and Y. Lung. An orthogonal genetic algorithm for multimedia multicast routing. *IEEE Transactions on Evolutionary Computation*, 3(I):53–62, 1999.