Improved scatter search for the global optimization of computationally expensive dynamic models

JOSE A. EGEA

Process Engineering Group. Instituto de Investigaciones Marinas (C.S.I.C.)
Eduardo Cabello 6, 36208 Vigo, Spain. jegea@iim.csic.es

EMMANUEL VAZQUEZ

Department of Signal and Electronic Systems, Supélec.
Plateau de Moulon, 3 rue Joliot-Curie, 91192 Gif sur Yvette, France.
emmanuel.vazquez@supelec.fr

JULIO R. BANGA

Process Engineering Group. Instituto de Investigaciones Marinas (C.S.I.C.)
Eduardo Cabello 6, 36208 Vigo, Spain. julio@iim.csic.es

RAFAEL MARTÍ

Departamento de Estadística e Investigación Operativa. Universitat de València.
Dr. Moliner 50, 46100 Burjassot (Valencia), Spain. rafael.marti@uv.es

Abstract

A new algorithm for global optimization of costly nonlinear continuous problems is presented in this paper. The algorithm is based on the scatter search metaheuristic, which has recently proved to be efficient for solving combinatorial and nonlinear optimization problems. A kriging-based prediction method has been coupled to the main optimization routine in order to discard the evaluation of solutions that are not likely to provide high quality function values. This makes the algorithm suitable for the optimization of computationally costly problems, as is illustrated in its application to two benchmark problems and its comparison with other algorithms.

KeyWords

Global Optimization, Expensive Functions, Scatter Search, Kriging.

1. Introduction

Many industrial and engineering problems can be formulated as optimization problems (Biegler and Grossmann 2004). These problems are often nonlinear and present dynamic behaviour due to their operating policies (i.e. batch or semi-batch operation) or to their inherent nonlinear dynamic nature (i.e. like in biotechnological processes, as reviewed by Banga et al. 2003). Further, in most real cases some specifications and/or constraints (which may also have nonlinear and/or dynamic nature) must be ensured. All these characteristics frequently result in non-convex problems, thus the use of global optimization methods becomes mandatory (Floudas et al. 2005).

Another relevant feature of this kind of problem, which has been the subject of recent research, is the significant computation time required by each function evaluation. Indeed, due to the complexity of the mathematical models representing real processes, the simulation of a complex system can take from

minutes to hours in a standard workstation. Therefore, the use of some kind of surrogate model, which substitutes the original one with enough accuracy, may help to alleviate this problem. Surrogate models are cheaper to evaluate, so their use will result in reductions of the total computation times, making them affordable from the industrial point of view. The taxonomy of global optimization methods based on response surfaces by Jones (2001a) states the problem and presents different methodologies to solve it. The most promising techniques to date seem to be kriging (being the most popular implementation the *EGO* algorithm of Jones et al. 1998) and interpolation by radial basis functions (*RBF's*; Gutmann 2001).

In this contribution, we present a methodology for the global optimization of (possibly dynamic, non-smooth) nonlinear problems with expensive evaluation. This methodology, and the associated software tool, *SSKm* (Scatter Search with Kriging for Matlab), is able to manage this class of problems by linking a scatter search method with a kriging interpolation.

The metaheuristic known as scatter search (Laguna and Martí 2003) is an evolutionary method founded on the premise that systematic designs and methods for creating new solutions afford significant benefits beyond those derived from recourse to randomization. This methodology has been successfully applied to a wide array of hard optimization problems. Our new procedure is an extension of a recent advanced design of this methodology (Egea et al. 2007) and treats the objective function as a black box, making the search algorithm context-independent. The kriging predictor implemented in *SSKm* avoids the evaluation of solution vectors that are likely to provide low quality function values, thus efficiently reducing the number of simulations needed to find the vicinity of the global solution.

The paper is organised as follows: Sections 2 and 3 present brief views of the general scatter search and kriging methodology respectively. Section 4 presents our algorithm *SSKm* explaining in detail its features. In section 5 illustrative examples of the algorithm application are presented, one of them being a real application of operational design of a waste water treatment plant (WWTP) benchmark. The final section contains the conclusions of this study.

2. Scatter Search

Scatter search (SS) was first introduced in Glover (1977) as a heuristic for integer programming. SS consists of five elements that can be implemented in different degrees of sophistication. The basic design to implement SS is based on the "five-method template" (Laguna and Martí 2003):

A *Diversification Generation Method* to generate a collection of diverse trial solutions within the search space.

An *Improvement Method* to transform a trial solution into one or more enhanced trial solutions

A *Reference Set Update Method* to build and maintain a reference set consisting of the *b* "best" solutions found (where the value of *b* is typically small e.g. no more than 20). Solutions gain membership to the reference set according to their quality or their diversity.

A *Subset Generation Method* to operate on the reference set, to produce several subsets of its solutions as a basis for creating combined solutions.

A *Solution Combination Method* to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solution vectors.

Figure 1 illustrates the main steps of the SS algorithm. The circles represent solutions and the darker circles represent improved solutions resulting from the application of the Improvement Method The algorithm starts (SS Initialization) with the creation of an initial set of solutions *P* generated with the Diversification Generation Method, and then extracts from it the reference set (*Refset).* The initial reference set is built according to the Reference Set Update Method, which takes the *b*/2 best solutions (as regards their quality in the problem solving) and the *b*/2 distinct and maximally diverse solutions from *P* to compose the *Refset*. Once the *Refset* has been built, its solutions are ordered according to quality. In this step, the Subset Generation Method creates sets of solutions in the *Refset* to be combined. In its simplest form, the Subset Generation Method generates all pairs of reference solutions. The sets of solutions in *Refset* are selected one at a time and the Solution Combination Method is applied to generate some trial solutions from each of those sets. These trial solutions are subjected to the Improvement Method. The Reference Set Update Method is applied once again to update the new *Refset* with the best solutions from the current *Refset* and the set of trial (possibly improved) solutions.

The SS Main Loop terminates after all the generated subsets are subjected to the Combination Method and none of the improved trial solutions are admitted to enter the *Refset* under the rules of the Reference Set Update Method. However, in advanced SS designs as this one shown in Figure 1, the *Refset* rebuilding is applied at this point keeping the best *b*/2 solutions in the *Refset* and selecting the other *b*/2 from *P*.

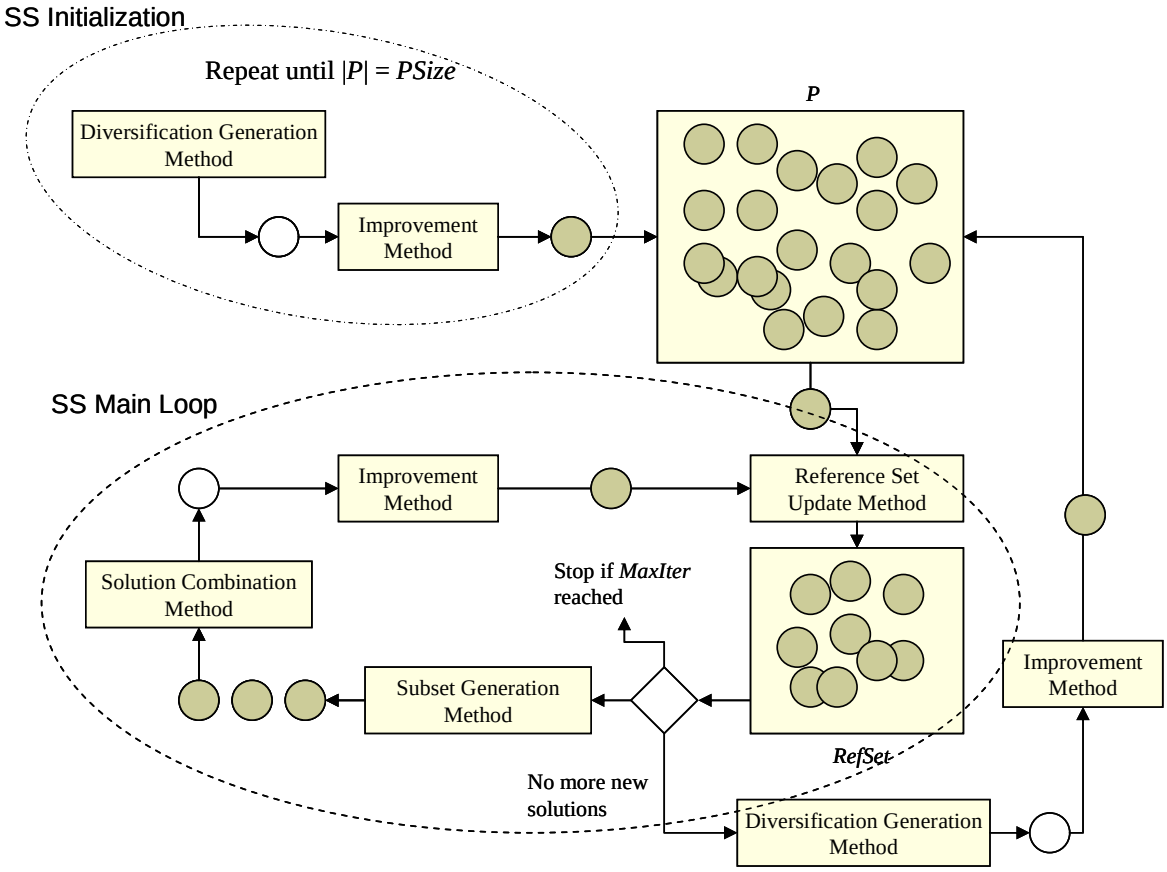


Figure 1: Schematic representation of the SS design where the shaded circles represent solutions that have been subjected to the *Improvement Method*

Of the five methods in SS methodology, only four are strictly required. The Improvement Method is usually needed if high quality outcomes are desired, but a SS procedure can be implemented without it as it occurs in some problems where the Improvement Method can not provide high quality solutions due to the problem's nature or when the computation budget is limited to a small number of function evaluations.

An advanced design of the SS methodology has recently been presented in Egea et al. (2007). Several strategies to surmount the problems arising in optimization problems from the biotechnological industry are implemented showing the flexibility of SS to be modified according to the difficulties of the problems to be solved. The algorithm presented in this paper is an extension of the method mentioned above, incorporating a kriging-based prediction mechanism. All these features are detailed in Section 4.

3. Kriging

The term *kriging* originates from geostatistics and the method was named and formalized by a French mathematician (Matheron 1963). Kriging can be defined as a probabilistic interpolation method to create cheap-to-evaluate surrogate models from scattered observations minimizing the expected squared prediction error subject to being unbiased and being linear in the observations (Jones 2001a). Many examples of kriging implementations that illustrate its superiority over other interpolation methods can be found in the literature (see for example Cox and John 1997, Jones et al. 1998, Sasena et al. 2002).

Consider a real function f to be interpolated. Assume that f is a sample path of a second-order Gaussian random process denoted by F. Kriging computes the best linear unbiased predictor of $F(x)$ using the observations of F on a set of points $S=\{x_1,\ldots,x_n\}$. Denote by F_S the vector of observations $(F(x_1),\ldots, F(x_n))^T$. The Kriging predictor is a linear combination of the observations, which may be written as

$$\hat{F}(x) = \lambda(x)^T F_S \tag{1}$$

with $\lambda(x)$ a vector of coefficients $\lambda_1,\ldots, \lambda_n$. These coefficients are chosen to obtain the smallest variance of prediction error among all unbiased predictors. This leads to a constrained minimization problem, which can be solved by a Lagrangian formulation (Matheron 1969). The vector $\lambda(x)$ can be computed as the solution of the system of linear equations

$$\begin{pmatrix} K & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \lambda(x) \\ \mu(x) \end{pmatrix} = \begin{pmatrix} k(x) \\ a(x) \end{pmatrix} \tag{2}$$

where K is the covariance matrix of the random vector F_S, A is a matrix of known functions $a_1,\ldots,a_q$ (usually polynomials of low degree) evaluated at the points of S, $k(x)$ is the covariance vector between $F(x)$ and F_S, $a(x)$ is the vector of $a_1,\ldots,a_q$ evaluated at x, $\mu(x)$is a vector of Lagrangian multipliers and 0 is a matrix of zeros.

Knowing the kriging coefficients, the predicted value of f given $f_S = (f(x_1),\ldots,f(x_n))^T$ can be written as

$$\hat{f}(x) = \lambda(x)^T f_S \tag{3}$$

The selection of a suitable covariance function is crucial for the success and accuracy of the kriging prediction. For this purpose, it is usual to choose a parameterized covariance model and to estimate its parameters based on the observations.

The use of a stationary, isotropic covariance model with one parameter to adjust regularity makes it possible to model a large class of functions (Vazquez 2005). Here we use the Matérn covariance, with the following parameterization (Yaglom 1986, Stein 1999)

$$k(h) = \frac{\sigma^2}{2^{\upsilon-1}\Gamma(\upsilon)} \left(\frac{2\upsilon^{1/2}h}{\rho} \right)^{\upsilon} \mathrm{K}_{\upsilon} \left(\frac{2\upsilon^{1/2}h}{\rho} \right) \tag{4}$$

where h is the Euclidean distance between two points, K_{υ} is the modified Bessel function of the second kind, υ controls the regularity, σ^2 is the variance and ρ represents the range of the covariance.

One of the advantages of kriging is that the variance of the prediction error at x can be computed even without any evaluation of f. This is one of the strongest points of this method compared to others: kriging provides a statistical framework that gives an idea of the uncertainty associated to each prediction. This also helps us to know which points are worth evaluating in different applications of the method (for example, in global optimization).

Figure 2 shows the kriging prediction of the sine function in the interval [-10 10]. The solid line is the real function whereas the dotted line is the kriging prediction based on the observations (dark circles). For a point x_i kriging provides a normal distribution function (dashed line). The mean of the distribution is the kriging prediction and the variance is also provided in the calculation process. With this distribution we can not only know which is the prediction in every point provided some observations but also the uncertainty associated to this prediction and thus the probability of finding a value lower than a threshold when evaluating the real function.

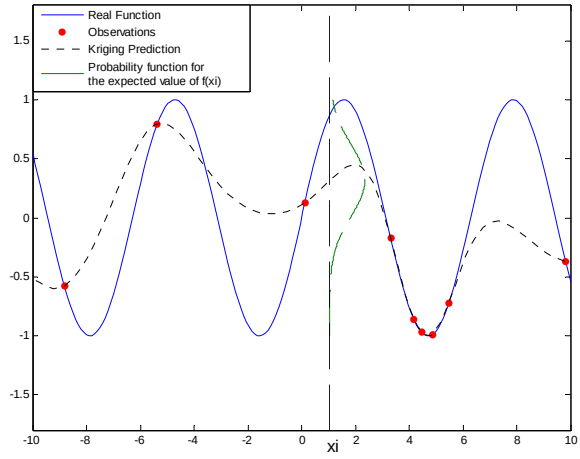


Figure 2: Kriging prediction for the function $y = sin(x)$ from a set of sampling points.
The Gaussian distribution for point x_i provided by kriging is shown.

In Figure 3, a 2-dimensional function, the *six-hump camel-back function*, is presented. The real function $f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + x_1^6/3 + x_1x_2 - 4x_2^2 + 4x_2^4$ within the interval [-5 5] is plotted in Figure 3a. Figures 3b, 3c and 3d plot the kriging prediction of the function in the same interval using n_0 = 20, 50 and 100 observations (i.e. real function evaluations) uniformly distributed in the same interval respectively. It can be observed that the larger number of observations, the higher accuracy in the prediction.

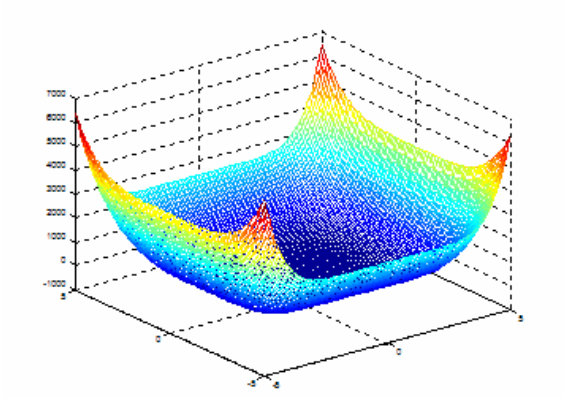


Figure 3a: *Six-hump camel-back* function

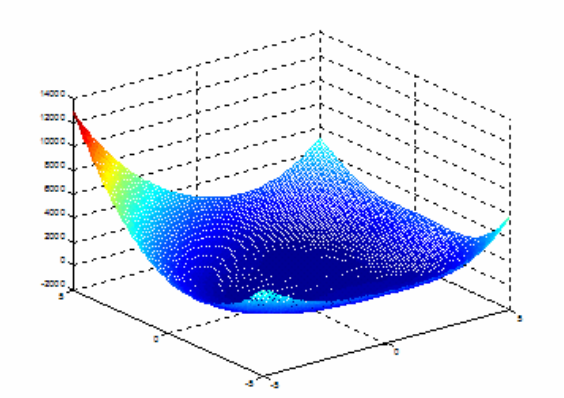


Figure 3b: Kriging prediction for n_0 = 20

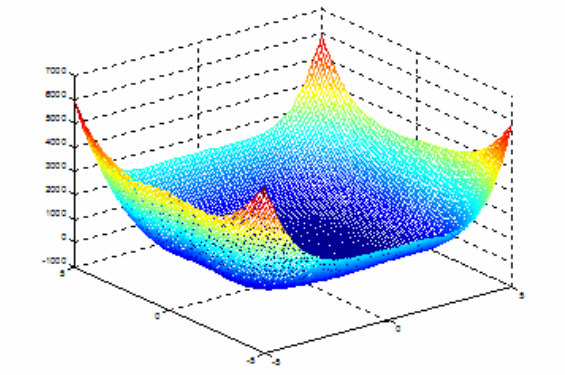


Figure 3c: Kriging prediction for n_0 = 50

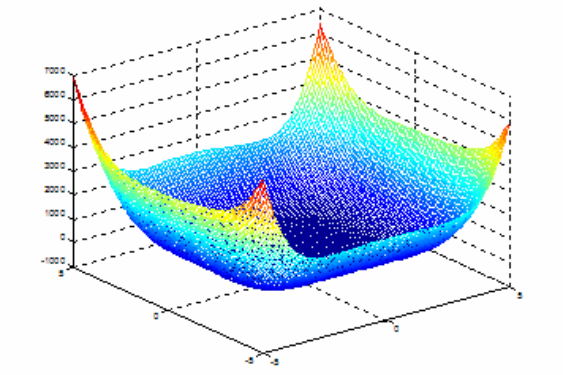


Figure 3d: Kriging prediction for n_0 = 100

4. *SSKm* Algorithm

In this section we present the *SSKm* algorithm (Scatter Search with Kriging for Matlab) which is a Matlab implementation of our proposed methodology for global optimization of computationally expensive problems. The algorithm is a scatter search implementation based on the one presented in Egea et al. (2007) extended to include a kriging predictor in order to increase the efficiency of the search. Thus, it combines the power of scatter search for exploring different parts of the search space in order to locate the global optimum, with the prediction of kriging to avoid the evaluation of solutions which are not likely to provide good objective function values in terms of quality.

There are a number of reasons that justify the combination of these two techniques. On the one hand, both of them have proved to be efficient in their respective fields (i.e. global optimization in the case of scatter search, and prediction in the case of kriging). On the other hand, kriging needs a careful selection of the points in which the prediction will be done, in order to avoid investing computational effort in calculating the kriging coefficients for points that will not be of interest. Scatter search operates on a set of solution vectors that evolve during the search process (new solutions replace old ones in the *Refset*). These solutions have by construction good objective function values, thus they are good candidates for the kriging prediction.

The scatter search algorithm proposed in this paper extends several of the innovative mechanisms presented in Egea et al. (2007), such as:

- Variable bounds with different orders of magnitude can be pointed out to generate solutions in all the orders of magnitude (in the initialization or rebuilding). This log-normalization may also help the prediction phase with kriging allowing a smoother correlation among variables.
- Combinations are based on hyper-rectangles around the solutions to be combined.
- The *Refset* update method takes into account a Euclidean distance to avoid duplication and clustering inside it. This Euclidean distance allows the method to replace other solutions in *Refset* different from the worst one as long as the distance is not violated.
- A strategy based on orthogonality is used together with the classical one of maximum diversity in the *Refset* rebuilding.
- No improvement method has been implemented since the purpose of the algorithm is to reduce as much as possible the number of function evaluations to locate the global optimum.

A pseudo code of our algorithm is presented in Table 1.

```
1.  Generate diverse solutions
2.  Form the first Refset and sort it by quality
3.  Compute the best observation, fbest
4.  Set variables estimation = 1, next_iter = 0
5.  while not termination criterion
6.      while uncombined pair of solutions in the Refset
7.          Generate solutions by combinations of pairs of solutions in the Refset
8.          while not next_iter
9.              if estimation
10.                 Estimate covariance parameters
11.             Compute kriging prediction over generated solutions
12.             Compute for each solution the probability of outperforming fbest
13.             if max(probabilites) > prob_min
14.                 Evaluate solution with the maximum probability
15.                 Add solution to observations
16.                 Update fbest
17.                 If diff(estimation, real value) < tol
18.                     estimate = 0
19.                 else
20.                     estimate = 1
21.                 do Refset replacement
22.                 Update uncombined pairs in the Refset
23.                 Delete evaluated solution from x^c
24.                 Check termination criterion
25.             else
26.                 next_iter = 1
27.         next_iter = 0
28.         Sort Refset
29.     Regenerate Refset and update the uncombined pairs in it
```

Table 1. *SSKm* pseudo-code

4.1 Initialization

SSKm starts by generating an initial set of solution vectors, *P*, to be evaluated, using a latin hypercube sampling. Users can choose the size of *P* having also the possibility of using their user-initial solution vectors (with their objective function values, if they are known, to avoid computing them). The balance between the size of *P* and the computational cost of the evaluation of this set is an open question. A large set may lead to good prior information and an accurate kriging surface but involves a high computation time for evaluating the initial set, whereas a small set is cheaper to evaluate but may not help the algorithm in the first iterations due to the lack of accuracy of the surface. For our algorithm we recommend a number of solution vectors in *P*, n_p, ten times higher than the size of the problem (i.e. the number of decision variables, *n*).

Once every solution in *P* is evaluated, we apply the *Refset* Update Method for the first time to form the initial *Refset* by selecting the *b*/2 best solutions of *P* in terms of quality (*b* being the total number of solutions in *Refset* whose most typical value is 10). The other *b/2* solutions are added to the *Refset* from the solutions remaining in *P* by maximizing their Euclidean distance with respect to the members already included in the *Refset*. That is, for each candidate solution *x* in *P* and each solution *z* in *Refset*, we calculate the Euclidean distance $d(x,z)$ and then select the solution that maximizes $d_{\min}(x)$, where

$$d_{\min}(x) = \min_{z \in RefSet} \{ d(x,z) \} \tag{5}$$

This criterion is applied in a sequential fashion. At each step we add to Refset the solution that maximizes $d_{\min}(x)$, remove it from P, and then re-calculate the Euclidean distances. Therefore, we add one solution at each step until the Refset has been completed (for *b*/2 steps).

4.2 Subset Generation and Solution Combination Methods

After the initialization, the *Refset* is sorted according to the quality of its solutions (i.e. the best solution is the first) and we apply the Subset Generation Method which, in our implementation, consists of selecting all pairs of solutions (not yet combined) in the *Refset* to combine them. The number of solutions created from each pair of solutions in the *Refset* depends on their position in the sorted *Refset*. These combinations are of the following three types, assuming that x' and x'' are the solutions to be combined and that x' is superior in quality to x'':

- Type 1: $c_1 = x' - d_1$
- Type 2: $c_2 = x' + d_2$
- Type 3: $c_3 = x'' + d_3$

where $d_i = r_i \cdot (x'' - x')/2$ with i = 1, 2, 3 or 4 depending on the number of solutions generated (see below); r_i is a vector of dimension n with all its components being uniformly distributed random numbers in the interval [0 1]. The notation above indicates that the vectors are multiplied component by component, thus it is not a scalar product. The vector d_i has the following form:

$$d_i = \begin{bmatrix} d_{i,1} \\ d_{i,2} \\ \vdots \\ d_{i,n} \end{bmatrix} = \begin{bmatrix} \dfrac{r_{i,1}\left(x_1'' - x_1'\right)}{2} \\ \dfrac{r_{i,2}\left(x_2'' - x_2'\right)}{2} \\ \vdots \\ \dfrac{r_{i,n}\left(x_n'' - x_n'\right)}{2} \end{bmatrix}. \tag{6}$$

Note that if both solutions, *x'* and *x''*, belong to the first *b*/2 elements of the sorted *Refset*, then 4 vectors are generated: one of type 1, one of type 3 and two of type 2 (where the second solution of type 2 is generated using $d_4 \neq d_2$). If only x' belongs to the first *b/2* elements of the sorted *Refset*, then 3 vectors are generated: one of each type. Finally, if neither x'' nor x' belong to the first *b*/2 elements of the sorted *Refset*, then 2 vectors are generated: one of type 2 and another one of type 1 or 3 (randomly chosen).

These vectors generated by combination of the *Refset* members will be named x^c, c=1, 2,… n_c, where n_c is the total number of vectors generated by combination. The parameter n_c may change every iteration depending on the number of combinations made among *Refset* members given that the method avoids doing combinations with pairs of solutions already combined.

4.3 *Refset* Update Method

In a classical evolutionary algorithm every solution obtained from combination is evaluated. In our algorithm only some of them verifying certain conditions (see section 4.6) are evaluated. After the evaluation of the selected vectors x^c, the *Refset* Update Method checks whether the replacement of any of the *Refset* members can be performed. The evaluated vector is compared with all solution vectors in *Refset*. If x^c outperforms any of the solutions in the *Refset* in terms of quality, the replacement is carried out as long as a minimum diversity is accomplished (i.e. the method avoids vector duplication in the *Refset* by computing Euclidean distances among all solution vectors and excluding to enter the *Refset* solutions that violate a user-defined distance-threshold). In the case where the candidate vector to enter the *Refset* is better in terms of quality than more than one vector in *Refset* and it complies with the Euclidean distance to avoid duplications in every case, the replaced solution vector will be the worst in terms of quality.

4.4 *Refset* Regeneration

When all possible new combinations have been done and none of the generated vectors in x^c can replace any of the vectors in the *Refset*, the SS procedure can either stop or continue by regenerating the *Refset*. The latter strategy is used in our algorithm. The worst g vectors in *Refset* (in terms of quality) are deleted. New diverse vectors are generated using the same strategy as in the initialization (i.e. latin hypercube sampling) and the *Refset* is refilled according to the diversity criterion of maximizing Euclidean distances performed in the first *Refset* formation. Normally g is equal to $b/2$ but in aggressive implementations it can be set to b-1 (i.e. all the solution vectors in the *Refset* except the best one are deleted).

The strategy for regenerating the *Refset* introduced in Egea et al. (2007) has been implemented in *SSKm*, because the standard diversity criterion based on Euclidean distances described in subsection 4.1 does not ensure that the search will be performed along the "different dimensions" of the space. In our new strategy the vectors refilling the *Refset* are chosen to maximize the number of relative directions defined by them and the existing vectors in the *Refset*.

After deleting the g worst solutions, the *Refset* is $(b\text{-}g)\times n$ dimensional. Let $j=b\text{-}g$ be the number of existing vectors in the current *Refset*. We introduce the new matrix $M^{(j-1)\times n}$ containing the vectors that define the segments formed by the best vector in *Refset* and the rest of vectors in it. The $(k\text{-}1)$*-th* row of M is $x^1\text{-}x^k$, being x^1 the best element not deleted in *Refset* in terms of quality, and x^k (k = 2,…,j) the rest of the elements in it (note that the *Refset* is sorted according to quality). For every diverse vector created with the Diversification Generator Method, x^v with $v \in [1, 2,...,n_p]$ in the regeneration phase, a vector Q^v of scalar products is also defined

$$Q^v = (x^1 - x^v) \cdot M^T \tag{7}$$

where x^1 is again the best not deleted element in *Refset* and M^T is the transpose matrix of M. For every x^v the maximum value of its vector Q^v is computed as $msp(x^v)$. The solution s will join the *Refset* in the regeneration phase if

$$msp(s) = \min\left\{msp(x^v)\right\} \quad \forall\, v \in \left[1, 2, ..., n_p\right] \tag{8}$$

At this stage, the value of j is increased one unit and the process continues until $j = b$. The application of this strategy results in a maximum diversity in search directions on the regenerated *Refset*.

4.5 Covariance parameter estimation and kriging prediction

After the initial sampling at the initialization (see Section 4.1), the kriging covariance parameters are estimated for the first time so that when the Solution Combination Method is applied, the kriging prediction provide information to proceed to the evaluation (or not) of the solutions in x^c.

Assuming that the mean of $F(x)$ is zero for the sake of simplicity, this parameter estimation can be done calculating the maximum-likelihood estimate of the vector of covariance parameters, ϕ, by minimizing the negative log-likelihood (Vechia 1998) that can be expressed as:

$$l(\phi) = \frac{n_o}{2}\log 2\pi + \frac{1}{2}\log\det K(\phi) + \frac{1}{2} f_s{}^T K(\phi)^{-1} f_s \quad (9)$$

with n_o being the number of observations, f_s the observations and $K(\phi)$ the covariance evaluated with the set of parameters ϕ. In the general case where the mean of $F(x)$ is not zero, the covariance parameters can be estimated using other methods, as for example using Restricted Maximum Likelihood (Stein 1999). For the parameter estimation of the covariance parameters the Nelder and Mead method *fminsearch* implemented in the Matlab Optimization Toolbox was used.

With the covariance parameters estimated the kriging prediction will be then performed over the set of created solutions x^c. The predictor provides the value of the prediction (mean) and the uncertainty associated to it (variance). With these two values and assuming a Gaussian distribution, it is possible to calculate the probability for the point x to have a function value lower than the best observation registered. For each vector in x^c that probability is calculated. If the maximum of those probabilities is higher than a value *prob_min* fixed by the user, the point having such probability is evaluated and added to the observations.

Once the new evaluation has been done, the value of the best observation is updated and the parameters are only re-estimated if the value of the prediction and the real function value of the new observation differ in a percentage fixed also by the user. The process is repeated with the rest of the vectors remaining in x^c up to the point where the maximum probability of improving the best observation is lower than *prob_min*. At that point we break the loop and start a new iteration with new combinations among the new pairs on the *Refset* not yet combined.

The algorithm, based on the SS and the Kriging methodologies as described in the subsections above, has been implemented in Matlab. The method stops when any of the following conditions is accomplished.

1. Maximum number of function evaluations achieved,
2. Maximum computation time achieved, or
3. Specified function value defined by the user achieved.

5. Experimentation

In this section some illustrative examples of the application of our algorithm are shown. Specifically we consider two well-known benchmark problems to test the efficiency of our *SSKm* solution method: The six-hump camel-back function introduced in Section 3 and the proportional-integral controller tuning of a waste water treatment. For the sake of comparison, we apply two traditional optimization algorithms, one deterministic, *Direct* (Jones 2001b), and one stochastic, *Differential Evolution* (*DE*, Storn and Price 1997). In the second example we consider a third method in the comparison: another surrogate-based method that uses radial basis function interpolation, *rbfSolve*, implemented in the Tomlab optimization toolbox for Matlab (Holmström and Edvall 2004). All the experiments were carried out on a PC Pentium-IV 3.06 GHz using Matlab 6.5, Release 13, under Windows XP Pro.

5.1 *Six hump camel-back* function

The first example is the optimization of the s*ix hump camel* function already presented in Section 3. This function has several local minima and two global optima. For solving this problem we have applied our algorithm in the search space defined by the bounds [-3, -2] and [3, 2] by using an initial sampling of 20 diverse points and calculating 80 extra points for a total number of 100 points. A total number of 100 function evaluations were also fixed for both algorithms.

Table 2 shows the values of the two global optima as well as the closest points to them achieved by each algorithm. Figure 4 presents the contour plots of the function and the 80 last points evaluated by each algorithm (represented as triangles).

	Solution vector	Function value
Global minimum #1	**[-0.0898 0.7126]**	**-1.0316**
Closest point of *Direct*	[0.0000 0.7407]	-0.9905
Closest point of *DE*	[-0.0960 0.6695]	-1.0168
Closest point of *SSKm*	[-0.0798 0.7123]	-1.0312
Global minimum #2	**[0.0898 -0.7126]**	**-1.0316**
Closest point of *Direct*	[0.0000 -0.7407]	-0.9905
Closest point of *DE*	[0.0881 0.5453]	-0.7569
Closest point of *SSKm*	[0.0856 -0.7084]	-1.0314

Table 2. Solutions for the optimization of the *six hump camel* function in 100 function evaluations

Table 2 and Figure 4 demonstrate the superiority of *SSKm* over the other two algorithms because it not only achieves the best function values but also locates most of the evaluations in the vicinity of both global minima, whereas the other methods present a bigger dispersion on their evaluations. This shows the power of *SSKm* not only for locating one global minimum but also for locating several global minima when they exist. This is a very interesting characteristic for robust optimization purposes.

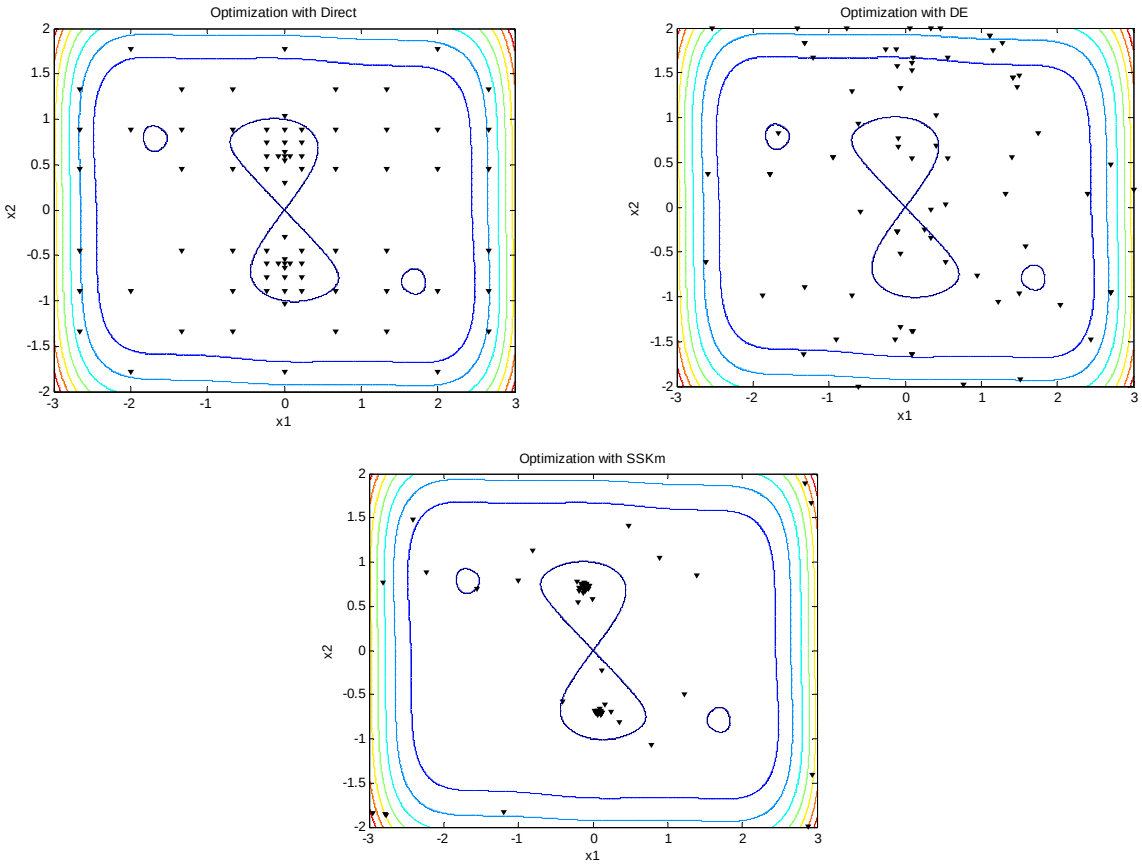


Figure 4. Contour plot of the *six hump camel* function with the evaluations done by: top left: *Direct*, top right: *DE*, bottom: *SSKm*

5.2 PI tuning of a costly waste water treatment plant

The next example of *SSKm* application is a challenging design problem consisting of the Proportional-Integral (*PI*) controller tuning of a computationally costly waste water treatment (WWT) plant. This model is formed by more than 150 ODE's and is detailed in the following website: http://www.ensic.u-nancy.fr/COSTWWTP/Benchmark/Benchmark1.htm

The model is costly because every function evaluation (or simulation) takes more than 2 minutes in a P-IV 3 GHz processor. This means that an optimization procedure can take days. For this reason, the use of surrogate model-based optimization algorithms is important for this problem. The plant, depicted in Figure 5, consists of five bioreactors (named *ASU* in the figure, which stands for Activated Sludge Unit) in which aqueous pollutants will be degraded by the actions of a microbial population (activated sludge). There are different parts in the plant. First there is an anoxic section (with lack of oxygen) in which denitrification takes place. Nitrates are reduced to gaseous nitrogen. The following part is an aerated area in which oxygen is introduced to help the microorganisms degrade the organic and nitrogenated compounds. Apart from oxidizing organic matter, nitrification takes part in this part of the plant. Since nitrates are being created, a recycling stream is necessary in the anoxic zone for the denitrification reaction. At the end of the plant there is clarifier in which the activated sludge is separated from the "clean" water. Part of this sludge is recirculated to the plant and the rest is thrown for disposal.

The control strategy consists of 2 PI controllers to control the nitrogen level in the second reactor by manipulating the internal recycle from the fifth to the first reactor, and the oxygen level in the fifth reactor by manipulating its aeration factor. Perturbations are introduced by means of different weather conditions. Different scenarios are considered (e.g. dry, rainy and stormy weather, see Copp 2001 for details) to choose the controller parameters that optimize the plant performance.

The objective function of this problem is a weighted sum of the *Integral Squared Error* (*ISE*) of both controllers for rainy weather although the study can be generalized to any kind of weather conditions (or their combinations) by introducing input data for such conditions available in the benchmark. Since this can be considered as a costly problem, it is a good candidate to be solved by a surrogate model-based method like *SSKm*.

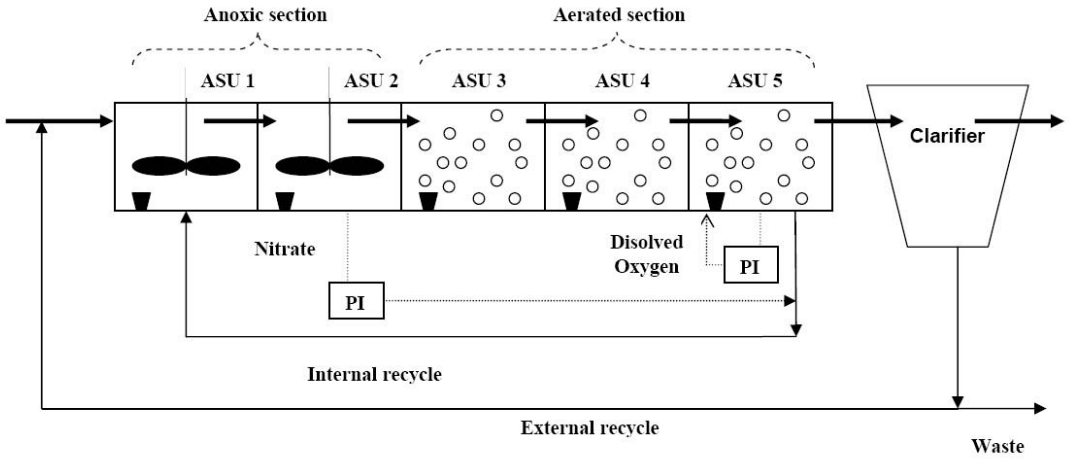


Figure 5. WWT plant layout

The problem dimension is four: the gain and the integral time of each controller. The bounds were selected taking into account the nominal values for the parameters given by the benchmark authors. The objective function was normalized so that its value using the parameters provided by default is 1. We want to demonstrate that using optimization techniques we can find better values in shorter times than using traditional PI tuning techniques, especially if we use surrogate model-based methods.

A maximum number of 400 function evaluations were fixed for all cases and the same initial point *x0* was used (except for *Direct*, which does not accept user initial points). Since *SSKm* and *rbfSolve* use an initial sampling phase to build the initial surrogate surface and to avoid bias, the same set of 42 initial points (41 points plus x_0) was used for both methods.

Convergence curves for the four methods are shown in Figure 6. It is clear that the two methods based on surrogate models achieve very good function values in a lower number of function evaluations than the other two. *SSKm* presents a slightly better convergence rate than *rbfSolve*. In the four cases 400 function evaluations (which translated in computation time means around 14 hours in our workstation) are enough

to improve the normalized value of 1 that results in the application of the parameters presented by default in the benchmark. These results show the successful application of optimization techniques for this problem and specially demonstrate that surrogate based-model algorithms outperform classical optimization algorithms for solving costly problems.

In table 2 bounds of the problem, initial point used, default parameters provided in the benchmark and final results of the optimization algorithms are presented, as well as their respective function values.

	x_1	x_2	x_3	x_4	**Function value**
Lower bound	$1.000 \cdot 10^{2}$	$7.000 \cdot 10^{-4}$	$1.000 \cdot 10^{2}$	$1.000 \cdot 10^{-2}$	13.5234
Upper bound	$1.000 \cdot 10^{3}$	$7.000 \cdot 10^{-1}$	$5.000 \cdot 10^{4}$	$1.000 \cdot 10^{0}$	100.2156
Nominal value	$5.000 \cdot 10^{2}$	$1.000 \cdot 10^{-3}$	$1.500 \cdot 10^{4}$	$5.000 \cdot 10^{-2}$	0.9985
x_0	$7.506 \cdot 10^{2}$	$5.069 \cdot 10^{-1}$	$2.783 \cdot 10^{4}$	$9.323 \cdot 10^{-2}$	35.9098
Final result *Direct*	$9.722 \cdot 10^{2}$	$5.017 \cdot 10^{-3}$	$2.259 \cdot 10^{4}$	$2.833 \cdot 10^{-2}$	0.6523
Final result *DE*	$7.126 \cdot 10^{2}$	$7.000 \cdot 10^{-4}$	$2.389 \cdot 10^{4}$	$5.482 \cdot 10^{-2}$	0.7077
Final result *rbfSolve*	$4.202 \cdot 10^{2}$	$7.000 \cdot 10^{-4}$	$2.019 \cdot 10^{4}$	$2.660 \cdot 10^{-2}$	0.5313
Final result *SSKm*	$5.828 \cdot 10^{2}$	$7.613 \cdot 10^{-4}$	$1.753 \cdot 10^{4}$	$2.280 \cdot 10^{-2}$	0.5314

Table 3. Data and results for the WWT plant PI tuning problem

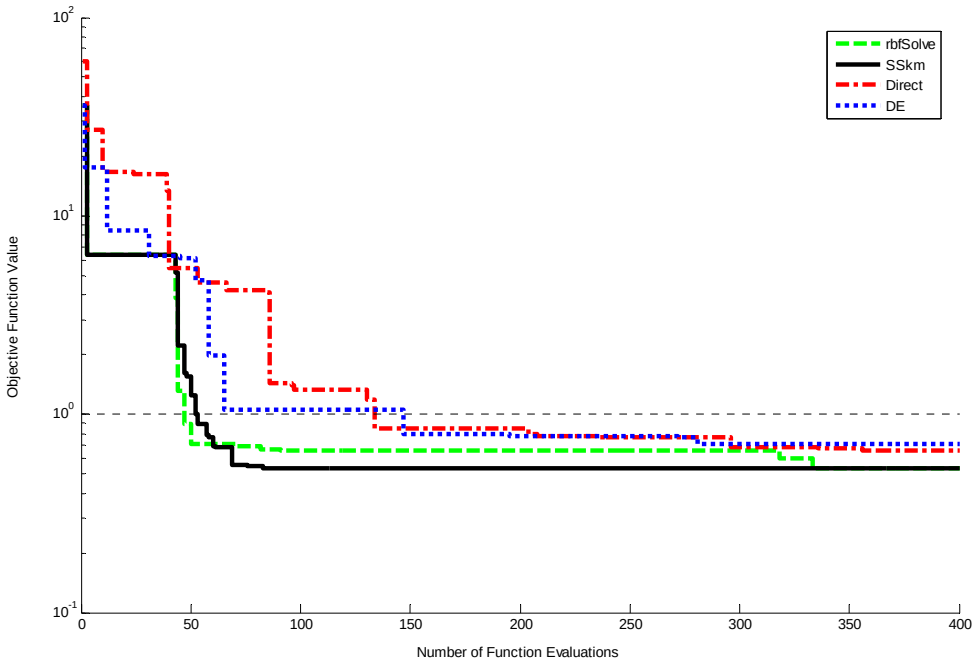


Figure 6. Convergence curves for the different optimization algorithms in the WWT plant PI Tuning

Conclusions and future work

In this paper a new global optimization algorithm for computationally costly problems has been presented. The algorithm is based on a synergistic approach combining the metaheuristic scatter search with the kriging interpolation technique. The evolutionary part of the algorithm provides efficient points for prediction and due to the statistical values provided by kriging, the method is able to filter low quality solutions. This allows a reduction in the number of function evaluations of the real model, making the algorithm suitable for solving computationally costly optimization problems. The capabilities and applicability of this new method have been illustrated with a challenging case study considering the integrated design and control of a waste water treatment plant.

Future research will be focused on the tuning of some parameters of the method, such as the minimum probability of improving the best observation demanded for evaluating a solution vector and the need of estimating the covariance parameters more or less often. The use of a dynamic Euclidean distance for avoiding duplication in *Refset* members may also be a key point of the algorithm. At present, a fixed distance is chosen by the user but a dynamic distance (larger at the beginning and smaller at the end of the optimization) may be more efficient, as well as its extension not only to *Refset* update phase but also to avoid function evaluation of solution vectors very close to others already evaluated.

Acknowledgments

The team at CSIC acknowledges financial support from the Spanish Government (MEC AGL2004-05206-C02-01/ALI) and Xunta de Galicia (PGIDIT05PXIC40201PN). Author Jose A. Egea gratefully acknowledges financial support (FPU fellowship) from the Spanish Ministry of Education and Science. Research by Rafael Martí is partially supported by the *Ministerio de Educación y Ciencia* under project code TIN2006-02696.

References

Banga, J.R., Moles, C.G., Alonso, A.A.: Global optimization of bioprocesses using stochastic and hybrid methods. In: Floudas, C.A., Pardalos, P.M. (eds.) Frontiers In Global Optimization. Nonconvex Optimization and Its Applications, vol. 74, pp. 45-70, Kluwer Academic Publishers, Hingham, MA, USA (2003)

Biegler, L.T., Grossmann, I.E.: Retrospective on optimization. Comput. Chem. Eng. **28(8)**, 1169-1192 (2004)

Copp, J. (ed.): The COST Simulation Benchmark – Description and Simulator Manual. COST (European Cooperation in the field of Scientific and Technical Research), Brussels (2001)

Cox, D.D., John, S.: SDO: A statistical method for global optimization. In: Alexandrov, N., Hussaini, M.Y. (eds.) Multidisciplinary Design Optimization: State of the Art, pp. 315-329. SIAM, Philadelphia (1997)

Egea, J.A., Rodríguez-Fernández, M., Banga, J.R., Martí, R.: Scatter search for chemical and bio-process optimization. J. Global Optim. **37(3)**, 481-503 (2007)

Floudas, C.A., Akrotirianakis, I.G., Caratzoulas, S., Meyer, C.A., Kallrath, J.: Global optimization in the 21st century: Advances and challenges. Comput. Chem. Eng. **29(6)**, 1185-1202 (2005)

Glover, F.: Heuristics for Integer Programming Using Surrogate Constraints. Decision Sci. **8**, 156-166 (1977)

Gutmann, H.M.: A radial basis function method for global optimization. J. Global Optim. **19(3)**, 201-227 (2001)

Holmström, K., Edvall, M.M.: The Tomlab Optimization Environment. In Kallrath, J. BASF AB (ed.) Modeling Languages in Mathematical Optimization, pp. 369-378. Kluwer Academic Publishers (2004)

Jones, D.R., Schonlau, M and Welch, W.J.: Efficient Global Optimization of expensive black-box functions. J. Global Optim. **13**, 455-492 (1998)

Jones, D.R.: A taxonomy of Global Optimization Methods Based on Response Surfaces. J. Global Optim. **21(4)**, 345-383 (2001a)

Jones, D.R.: DIRECT global optimization algorithm. .In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of optimization, pp. 431-440. Kluwer Academic Publishers, Dordrecht (2001b)

Laguna, M., Martí, R.: Scatter Search: Methodology and Implementations in C. Kluwer Academic Publishers, Boston (2003)

Matheron, G.: Principles of geostatistics. Econ. Geol.. **58**, 1246-1266 (1963)

Sasena, M.J., Papalambros, P., Goovaerts, P: Exploration of metamodeling sampling criteria for constrained global optimization. Eng. Optim. **34**, 263-278 (2002)

Stein, M.L.: Interpolation of Spatial Data: Some Theory for Kriging. Springer, New York (1999)

Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J. Global Optim. **11**, 341-359 (1997)

E. Vazquez: Modélisation comportementale de systèmes non-linéaires multivariables par méthodes à noyaux et applications. Ph.D. thesis, Paris XI Orsay University (2005)

Vecchia, A.V.: Estimation and model identification for continuous spatial processes. J. Royal Statist. Soc. **B(50)**, 297-312 (1998)

Yaglom, A.M.: Correlation Theory of Stationary and Related Random Functions I: Basic results. Springer Series in Statistics, Springer-Verlag, New York (1986)

The MathWorks Inc.: Optimization Toolbox for Use with Matlab®. User's guide. Version 2.

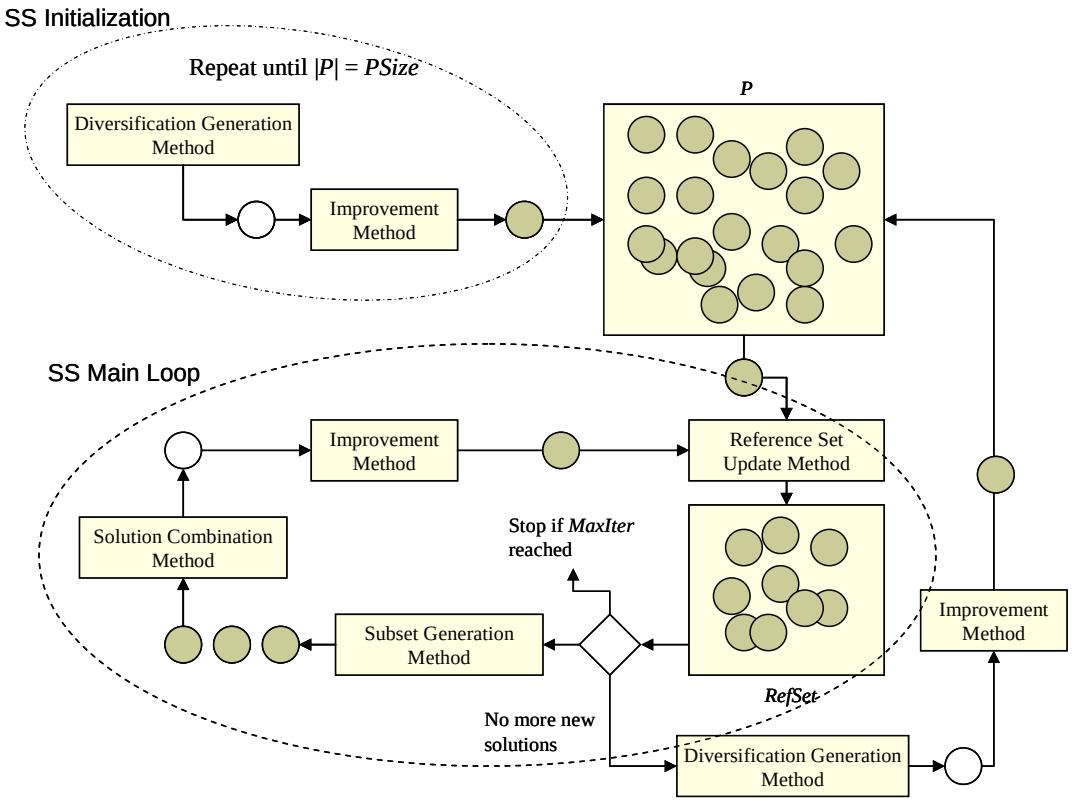


Figure 1: Schematic representation of a basic SS design. Shaded circles represent solutions that have been subjected to the *Improvement Method*

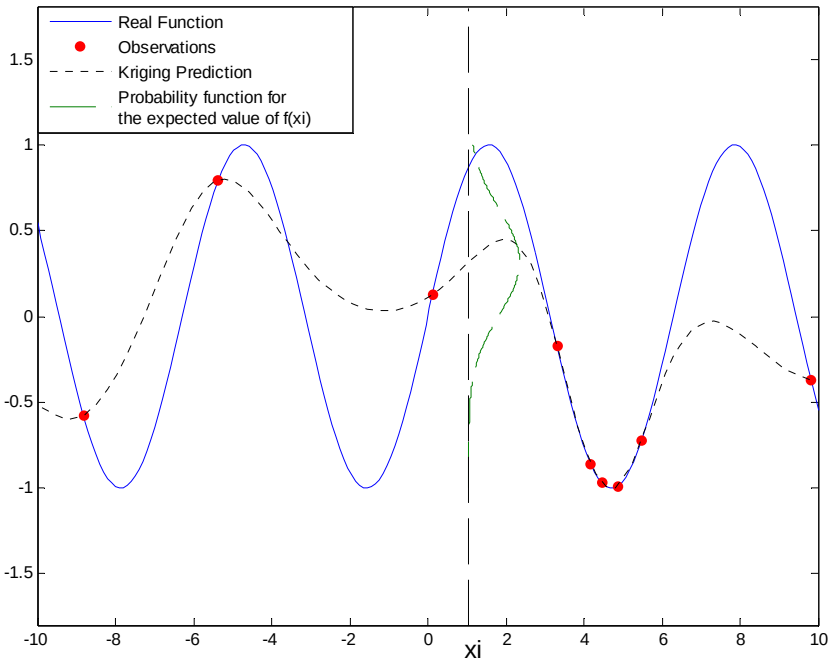


Figure 2: Kriging prediction for the function $y = sin(x)$ from a set of sampling points. The Gaussian distribution for point x_i provided by kriging is shown.

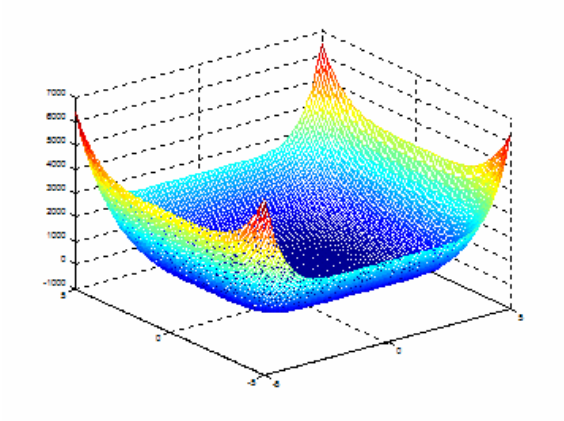


Figure 3a: *Six-hump camel-back* function

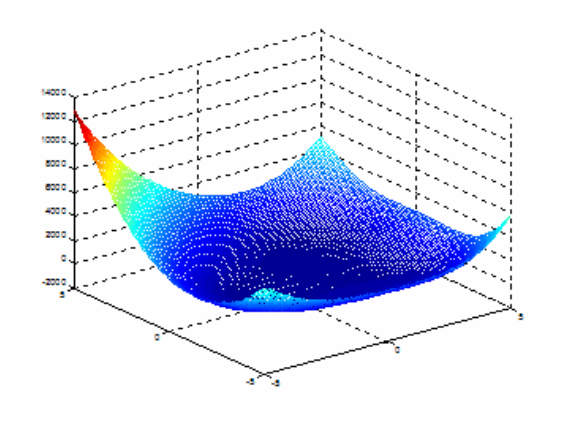


Figure 3b: Kriging prediction for $n_o = 20$

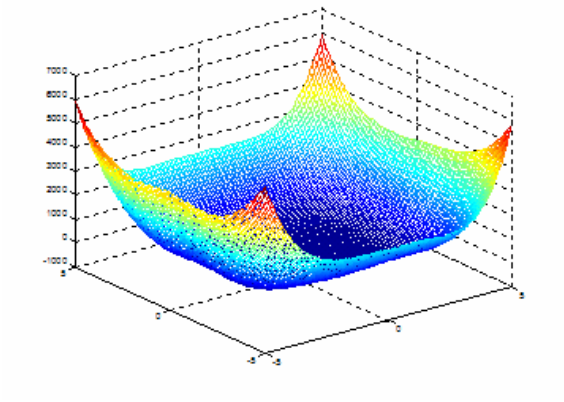


Figure 3c: Kriging prediction for $n_o = 50$

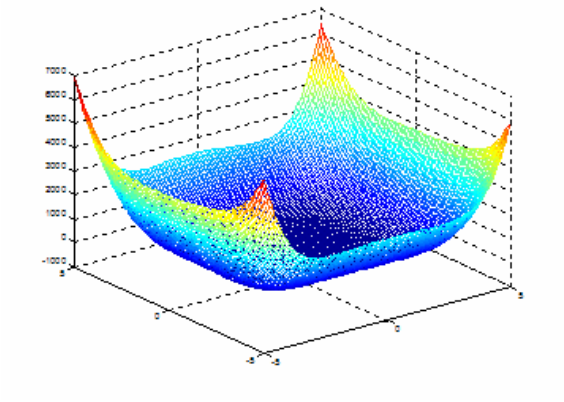
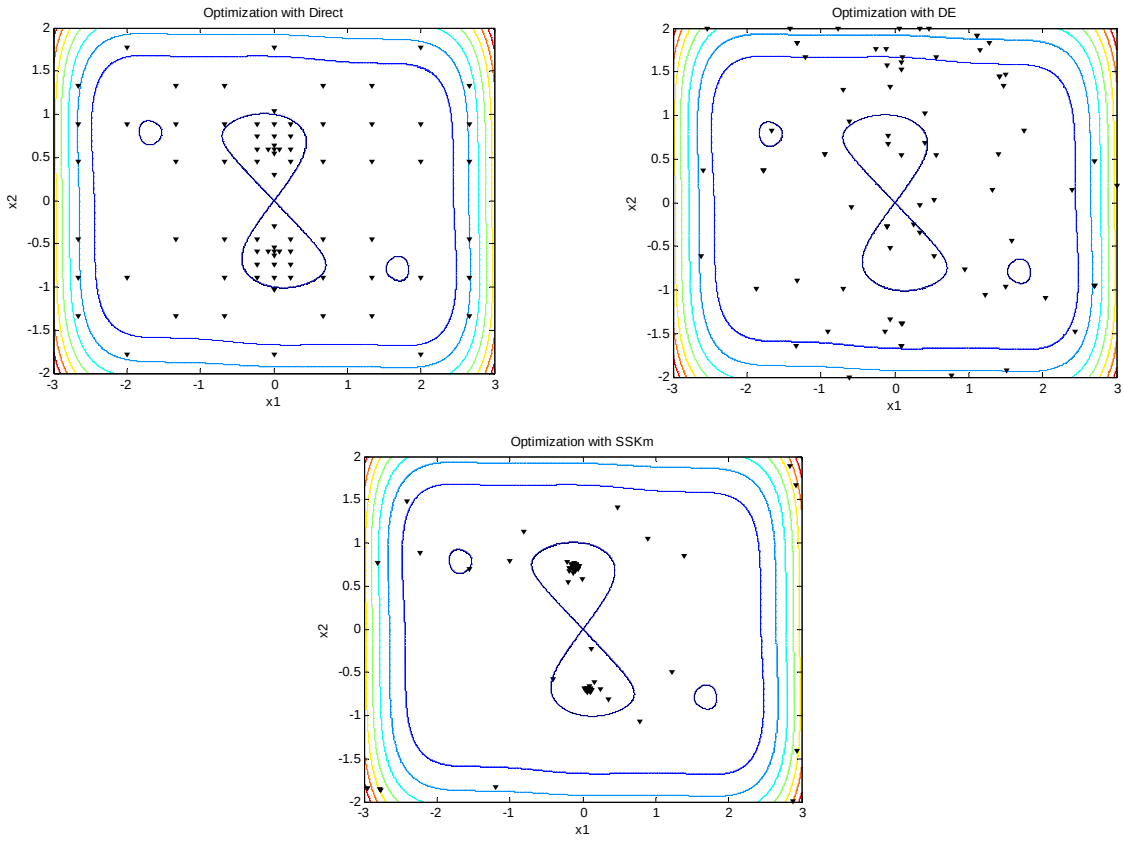


Figure 3d: Kriging prediction for $n_o = 100$



Figures 4. Contour plot of the *six hump camel* function with the evaluations done by: top left: *Direct*, top right: *DE*, bottom: *SSKm*

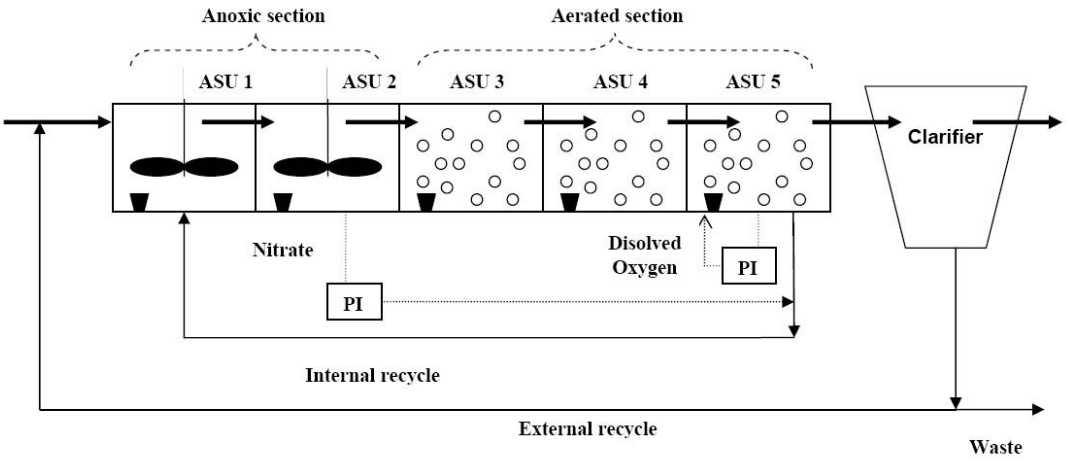


Figure 5. WWT plant layout

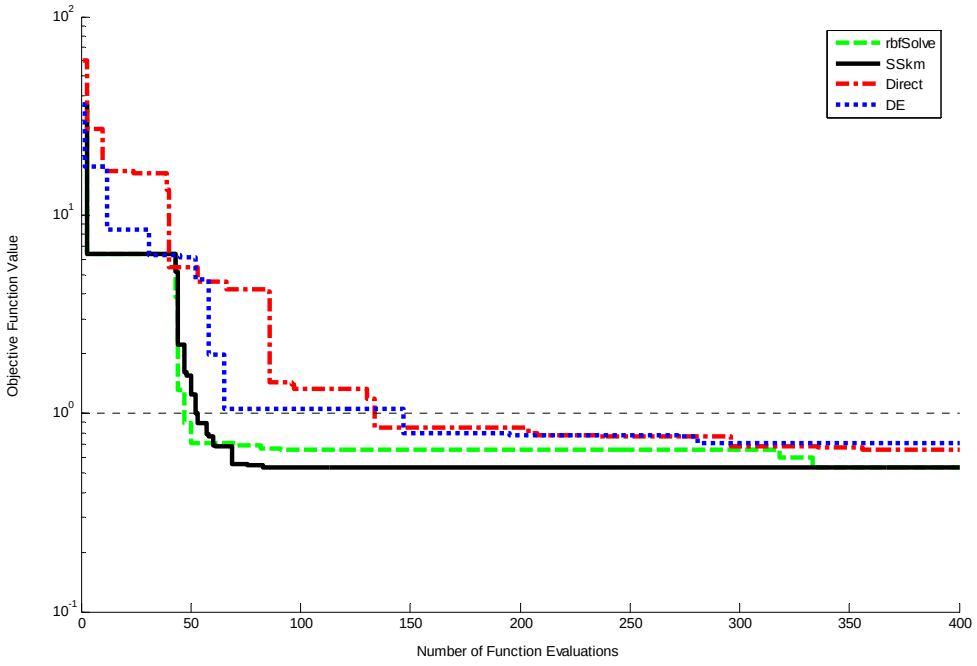


Figure 6. Convergence curves for the different optimization algorithms in the WWT plant PI Tuning