An evolutionary method for complex-process optimization

Jose A. Egea[a,*], Rafael Martí[b], Julio R. Banga[a]

[a] (Bio)Process Engineering Group, Instituto de Investigaciones Marinas (IIM-CSIC), Vigo, Spain
[b] Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain

Abstract

In this paper we present a new evolutionary method for complex-process optimization. It is partially based on principles of the scatter search methodology, but it makes use of innovative strategies to be more effective in the context of complex-process optimization using a small number of tuning parameters. In particular, we introduce a new combination method based on path relinking, which considers a broader area around the population members than previous combination methods. We also use a population-update method which improves the balance between intensification and diversification. New strategies to intensify the search and to escape from suboptimal solutions are also presented. The application of the proposed evolutionary algorithm to different sets of both state-of-the-art continuous global optimization and complex-process optimization problems reveals that it is robust and efficient for the type of problems intended to solve, outperforming the results obtained with other methods found in the literature.

Key words: evolutionary algorithms, complex-process optimization, continuous optimization, global optimization, metaheuristics

1. Introduction

Many optimization problems arising from engineering applications are described by complex mathematical models (e.g., sets of differential-algebraic equations). A general complex-process optimization problem may be formulated as follows:

Find $\mathbf{x}$ to minimize:

$$C = \phi(\dot{\mathbf{y}}, \mathbf{y}, \mathbf{x}) \tag{1}$$

*Corresponding author. Tel. +34 986 231 930 (ext. 231); fax: +34 986 292 762
Email addresses: jegea@iim.csic.es (Jose A. Egea), rafael.marti@uv.es (Rafael Martí), julio@iim.csic.es (Julio R. Banga)

Preprint submitted to Computers & Operations Research *January 27, 2009*

subject to

$$\mathbf{f}(\dot{\mathbf{y}}, \mathbf{y}, \mathbf{x}) = 0 \tag{2}$$

$$\mathbf{y}(t_0) = \mathbf{y}_0 \tag{3}$$

$$\mathbf{h}(\mathbf{y}, \mathbf{x}) = 0 \tag{4}$$

$$\mathbf{g}(\mathbf{y}, \mathbf{x}) \leq 0 \tag{5}$$

$$\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \tag{6}$$

where $\mathbf{x}$ is the vector of decision variables; C is the cost (objective function) to minimize; $\mathbf{f}$ is a functional describing the complex-process model (e.g., a system of differential algebraic equations); $\mathbf{y}$ is the vector of the states (and $\dot{\mathbf{y}}$ is its derivative); t_0 the initial time for the integration of the system of differential algebraic equations (and, consequently, $\mathbf{y}_0$ is the vector of the states at that initial time); $\mathbf{h}$ and $\mathbf{g}$ are possible equality and inequality constraint functions which express additional requirements for the process performance; and, finally, $\mathbf{x}^L$ and $\mathbf{x}^U$ are the upper and lower bounds for the decision variables.

Due to their complexity, these models have to be treated as "black-boxes" and they often present high nonlinearity and multimodality, thus the solution of this type of problems is usually a difficult task. Moreover, in many instances, complex-process models present noise and/or discontinuities which make traditional deterministic methods (e.g., gradient-based methods) inefficient to find the global solutions. Global optimization methods are robust alternatives to solve complex-process optimization problems. They can be roughly divided into deterministic (or exact) methods [1] and stochastic (or heuristic) methods [2]. Among stochastic methods, metaheuristics [3] and in particular population-based algorithms [4, 5], seem to be the most promising methods to deal with complex-process optimization since they usually provide excellent solutions (quite often the global optimum) in reasonable computation times. Some recent applications of population-based algorithms to complex-process optimization can be found in [6, 7, 8, 9, 10, 11, 12].

Here we propose an evolutionary method for global optimization of complex-process models, which employs some elements of two well-established methodologies: scatter search [13] and path relinking [14]. Regarding scatter search, the method uses a relatively small population size, partially chosen by a quality criterion from an initial set of diverse solutions. It also performs systematic combinations among the population members. Regarding path relinking, the new solutions are generated within the areas defined by every pair of solutions in the population, introducing a bias to generate new solutions which share more properties with the best population members than with the rest. However, we have introduced new strategies and modified some standard scatter search designs in such a way that we prefer to label our method as "Evolutionary Algorithm for Complex-process Optimization" (*EACOP*). Specifically, our contributions are:

- A small population without memory structures (repeated sampling is allowed).

- A new combination method based on wide hyper-rectangles.
- An aggressive population update for a quick convergence.
- A search intensification strategy called the "go-beyond".

On the other hand, our algorithm does not incorporate an improvement or local search method, as it is customary in scatter search and other population based methodologies. We have empirically found that in complex process optimization the marginal improvement obtained by the local search does not justify its inclusion in the algorithm, and its associated running time can be better invested in the generation and combination of solutions for a better overall performance.

This paper is organized as follows: Section 2 presents our proposed algorithm for complex-process optimization. Section 3 presents the results obtained by applying the methodology to different sets of benchmark problems and compare them with those obtained by applying other state-of-the-art methods. The paper finishes with some conclusions.

2. The evolutionary algorithm

In this section we present a novel evolutionary algorithm for optimization of complex-process models. It shares some elements of scatter search, but we have introduced a set of changes with respect to the classical SS design to make the algorithm more robust and efficient, obtaining a better balance between diversification and intensification (which is the key point of global optimization algorithms) and using less tuning parameters.

To illustrate how the algorithm works, during the following sections we will consider a 2-D dimensional unconstrained function to be minimized, shown as contour plots. In particular, we consider the function $f(x_1, x_2) = 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + 7\sin(0.5x_1)\sin(0.7x_1x_2)$ in the range $x_1 \in [-6, 6]$, $x_2 \in [-2, 7]$, which presents several minima.

2.1. Building the initial population

In this subsection we follow the standard SS design generating an initial set S of m diverse vectors (normally $m = 10 \times nvar$, being $nvar$ the problem size). Here we use a latin hypercube uniform sampling [15] to generate them. All these vectors are evaluated and the $b/2$ best ones in terms of quality (being b the population size) are selected as members of the initial population, Pop_0. For example, in a minimization problem, provided the diverse vectors are sorted according to their function values (the best one first), the initial selection is $Pop_0 = \left[x^1, x^2, \dots, x^{b/2}\right]^T$ with $x^i \in S$ and $i \in [1, 2, \dots, m]$, such that

$$f(x^i) \leq f(x^j) \ \ \forall \ \ j > i \ , \ i \in [1, 2, \dots, b/2 - 1] \ , \ j \in [2, 3, \dots, b/2] \qquad (7)$$

Pop_0 is completed selecting randomly $b/2$ additional vectors from the remaining $m - b/2$ diverse vectors in S. This completion strategy, although less sophisticated than others traditionally used in SS, which take into account relative distances to maximize the diversity of the solutions added to the initial population, has empirically shown to be as effective as the latter. Moreover, for large-scale optimization problems, more sophisticated strategies can lead to high computational efforts to calculate relative distances amongst vectors.

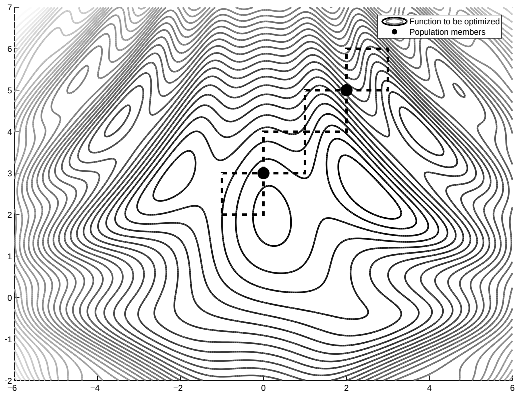
2.2. Combination method

After the initial population has been built, its solutions are sorted according to their quality (i.e., the best solution is the first) and the combination method is applied. In the context of SS, Laguna and Martí [13] checked that most of the quality solutions obtained by combination arise from sets of two solutions, thus, in our implementation, we restrict the combinations to pairs of solutions.

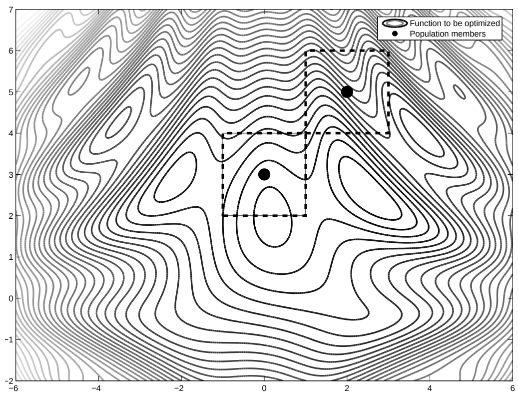
The combination method is a key element in many optimization algorithms. In evolutionary algorithms, this combination method is represented by the crossover and mutation operators. In the SS framework, linear combinations of two solutions were suggested by Glover [16]. Herrera et al. [17] studied different types of combination procedures for SS applied to continuous problems. They concluded that the BLX-α algorithm (with $\alpha = 0.5$) is a suitable combination method for continuous scatter search. Using concepts from path relinking, Laguna and Martí [18] already used this idea and extended it to avoid generating solutions in the same area by defining up to four different regions within and beyond the segments linking every pair of solutions. These authors changed the number of generated solutions from each pair of solutions in the population depending on their relative position. Ugray et al. [19] and Egea et al. [20] used the same principles, but instead of performing linear combinations between solutions, they performed a type of combination based on hyper-rectangles covering broader spaces and allowing different paths between pairs of solutions. However, these hyper-rectangles were created along the directions defined by every pair of population members, thus restricting possible promising search areas (Figure 1(a)). In our design, we define the hyper-rectangles around the population members, which allows the number of search directions to increase. Besides, we consider a larger area covered by the hyper-rectangles, which enhances diversification not only regarding search directions but also regarding search distance (Figure 1(b)).

The areas containing high quality solutions should be more deeply explored with respect to other areas. We therefore use the relative quality of every pair of solutions (regarding their position in the sorted population) as a measure of bias to create the hyper-rectangles.

Every population member defines $b - 1$ hyper-rectangles. A new solution is created inside every hyper-rectangle, which means that $b^2 - b$ new solutions are created in every iteration. It must be noted that the population members are sorted according to their function values (the best one first) in every iteration. Considering minimization, this means:



(a) *Egea et al. (2007)*



(b) Our algorithm

Figure 1: Hyper-rectangles defining the areas for generating new solutions

$$f(x^1) \leq f(x^2) \leq \ldots \leq f(x^b) \tag{8}$$

Let us consider a solution, x^i, to be combined with the rest of solutions in the population, x^j, $\forall i, j \in [1, 2, \cdots, b], i \neq j$. Two new points within the search space are defined:

$$c_1 = x^i - d\,(1 + \alpha \cdot \beta) \tag{9}$$

$$c_2 = x^i + d\,(1 - \alpha \cdot \beta) \tag{10}$$

where

$$d = \frac{x^j - x^i}{2}, \tag{11}$$

$$\alpha = \begin{cases} 1 & \text{if} \quad i < j \\ -1 & \text{if} \quad j < i \end{cases} \tag{12}$$

and

$$\beta = \frac{|j - i| - 1}{b - 2} \tag{13}$$

The new solution, x^{new}, will be created in the hyper-rectangle defined by c_1 and c_2:

$$x^{new} = c_1 + (c_2 - c_1) \bullet r \tag{14}$$

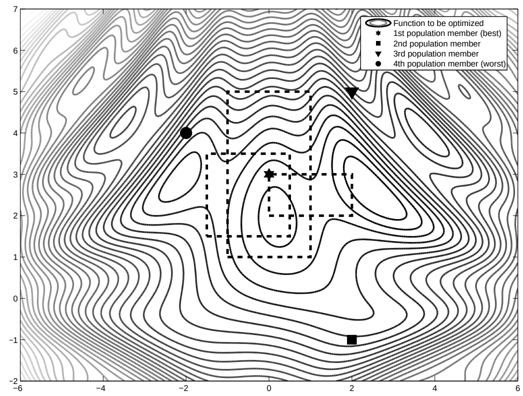
where r is a vector of dimension $nvar$ with all its components being uniformly distributed random numbers in the interval $[0, 1]$. The notation $(\bullet)$ above indicates an entrywise product (i.e., the vectors are multiplied component by component), thus it is not a scalar product.

"Bad" population members will generate new solutions close to "good" population members with higher probability whereas the latter will generate new solutions far from the former with higher probability. The higher the difference of quality between solutions, the higher the bias (β) is introduced. Figure 2(a) shows the hyper-rectangles generated by the best solution in the population. They are defined by its relative position with respect to the rest of solutions in the population: the higher the difference of quality, the further the hyper-rectangle from the "bad" solution is created. Similarly, Figure 2(a) shows the hyper-rectangles generated by the worst solution in the population. In this case, they are generated to create solutions close to high quality solutions with increasing probability according to their quality.

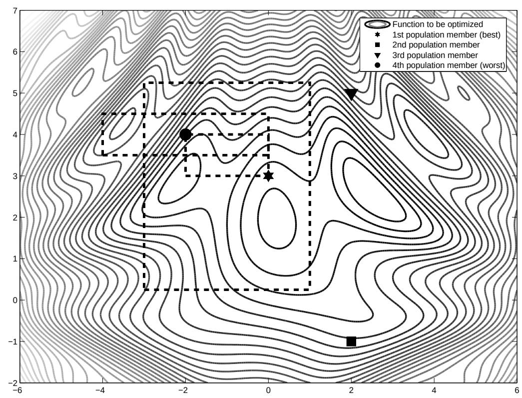
Although the incorporation of a memory structure is quite common in scatter search implementations to avoid combinations among population members previously combined, we have empirically found that our combination method based on wide hyper-rectangles and random sampling, benefits from multiple combinations of the same solutions. When the memory structure is present, the method does not explore any more a promising area around a pair of solutions if they did not generate a high quality solution in a previous iteration. However, we can consider the situation illustrated in Figure 3, in which the solution generated in iteration $i+1$ is much better than the generated in iteration i from the same parents (and could eventually be the best so far). For this reason, we ignore in our method this memory structure.

2.3. Population update

The most used strategies to update the population in evolutionary algorithms are the $(\mu + \lambda)$ and (μ, λ) updating schemes [21]. In the $(\mu + \lambda)$-ES the new population is selected by choosing μ solutions from the μ parents and λ



(a) Hyper-rectangles defined by the best population member



(b) Hyper-rectangles defined by the best population member

Figure 2: Biased hyper-rectangles

offspring from the previous generation. In the (μ, λ)-ES the new μ population members are selected from the λ offspring in the previous generation. In general, $(\mu + \lambda)$ updating strategies may rapidly converge to sub-optimal solutions in continuous problems, specially in the case of methods using a small number of population members, like scatter search or our proposed method. On the other hand, (μ, λ) strategies do not present this effect, but they may need a much higher number of function evaluations to achieve the optimal solutions. Here we propose a $(1+1)$ strategy applied to every population member, similar to that used in other evolutionary algorithms [22], which turns to be a good trade-off point between both methods in our context. It can be expressed by

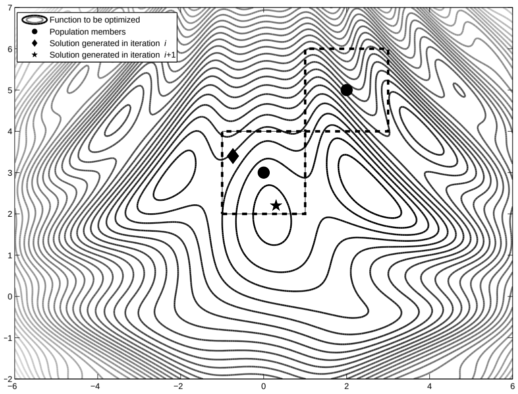


Figure 3: Two solutions generated in the same hyper-rectangle in two consecutive iterations

saying that *a solution can only enter the population by replacing its parent.*

As stated in Section 2.2, every population member is combined with the rest of population members, thus it performs $b-1$ combinations creating $b-1$ new solutions (the offspring). Amongst these new solutions, we identify the best one in terms of quality. If it outperforms its parent (i.e., the population member which was being combined), the former replaces the latter in the population. Provided the combination method mentioned above, this strategy acts by performing individual movements of the population members along the paths contained in the areas defined by each pair of solutions, instead of performing movements of the whole population at once as considered in $(\mu+\lambda)$ and (μ,λ) strategies. Although these individual movements are conditioned by the position and distance of the population members, we could consider that every solution follows a self-tuned annealing scheme, in which big steps are allowed at the beginning of the search whereas the solution moves much more locally in the end, due to the proximity of the population members in the final stages.

2.4. Exploiting promising directions: the go beyond strategy

We have implemented an advanced strategy to enhance the search intensification named the *go-beyond* strategy, which consists in exploiting promising directions. When performing the combination method all the new solutions created around a population member are sorted by quality. If the best of them outperforms its parent, a new non-convex solution in the direction defined by the child and its parent is created. The child becomes the new parent and the new generated solution is the new child. If the improvement continues, we might be in a very promising area, thus we apply this strategy again doubling the area for creating new solutions.

A straightforward question arises from the last paragraph: *how do we identify the parent of a generated solution*? As explained in Section 2.2, new solutions are created in hyper-rectangles defined by the pair of population members combined and around one of the solutions of the pair. The parent of a solution will be the population member around which the hyper-rectangle containing the new solution has been generated. Figure 4 depicts how the *go-beyond* strategy works: from a pair of population members, two new solutions are generated in the corresponding hyper-rectangles. The squared solution is the child whose parent is the population member closest to it. Since the child outperforms the parent in quality we apply the *go-beyond* strategy and consider a new hyper-rectangle (solid line) defined by the distance between the parent and the child. A new solution (triangle) is created in this hyper-rectangle. This new solution becomes the child and the old child (i.e., the squared solution) becomes the parent. Since the new child (triangle) outperforms again its parent (square), the process is repeated, but the size of the new hyper-rectangle created (dotted line) is double-sized because of the improvement experienced during two consecutive combinations. Finally, a new solution (starred) is created in an area very close to the global minimum. Algorithm 1 shows a pseudocode of the *go beyond* strategy procedure.

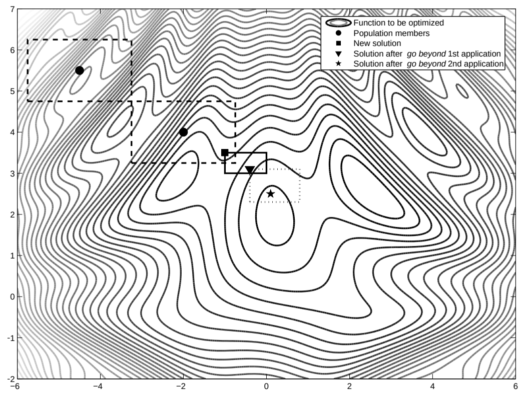
Algorithm 1 *go beyond* strategy

Apply the combination
for $i = 1$ to b **do**
  Identify the best child, $x_{best\_child}(i)$, outperforming its parent, $x_{parent}(i)$
  $x_{ch} = x_{best\_child}$
  $x_{pr} = x_{parent}$
  $improvement = 1$
  $\Lambda = 1$
  while $f(x_{ch}) < f(x_{pr})$ **do**
    Create a new solution, x_{child_new}, in the rectangle defined by $[x_{ch} - \frac{x_{pr}-x_{ch}}{\Lambda}, x_{ch}]$
    $x_{pr} = x_{ch}$
    $x_{ch} = x_{child\_new}$
    $improvement = improvement + 1$
    if $improvement = 2$ **then**
      $\Lambda = \Lambda/2$
      $improvement = 0$
    end if
  end while
end for

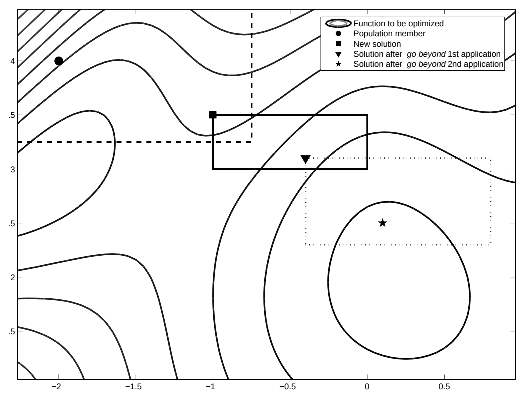
Although the *go-beyond* strategy has been mainly designed to enhance the search intensification, the fact that the size of the hyper-rectangles increases if the new solutions improve the old ones during consecutive iterations induces a diversification strategy, exploring regions where different minima can be found.

2.5. Escaping from local optima

Our algorithm does not implement any rebuilding mechanism as it is customary in advanced evolutionary designs [13] to replace the worst solutions



(a)



(b) (Zoom)

Figure 4: The *go-beyond* strategy

which are not likely to produce high quality offspring. Instead of this, we define a vector $\mathbf{n_{stuck}}$ which computes the number of consecutive iterations that every population member does not produce any new solution outperforming its function value. If the corresponding $n_{stuck}(i)$ value for a population member i exceeds a predefined number, *nchange*, we consider that this solution is stuck in a local optima and we replace it with another solution randomly generated within the search space. The number of consecutive iterations to perform the replacement will be experimentally determined in Section 3. When a population member is replaced, its $n_{stuck}(i)$ value is reset to zero.

Algorithm 2 summarizes in pseudo-code how our algorithm works.

Algorithm 2 Pseudo code of our algorithm

```
Set parameters
Initialize n_stuck
Create set of diverse solutions (latin hypercube)
Generate initial population with high quality and random solutions
repeat
  for i = 1 to b do
    Combine x^i with the rest of population members
    if best child outperforms x^i then
      Label x^i
      Apply go beyond strategy (Algorithm 1)
    end if
  end for
  Replace labeled population members by their corresponding best children and
  reset their corresponding n_stuck(i)
  Add one unit to the corresponding n_stuck(j) of the not labeled population mem-
  bers
  if any of the n_stuck values > nchange then
    Replace those population members by random solutions and reset their n_stuck
    values
  end if
until Stopping criterion is met
```

---

3. Computational experience

To test our algorithm's performance, we have carried out three different sets of experiments. In the first one we consider a set of 40 well known unconstrained global optimization problems of different dimensions (we will call them *LM* problems) that have usually been used as benchmark problems in the literature for testing optimization software [18, 23]. In this instance we will select a value for *nchange* (i.e., the number of consecutive iterations that a population member has not being updated before replacing it by a random solution). In the second set of experiments we will consider the set of 24 "never solved" functions used as benchmarks in the IEEE Congress on Evolutionary Computation 2005 (CEC'2005) [24]. In the final set of experiments we will consider two complex-process optimization problem arising from bioprocess engineering. In both the second and the third set of experiments we will compare our algorithm with other state-of-the-art global optimization methods.

All the computational experiments were conducted on a Pentium IV computer at 2.66 GHz. Both our algorithm and the methods used in the third set of experiments (see Section 3.3) were implemented in Matlab. Results for the second set of experiments (i.e., CEC'2005 problems) were taken from the references shown in Table 3.

The number of population members depends on the problem size in our algorithm. Here we generate approximately a number of new solutions of $10 \cdot nvar$ per iteration. This means that the number of population members is the first even number, n, which accomplishes $n^2 - n \geq 10 \cdot nvar$. Table 1 shows the number of population members used for the different dimensions considered in the test problems.

Population size	Problem dimension
6	2-3
8	4
10	6
12	10
16	20-24
18	25-30
22	40

Table 1: Number of population members used depending on the problem dimension

3.1. LM problems

The mathematical equations of the 40 test problems in the first data set are described in [18] and [23]. Table 2 provides information about all these problems.

Following the same procedure as in [18], we have defined an optimality gap as:

$$GAP = |f(x) - f(x^*)| \tag{15}$$

where x is a heuristic solution and x^* is the optimal solution. We say that a heuristic solution is satisfactory if:

$$GAP \leq \begin{cases} \varepsilon & \text{if} \quad f(x^*) = 0 \\ \varepsilon\,|f(x^*)| & \text{if} \quad f(x^*) \neq 0 \end{cases} \tag{16}$$

We set $\varepsilon = 0.001$. For each test function we performed 25 independent runs with a limit of 50000 function evaluations. We tested values of *nchange* from 1 to 50 and computed the following indexes:

- Number of different problems solved.
- Number of total problems solved (regarding the 25 runs per problem).
- Number of different solved problems in an independent run (and its frequency).

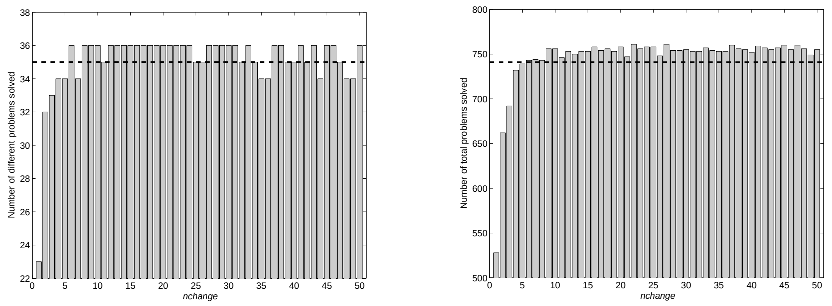
Figures 5 shows the influence of *nchange* over the number of different problems solved and the number of total problems solved considering the 25 runs per problem performed. The dashed lines represent the results obtained ignoring any type of replacement (i.e., for $nchange = \infty$).

Number of variables	Problem Number	Problem Name	x^*	$f(x^*)$
2	1	Branin	(9.42478, 2.475)[a]	0.397887
	2	B2	(0, 0)	0
	3	Easom	(π, π)	-1
	4	Goldstein and Price	(0, -1)	3
	5	Shubert	(-7.7083, -7.0835)[a]	-186.7309
	6	Beale	(3, 0.5)	0
	7	Booth	(1, 3)	0
	8	Matyas	(0, 0)	0
	9	SixHumpCamelback	(0.089840, -0.712659)[a]	-1.031628
	10	Schwefel(2)	(420.9687, 420.9687)	0
	11	Rosenbrock(2)	(1, 1)	0
	12	Zakharov(2)	(0, 0)	0
3	13	De Joung	(0, 0, 0)	0
	14	Hartmann(3,4)	(0.114614, 0.555649, 0.852547)	-3.862782
4	15	Colville	(1, 1, 1, 1)	0
	16	Shekel(5)	(4, 4, 4, 4)	-10.1532
	17	Shekel(7)	(4, 4, 4, 4)	-10.40294
	18	Shekel(10)	(4, 4, 4, 4)	-10.53641
	19	Perm(4,0.5)	(1, 2, 3, 4)	0
	20	Perm0(4,10)	(1, 1/2, 1/3, 1/4)	0
	21	Powersum	(1, 2, 2, 3)	0
6	22	Hartmann(6,4)	(0.20169, 0.150011, 0.47687, 0.275332, 0.311652, 0.6573)	-3.322368
	23	Schwefel(6)	(420.9687,. . . , 420.9687)	0
	24	Trid(6)	$x_i = i * (7 - i)$	-50
10	25	Trid(10)	$x_i = i * (11 - i)$	-210
	26	Rastrigin(10)	(0,. . . , 0)	0
	27	Griewank(10)	(0,. . . , 0)	0
	28	Sum Squares(10)	(0,. . . , 0)	0
	29	Rosenbrock(10)	(1,. . . , 1)	0
	30	Zakharov(10)	(0,. . . , 0)	0
20	31	Rastrigin(20)	(0,. . . , 0)	0
	32	Griewank(20)	(0,. . . , 0)	0
	33	Sum Squares(20)	(0,. . . , 0)	0
	34	Rosenbrock(20)	(1,. . . , 1)	0
	35	Zakharov(20)	(0,. . . , 0)	0
>20	36	Powell(24)	(3, -1, 0, 1, 3,. . . , 3, -1, 0, 1)	0
	37	Dixon and Price(25)	$x_i = 2^{-\frac{z-1}{z}}$, $z = 2^{i-1}$	0
	38	Levy(30)	(1,. . . , 1)	0
	39	Sphere(30)	(0,. . . , 0)	0
	40	Ackley(30)	(0,. . . , 0)	0

[a]This is one of several multiple optimal solutions.

Table 2: *LM* test problems

According to the results in Figure 5 we can conclude that the replacement described in Section 2.5 helps to obtain better results. However, it is not obvious to choose an optimal value for *nchange*. Values under 10 seem to provide poor results, whereas there is not a clear trend for the rest of values in the tested range. According to the criteria mentioned above, we have chosen a value of $nchange = 22$ because it is in the group of values solving the highest number



(a) Number of different problems solved (b) Number of total problems solved

Figure 5: Influence of *nchange*

of different problems (36), and it solves the highest number of total problems (761). A value of *nchange* = 27 provides the same results but it solves 32 problems in its best run (4 times out of 25) whereas the test with *nchange* = 22 solves 33 problems in 2 out of the 25 independent runs performed. Results in this experiment compare favorably with the results reported by Laguna and Martí [18], which tested different advanced scatter search designs reporting 30 different solved problems, and Hedar and Fukushima [23] which presented a directed tabu search method, reporting 32 different solved problems.

3.2. CEC'2005 problems

In this experiment we will consider some of the functions used as benchmarks in the IEEE Congress on Evolutionary Computation 2005 (CEC'2005) and described in [24]. In particular, these function are F_8, F_{13}, F_{14}, F_{16}, F_{17}, F_{18}, F_{19}, F_{20}, F_{21}, F_{22}, F_{23}, and F_{24} with dimensions $N = 10$ and $N = 30$, for a total of 24 test problems. These functions were reported in [25] under the section "Never solved multimodal functions" and are considered the most difficult instances used a global optimization benchmark problems up to now.

In our second experiment we run 25 independent times each instance and record the best, worst and mean value obtained considering all the runs. The budget of function evaluations is 100,000 for problems with dimension $N = 10$, and 300,000 for problems with dimension $N = 30$. We compare our results with those obtained by a set of methods, most of them based on hybrid evolutionary strategies, shown in Table 3.

Table 4 reports the sorted average of the minimum optimality gap (i.e., the gap of the best run out of 25) across the 24 instances.

In this second set of experiments, our algorithm achieves a value very close to *L-CMA-ES* (which is in the first place) for $N = 10$, and the best value for $N = 30$. These results reveal that our method is competitive for solving difficult problems.

Name	Description	Reference
BLX-GL50	Hybrid real coded genetic algorithm	[26]
BLX-MA	Real coded memetic algorithm	[27]
CoEVO	Cooperative co-evolutionary algorithm	[28]
DE	Differential evolution algorithm	[29]
DMS-L-PSO	Particle multi-swarm optimizer	[30]
EDA	Continuous estimation of distribution algorithm	[31]
G-CMA-ES	Covariance matrix adaptation evolution strategy	[32]
K-PCX	Population based steady-state algorithm	[33]
L-CMA-ES	Advanced local search evolutionary algorithm	[34]
L-SADE	Self adaptive differential evolution algorithm	[35]
SPC-PNX	Real parameter genetic algorithm	[36]

Table 3: Methods considered for the comparison

(a)

$N = 10$	
Method	Avg. GAP
L-CMA-ES	202.7
DE	203.4
L-SaDE	205.6
SPC-PNX	206.0
EACOP	208.3
DMS-L-PSO	244.4
EDA	249.8
G-CMA-ES	256.0
BLX-GL50	257.2
K-PCX	257.4
CoEVO	268.2
BLX-MA	306.2

(b)

$N = 30$	
Method	Avg. GAP
EACOP	385.1
L-CMA-ES	392.6
G-CMA-ES	402.1
BLX-MA	407.2
EDA	408.1
BLX-GL50	408.6
SPC-PNX	410.4
DE	412.6
K-PCX	419.3
CoEVO	549.2
L-SaDE	N/A
DMS-L-PSO	N/A

Table 4: Comparison over the "Never solved" *CEC'2005* test problems

3.3. Complex-process problems

In this last set of experiments we will consider two complex-process models arising from bioprocess engineering. For the sake of comparison, we have considered three methods for solving this type of problems:

- **DE:** *Differential Evolution.* This is a heuristic algorithm for the global optimization of nonlinear and (possibly) non-differentiable continuous functions presented by [22]. This population-based method handles stochastic variables by means of a direct search method which outperforms other popular global optimization algorithms, and it is widely used by the evolutionary computation community.

- **G-CMA-ES:** *Covariance Matrix Adaptation Evolutionary Strategy.* This is an evolutionary algorithm that makes use of the covariance matrix in a similar way to the inverse Hessian matrix in a quasi-Newton method, and it is particularly interesting for solving ill-conditioned and non-separable problems. This method [32] was ranked in the first place in the CEC'2005 (see Section 3.2) [25].

- **SSm:** *Scatter search for Matlab*. This advanced scatter search implementation was recently developed in the context of complex-process optimization, outperforming other state-of-the-art methods [20].

The problems considered in this set of experiments contain additional constraints apart from bound constraints in the decision variables. To handle them, we have modified the objective functions using a static penalty term. The objective function evaluated by the tested algorithms has the following form:

$$F(\mathbf{x}) = C(\mathbf{x}) + w \cdot \max\{\max\{viol(\mathbf{h}), viol(\mathbf{g})\}\} \tag{17}$$

where $\mathbf{x}$ is the vector of decision variables being evaluated, $C(\mathbf{x})$ is the original objective function value (Eq. 1), $\mathbf{h}$ is the set of equality constraints (Eq. 4) and $\mathbf{g}$ is the set of inequality constraints (Eq. 5). w is a penalization parameter selected by the user, which is constant during the optimization procedure (and usually has a high positive value). We use the $L-\infty$ norm of the constraints set to penalize the original objective function.

We have performed 10 independent runs for each instance and the best and mean values achieved by each method are reported.

3.4. Integrated design and control of a wastewater treatment plant

This case study represents a configuration of a real wastewater treatment plant placed in Manresa (Spain), as described by Moles et al. [37].

The overall model consists of 33 DAEs (14 of them are ODEs) and the optimization problem has 8 design variables. The integrated design problem is formulated as an NLP-DAEs, where the objective function to be minimized is a weighted sum of economic and controllability cost terms.

The minimization is subject to several sets of constraints:

- The 33 model DAEs (system dynamics), acting as differential-algebraic equality constraints.
- 32 inequality constraints which impose limits on some process magnitudes.
- An additional set of 120 double inequality constraints on the state variables.

To prove the inefficiency of local search methods for solving this problem we have applied a multistart procedure (using 100 different initial points) using a SQP method. The histogram of the local solutions found is shown in Figure 6. Only solutions with function values lower than 10000 are plotted in the histogram.

The histogram shows the practical non-convexity of the problem and the best value reported by the multistart ($f(x) = 1738.7$) is far from the best known solution of 1537.8 reported by Moles et al. [37] and Egea et al. [20].

Table 5 shows the results obtained by each algorithm in a budget of 15,000 function evaluations.

Every method finds the best known solution for this problem along the 10 runs performed, but only DE and EACOP find it in all the runs.

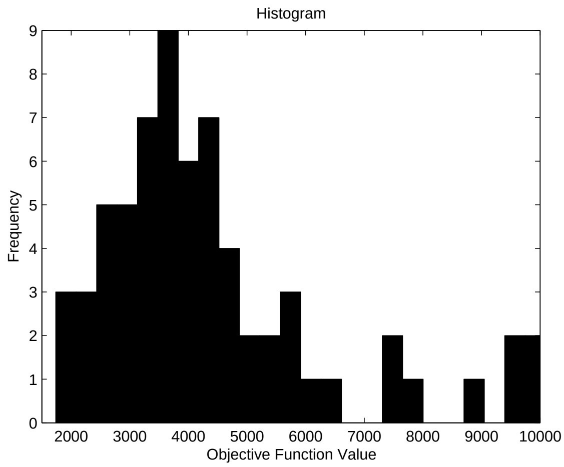


Figure 6: Histogram of solutions obtained from the multistart procedure for the integrated design and control problem

	DE	G-CMA-ES	SSm	EACOP
Best	1537.8	1537.8	1537.8	1537.8
Mean	1537.8	1540.7	1538.2	1537.8

Table 5: Results for the integrated design and control problem

3.5. Drying operation

This case study deals with the optimization of a bioproduct drying process, similar to the one formulated by Banga and Singh [38]. In particular, the aim is to dry a cellulose slab maximizing the retention of a nutrient (ascorbic acid). The dynamic optimization problem associated with the process consists of finding the dry bulb temperature along the time to maximize the ascorbic acid retention at the final time.

The models is described by a systems of partial differential equations (PDE's) which is transformed to a system of ODE's using a collocation method [39]. The number of decision variables for this problem is 40. Like in the previous example, we have applied a multistart procedure (using 100 different initial points) using a SQP method. The histogram of the local solutions found is shown in Figure 7. Only values corresponding to feasible solutions are presented.

Again, the histogram shows the practical non-convexity of the problem and the best value reported by the multistart is very far from the best known solution for this problem.

Table 6 shows the results obtained by each algorithm in a budget of 200,000 function evaluations.

In this example our algorithm obtains the best results regarding both best and mean values along the 10 runs performed (note that this is a maximization problem).

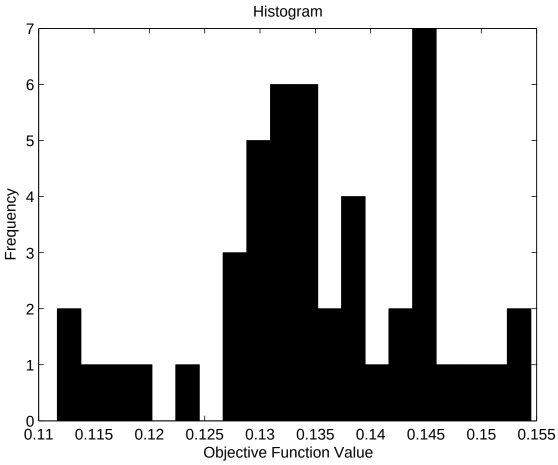


Figure 7: Histogram of solutions obtained from the multistart procedure for the drying operation problem

	DE	G-CMA-ES	SSm	EACOP
Best	0.1986	0.1995	0.1979	0.2001
Mean	0.1944	0.1975	0.1962	0.1991

Table 6: Results for the drying operation problem

Conclusions

We have developed an evolutionary method for optimization of complex-process models which makes use of some elements of the scatter search and path relinking metaheuristics. However, our method incorporates several innovative mechanisms and strategies that constitute a different evolutionary design.

We have applied the proposed methodology over different sets of nonlinear global optimization problems. For the first set of problems, the results outperformed those found in the literature. For the second set of problems (i.e., the "never solved" problems of the *CEC'2005* conference), our algorithm ranks in the first positions regarding the minimum gap with respect to the global solution compared to other state-of-the-art solution methods. In the third set of experiments we consider two complex-process models and our algorithm is competitive with previous methods. In summary, our proposed method proves to be efficient for solving complex-process models, and it is specially interesting in those cases in which standard local search methods fail to locate the global solution.

Acknowledgments

The team at CSIC acknowledges financial support from EU project BaSysBio LSHG-CT-2006-037469 and Spanish MICINN project MultiSysBio DPI2008-

06880-C03-02. Research by Rafael Martí has been partially supported by the *Ministerio de Ciencia e Innovación* of Spain (Grant Ref. TIN2006-02696).

References

[1] C. A. Floudas, Deterministic Global Optimization: Theory, Methods and Applications, Vol. 37 of Nonconvex Optimization and Its Applications, Kluwer Academic Publisher, 2000.

[2] A. Zhigljavsky, A. Zilinskas, Stochastic Global Optimization, Vol. 9 of Springer Optimization and Its Applications, Springer, 2008.

[3] F. W. Glover, G. A. Kochenberger, Handbook of Metaheuristics (International Series in Operations Research & Management Science), Springer, 2003.

[4] D. V. Arnold, H. G. Beyer, A comparison of evolution strategies with other direct search methods in the presence of noise, Computational Optimization and Applications 24 (1) (2003) 135–159.

[5] D. V. Arnold, H. G. Beyer, On the benefits of populations for noisy optimization, Evolutionary Computation 11 (2) (2003) 111–127.

[6] C. O. Ourique, E. C. Biscaia Jr., J. C. Pinto, The use of particle swarm optimization for dynamical analysis in chemical processes, Computers & Chemical Engineering 26 (12) (2002) 1783–1793.

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

[8] D. Sarkar, J. Modak, Optimization of fed-batch bioreactors using genetic algorithm: Multiple control variables, Computers & Chemical Engineering 28 (5) (2004) 789–798.

[9] G. Onwubolu, B. Babu, New Optimization Techniques in Engineering, Springer, 2004.

[10] B. Zhang, D. Chen, W. Zhao, Iterative ant-colony algorithm and its application to dynamic optimization of chemical process, Computers & Chemical Engineering 29 (10) (2005) 2078–2086.

[11] S. Balku, R. Berber, Dynamics of an activated sludge process with nitrification and denitrification: Start-up simulation and optimization using evolutionary algorithm, Computers & Chemical Engineering 30 (3) (2006) 490–499.

[12] R. Angira, A. Santosh, Optimization of dynamic systems: A trigonometric differential evolution approach, Computers & Chemical Engineering 31 (9) (2007) 1055–1063.

[13] M. Laguna, R. Martí, Scatter Search: Methodology and Implementations in C, Kluwer Academic Publishers, Boston, 2003.

[14] F. Glover, M. Laguna, R. Martí, Scatter search and path relinking: Foundations and advanced designs, in: G. C. Onwubolu, B. V. Babu (Eds.), New Optimization Techniques in Engineering, Springer-Verlag, 2004, pp. 87–100.

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

[16] F. Glover, Tabu search for nonlinear and parametric optimization (with links to genetic algorithms), Discrete Applied Mathematics 49 (1-3) (1994) 231–255.

[17] F. Herrera, M. Lozano, D. Molina, Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies, European Journal of Operational Research 169 (2) (2006) 450–476.

[18] M. Laguna, R. Martí, Experimental testing of advanced scatter search designs for global optimization of multimodal functions, Journal of Global Optimization 33 (2) (2005) 235–255.

[19] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, R. Martí, A multistart scatter search heuristic for smooth NLP and MINLP problems, in: C. Rego, B. Alidaee (Eds.), Adaptive Memory and Evolution: Tabu Search and Scatter Search, Kluwer Academic Publishers, 2005, pp. 25–58.

[20] J. A. Egea, M. Rodríguez-Fernández, J. R. Banga, R. Martí, Scatter search for chemical and bio-process optimization, Journal of Global Optimization 37 (3) (2007) 481–503. doi:10.1007/s10898-006-9075-3.

[21] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, 1992.

[22] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.

[23] A.-R. Hedar, M. Fukushima, Tabu search directed by direct search methods for nonlinear global optimization, European Journal of Operational Research 170 (2) (2006) 329–349.

[24] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization, Tech. rep., Nanyang Technological University of Singapore (2005).

[25] N. Hansen, Compilation of results on the 2005 cec benchmark function set, Tech. rep., CoLAB Institute of Computational Sciences, ETH, Zurich (2006).

[26] C. García-Martínez, M. Lozano, Hybrid real-coded genetic algorithms with female and male differentiation, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 896–903.

[27] D. Molina, F. Herrera, M. Lozano, Adaptive local search parameters for real-coded memetic algorithms, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 888–895.

[28] P. Posik, Real-parameter optimization using the mutation step co-evolution, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 872–879.

[29] J. Rönkkönen, S. Kukkonen, K. Price, Real-parameter optimization with differential evolution, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 506–513.

[30] J. Liang, P. Suganthan, Dynamic multi-swarm particle swarm optimizer with local search, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 522–528.

[31] B. Yuan, M. Gallagher, Experimental results for the special session on real-parameter optimization at cec 2005: a simple, continuous eda, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 1792–1799.

[32] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005a, pp. 1769–1776.

[33] A. Sinha, S. Tiwari, K. Deb, A population-based, steady-state procedure for real-parameter optimization, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 514–521.

[34] A. Auger, N. Hansen, Performance evaluation of an advanced local search evolutionary algorithm, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005b, pp. 1777–1784.

[35] A. Qin, P. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 1785–1791.

[36] P. Ballester, J. Stephenson, J. Carter, K. Gallagher, Real-parameter optimization performance study on the cec-2005 benchmark with spc-pnx, in: Procs. of 2005 IEEE Congress on Evol. Comput. (CEC'2005), 2005, pp. 498–505.

[37] C. G. Moles, G. Gutierrez, A. A. Alonso, J. R. Banga, Integrated process design and control via global optimization: a wastewater treatment plant case study, Chemical Engineering Research & Design 81 (2003) 507–517.

[38] J. R. Banga, R. P. Singh, Optimization of air drying of foods, Journal of Food Engineering 23 (2) (1994) 189–211.

[39] W. E. Schiesser, The numerical method of lines, Academic Press, New York, USA, 1991.