On Challenging Techniques for Constrained Global Optimization

Isabel A. C. P. Espírito Santo, Lino Costa, Ana Maria A. C. Rocha, M. A. K. Azad and Edite M. G. P. Fernandes

Abstract This chapter aims to address the challenging and demanding issue of solving a continuous nonlinear constrained global optimization problem. We propose four stochastic methods that rely on a population of points to diversify the search for a global solution: genetic algorithm, differential evolution, artificial fish swarm algorithm and electromagnetism-like mechanism. The performance of different variants of these algorithms is analyzed using a benchmark set of problems. Three different strategies to handle the equality and inequality constraints of the problem are addressed. An augmented Lagrangian-based technique, the tournament selection based on feasibility and dominance rules, and a strategy based on ranking objective and constraint violation are presented and tested. Numerical experiments are reported showing the effectiveness of our suggestions. Two well-known engineering design problems are successfully solved by the proposed methods.

1 Introduction

The problem that is addressed in this chapter is a continuous nonlinear constrained global optimization problem with the general form

$$\min_{x\in\Omega} f(x), \text{ subject to } h(x)=0\,,\ g(x)\leq 0\,, \tag{1}$$

where some of the functions $f:\mathbb{R}^n\to\mathbb{R}$, $h:\mathbb{R}^n\to\mathbb{R}^m$ and $g:\mathbb{R}^n\to\mathbb{R}^p$ are nonlinear and $\Omega=\{x\in\mathbb{R}^n : l\leq x\leq u\}$. The function $f(x)$ is the objective function, the equa-

Isabel A. C. P. Espírito Santo, Lino Costa and Ana Maria A. C. Rocha
Department of Production and Systems, University of Minho, 4710-057 Braga, Portugal,
e-mail: {iapinho,lac,arocha}@dps.uminho.pt

M. A. K. Azad and Edite M. G. P. Fernandes
Algoritmi R&D Centre, University of Minho, 4710-057 Braga, Portugal,
e-mail: {akazad,emgpf}@dps.uminho.pt

tions $h_j(x) = 0, j = 1,\ldots,m$ and the inequalities $g_j(x) \leq 0, j = 1,\ldots,p$ define the equality and inequality constraints of the problem, respectively. The vectors l and u are the lower and upper bounds, respectively, on the continuous decision variables x, and n represents the number of variables. The feasible region of the problem is the set $\mathscr{F} = \{x \in \mathbb{R}^n : h(x) = 0, g(x) \leq 0, l \leq x \leq u\}$. A point x is a feasible point if $x \in \mathscr{F}$. If the function f is not assumed to be convex then the problem (1) may have many minima in the feasible region. In this chapter, we look for a minimizer, $x^* \in \mathscr{F}$, such that $f(x^*) \leq f(x)$ for all feasible points $x \neq x^*$. This class of global optimization problem arises frequently in real-world applications of different fields. Specially for large-scale problems and non-smooth problems, derivative-free and stochastic methods are the most appropriate methods. Some well-known derivative-free methods like the deterministic pattern search method [35, 36] cannot guarantee convergence to the global minimum. Stochastic algorithms based on a point-to-point search, like the simulated annealing [28], and on a population of points, like particle swarm optimization [42], evolutionary algorithm [57], genetic algorithm [20, 25], differential evolution [4, 6, 38, 54], electromagnetism-like mechanism [2, 50] and artificial fish swarm algorithm [51] are available in the literature. Global optimization algorithms are usually divided into two main classes, the deterministic and the stochastic one. Deterministic methods provide a theoretical guarantee of locating the global minimizer, or at least an approximate global minimizer to within a prescribed tolerance, in a finite number of steps. The DIviding RECTangles based method [32], branch-and-bound based methods [10, 29] and interval analysis based techniques [27, 29, 59] are the most known in the literature. Stochastic methods, on the other hand, incorporate probabilistic (stochastic) elements, either in the problem data, for instance in the objective function and constraints, or in the algorithm itself, or in both. The convergence proofs for this type of methods involve the use of probability theory [51]. Stochastic algorithms are guaranteed to converge in infinite steps with probability one, but in practice there is no guarantee that the obtained solution will actually be the global optimum, or by how far the algorithm has failed to locate the true global optimum. Because global optimization problems are undecidable on unbounded search spaces and NP-hard on bounded spaces, there may exist practical limits during the solution process. For example, methods based on analytic transformations lead to transformed problems of exponentially increasing size, while the branch-and-bound method may split the problem into an exponential number of subproblems. Usually, stochastic methods are faster in locating a global optimum than deterministic ones. For the study herein presented, four well-known stochastic methods were chosen. They are population-based methods and mainly inspired by evolutionary and swarm intelligence theories.

In most global optimization algorithms (both deterministic and stochastic) it is possible to identify two separate phases. The exhaustive exploration of the search space is delegated to the global phase and most advanced techniques use stochastic methods to search for promising regions where global minima do exist. Then, the algorithms employ deterministic methods to exploit locally the promising regions found so far. This is known as the local phase of the algorithms.

While addressing the issue of solving global optimization problems we will analyze the practical behavior of the chosen solution methods and compare results with others in the literature. A set of benchmark simple bound constrained problems and another one with general constrained problems are used in the comparisons.

This chapter is structured in five sections. Section 2 aims at describing some popular and interesting population-based stochastic methods for simple bound constrained global optimization: (i) genetic algorithm; (ii) differential evolution; (iii) artificial fish swarm algorithm; (iv) electromagnetism-like mechanism. Section 3 is devoted to describing and discussing approaches for effective constraint-handling, namely: augmented Lagrangian-based techniques, tournament selection based on feasibility and dominance rules, and the ranking of objective function and constraint violation. A numerical comparison with other methods available in the literature, when solving two well-known engineering design problems, is shown in Section 4. We conclude the chapter in Section 5.

2 Population-Based Methods for Bound Constrained Problems

In population-based methods, a set (population) of candidate solutions (points) is randomly generated at the first iteration and is manipulated throughout the iterative process by means of a class of operations. The used operations depend on the theoretical principles that define the algorithmic structure, for example, they may use nature-inspired principles, copy physical processes and follow swarm intelligence principles. Although they may be computationally demanding, they are the most capable of performing the exploration of the search space for global minima, when compared with point-to-point search methods.

In the remaining part of the chapter, the following notation is used: $x^i \in \mathbb{R}^n$ denotes the ith point of a population; x^{best} is the point that has the least objective function value, when compared with all the other points in the set $\mathscr{F}$; $x^i_j \in \mathbb{R}$ is the jth ($j = 1, \ldots, n$) component of the point x^i of the population and p_{size} is the number of points in the population. Hereafter PI represents the set of indices $\{1, 2, \ldots, p_{\text{size}}\}$ and a random number ξ uniformly distributed between 0 and 1 is represented by $\xi \sim \text{U}[0, 1]$. In the iterative processes k represents the iteration counter.

2.1 Genetic algorithm

A Genetic Algorithm (GA) is a population-based algorithm that uses techniques inspired by evolutionary biology such as inheritance, selection, crossover and mutation [26]. GAs start from a population of points of size p_{size}. Traditionally, the potential optimal solutions to the optimization problem are represented as binary strings, but other encodings are also possible such as real representation of solutions for continuous problems. In this implementation, a real instead of a binary

representation is used since it is more suitable for continuous problems. Therefore, each point of the population x^i, for $i \in \text{PI}$, is an n dimensional vector. The initial population is randomly generated. A fitness function is defined in order to compare the points of the population (in terms of the objective function value alone if the problem is unconstrained and in terms of objective function value and constraint violation when the problem has equality and inequality constraints). A stochastic selection is implemented, guaranteeing that better points are more likely to be selected. New points in the search space are generated by the application of genetic operators - crossover and mutation - to the selected points from the population. Mutation introduces diversity in the population since crossover, exclusively, could not assure the exploration of new regions of the search space. Elitism is implemented by maintaining, during the search, a given number e, of the best points in the population.

Crossover combines two points in order to generate new ones. A Simulated Binary Crossover (SBX) [17], that simulates the working principle of single-point crossover operator for binary strings, is implemented. Two points, x^{r_1} and x^{r_2}, are randomly selected from the pool, i.e., indices r_1, r_2 are randomly chosen from the set of indices that contains the remaining $p_{\text{size}} - e$ points, mutually different, and with probability p_{cross}, two new points, w^{r_1} and w^{r_2}, are generated according to

$$\begin{aligned} w_j^{r_1} &= 0.5\left((1+\beta_j)x_j^{r_1} + (1-\beta_j)x_j^{r_2}\right) \\ w_j^{r_2} &= 0.5\left((1-\beta_j)x_j^{r_1} + (1+\beta_j)x_j^{r_2}\right), \end{aligned} \quad \beta_j = \begin{cases} (2\xi_j)^{\frac{1}{\eta_{\text{cross}}+1}} & \text{if } \xi_j \leq 0.5 \\ \left(\dfrac{1}{2(1-\xi_j)}\right)^{\frac{1}{\eta_{\text{cross}}+1}} & \text{if } \xi_j > 0.5 \end{cases}$$

for $j = 1, \ldots, n$, where $\xi_j \sim \text{U}[0,1]$ and $\eta_{\text{cross}} > 0$ is an external parameter of the distribution of β_j. This procedure is repeated until the number of generated points equals the number of points in the pool.

A Polynomial Mutation is applied, with a probability p_{mut}, to the points produced by the crossover operator. This operator guarantees that the probability of creating a new point v^i, closer to the previous one w^i ($i \in \text{PI}$) is greater than the probability of creating one away from it. It can be componentwise expressed by:

$$v_j^i = w_j^i + (u_j - l_j)\iota_j, \quad \iota_j = \begin{cases} (2\xi_j)^{\frac{1}{\eta_{\text{mut}}+1}} - 1 & \text{if } \xi_j < 0.5 \\ 1 - (2(1-\xi_j))^{\frac{1}{\eta_{\text{mut}}+1}} & \text{if } \xi_j \geq 0.5 \end{cases}$$

for $j = 1, \ldots, n$, where $\xi_j \sim \text{U}[0,1]$ and $\eta_{\text{mut}} > 0$ is an external parameter of the distribution of ι_j and l_j and u_j are the bounds of the decision variables ($l_j \leq x_j \leq u_j$). The GA proceeds according to Algorithm 1.

Modified GA with diversity preserving mechanism. Maintaining diversity among solutions in population is an important task, which will allow a crossover operator to constantly search for better solutions. There exist a number of ways to maintain and promote diversity in a population such as the niching methods or sharing mechanisms [18] and the mutation [26]. In this experiment, we introduce a niching method to preserve diversity among the solutions in the population. A simple niching stra-

Algorithm 1 Genetic algorithm

1: Randomly generate $x^i \in \Omega$, for $i \in$ PI; set $k = 0$; set parameters
2: **While** the stopping condition is not met **do**
2.1 evaluate the population and select the best e points
2.2 select by tournaments $p_{\text{size}} - e$ points from the population
2.3 apply SBX crossover to $p_{\text{size}} - e$ points
2.4 apply mutation to $p_{\text{size}} - e$ points
2.5 replace the worst $p_{\text{size}} - e$ points
2.6 set $k = k + 1$

tegy is implemented in the tournament selection operator. When comparing two solutions, x^{i_1} and x^{i_2}, a normalized distance d_{i_1,i_2} is measured between them. If this distance is smaller than a critical distance σ, the solutions are compared with their objective function values. Otherwise, they are not compared and another solution x^{i_3} is checked. If a specified number (n_t) of solutions are checked and none of them satisfy the critical distance, the i_1th solution is declared as winner. The normalized distance is calculated in the variable space by $d_{i_1,i_2} = \left(\sum_{j=1}^{n} (x_j^{i_1} - x_j^{i_2})^2/(u_j - l_j)\right)^{1/2}$, where l and u are the bounds of the decision variables. Thus, the solutions that are far away from each other are not compared and diversity among solutions in population is preserved.

Benchmark problems. We use three classical and well-known nonlinear optimization benchmark problems [21] (see Table 1) to compare the performance of various solution methods. The table reports the name of the problem, the objective function $f(x)$, the number of variables 'n', the default set Ω, the known (global) optimal solution 'f_{opt}', the global minimizer and the number of local minima. To observe the consistency of the outcome due to the stochastic nature of the algorithms, we run each problem 30 times. Each run uses a different seed to randomly generate the starting points of the population in the set Ω. Population size is set to be dependent on the dimension of the problem $p_{\text{size}} = \min\{100, 10n\}$.

Stopping condition. All the algorithms presented hereafter in this section stop when the best solution, $f(x^{\text{best}})$, is within 0.01% accuracy of the known optimal solution f_{opt}, in relative terms, or the number of function evaluations exceeds a limit $nf_{\max}$, i.e., when

$$\left| f(x^{\text{best}}) - f_{\text{opt}} \right| \leq \varepsilon^* \left| f_{\text{opt}} \right| + (\varepsilon^*)^2 \quad \text{OR} \quad nf_{\text{eval}} > nf_{\max} \tag{2}$$

holds, where nf_{eval} is the number of function evaluations required to reach the optimal solution with a specified tolerance. In this section, we set ε^* to 10^{-4} and $nf_{\max} = 50000$. Based on this stopping condition, a comparison of several solution methods may be performed using the three criteria:

- 'accuracy', which measures how close the method gets to the known optimal solution;

- 'efficiency', which is measured by the required number of function evaluations to reach the optimal solution;
- 'reliability', which is measured by the number of successful runs.

Here, a run is called a 'success' if the best solution obtained by the algorithm is within 0.01% accuracy of the known optimal solution.

Results. The results presented in subsequent tables refer to the best of the best function values obtained in the 30 runs, 'f_{best}', the average value of the best solutions in the 30 runs, 'f_{avg}', the standard deviation of the best solutions, 'st. dev.', and the average number of function evaluations obtained over the 30 runs, 'nfe_{avg}'.

Table 1 Selected test problems for bound constrained optimization.

Problem	
Goldstein and Price	$f(x) = (1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2))$ $(30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2))$
(GP) $n=2$	$\Omega = [-2,2]^2$, f_{opt} = 3, global at $(0,-1)$, 4 locals
Modified Himmelblau	$f(x) = (x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2+0.1((x_1-3)^2+(x_2-2)^2)$
(MHB) $n=2$	$\Omega = [-6,6]^2$, f_{opt} = 2.8379e-11, global at $(3,2)$, 4 locals
Rastrigin	$f(x) = 10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i))$
(RA-n) $n=2,5,10$	$\Omega = [-5.12,5.12]^n$, f_{opt} = 0, global at $(0,\ldots,0)$, >50 locals

Table 2 shows the results obtained by a basic GA framework, as outlined in Algorithm 1, as well as those obtained by the modified GA with diversity preserving mechanism. The values set to the parameters of the algorithms are displayed at the bottom row of the table. We may observe that both GA reached the 50000 function evaluations in all 30 runs (0% of successful runs) with the problem RA-10, but solved the remaining problems GP, MHB, RA-2 and RA-5 with 100% of successful runs. The modified GA needs in general less function evaluations to achieve the required accuracy.

To show the evolution of the population of solutions throughout the iterative process and how they converge to the global minimum, Fig. 1 shows three plots relative to the problem RA-2. For a better view, the population here has only twenty points. The plot on the left, shows the population spread all over the set $[-5.12,5.12]^2$ at the initial iteration; the plot in the center shows the population at iteration 40 and the plot on the right presents the population at iteration 80 converging to the optimal solution.

Table 2 Results using GA versions.

Prob.	Algorithm 1				Modified GA			
	f_{best}	f_{avg}	st. dev.	nfe_{avg}	f_{best}	f_{avg}	st. dev.	nfe_{avg}
GP	3.00e+00	3.00e+00	4.88e-05	675	3.00e+00	3.00e+00	4.05e-01	795
MHB	1.14e-10	3.61e-09	2.10e-09	8422	8.72e-11	3.14e-09	1.86e-09	7448
RA-2	1.92e-10	2.46e-09	1.51e-09	10394	1.32e-10	2.35e-09	1.50e-09	7965
RA-5	1.54e-09	6.07e-09	2.22e-09	38173	7.11e-10	6.01e-09	2.04e-09	35458
RA-10	2.18e-06	9.47e-06	4.22e-06	50000	2.04e-06	8.27e-06	2.39e-06	50000

Parameter values: $e = 0.1p_{\text{size}}$, $p_{\text{cross}} = 0.9$, $\eta_{\text{cross}} = 20$, $p_{\text{mut}} = \frac{1}{n}$, $\eta_{\text{mut}} = 20$, $\sigma = \frac{1}{\sqrt{n}}$, $n_{\text{t}} = 0.5p_{\text{size}}$

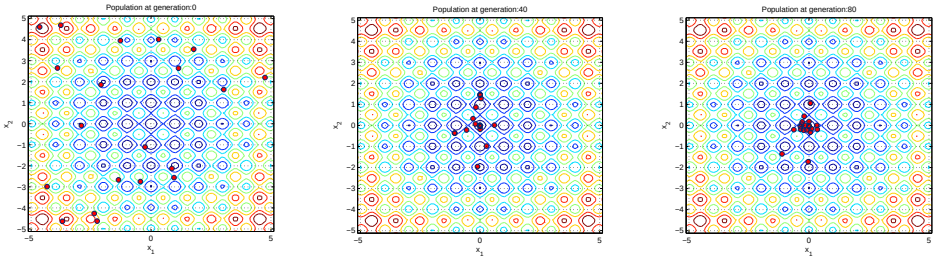

Fig. 1 Three movement graphs of 20 points at $k = 0, k = 40, k = 80$ using GA for problem RA-2.

2.2 Differential evolution

Differential evolution (DE) is a simple yet powerful evolutionary algorithm proposed by Storn and Price [56] for global optimization problems with simple bounds. DE is a floating point encoding that creates a new candidate point by combining the current point and several other points of the same population. DE has three parameters: amplification factor of the differential variation M, crossover control parameter CR, and population size p_{size}. The initial population is generated randomly and should cover the entire search space Ω. Let x_k^i represent a point at iteration k, herein denoted by target point. DE follows three operations to generate the population for the next iteration. It performs mutation to create the mutant points v_{k+1}^i, $i = 1, \ldots, p_{\text{size}}$, each relative to the corresponding target point x_k^i. The most commonly used mutation, refereed as DE/rand/1, is

$$v_{k+1}^i = x_k^{r_1} + M\,(x_k^{r_2} - x_k^{r_3}) \tag{3}$$

with indices r_1, r_2, r_3 randomly chosen from the set PI, mutually different and different from the running index i, so that p_{size} must be greater or equal to four to allow for this condition. M is a real and constant parameter of the set $[0,2]$ which controls the amplification of the differential variation $(x_k^{r_2} - x_k^{r_3})$. $x_k^{r_1}$ is called the base point. There are other frequently used mutation strategies available in the literature

[15, 43, 56, 60]. In order to increase the diversity, crossover is introduced. A trial point u^i_{k+1} is componentwise formed by

$$u^i_{j,k+1} = \begin{cases} v^i_{j,k+1} & \text{if } (\xi_j \leq CR) \text{ or } j = s_i \\ x^i_{j,k} & \text{if } (\xi_j > CR) \text{ and } j \neq s_i \end{cases}, \; j = 1,2,\ldots,n. \tag{4}$$

In (4), the random number $\xi_j \sim \mathrm{U}[0,1]$, and aims to perform the mixing of jth component of the points. Further, $CR \in [0,1]$ is a constant parameter for crossover which has to be determined by the user, and uniformly chosen random index s_i from the set $\{1,2,\ldots,n\}$ ensures that u^i_{k+1} gets at least one component from v^i_{k+1}. When generating the mutant point, some components can be generated outside the domain Ω. So, the components of each point are checked and if required they are projected onto the boundary of the search space Ω.

Finally, a selection is performed. The trial point replaces the target point for the next iteration only if it has better or equal fitness. The above three operations are repeated until a stopping condition is verified. It is not an easy task to set appropriate values for the parameters since these depend on the nature and size of the optimization problems. DE performance depends on the amplification factor of differential variation and crossover control parameter. Hence adaptive control parameters have been implemented in DE in order to obtain a competitive algorithm. The self-adaptive technique proposed by [11] for parameters M and CR generates a different set M^i, CR^i for each point in the population:

$$M^i_{k+1} = \begin{cases} M_l + \xi_1(M_u - M_l) & \text{if } \xi_2 < \tau_1 \\ M^i_k & \text{otherwise} \end{cases} \quad \text{and} \quad CR^i_{k+1} = \begin{cases} \xi_3 & \text{if } \xi_4 < \tau_2 \\ CR^i_k & \text{otherwise} \end{cases} \tag{5}$$

where M_l and M_u are the lower and upper bounds of M, respectively, $0 < \tau_1, \tau_2 < 1$ and each $\xi_j \sim \mathrm{U}[0,1], j = 1,\ldots,4$. The values of CR are generated randomly from $[0,1]$.

Modified DE with mixing mutation. The most commonly used mutation (3) has an exploratory effect but it slows down the convergence of DE. In this context several modifications have been proposed for an effective DE for global optimization problems [11, 15, 33, 43, 60].

In a population-based solution method, it is very important to obtain optimal solutions in a minimum time period. The DE algorithm should be capable of exploring the whole search space as well as exploiting around the neighborhood of a reference point (this can be the best point). With these arguments, other modifications in DE are proposed. The new modified DE, 'mDE', uses: (i) a combination of two mutation strategies with a weight factor ω; (ii) a cyclical usage of the overall best point as base point in mutation; (iii) self-adaptive techniques for parameters M, CR, as well as for the weight ω. Hence, two intermediate mutant points, $v^{i,1}_{k+1}$ and $v^{i,2}_{k+1}$, are generated and combined using a scalar weight factor $\omega \in [0,1]$ to form actual mutant point v^i_{k+1}:

$$v^i_{k+1} = \omega^i_{k+1} v^{i,1}_{k+1} + (1 - \omega^i_{k+1}) v^{i,2}_{k+1} \tag{6}$$

where ω_{k+1}^i is a self-adaptive weight factor, which aims to balance the combination of the two intermediate mutant points, randomly generated from $[0,1]$ in a way similar to that described in (5) for parameter CR, and

$$v_{k+1}^{i,1} = x_k^{r_1} + M_{k+1}^i (x_k^{r_2} - x_k^{r_3}) \text{ and } v_{k+1}^{i,2} = x_k^{r_4} + M_{k+1}^i (x_k^{r_5} - x_k^{r_6}). \tag{7}$$

The indices r_1, r_2, r_3 are randomly chosen from the set PI, mutually different and also different from the running index i and r_1 is the index of the best point among the three. Indices r_4, r_5, r_6 are also randomly chosen from PI, mutually different and also different from i. Furthermore, at every R iterations, the mutant point is alternatively generated by

$$v_{k+1}^i = x^{\text{best}} + M_{k+1}^i (x_k^{r_1} - x_k^{r_2}) \tag{8}$$

where the best point found so far is used as the base point and the other two points for the differential variation are randomly chosen as previously described. Algorithm 2 below contains the main steps of this mDE.

Algorithm 2 Modified differential evolution algorithm

1: Randomly generate $x^i \in \Omega$, for $i \in$ PI; set $k = 0$; set parameters
2: Evaluate the population and select x^{best}
3: **While** the stopping condition is not met **do**
  3.1 **for** $i = 1, \ldots, p_{\text{size}}$ **do**
    if MOD$(k,R) = 0$ **then** compute mutant point v^i using (8)
      else using (6) and (7)
    compute trial point u^i by crossover
    evaluate new population
    perform selection and select x^{best}
  3.2 set $k = k + 1$

Experimental results. Table 3 shows the results obtained by our proposal of a DE framework, presented in Algorithm 2, as well as those obtained by the variant of DE method proposed in [11], usually denoted by 'jDE', when solving the above described set of problems in Table 1. The algorithms stop when the stopping condition (2) is satisfied. Both algorithms stopped with successful runs for all problems except jDE that did not stop successfully for problem RA-10 in all 30 runs. It is shown that mDE is more efficient and reliable than jDE. mDE is able to reach the optimal solutions with the proposed accuracy requiring less function evaluations than jDE.

We include Fig. 2 to show the progress of average of the best objective function values of RA-10 with respect to number of function evaluations, after 30 runs. To analyze the progress of both mDE and jDE, we run the algorithms until the number of function evaluations reaches $nf_{\max} = 50000$. We observe that the progress towards f_{opt} by mDE is better than that of jDE where mDE converged to the best solution before $nf_{\max}$ reached in all 30 runs. The value of f_{avg} obtained here by mDE was 3.50×10^{-9} whereas by jDE it was 5.58×10^{-3}.

Table 3 Results using DE algorithms.

Prob.	jDE (in [11])				mDE (Algorithm 2)			
	f_{best}	f_{avg}	st. dev.	nfe_{avg}	f_{best}	f_{avg}	st. dev.	nfe_{avg}
GP	3.00e+00	3.00e+00	1.00e-04	739	3.00e+00	3.00e+00	8.23e-05	509
MHB	2.84e-10	5.31e-09	2.69e-09	1358	1.55e-10	4.29e-09	2.76e-09	1010
RA-2	6.06e-10	5.49e-09	2.71e-09	1350	4.64e-10	4.14e-09	2.66e-09	1057
RA-5	2.64e-09	7.17e-09	2.02e-09	11378	1.97e-09	6.41e-09	2.48e-09	8130
RA-10	9.97e-06	5.58e-03	1.17e-02	50000	5.09e-09	7.94e-09	1.22e-09	43560

Parameter values: $M_l = 0.1$, $M_u = 1$, $\tau_1 = \tau_2 = 0.1$, $R = 10$

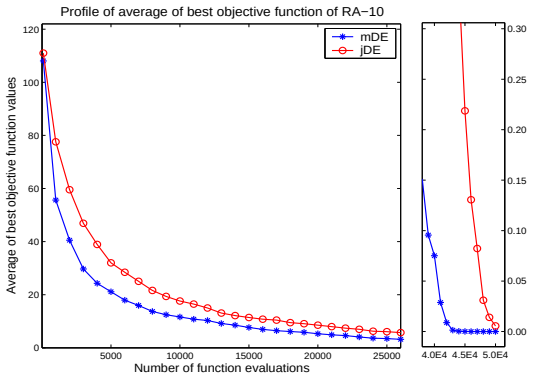

Fig. 2 Progress of mDE and jDE towards the optimum solution of RA-10.

2.3 Artificial fish swarm algorithm

The artificial fish swarm algorithm is a recent and easy to implement artificial life computing algorithm that simulates fish swarm behaviors inside water [58]. Fishes desire to stay close to the swarm, protecting themselves from predators and looking for food, and to avoid collisions within the group. Fish behaviors inside water can be summarized as follows [31]: (i) random behavior - fish swims randomly in water looking for food and other companions; (ii) searching behavior - fish tends directly and quickly to regions with more food, by vision or sense; (iii) swarming behavior - fish naturally assembles in groups which is a living habit in order to guarantee the existence of the swarm and avoid dangers; (iv) chasing behavior - fish finds the food dangling quickly after a fish, or a group of fishes, in the swarm that discovered food; (v) leaping behavior - fish leaps to look for food in other regions when it stagnates in a region.

The environment in which the artificial fish moves, searching for the minimum, is the feasible search space Ω of the minimization problem (1). The position of an artificial fish in the solution space is herein denoted by a point x (a vector in $\mathbb{R}^n$).

The artificial fish swarm (AFS) algorithm uses a population of points to identify promising regions looking for a global solution [58].

The crucial issue of the artificial fish swarm algorithm is the 'visual scope' of each point x^i. This represents a closed neighborhood centered at x^i with a positive radius defined by $v = \varsigma \max_{j\in\{1,\ldots,n\}}(u_j - l_j)$, where ς is a positive visual parameter. In general, this parameter is maintained fixed over the iterative process. However, experiments show that a slow reduction accelerates the convergence to the solution [50]. The following update $\varsigma = \max\left\{\varsigma_{\min}, \mu_\varsigma \varsigma\right\}$, every V iterations, where $0 < \mu_\varsigma < 1$ and $\varsigma_{\min} > 0$, is proposed.

Let I^i be the set of indices of the points inside the 'visual scope' of x^i ($i \notin \mathrm{I}^i$ and $\mathrm{I}^i \subset \mathrm{PI}$) and n_p^i be the number of points in the 'visual scope'. If the condition $n_\mathrm{p}^i / p_{\mathrm{size}} \leq \theta$ holds, where $\theta \in (0,1]$ is the crowd parameter, the 'visual scope' of x^i is said to be not crowded. Depending on the relative positions of the points in the population, three possible situations may occur:

- when $n_\mathrm{p}^i = 0$, the 'visual scope' is empty, and the point x^i, with no other points in its neighborhood to follow, has a random behavior moving randomly inside the 'visual scope';
- when the 'visual scope' is crowded, the point has some difficulty in following any particular point, and has a searching behavior choosing randomly another point (from the 'visual scope') and moving towards it;
- when the 'visual scope' is not crowded, the point is able either to swarm moving towards the central point of the 'visual scope', or to chase moving towards the best point of the 'visual scope'. The algorithm simulates both movements and chooses the best in the sense that a better objective function value is obtained.

The swarming behavior is characterized by a movement towards the central point in the 'visual scope' of x^i, defined by $c = \sum_{j\in\mathrm{I}^i} x^j / n_\mathrm{p}^i$. However, the swarming behavior is activated only if the central point has a better objective function value than that of x^i. Otherwise, the point x^i follows the searching behavior. In the searching behavior, a point is randomly chosen inside the 'visual scope', x^{rand}, and a movement towards it is carried out if the random point improves over x^i. Otherwise, the point has a random behavior.

The chasing behavior is carried out when a point, denoted by $x^{\min}$, with the minimum objective function value inside the 'visual scope' of x^i, satisfies $f(x^{\min}) \equiv \min\left\{f(x^j) : j \in \mathrm{I}^i\right\} < f(x^i)$. However, if this last condition is not satisfied then the point activates the searching behavior.

Finally, when the best point of the population, x^{best}, does not change for a certain number of iterations, the algorithm may have fallen into a local minimum. To be able to overcome this 'stagnation' and try to converge to the global minimum, the leaping behavior is implemented. At every L iterations, a point is randomly selected from the population and a random movement is carried inside the set Ω. We refer to [50, 58] for details.

Priority-based modified AFS. New heuristics have been incorporated into the AFS algorithm aiming to improve accuracy and reduce computational costs. They are

mainly focused on: (i) a priority-based AFS strategy, aiming to speed convergence; (ii) a selection behavior aiming to define the population for the next iteration; (iii) a local search, aiming to refine the search around x^{best}, at the end of each iteration.

The proposed priority-based strategy is implemented only when the 'visual scope' of a point x^i is not crowded. Instead of simulating both swarming and chasing behaviors at the same time, it tries one behavior at each time. Ranking the chasing behavior with highest priority, the movement in direction to $x^{\min}$ is carried out first if $f(x^{\min}) < f(x^i)$. Otherwise, the swarming behavior will be the alternative. So, the movement in direction to c is then carried out if $f(c) < f(x^i)$. However, if the latter condition does not hold then the point has a searching behavior. Algorithm 3 contains the main steps of this modified AFS algorithm, herein denoted by 'mAFS-P'. We remark that this modified AFS algorithm has been devised to consider the bound constraints of the problem, i.e., all movements are maintained inside Ω. Thus, each point movement, either towards the central c, the $x^{\min}$ or x^{rand}, defines a trial point, herein denoted by y^i, that is defined componentwise by $y^i_j = x^i_j + \xi d^i_j$, $j = 1, \ldots, n$, where d^i represents the vector with the direction of movement and ξ is a uniformly distributed real value from the interval $[0, 1]$. Then any point's component outside the bounds, is projected onto Ω. At the end of each iteration, the algorithm implements a selection operation aiming to accept the trial point y^i only if it improves over the current point x^i. Details of these new proposals can be found in [50, 51].

After defining the population for the next iteration, the best point is selected and a local search procedure is used to refine locally the search around x^{best}. We propose the well-known Hooke and Jeeves pattern search algorithm [30]. This is a derivative-free deterministic method that exploits the neighborhood of a point for a better approximation using two types of moves: the exploratory move and the pattern move. It is a variant of the well-known coordinate search method (a search along the coordinate axes) and it incorporates a pattern move to accelerate the progress of the algorithm, by exploiting information obtained from the search in previous successful iterations.

Algorithm 3 Modified artificial fish swarm algorithm

1: Randomly generate $x^i \in \Omega$, for $i \in$ PI; set $k = 0$; set parameters
2: Evaluate the population and select x^{best}
3: **While** the stopping condition is not met **do**
  3.1 **For** $i = 1, \ldots, p_{\text{size}}$ **do**
    if 'visual scope' of x^i is empty **then** random behavior **else**
      if 'visual scope' is crowded **then** searching behavior **else**
        if $f(x^{\min}) < f(x^i)$ **then** chasing behavior **else**
          if $f(c) < f(x^i)$ **then** swarming behavior **else** searching behavior
  3.2 selection operation
  3.3 select x^{best}
  3.4 **if** 'stagnation' occurs **then** leaping behavior
  3.5 apply a local search to x^{best}
  3.6 set $k = k + 1$

Experimental results. To assess the performance of the modified AFS algorithm, Algorithm 3 is tested with the previously described set of problems (see Table 1) and compared with the basic version of AFS. when solving the above described set of problems in Table 1. The stopping condition (2) is used once more to terminate the algorithms. The results are displayed in Table 4. We observe that mAFS-P was able to solve the problems GP, MHB, RA-2 and RA-5 with 100% of successful runs, and the problem RA-10 with 38% of successful runs, while the basic AFS reached the 50000 function evaluations in all 30 runs (0% of successful runs) with the problems RA-5 and RA-10.

Table 4 Results using AFS algorithms.

Prob.	AFS				mAFS-P (Algorithm 3)			
	f_{best}	f_{avg}	st. dev.	nfe_{avg}	f_{best}	f_{avg}	st. dev.	nfe_{avg}
GP	3.00e+00	3.00e+00	2.27e-05	914	3.00e+00	3.00e+00	1.50e-06	1760
MHB	1.64e-10	4.04e-09	2.53e-09	30056	2.88e-11	2.93e-10	1.65e-10	1882
RA-2	1.28e-10	1.10e-08	8.16e-09	38012	1.44e-10	1.22e-09	7.54e-10	4017
RA-5	1.03e-07	1.66e-06	1.32e-06	50094	4.66e-11	1.85e-09	2.40e-09	8890
RA-10	2.08e-05	7.10e-01	4.48e-01	50080	1.43e-10	6.30e-01	4.88e-01	36198

Parameter values: initial $\varsigma = n$, $\varsigma_{\min} = 10^{-6}$, $\mu_{\varsigma} = 0.9$, $V = n$, $\theta = 0.8$, $L = p_{\text{size}}$

To complement our study, we show the profiles of the relative performance of the two algorithms/solvers (AFS and mAFS-P) in comparison, when solving a set of problems, as proposed by Dolan and Moré [22]. The profiles are generated considering a set of 25 bound constrained problems selected from the benchmark set presented in Appendix B of [3]. These profiles plot the values of the cumulative distribution function $\rho(\tau)$, for each value $\tau \in \mathbb{R}$, of the ratios r_j, which for $j = 1,2$ are given by: $1 + m_{p_j} - \min\{m_{p_1}, m_{p_2}\}$, if $\min\{m_{p_1}, m_{p_2}\} < 0.00001$, or $m_{p_j} / \min\{m_{p_1}, m_{p_2}\}$, otherwise, where m_{p_1} and m_{p_2} represent the metric of solver 1 and solver 2 respectively. The value of $\rho_j(1)$ gives the probability that the solver j will win over the other in comparison. The higher the ρ the better the solver is. Figure 3 contains the profiles that correspond to the metrics f_{best} (plot on the left) and f_{avg} (plot on the right). The plot on the left shows that mAFS-P outperforms AFS in 92% of the tested problems. This means that in 92% of the problems the values of f_{best} obtained by mAFS-P are smaller than or equal to those obtained by AFS, while in 76%, AFS gives values of f_{best} smaller than or equal to those of mAFS-P. When the metric f_{avg} is considered (plot on the right) we can also conclude that mAFS-P outperforms AFS.

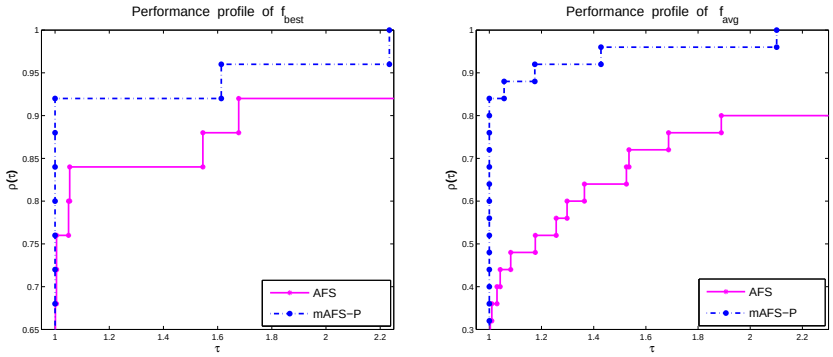

Fig. 3 Comparison between AFS and mAFS-P based on performance profiles.

2.4 Electromagnetism-like mechanism

The electromagnetism-like mechanism (EM) is a stochastic population-based algorithm that mimics the behavior of electrically charged particles and is specifically designed for solving bound constrained problems. In other words, a set of points is sampled from the feasible region Ω and these points imitate the role of the charged particles in basic electromagnetism theory [9]. The EM algorithm starts with a population of randomly generated points from the feasible region and, analogous to electromagnetism theory, each sample point of the population is considered as a charged particle that is released to the space. The charge of each point is related to its objective function value and determines the magnitude of attraction or repulsion of the point over the sample population. The charge of point x^i is then computed by $q^i = \exp\left(-n(f(x^i) - f(x^{\text{best}}))/\sum_{j=1}^{p_{\text{size}}}(f(x^j) - f(x^{\text{best}}))\right)$.

This way the points that have smaller objective function values possess higher charges. Different approaches to evaluate the charges could be found in [2, 19, 33, 45]. The charges are then used to find a direction to move the points in the population. The direction is evaluated as a combination of the forces exerted on a particular point via other points. Like the electromagnetic forces, this force is calculated by adding vectorially the forces from each of the other points calculated separately. Hence,

$$F^i = \sum_{j \neq i}^{p_{\text{size}}} F^i_j \equiv \begin{cases} (x^j - x^i)\dfrac{q^i q^j}{\|x^j - x^i\|^2} & \text{if } f(x^j) < f(x^i) \text{ (attraction)} \\ (x^i - x^j)\dfrac{q^i q^j}{\|x^j - x^i\|^2} & \text{if } f(x^j) \geq f(x^i) \text{ (repulsion)} \end{cases}, \tag{9}$$

for $i = 1, 2, \ldots, p_{\text{size}}$. Note that the force computation (9) is based on the Coulomb's law of the electromagnetism theory that states that the total force exerted on a point

via other points is inversely proportional to the square of the distance between the points and directly proportional to the product of their charges.

Then, each point x^i is moved according to the direction of the total force F^i by a random step length. To maintain feasibility, the force exerted on each point is normalized and scaled by the allowed range of movement towards the lower bound l_j, or the upper bound u_j, of the set Ω, for each component j. Thus, for $i \in$ PI but $i \neq$ best

$$x_j^i = \begin{cases} x_j^i + \xi \dfrac{F_j^i}{\|F^i\|}(u_j - x_j^i) \text{ if } F_j^i > 0 \\ x_j^i + \xi \dfrac{F_j^i}{\|F^i\|}(x_j^i - l_j) \text{ otherwise} \end{cases}, \quad j = 1,2,\ldots,n. \tag{10}$$

The random step length ξ is assumed to be uniformly distributed between 0 and 1.

Finally the EM algorithm also performs a local refinement that is applied to the best point of the population. x^{best} is componentwise assigned to a temporary point y. Then a random movement of maximum length $\sigma \max_j(u_j - l_j)$, where $\sigma > 0$, is carried out and if a better position is obtained within M_{local} iterations, x^{best} is replaced by y, the search ends for that component and proceeds to another one. In [9] it is shown that this simple random local search improves accuracy at a reduced computational cost. More sophisticated local search procedures have been implemented to enhance the EM algorithm: a deterministic local search procedure [47] and a stochastic one [48]. The formal description of the EM algorithm is depicted in Algorithm 4 below.

Algorithm 4 Electromagnetism-like mechanism algorithm

1: Randomly generate $x^i \in \Omega$, for $i \in$ PI; set $k = 0$; set parameters
2: Evaluate the population and select x^{best}
3: **While** the stopping condition is not met **do**
  3.1 compute the charges $c^i, i \in$ PI
  3.2 compute the total forces $F^i, i \in$ PI
  3.3 move the points except x^{best}
  3.4 evaluate the new population and select x^{best}
  3.5 apply a local search to x^{best}
  3.6 set $k = k + 1$

Modified EM based on memory force vector. In order to improve its search ability and efficiency and to extend to larger dimensional problems, a simple modification has been introduced in the original EM algorithm. The force F^i, used to move the point x^i as outlined in (10), is a linear combination of the force exerted on that point at the current iteration k, F_k^i, with the total force of the previous iteration, F_{k-1}^i, as follows: $F^i = F_k^i + \beta F_{k-1}^i$, where β is a positive memory constant which adjusts the change in the movement force vector. The point can memorize the previous force and adjust the current force to move the point. Further, the Hooke and Jeeves (HJ) pattern search algorithm [30, 35] is used to refine the best point of the population

instead of the random line search of the original EM. This is a derivative-free method that searches in the neighborhood of x^{best} for a better approximation using two types of moves: the exploratory move and the pattern move. We refer to [47] for details concerning the use of HJ in the EM context.

To analyze the influence of the parameter β to the final results, we use the mean absolute error $MAE = |f_{\text{opt}} - f_{\text{avg}}|/n$, a scaled distance between the average performance and the optimal value [39], that aims to measure the accuracy of the solutions found by the algorithm, and tested three values of β: $0.1, 0.5$ and 0.9. Figure 4 is a 100% stacked column chart for the values of MAE obtained by the selected β values. For each problem, we can compare the percentage that each β value contributes to the total. The smaller the percentage is the better. Overall, the area of the bars corresponding to $\beta = 0.1$ is smaller than the others - 13% in contrast with 42% (for $\beta = 0.5$) and 45% (for $\beta = 0.9$). This indicates that 0.1 is a better choice for β.

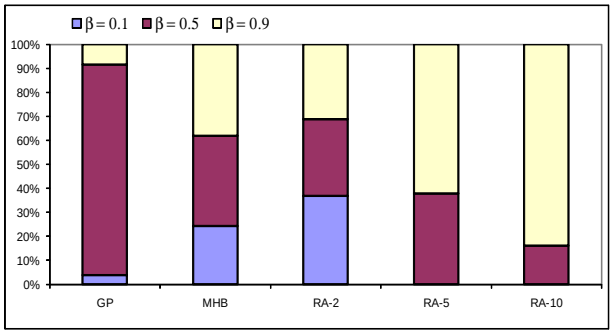

Fig. 4 *MAE* comparison for the three values of β.

Experimental results. The numerical experiments carried out with both EM algorithms are summarized in Table 5. The set of tested problems is described in Table 1. As previously referred, the values set to the parameters are displayed at the bottom row of the table. The results show that the modified EM based on the HJ local search algorithm performs well and has 100% of successful runs with all the tested problems, according to the stopping condition in (2). When solving the problems RA-5 and RA-10, the EM algorithm (in Algorithm 4) only stops when the maximum number of function evaluations is reached in all 30 runs.

3 Constraint-Handling Techniques

Most stochastic methods for global optimization are primarily developed for unconstrained or simple bound constrained problems. After that, they are extended to more general constrained problems by modifying the solution procedures or by applying

Table 5 Results using EM algorithms.

Prob.	Algorithm 4				Modified EM (with local HJ)			
	f_{best}	f_{avg}	st. dev.	nfe_{avg}	f_{best}	f_{avg}	st. dev.	nfe_{avg}
GP	3.00e+00	3.00e+00	2.49e-05	420	3.00e+00	3.00e+00	1.14e-06	357
MHB	6.53e-11	1.71e-09	1.18e-09	19249	2.80e-12	1.35e-10	1.08e-10	855
RA-2	4.48e-11	2.08e-09	1.57e-09	25657	7.25e-11	1.03e-09	5.54e-10	3490
RA-5	3.74e-08	4.43e-07	4.25e-07	50027	7.58e-11	6.74e-09	3.59e-09	13582
RA-10	3.52e-07	1.59e-06	7.88e-07	50052	1.20e-10	2.69e-10	1.29e-10	14143

Parameter values: $\sigma = 0.001$, $M_{\text{local}} = 10$, $\beta = 0.1$

penalty function methods. The penalty technique is by far the most used strategy, but others deserve attention for their efficiency [40]. The four main categories of constraint-handling techniques are: (i) methods based on penalty functions - where the constraint violation is combined with the objective function to define the penalty function that aims at penalizing infeasible solutions [7, 14, 25, 39, 42, 49, 51]; (ii) methods based on multiobjective optimization concepts - where both constraint violation and objective function are goals to be minimized separately [1, 2, 28]; (iii) methods based on biasing feasible over infeasible solutions - where constraint violation and objective function are used separately and optimized by some sort of order being the violation the most important [4, 16, 44, 52, 53]; (iv) methods based on preserving feasibility of solutions - where infeasible points are discarded or repaired [12]. Three classes of challenging constraint-handling techniques are herein described and tested.

3.1 Augmented Lagrangian-based techniques

Methods based on penalty functions transform the constrained problem into a sequence of unconstrained subproblems by adding a weighted constraint violation term to the objective function. This term is known as penalty term and the main goal is to penalize the objective function when constraints are violated. The penalty function is the objective to be minimized in the unconstrained subproblem. It depends on a positive penalty parameter that should be updated throughout the iterative process, so that the sequence of the solutions of the unconstrained subproblems converges to the solution of the constrained problem. With most penalty functions, the solution of the constrained problem is reached for an infinite value of the penalty [8]. In fact, the penalty parameter aims at balancing objective and constraint violation. Small values of the penalty can produce almost optimal but infeasible solutions, whereas large values can give feasible solutions although an optimal solution at the boundary may not be found. Different penalties may emerge depending on the way the penalty parameter varies throughout the iterative process. The most used are an-

nealing penalties, dynamic penalties and adaptive penalties [7, 39, 54]. Annealing penalties use an annealing criterion to attenuate the fitness of infeasible solutions. With dynamic penalties, a penalty parameter changes over time and depends on the iteration counter. The updating formula may even depend on user defined constants. On the other hand, in the adaptive penalty strategy, the user does not need to specify any constant value to tune the penalty updating formula. This formula depends on information gathered from the population, for instance the level of constraint violation and the mean objective function values.

We are interested in a particular class of penalty functions known as augmented Lagrangian functions to handle the equality and inequality constraints of the problem (1). An augmented Lagrangian is a more sophisticated penalty function for which a finite penalty parameter value is sufficient to yield convergence to the solution of the constrained problem [8], avoiding the side-effects associated with ill-conditioning of simpler penalty functions.

Augmented Lagrangian functions. Augmented Lagrangian functions depend on a penalty parameter as well as on the Lagrange multiplier vectors associated with the constraints of the problem. With augmented Lagrangian functions, the estimation of the multiplier vectors ought to be considered during the iterative process. Augmented Lagrangians are quite common in deterministic type methods [10, 13, 23, 24, 36] in both local and global optimization contexts, but more rare and only recently applied to heuristics that rely on a population of points to converge to an approximate solution of the subproblems [14, 25, 49, 51]. Conn et al. [13] proposed an augmented Lagrangian method for nonconvex optimization with equality constraints and proved global convergence results. Lewis and Torczon [36] extended the algorithm with the augmented Lagrangian

$$\mathscr{P}(x) = f(x) + \sum_{j=1}^{m} \lambda_j h_j(x) + \frac{\mu}{2} \sum_{j=1}^{m} h_j(x)^2 \tag{11}$$

to derivative-free pattern search methods where $\mu > 0$ is the penalty parameter and $\lambda \in \mathbb{R}^m$ is the multiplier vector associated with the constraints $h(x) = 0$. It is expected that as iterations proceed μ increases and the sequence of minimizers of (11) converges to the solution of the constrained problem [8]. The choice of values for the sequence of penalty parameters is a nontrivial issue. If the algorithm is struggling to generate feasible solutions, it is convenient to increase the penalty parameter. However, very large values can cause a slow convergence. This augmented Lagrangian framework may be extended to the more general problem like (1) adding to (11) the term related with the inequality constraint violation [8, 23, 24]:

$$\mathscr{L}_{a,1}(x) = \mathscr{P}(x) + \frac{1}{2\mu} \sum_{j=1}^{p} \left(\left[\max \left\{ 0, \delta_j + \mu g_j(x) \right\} \right]^2 - \delta_j^2 \right) \tag{12}$$

where $\delta \in \mathbb{R}^p$ is the multiplier vector associated with the constraints $g(x) \leq 0$. The main drawback of $\mathscr{L}_{a,1}(x)$ is that the function is not twice continuously differentiable. Thus derivative-free methods are the most appropriate for computing the se-

quence of minimizers of the objective function $\mathscr{L}_{a,1}(x)$ for a sequence of μ values. The estimation of the multiplier vectors λ and δ in this iterative process may follow the well-known first-order updating formulae: $\lambda_{j,k+1} = \lambda_{j,k} + \mu_k h_j(x_{k+1})$ for $j = 1,\ldots,m$ and $\delta_{j,k+1} = \max\{0, \delta_{j,k} + \mu_k g_j(x_{k+1})\}$ for $j = 1,\ldots,p$, where a safeguarded scheme is used to maintain the multiplier vectors bounded throughout the process, $\lambda_{j,k} \in [\lambda^-, \lambda^+]$ for $j = 1,\ldots,m$ and $\delta_{j,k} \in [0, \delta^+](j = 1,\ldots,p)$, for all k. Other papers available in the global optimization area consider the Powell-Hestenes-Rockafellar (PHR) augmented Lagrangian function

$$\mathscr{L}_{a,2}(x) = f(x) + \frac{\mu}{2}\left\{\sum_{j=1}^{m}\left[h_j(x) + \frac{\lambda_j}{\mu}\right]^2 + \sum_{j=1}^{p}\left[\max\left\{0, g_j(x) + \frac{\delta_j}{\mu}\right\}\right]^2\right\} \quad (13)$$

where, as previously stated, $\lambda_j, j = 1,\ldots,m$ and $\delta_j, j = 1,\ldots,p$ are the multipliers that are tuned according to the previously described updating formulae (see [10]). Birgin et al. [10] combine an augmented Lagrangian approach with the deterministic global αBB optimization method and its convex α-underestimation. A stochastic population-based method has also been implemented in the PHR augmented Lagrangian context [50] when computing the minimizers of (13). We remark that traditional augmented Lagrangian methods are locally convergent if the subproblems are solved within a prescribed tolerance, yielding ε_k-approximate global minimizers of the subproblems, and the sequence of tolerances converges to zero as $k \to \infty$ [8, 36]. In this type of augmented Lagrangian penalty algorithm, the penalty parameter is usually increased when the constraint violation has not been reduced, otherwise it is not changed. Algorithm 5 describes the main steps of the herein implemented augmented Lagrangian algorithm.

Algorithm 5 Augmented Lagrangian algorithm

1: **Given:** $\mu_0, \lambda_0, \delta_0, \varepsilon_0$; set $k = 0$; set parameters
2: Randomly generate x_0 in Ω
3: **While** the stopping condition is not met **do**
3.1 For the fixed values $\mu_k, \lambda_k, \delta_k$, compute x^{best} an ε_k-approximation to

$$\arg\min \mathscr{L}_a(x) \text{ subject to } x \in \Omega$$

3.2 set $x_{k+1} = x^{\text{best}}$
3.3 reduce ε_k and increase μ_k if appropriate
3.4 update λ_k and δ_k
3.5 set $k = k + 1$

Since equality constraints are the most difficult to be satisfied, a common procedure in stochastic methods for global optimization is to convert the equality constraints of the problem into inequality constraints: $|h_j(x)| \leq \zeta$, where ζ is a positive relaxation parameter. In general, the relaxation parameter is fixed over the entire iterative process. Typically, 10^{-3}, 10^{-4} and 10^{-5} are common values in the literature. This strategy directly affects the accuracy of the computed solution. The smaller the

ζ value the more difficult the problem is. The vector of the $mp = m + p$ inequality constraints is then defined by $G(x) = (|h_1(x)| - \zeta, \ldots, |h_m(x)| - \zeta, g_1(x), \ldots, g_p(x))^T$, the corresponding multipliers vector is represented by $\Delta \in \mathbb{R}^{mp}$ and the augmented Lagrangian has the form

$$\mathscr{L}_{a,3}(x) = f(x) + \frac{\mu}{2} \left\{ \sum_{j=1}^{mp} \left[\max \left\{ 0, G_j(x) + \frac{\Delta_j}{\mu} \right\} \right]^2 \right\}. \tag{14}$$

The augmented Lagrangian algorithm to be implemented with the function (14) is similar to the Algorithm 5 considering appropriate modifications relative to the multiplier vector. We refer the reader to [49, 51] for details.

Benchmark problems. For the numerical experiments presented in this section we consider 13 benchmark constrained nonlinear programming problems described in full detail in [37, 52]. The problems are known as g01-g13 (the 'g' suit). Some characteristics of these test problems are shown in Table 6 including the best known optimal solutions f_{opt} [37].

Stopping condition. Let $\vartheta(x) = \sum_{j=1}^{m} \left| h_j(x) \right| + \sum_{j=1}^{p} \max\{0, g_j(x)\}$ be the herein used measure of constraint violation. The numerical results obtained with each one of the constraint-handling techniques presented in this section are based on the following stopping condition. The algorithms stop when

$$\left(\vartheta(x^{\text{best}}) \leq \varepsilon^* \text{ AND } \left| f(x^{\text{best}}) - f_{\text{opt}} \right| \leq 10^2 \varepsilon^* \left| f_{\text{opt}} \right| + \varepsilon^* \right) \text{ OR } nf_{\text{eval}} > nf_{\max} \tag{15}$$

is true, where ε^* is set to 10^{-6} and the maximum number of function evaluations, $nf_{\max}$, is set to 300000.

Results. We run Algorithm 5, using GA (based on Algorithm 1) with the augmented Lagrangian (12), using EM (based on the Algorithm 4) with the Lagrangian (13) and using the AFS algorithm with the Lagrangian (14) to solve the bound constrained subproblems in Step 3.1 of the algorithm. Table 7 presents a synthesis of the computational results: f_{best} and f_{avg} for the problems g01-g13. The values set to the parameters of the algorithms are displayed in the bottom row of the table. From the table, we may conclude that the results obtained by EM-$\mathscr{L}_{a,2}$ for the set g01-g13 are slightly better than the others in comparison.

3.2 Tournament selection based on feasibility and dominance rules

Taking into consideration that evolutionary algorithms are based on a population of points, Deb [16] proposed a penalization scheme that ensures that an infeasible solution can never be better than a feasible one. The technique has been widely used in the context of differential evolution, genetic algorithm, particle swarm optimization, artificial bee colony and electromagnetism-like mechanism [5, 16, 46]. This technique is appropriate to tackle inequality constraints. In order to deal with equality

Table 6 Test problems for constrained optimization.

Prob.	Type of f	f_{opt}	n	m	p	Prob.	Type of f	f_{opt}	n	m	p
g01	quadratic	-15.00000000	13	0	9	g08	general	-0.09582504	2	0	2
g02	general	-0.80361910	20	0	2	g09	general	680.630057	7	0	4
g03	polynomial	-1.00050010	10	1	0	g10	linear	7049.24802	8	0	6
g04	quadratic	-30665.53867	5	0	6	g11	quadratic	0.74990000	2	1	0
g05	cubic	5126.496714	4	3	2	g12	quadratic	-1.00000000	3	0	1
g06	cubic	-6961.813876	2	0	2	g13	general	0.05394151	5	3	0
g07	quadratic	24.30620907	10	0	8						

Table 7 Results obtained with augmented Lagrangian techniques.

	GA-$\mathscr{L}_{a,1}$		EM-$\mathscr{L}_{a,2}$		AFS-$\mathscr{L}_{a,3}$	
	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}
g01	-14.9995	-14.6320	-14.9997	-14.7323	-14.9995	-14.9989
g02	-0.5728	-0.4523	-0.5157	-0.4288	-0.5558	-0.5043
g03	-1.0000	-0.9934	-0.9965	-0.9943	-1.0000	-0.9995
g04	-30650.546	-30645.783	-30665.539	-30665.422	-30664.798	-30663.487
g05	5131.4481	5156.3980	5126.5181	5131.2619	5126.5871	5128.5040
g06	-6944.8840	-6939.8921	-6958.7753	-6954.5228	-6961.7921	-6961.4422
g07	24.8900	25.1332	24.3080	24.3095	25.1260	25.7707
g08	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958
g09	680.6493	680.8867	680.6319	680.6488	680.6303	680.6349
g10	7124.2540	7283.9911	7066.0691	7212.2501	7054.4879	7074.6165
g11	0.7500	0.7500	0.7836	0.9891	0.7500	0.7500
g12	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
g13	0.0539	0.3142	0.0539	0.0548	0.0540	0.0543

Parameter values: $\mu_0 = 1$, ε_0=0.1, $\lambda^- = -10^{12}$, $\lambda^+ = \delta^+ = 10^{12}$, $\zeta = 10^{-5}$,
In GA: λ_0=δ_0=1; in AFS and EM: λ_0=δ_0=0 (componentwise)

constraints a reformulation into inequality constraints should be considered, as previously explained in this section. Hence, based on the vector of constraints, $G(x)$, constraint violation is measured by $\vartheta_G(x) = \sum_{j=1}^{mp} \max\{0, G_j(x)\}$. A fitness function for a comparison between points that uses constraint violation for infeasible solutions is proposed. The tournament based constraint method to handle inequality constraints is based on a penalty function which does not require any penalty parameter. Further, another important advantage of this approach is that it does not require the computation of the objective function value for infeasible points. The fitness function then considers the constraint violation to assess infeasible points and is expressed by:

$$F(x) = \begin{cases} f(x), & \text{if the point is feasible} \\ f_{\max} + \vartheta_G(x), & \text{otherwise} \end{cases}$$

where $f_{\max}$ is the objective function value of the worst feasible solution in the population. When all points are infeasible then its value is set to zero. Here, a point is considered feasible if $\vartheta_G(x) \leq 10^{-6}$. Thus, the fitness of an infeasible point depends not only on the constraint violation, but also on the population of points at hand. Figure 5 depicts the behavior of a typical fitness $F(x)$ in both feasible and infeasible regions. To handle simple bound constraints, a projection technique is im-

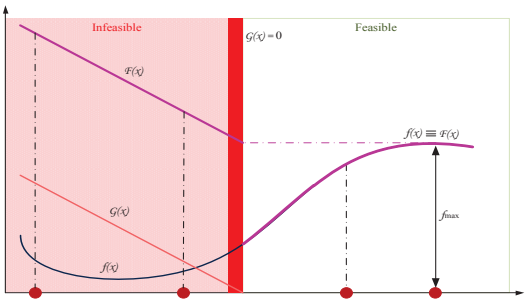

Fig. 5 Typical fitness $F(x)$ in both feasible and infeasible regions.

plemented, i.e., each generated point is componentwise projected onto the boundary of the set Ω.

To select the best solution, a tournament selection is exploited to make sure that: (i) when two feasible solutions are compared, the one with better objective function value is chosen; (ii) when one feasible and one infeasible solutions are compared, the feasible solution is chosen; (iii) when two infeasible solutions are compared, the one with smaller constraint violation is chosen.

Experimental results. The tournament selection technique based on the feasibility and dominance rules was implemented in the GA, DE and EM algorithms context. To simplify the notation, these solvers are herein denoted by 'GA-f&d', 'DE-f&d' and 'EM-f&d' respectively. The results obtained when solving the problems g01-g13 are listed in Table 8. The results show that DE-f&d outperforms GA-f&d and EM-f&d.

3.3 Ranking the objective function and constraint violation

Using the relationship between the objective function and constraint violation, a stochastic ranking (SR) for constraint-handling in global optimization is proposed in [52]. This ranking ensures that the good feasible points as well as promising infeasible ones are ranked in the top of the population. A probability p_f of using

Table 8 Results using feasibility and dominance rules.

	GA-f&d		DE-f&d		EM-f&d	
	f_{best}	f_{avg}	f_{best}	f_{avg}	f_{best}	f_{avg}
g01	-14.9999	-14.8873	-14.9993	-14.9987	-14.9999	-14.9998
g02	-0.7408	-0.5230	-0.7949	-0.7561	-0.7777	-0.4948
g03	-1.0000	-1.0000	-1.0001	-1.0001	-0.9992	-0.9959
g04	-30651.343	-30644.545	-30664.503	-30663.166	-30648.166	-30628.347
g05	5128.3653	5198.4028	5126.3579	5206.1841	5126.6708	5143.5152
g06	-6961.7988	-6961.7020	-6961.8362	-6614.2729	-6961.7885	-6961.4934
g07	24.5523	25.2021	24.2316	24.2690	28.9110	35.6513
g08	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958
g09	680.7745	680.8148	680.6565	680.6886	680.6839	681.3311
g10	7053.6216	7228.9648	7049.9227	7074.4042	7097.1938	7400.9558
g11	0.7490	0.7490	0.7500	0.8566	0.7500	0.7500
g12	-1.0000	-1.0000	-1.0000	-0.9999	-1.0000	-1.0000
g13	0.4418	0.6992	0.5532	0.8619	0.0902	0.3866

Parameter value: $\zeta = 10^{-5}$

only the objective function for comparison in ranking in the infeasible regions of the search space is introduced. Thus, given any pair of two adjacent points, the probability of comparing them (to see which one is fitter) according to the objective function is 1 if both individuals are feasible; otherwise it is p_f.

Runarsson and Yao [53] proposed another constraint-handling technique for constrained problems in order to find the right balance between the objective function and the constraint violation. This method is called global competitive ranking and is herein denoted by 'GcR'. In this method, a point is ranked by comparing it against all other points of the population. This is different from the stochastic ranking where two adjacent points compete for a given rank. In this ranking method, $\vartheta(x)$ measures constraint violation of a point x. We consider an individual point as a feasible one if $\vartheta(x) \leq 10^{-6}$. After calculating f and ϑ for all points in the population, they are sorted separately in ascending order (since we consider the minimization problem) and given ranks. Special consideration is given to tied values. In the case of tied values the same higher rank will be given. For example, in these eight individuals, already in ascending order, $\langle 6,(5,8),1,(2,4,7),3\rangle$ (individuals in parentheses have the same value) the corresponding ranks are $r(6) = 1, r(5) = r(8) = 2, r(1) = 4, r(2) = r(4) = r(7) = 5, r(3) = 8$. After the ranking of all the points based on the objective function f and the constraint violation ϑ, separately, the fitness function of each point x^i is given by

$$F_{\text{GcR}}(x^i) = p_f \frac{r_{i,f} - 1}{p_{\text{size}} - 1} + (1 - p_f) \frac{r_{i,\vartheta} - 1}{p_{\text{size}} - 1} \tag{16}$$

where F_{GcR} means fitness based on the global competitive ranking, and $r_{i,f}$ and $r_{i,\vartheta}$ are the ranks of point x^i based on the objective function and constraint violation,

respectively. p_f indicates the probability that the fitness is calculated based on the rank of objective function. It is clear from the above that p_f can be used easily to bias the calculation of fitness according to the objective function or the constraint violation. The probability should take a value $0.0 < p_f < 0.5$ in order to guarantee that a feasible solution may be found. From (16), the fitness of a point is a value between 0 and 1, and the best point in a population has the lowest fitness value.

Implementation within DE. When implementing the global competitive ranking technique in the differential evolution framework, the selection operation relies on the fitness (16). Thus, when evaluating the points at the current iteration k (x_k^i), and additionally the trial points for iteration $k+1$ (u_{k+1}^i), all of them are ranked together as above and their corresponding fitness F_{GcR} are calculated. Then, to decide which will be the points for the new population (at iteration $k+1$), a selection based on their calculated fitness is performed

$$x_{k+1}^i = \begin{cases} u_{k+1}^i & \text{if } F_{\text{GcR}}(u_{k+1}^i) \leq F_{\text{GcR}}(x_k^i) \\ x_k^i & \text{otherwise} \end{cases}.$$

After performing selection, the best point, x^{best}, is chosen based on the lowest fitness function of the new points. See [4, 5] for details concerning GcR implementation in a DE context.

Experimental results. To analyze the performance of the GcR technique we solved the previously referred 13 constrained nonlinear programming problems using the differential evolution technique, as described in Algorithm 2. The solver is hereafter denoted by 'DE-GcR'. For comparative purposes, Table 9 shows the results obtained by DE-GcR and those obtained by an evolutionary strategy combined with the stochastic ranking and global competitive ranking techniques, denoted by 'ES-SR' [52] and 'ES-GcR' [53] respectively. The values of f_{best} and f_{avg} in the second and third columns of the table are obtained using the stopping condition (15). The fourth column contains the values of f_{best} obtained with DE-GcR using the stopping condition suggested in [52, 53], as registered in the bottom row of the table. The values used for $k_{\max}$ are 175 for g12 and 1750 for the remaining problems. We may conclude that DE-GcR performs better than the other two in comparison. Based on the stopping condition (15), we may conclude that the strategy GcR is more effective in reaching a good solution than DE-f&d, when implemented in a DE context (see second and third columns of Table 9 *versus* fourth and fifth columns of Table 8).

4 Engineering Design Problems

This section is devoted to presenting a comparative study of the previously described stochastic algorithms combined with the presented constraint-handling approaches, when solving two well-known engineering design problems. First, a non-smooth problem is considered. It belongs to a class of economic dispatch problems. Then,

Table 9 Results using ranking techniques (with $p_f = 0.45$).

	DE-GcR		DE-GcR†	ES-SR†	ES-GcR†
	f_{best}	f_{avg}	f_{best}	f_{best}	f_{best}
g01	-14.9994	-14.9987	-15.0000	-15.0000	-15.0000
g02	-0.8035	-0.7409	-0.8036	-0.8035	-0.8035
g03	-1.0001	-1.0001	-1.0001	-1.0000	-1.0000
g04	-30664.507	-30663.181	-30665.540	-30665.539	-30665.539
g05	5126.4625	5129.0287	5126.3531	5126.4970	5126.4970
g06	-6961.9236	-6961.9126	-6961.9236	-6961.8140	-6943.5600
g07	24.2316	24.2716	24.2316	24.3070	24.3080
g08	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958
g09	680.6570	680.6845	680.6301	680.6300	680.6310
g10	7049.9309	7074.5823	7049.2461	7054.3160	$*$
g11	0.7500	0.8667	0.7500	0.7500	0.7500
g12	-1.0000	-0.9999	-1.0000	-1.0000	-1.0000
g13	0.0539	0.0539	0.0539	0.0539	0.0539

† Stopping condition: $(k > k_{\max})$ OR $(|f_{\text{best}} - f_{\text{opt}}| \leq 10^{-5})$; $(*)$ not solved

a smooth problem concerned with a pressure vessel design problem is presented, where two of the variables should take values from a specified set.

4.1 A non-smooth economic dispatch problem

The objective of the economic dispatch problem is to find the optimal combination of power dispatches from different power generating units in a given time period to minimize the total generation cost while satisfying the specified load demands and the generating units operating conditions. Generally, the cost function for each generating unit can be represented by a quadratic function, but due to valve-point loading effects the resulting cost function has additional nondifferentiable terms. We consider and solve the following nonsmooth economic dispatch problem to verify the effectiveness of the proposed solution methods. The problem has three generating units and the hourly power demand is equal to 850 (see other data in [55]):

$$\min \sum_{i=1}^{3} a_i x_i^2 + b_i x_i + c_i + \left| d_i \sin(e_i(x_{i,\min} - x_i)) \right|$$
$$\text{subject to } \sum_{i=1}^{3} x_i = 850 \text{ and } x_{i,\min} \leq x_i \leq x_{i,\max}, i = 1,2,3.$$

We run Algorithm 1 combined with the technique based on the feasibility and dominance rules, denoted by 'GA-f&d', Algorithm 2 combined with the feasibility and dominance rules - 'DE-f&d' - and combined with the global competitive ranking

technique as well - 'DE-GcR', Algorithm 3 combined with the augmented Lagrangian technique, herein denoted by 'AFS-$\mathscr{L}_{a,3}$' and Algorithm 4 combined with the feasibility and dominance rules denoted by 'EM-f&d'. Values listed in the Table 10 correspond to: (i) optimal values for the design variables; (ii) the optimal objective function value obtained from the best run, 'f_{best}'. In the tables, '-' means unavailable information. The algorithms are allowed to run until the number of function evaluations reaches $nf_{\max} = 15\,000$. Our results are compared with the results listed in [41] and those obtained by two well-known solvers available through NEOS server (http://www.neos-server.org/neos) - LINDO is a global solver and IPOPT is a local one. The results achieved by the herein proposed population-based algorithms are competitive with those of the literature. GA-f&d is able to achieve the least value of all.

Table 10 Results for the economic dispatch problem.

	Best solution found							
	GA-f&d	DE-f&d	DE-GcR	AFS-$\mathscr{L}_{a,3}$	EM-f&d	in [41]	LINDO	IPOPT
x_1	300.27	498.93	300.27	300.27	299.68	300.27	598.67	251.20
x_2	400.00	251.20	400.00	399.98	399.96	400.00	101.60	399.20
x_3	149.73	99.87	149.73	149.75	150.36	149.73	149.73	199.60
f_{best}	8234.06	8241.17	8234.07	8234.09	8234.44	8234.07	8382.73	8562.41

4.2 Pressure vessel design problem

This example corresponds to the design of a cylindrical pressure vessel with both ends capped with a hemispherical head [28, 34]. The problem consists of minimizing the total cost of the material, forming and welding of the cylindrical vessel, and has four design variables subject to four inequality constraints. The herein denoted variables x_1 and x_2 are integer multiples of 0.0625. We then consider $x_i = 0.0625 n_i$ and work with the integer variables n_1 and n_2.

The heuristic herein implemented to deal with the integer variables can be summarized as follows. Whenever new trial points are evaluated, the components that correspond to integer variables are rounded to the nearest integer. Then, the corresponding constraint violation and objective function values are computed, since they are crucial to evaluate fitness for comparative purposes. All the other procedures inside the algorithms proceed as if those variables were continuous. The results obtained with the 'GA-f&d', 'DE-f&d', 'DE-GcR' and 'EM-f&d', when solving the pressure vessel design problem, are reported in Table 11. The results available in [28, 34] are also listed. When solving this problem, the algorithms are allowed to run until the number of function evaluations reaches $nf_{\max} = 50\,000$. The reported solutions are competitive with others available in the literature. Between the four

algorithms herein tested, GA-f&d gives the least value of f. The proposed methods are able to solve a difficult nonlinear constrained optimization problem using a simple heuristic to tackle integer variables.

Table 11 Results for the vessel design problem.

	Best solution found					
	GA-f&d	DE-f&d	DE-GcR	EM-f&d	in [28]	in [34]
x_1	0.8125	0.8125	0.8125	0.8125	0.768326	1.125
x_2	0.3750	0.4375	0.4375	0.4375	0.379784	0.625
x_3	41.8844	42.0985	42.0985	42.0841	39.80962	58.2789
x_4	179.3074	176.6360	176.6360	176.8150	207.2256	43.7549
f_{best}	5 890.999	6 059.708	6 059.708	6 061.472	5 868.765	7 198.433

5 Conclusions

To address the solving of nonlinear constrained global optimization problems, this chapter is organized into three parts, excluding the Introduction and this section. The first part consists of Section 2 where global optimization problems with only simple bound constraints are analyzed. The second part includes the Section 3 where general constrained problems are addressed and the final part resumes to the Section 4 where two engineering design problems are presented and solved.

Constraint-handling involving population-based algorithms is a challenging issue. This study describes and tests three strategies. Our selection contains: the technique based on augmented Lagrangian functions, the tournament selection based on feasibility and dominance rules and a technique that relies on ranking the objective function and constraint violation (in Section 3). These methodologies are combined with four population-based stochastic methods. Two of them are inspired by evolutionary theories, namely the genetic algorithm and the differential evolution, the other uses swarm intelligence approaches, namely the artificial fish swarm algorithm, and another is based on basic electromagnetism theory, known as electromagnetism-like mechanism. Besides introducing the basic ideas behind these well-known stochastic methods, other novel and recent variants are presented, analyzed and compared (in Section 2). For each of these sections, we selected a set of benchmark problems which are solved with all the proposed strategies. The reported numerical results show that our choices are effective in solving typical global optimization problems.

Acknowledgments. The fourth author acknowledges Ciência 2007 of FCT, Fundação para a Ciência e a Tecnologia (Foundation for Science and Technology), Portugal for the financial support under fellowship grant: C2007-UMINHO-ALGORITMI-04. The other authors acknowledge

FEDER COMPETE, Programa Operacional Fatores de Competitividade (Operational Programme Thematic Factors of Competitiveness) and FCT for the financial support under project grant: FCOMP-01-0124-FEDER-022674.

References

1. Aguirre AH, Rionda SB, Coello Coello CA, Lizárraga GL, Montes EM (2004) Handling constraints using multiobjective optimization concepts. International Journal for Numerical Methods in Engineering 59:1989–2017
2. Ali MM, Golalikhani M (2010) An electromagnetism-like method for nonlinearly constrained global optimization. Computers and Mathematics with Applications 60:2279–2285
3. Ali MM, Khompatraporn C, Zabinsky ZB (2005) A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of Global Optimization 31:635–672
4. Azad MAK, Fernandes EMGP (2011) Modified differential evolution based on global competitive ranking for engineering design optimization problems. Lecture Notes in Computer Science, Vol 6784, Part III, B Murgante et al. (eds.) 245–260
5. Azad MAK, Fernandes EMGP (2011) Global Competitive Ranking for Constraints Handling with Modified Differential Evolution. In Proceedings of International Conference on Evolutionary Computation Theory and Applications, 42–51 SciTePress, Paris, France
6. Azad MAK, Fernandes EMGP, Rocha AMAC (2010) Nonlinear continuous global optimization by modified differential evolution. In International Conference of Numerical Analysis and Applied Mathematics 2010, TE Simos et al. (eds.) Vol. 1281, 955–958
7. Barbosa HJC, Lemonge ACC (2008) An adaptive penalty method for genetic algorithms in constrained optimization problems. In Frontiers in Evolutionary Robotics, Iba H (ed.) 34 pag. I-Tech Education Publ., Austria
8. Bertsekas DP (1999) Nonlinear Programming, 2nd edn. Athena Scientific, Belmont
9. Birbil SI, Fang S (2003) An electromagnetism-like mechanism for global optimization. Journal of Global Optimization 25:263–282
10. Birgin EG, Floudas CA, Martinez JM (2010) Global minimization using an augmented Lagrangian method with variable lower-level constraints. Mathematical Programming 125:139-162
11. Brest J, Greiner S, Bošković B, Mernik M, Žumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Transaction on Evolutionary Computation 10:646–657
12. Chootinan P, Chen A (2006) Constrained handling in genetic algorithms using a gradient-based repair method. Computers and Operations Research 33:2263–2281
13. Conn AR, Gould NIM, Toint PhL (1991) A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. Journal on Numerical Analysis 28:545–572
14. Costa L, Espírito Santo IACP, Denysiuk R, Fernandes EMGP (2010) Hybridization of a genetic algorithm with a pattern search augmented Lagrangian method. Proc. of 2nd International Conference on Engineering Optimization, H Rodrigues et al. (eds), 10 pag. Lisbon
15. Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood-based mutation operator. IEEE Transactions on Evolutionary Computation 13:526–553
16. Deb K (2000) An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186:311–338
17. Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Systems 9:115–149

18. Deb K, Goldberg D (1989) An investigation of niche and species formation in genetic function optimization. Proc. of the Third International Conference on Genetic Algorithms, 42–50
19. Debels D, DeReyck B, Leus R, Vanhoucke M (2005) A hybrid scatter search/electromagnetism metaheuristic for project scheduling. European Journal of Operational Research 169:638–653
20. Deep K, Dipti (2008) A self-organizing migrating genetic algorithm for constrained optimization. Applied Mathematics and Computation 198:237–250
21. Dixon LCW, Szegö GP (1978) The global optimization problem: an introduction. In: Dixon LCW, Szegö GP (eds.) Towards Global Optimisation 2. North-Holland, Amsterdam, 1–15
22. Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Mathematical Programming 91:201–213
23. Espírito Santo IACP, Fernandes EMGP (2010) Simplified model for the activated sludge system: WWTP cost minimization via an augmented Lagrangian pattern search method. In International Conference of Numerical Analysis and Applied Mathematics 2010, TE Simos et al. (eds.), Vol. 1281, 971–974
24. Espírito Santo IACP, Fernandes EMGP, Araújo MM, Ferreira EC (2007) Cost minimization of a WWTP using an augmented Lagrangian pattern search based solver. In 10 th. IWA Specialised Conference on Design, Operation and Economics of Large Wastewater Treatment Plants Publishing, Wien 17–20
25. Espírito Santo IACP, Costa L, Denysiuk R, Fernandes EMGP (2010) Hybrid genetic pattern search augmented Lagrangian algorithm: application to WWTP optimization. Lecture Notes in Management Science, Proceedings of 2nd International Conference on Applied Operational Research, Collan M (ed.) Vol. 2, 45–56
26. Goldberg D (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley
27. Hansen ER, Walster GW (2004) Global Optimization Using Interval Analysis, CRC Press
28. Hedar A-R, Fukushima M (2006) Derivative-free filter simulated annealing method for constrained continuous global optimization. Journal of Global Optimization 35:521–549
29. Hendrix EMT, G.-Toth B (2010) Introduction to Nonlinear and Global Optimization, Optimization and its Applications 37, Springer-Verlag
30. Hooke R, Jeeves TA (1961) Direct search solution of numerical and statistical problems. Journal of the Association for Computing Machinery 8:212–229
31. Jiang M, Wang Y, Pfletschinger S, Lagunas MA, Yuan D (2007) Optimal multiuser detection with artificial fish swarm algorithm. CCIS 2, ICIC 2007, D-S Huang et al. (eds.) Springer-Verlag 1084–1093
32. Jones DR, Perttunen CD, Stuckman BE (1993) Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications 79:157–181
33. Kaelo P, Ali MM (2006) A numerical study of some modified differential evolution algorithms. European Journal of Operational Research 169:1176–1184
34. Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering 194:3902–3933
35. Lewis RM, Torczon V (1999) Pattern search algorithms for bound constrained minimization. SIAM Journal on Optimization 9:1082–1099
36. Lewis RM, Torczon V (2002) A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. SIAM Journal on Optimization 12:1075–1089
37. Liang JJ, Runarsson TP, Mezura-Montes E, Clerc M, Suganthan PN, Coello Coello CA, Deb K (2006) Problem definitions and evaluation criteria for the CEC2006 (http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-06/CEC06.htm)
38. Liao TW (2010) Two hybrid differential evolution algorithms for engineering design optimization. Applied Soft Computing 10:1188–1199
39. Liu J-L, Lin J-H (2007) Evolutionary computation of unconstrained and constrained problems using a novel momentum-type particle swarm optimization. Engineering Optimization 39:287–305

40. Mallipeddi R, Suganthan PN (2010) Ensemble of constraint handling techniques. IEEE Transactions on Evolutionary Computation 14:561–579
41. Park JB, Lee KS, Shin JR, Lee KY (2005) A particle swarm optimization for economic dispatch with nonsmooth cost functions. IEEE Transactions on Power Systems 20:34–42
42. Petalas YG, Parsopoulos KE, Vrahatis MN (2007) Memetic particle swarm optimization. Annals of Operations Research 156:99–127
43. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13:398–417
44. Ray T, Liew KM (2003) Society and Civilization: an optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation 7:386–396
45. Rocha AMAC, Fernandes EMGP (2008) On charge effects to the electromagnetism-like algorithm. In Euro Mini Conference Continuous Optimization and Knowledge-Based Technologies, Sakalauskas L, Weber GW, Zavadskas EK (eds.) ISBN: 978-9955-28-283-9, 198–203
46. Rocha AMAC, Fernandes EMGP (2008) Feasibility and dominance rules in the electromagnetism-like algorithm for constrained global optimization problems. Lecture Notes in Computer Science, Computational Science and Its Applications, O Gervasi et al. (eds.) 5073:768–783
47. Rocha AMAC, Fernandes EMGP (2009) Modified movement force vector in a electromagnetism-like mechanism for global optimization. Optimization Methods and Software 24:253–270
48. Rocha AMAC, Fernandes EMGP (2009) Hybridizing the electromagnetism-like algorithm with descent search for solving engineering design problems. International Journal of Computer Mathematics, 86:1932-1946
49. Rocha AMAC, Fernandes EMGP (2011) Numerical study of augmented Lagrangian algorithms for constrained global optimization. Optimization 60(10-11):1359–1378
50. Rocha AMAC, Fernandes EMGP, Martins TFMC (2011) Novel fish swarm heuristics for bound constrained global optimization problems. Lecture Notes in Computer Science, Vol 6784, Part III, B Murgante et al. (eds.) 185–199
51. Rocha AMAC, Martins TFMC, Fernandes EMGP (2011) An augmented Lagrangian fish swarm based method for global optimization. Journal of Computational and Applied Mathematics 235:4611-4620
52. Runarsson TP, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. IEEE Transaction on Evolutionary Computation 4:284–294
53. Runarsson TP, Yao X (2003) Constrained evolutionary optimization – the penalty function approach. In: Evolutionary Optimization: International Series in Operations Research and Management Science, R Sarker et al. (eds.) 87–113
54. Silva EK, Barbosa HJC, Lemonge ACC (2011). An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization. Optimization and Engineering 12:31–54
55. Sinha N, Chakrabarti R, Chattopadhyay PK (2003) Evolutionary programming techniques for economic load dispatch. IEEE Transactions on Evolutionary Computation 7:83–94
56. Storn R, Price K (1997) Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11:341–359
57. Tahk M-J, Woo H-W, Park M-S (2007) A hybrid optimization method of evolutionary and gradient search. Engineering Optimization 39:87–104
58. Wang X, Gao N, Cai S, Huang M (2006) An artificial fish swarm algorithm based and ABC supported QoS unicast routing scheme in NGI. Lecture Notes in Computer Science, ISPA, G Min et al.(eds.) 4331:205–214
59. Zhang X, Liu S (2008) Interval algorithm for global numerical optimization. Engineering Optimization 40:849–868
60. Zhang J, Sanderson AC (2009) JADE: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13:945–958