Multilocal Programming and Applications

A. I. Pereira, O. Ferreira, S. P. Pinho and Edite M. G. P. Fernandes

Abstract Multilocal programming aims to identify all local minimizers of unconstrained or constrained nonlinear optimization problems. The multilocal programming theory relies on global optimization strategies combined with simple ideas that are inspired in deflection or stretching techniques to avoid convergence to the already detected local minimizers. The most used methods to solve this type of problems are based on stochastic procedures and a population of solutions. In general, population-based methods are computationally expensive but rather reliable in identifying all local solutions. In this chapter, a review on recent techniques for multilocal programming is presented. Some real-world multilocal programming problems based on chemical engineering process design applications are described.

1 Introduction

The purpose of this chapter is to present recent techniques for solving constrained Multilocal Programming Problems (MPP for short) of the following form

A. I. Pereira
Polytechnic Institute of Bragança, Bragança, and Algoritmi R&D Centre, University of Minho, Braga, Portugal,
e-mail: apereira@ipb.pt

O. Ferreira and S. P. Pinho
LSRE/LCM Laboratory of Separation and Reaction Engineering, Polytechnic Institute of Bragança, Bragança, Portugal,
e-mail: {oferreira,spinho}@ipb.pt

E. M. G. P. Fernandes
Algoritmi R&D Centre, University of Minho, Braga, Portugal,
e-mail: emgpf@dps.uminho.pt

$$\begin{aligned} &\max\ f(x) \\ &\text{s.t. } g_j(x) \leq 0,\ j = 1,\ldots,m \\ &\quad l_i \leq x_i \leq u_i,\ i = 1,\ldots,n \end{aligned} \tag{1}$$

where at least one of the functions $f, g_j : \mathbb{R}^n \to \mathbb{R}$ is nonlinear, and $\mathscr{F} = \{x \in \mathbb{R}^n : l_i \leq x_i \leq u_i, i = 1,\ldots,n,\ g_j(x) \leq 0, j = 1,\ldots,m\}$ is the feasible region. Problems with equality constraints, $h(x) = 0$, can be reformulated into the above form by converting into a couple of inequality constraints $h(x) - \upsilon \leq 0$ and $-h(x) - \upsilon \leq 0$, where υ is a small positive relaxation parameter. Since concavity is not assumed, f may possess many global and local (non-global) maxima in $\mathscr{F}$. In MPP, the aim is to find all points $x^* \in \mathscr{F}$ such that $f(x^*) \geq f(x)$ for all $x \in \mathscr{V}_\varepsilon(x^*) \cap \mathscr{F}$, where $\mathscr{V}_\varepsilon(x^*)$ represents the neighborhood of x^* with radius $\varepsilon > 0$. It is also assumed that problem (1) has a finite number of isolated global and local maximizers. The existence of local maximizers other than global ones makes this problem a great challenge. Here, we use the following notation: N is the number of solutions of the problem (1) and $X^* = \{x_1^*, x_2^*, \ldots, x_N^*\}$ is the set that contains those solutions. The algorithms herein presented for MPP aim at finding all the maximizers $x_1^*, x_2^*, ..., x_r^* \in \mathscr{F}$ such that

$$|f_{\max} - f(x_s^*)| \leq \delta_0 \text{ for all } s = 1, ..., r\ \ (r \leq N) \tag{2}$$

where δ_0 is a small positive constant and $f_{\max} = \max\{f(x_1^*), \ldots, f(x_r^*)\}$.

The MPP can be considered as defining a class of global optimization problems and are frequently encountered in engineering applications (e.g. [8, 15, 32]). Some algorithms for solving this type of problem require substantial gradient information and aim to improve the solution in a neighborhood of a given initial approximation. When the problem has global as well as local solutions, classical local optimization techniques can be trapped in any local (non-global) solution. A global optimization strategy is indeed the most appropriate to solve multilocal programming problems. When the objective function is multimodal, the probability of convergence to an already detected local solution is very high and depends very closely on the provided initial approximation. Methods that avoid converging to already identified solutions have been developed and integrated into a variety of classical global methods.

This study is focused on the analysis of the practical behavior of stochastic and deterministic methods for the computation of multiple solutions of the problem in the form (1). A penalty technique is chosen to tackle the constraints of the problem. Furthermore, challenging problems in the chemical engineering area, such as those that aim to evaluate if a multicomponent liquid mixture is globally stable regarding the separation in two or more liquid phases, by minimizing the tangent plane distance function for the Gibbs free energy of mixing, are fully described and solved.

The remainder of this paper is organized as follows. Section 2 provides a review on two particular classes of global optimization methods that can be extended to solve bound constrained MPP, presents the corresponding algorithms and illustrates their performance using three examples. In Section 3, the penalty function-based technique is addressed and various penalty functions are presented, tested and compared using a selected set of problems. Section 4 illustrates the use of numerical methods to solve very demanding real problems in the chemical engineering area.

2 Bound Constrained Multilocal Programming

In this section, we address a simpler problem known as bound constrained multilocal programming problem. The problem is presented in the following form

$$\begin{aligned} &\max\ f(x) \\ &\text{s.t. } l_i \leq x_i \leq u_i,\ i = 1,\ldots,n \end{aligned} \tag{3}$$

where the feasible region is just defined by $\mathscr{F} = \{x \in \mathbb{R}^n : l_i \leq x_i \leq u_i,\ i = 1,\ldots,n\}$. The two main classes of methods for solving the multilocal programming problem (3) are the stochastic and the deterministic, which are presented below [16, 20, 28, 39, 44, 45].

2.1 Stochastic methods

A stochastic method available in the literature to solve unconstrained and bound constrained global optimization problems will be described. In general, each run of a stochastic global method finds just one global solution. A survey on stochastic methods is presented in the textbook [62]. To be able to compute multiple solutions in just one run, where each of them is found only once, special techniques have to be incorporated into the global methods. These techniques aim at avoiding repetitive identification of the same solutions. Well-known examples are the clustering methods [50, 51, 52]. Other techniques that aim to escape from previously computed solutions, in general local solutions, are based on constructing auxiliary functions via a current local solution of the original problem [55, 63, 64]. Deflecting function and function stretching techniques can also be applied to prevent convergence to an already detected local solution [38, 39, 40, 53].

Clustering techniques rely on the multistart algorithm. The multistart is a stochastic algorithm where in a repetitive manner a local search is applied to a point that is randomly selected from the feasible region. Since the same local solution may be selected over and over again, the clustering technique aims to avoid the location of already detected solutions. A cluster contains a set of points, defining the so-called region of attraction, that terminate in a particular solution after applying a local search procedure. In this way only one local search is required to locate that solution. This process is able to limit the number of local search applications [50]. Another use of region of attractions based on a multistart algorithm is the therein called Ideal Multistart [52]. This method applies a local search procedure to an initial randomly generated point to reach the first solution, x_1^*, and the corresponding region of attraction is then defined, A_1. Then points are successively randomly generated from the feasible region until a point that does not belong to A_1 is found. The local search is then applied to obtain the second solution x_2^* and then the region of attraction A_2 is defined. After this, points are randomly generated and a local search is applied to the first point that does not belong to $A_1 \cup A_2$ to obtain x_3^* (and then A_3),

and so on. The definition of the so-called critical distance to construct the cluster is an important issue in clustering-based multistart methods. In some cases, the second derivative information of the objective function is required. In others, like [3], the critical distance becomes adaptive and does not require any special property of the objective function. The therein proposal is embedded within a simulated annealing (SA) algorithm to obtain a global algorithm that converges faster than the SA itself.

Deflection and stretching techniques rely on the concept of transforming the objective function in such a way that the previously detected solution is incorporated into the form of the objective function of the new problem. These techniques were mainly developed to provide a way to escape from local solutions and to drive the search to a global one. For example, in [53], a deflecting function technique was proposed in a simulated annealing context. The transformation of the objective function $f(x)$ works as follows. The deflecting function of the original f at a computed maximizer x^*, herein denoted as f_d, is defined by

$$f_d = f(x^*) - 0.5[\text{sign}(f(x^*) - f(x)) - 1](f(x) - f(x^*)). \tag{4}$$

All the maximizers which are located below $f(x^*)$ disappear although the maximizers with function values higher than $f(x^*)$ are left unchanged. An example is provided to show the defected effect.

Example 1. Consider the one-dimensional problem where the objective function is

$$f(x) = -x\sin(x), \text{ for } x \in [-8,8],$$

which has 3 maxima in the set $[-8,8]$.

Figure 1 shows the plot of $f(x)$ using a solid line. Let $x^* = -4.9132$ be the first computed maximizer, where $f(x^*) = 4.8145$. The plot of the deflecting function, $f_d(x)$, at $x^* = -4.9132$ is shown with a dashed line in the left plot, where all the values with $f(x) < f(x^*)$ are deflected. All the maximizers are alleviated and the function becomes a line when the deflecting function technique is applied on a global maximizer. In the right plot, the deflecting technique is applied to f at the local maximizer $x^* = 0$, with $f(x^*) = 0$ and as can be seen $f_d(x)$, represented by a dashed line, keeps the f values of points that have $f(x) \geq f(x^*)$.

On the other hand, the function stretching technique consists of a two-phase transformation [38, 39, 40]. The first transformation stretches the objective function downwards in a way that all the maxima with smaller values than the previously detected maximum are eliminated. Then the second phase transforms the detected maximum into a minimum. All the other maxima (with larger values than the detected maximum) are unaltered. If x^* is an already detected maximum of f, then the first transformation is defined by

$$f_1(x) = f(x) - \frac{\delta_1}{2}\|x - x^*\|[\text{sign}(f(x^*) - f(x)) + 1] \tag{5}$$

and the second by

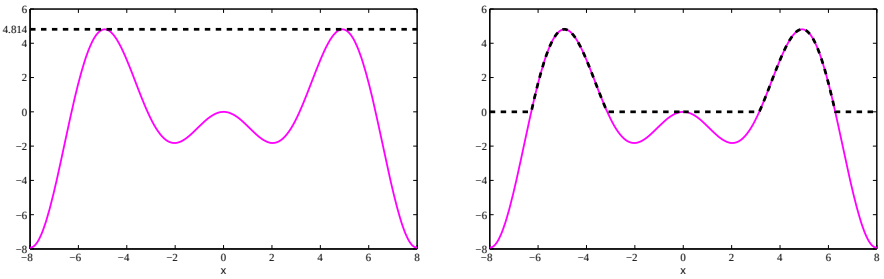

Fig. 1 Plot of f and f_d at $x^* = -4.9132$ (left plot) and at $x^* = 0$ (right plot).

$$f_2(x) = f_1(x) - \frac{\delta_2[\text{sign}(f(x^*) - f(x)) + 1]}{2\tanh(\kappa(f_1(x^*) - f_1(x)))} \tag{6}$$

where δ_1, δ_2 and κ are positive constants. To illustrate the effects of these transformations as the parameters vary, we use Example 1. Figure 2 shows the plot of $f(x)$ using a solid line. Based on the computed local maximizer $x^* = 0$ and applying the transformation (5) with $\delta_1 = 1.5$, we get the function $f_1(x)$ which is plotted in the figure with a dotted line, and applying (6), with $\delta_2 = 0.5$ we get the function $f_2(x)$, displayed in both plots of the figure with a dashed line. The plot on the left corresponds to $\kappa = 0.1$ and the one on the right corresponds to $\kappa = 0.05$. Function $f_1(x)$ comes out after the first transformation (5) and the bigger the δ_1 the greater the stretch is. See the plots on the right of Figs. 2 and 3. Parameter δ_2 defines the range of the effect (see the plots on the left of Figs. 2 and 3) and the parameter κ defines the magnitude of the decrease on f at x^* (see both plots of Fig. 2).

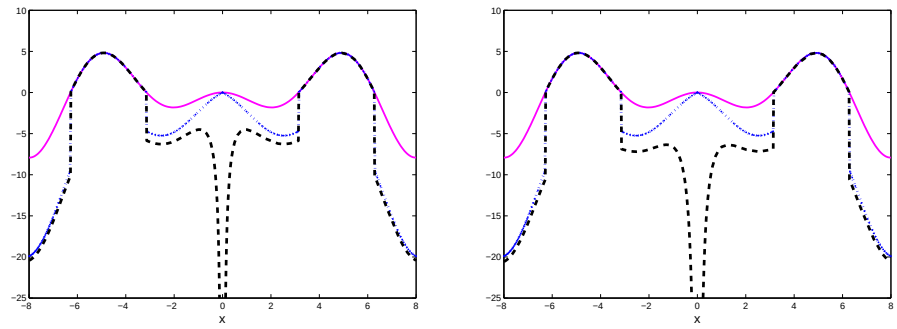

Fig. 2 Plot of f, f_1, f_2 with $\delta_1 = 1.5, \delta_2 = 0.5$, $\kappa = 0.1$ (on the left) and $\kappa = 0.05$ (on the right).

In a multilocal programming context, global as well as local (non-global) solutions need to be computed. Implementing the function stretching technique locally

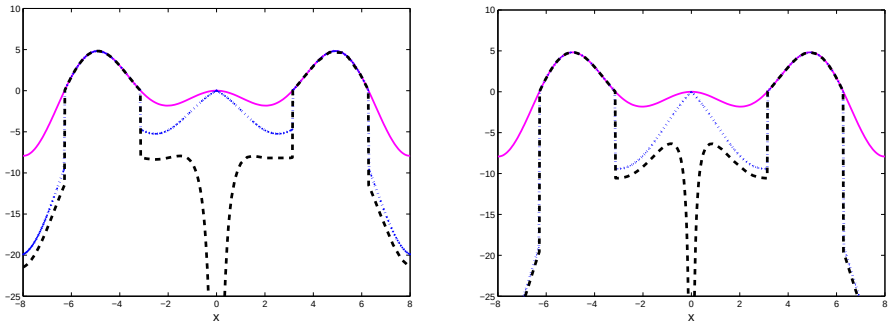

Fig. 3 Plot of f, f_1, f_2 with $\delta_1 = 1.5, \delta_2 = 1.5, \kappa = 0.1$ (on the left) and $\delta_1 = 3, \delta_2 = 0.5, \kappa = 0.05$ (on the right).

aims at stretching downwards the objective function f only in a neighborhood of an already detected maximizer, leaving all the other maxima unchanged. The successive application of this technique prevents the convergence to the solutions computed thus far. Therefore, this local stretching technique can be used when both global and local solutions are required since the strategy alleviates only the detected solutions. We now accept that the following assumption holds.

Assumption 1 *All optimal solutions of problem (3) are isolated points.*

Here we aim at presenting a proposal that applies locally the function stretching technique and uses a simulated annealing algorithm. The method is able to detect sequentially the global and local solutions instead of rambling over the feasible region attracted by previously identified solutions. After the computation of a solution, the objective function of the current problem is transformed using the function stretching technique. A sequence of global optimization problems with stretched objective functions is iteratively defined and solved by the SA algorithm [44, 45].

The SA is a point-to-point stochastic algorithm that does not require derivative information and is able to guarantee convergence to a global solution with probability one [22]. In fact, the practical implementation of the herein presented Stretched Simulated Annealing (SSA) method makes use of one of the most effective variants of SA known as Adaptive Simulated Annealing (ASA) algorithm [24].

The main steps of the ASA algorithm are resumed in Algorithm 1 below. For details on the algorithm convergence analysis, see [23, 24]. The ASA method can be easily described using five phases: the generation of a trial point, the 'acceptance criterion', the redefinition of the control parameters, the reduction of the control parameters and the stopping condition.

The generation of a trial point is one of its crucial phases and it should provide a good exploration of the search region as well as a feasible point. The parameter N_c^k in the Algorithm 1 aims at adapting the method to the problem. The 'acceptance criterion' allows the ASA algorithm to avoid getting stuck in local solutions when searching for a global one. For that matter, the process accepts points whenever an

Algorithm 1 ASA algorithm

1: **Given:** x^0, N_c^0 and the initial control parameter values. Set $k=0$ and $j=0$
2: **While** the stopping condition is not verified **do**

2.1 Based on x^k, randomly generate a trial point $y \in [l,u]$ and $j = j+1$
2.2 Verify the 'acceptance criterion'
2.3 **If** $j < N_c^k$ **then** $j = j+1$ and go to 2.2
else update N_c^k and $j=0$
2.4 Update control parameters
2.5 Set $k = k+1$

increase of the objective function is verified

$$x^{k+1} = \begin{cases} y & \text{if } \xi \leq A_{x^k,y}(c_A^k) \\ x^k & \text{otherwise} \end{cases}$$

where x^k is the current approximation to the global maximum, y is the trial point, ξ is a random number drawn from $U(0,1)$ and $A_{x^k,y}(c_A^k)$ is the acceptance function. This function represents the probability of accepting the point y when x^k is the current point, and it depends on a positive control parameter c_A^k. An usual acceptance function is

$$A_{x^k,y}(c_A^k) = \min\left\{1, e^{-\frac{f(x^k)-f(y)}{c_A^k}}\right\},$$

known as Metropolis criterion. This criterion accepts all points with objective function values equal or greater than $f(x^k)$. However, if $f(y) < f(x^k)$, the point y might be accepted with some probability. During the iterative process, the probability of descent movements decreases slowly to zero. Different acceptance criteria are proposed in [24]. The control parameter c_A^k, also known as temperature or cooling schedule, must be updated in order to define a positive decreasing sequence. To speed up the search, the ASA algorithm considers the reannealing of the process, meaning that the control parameters are redefined during the iterative process (see details in [24]). In general, the stopping condition for the ASA method is based on the idea that the algorithm should terminate when no further changes occur. Another stopping criterion limits the number of function evaluations, or defines a lower limit for the value of the control parameter.

We now describe the details concerning the SSA algorithm. The local application of the function stretching technique aims to prevent the convergence of the ASA algorithm to previously detected solutions. Let x_1^* be the first detected solution. Function stretching technique is then applied only locally, in order to transform $f(x)$ in a neighborhood of x_1^*, $V_{\varepsilon_1}(x_1^*)$, with radius $\varepsilon_1 > 0$. Thus, $f(x)$ is reduced only inside the region $V_{\varepsilon_1}(x_1^*)$ leaving all the other maxima unchanged. The maximum $f(x_1^*)$ disappears but all the others remain unchanged. Each global optimization problem of the sequence is solved by ASA. The multilocal procedure terminates when for a

predefined set of consecutive iterations no more solutions are detected [42, 44]. To illustrate this SSA procedure the following problem is considered.

Example 2. Consider the function

$$f(x) = -\cos^2(x_1) - \sin^2(x_2) \text{ where } x \in [-5,5]^2,$$

which has 12 global maxima in the set $[-5,5]^2$. In Fig. 4, the objective function of Example 2 and the function f_2 that comes out after applying transformations (5) and (6) to the previously computed global maximizer $x_1^* = (\frac{\pi}{2}, 0)$ are displayed. Transformations (5) and (6) stretch the neighborhood of x_1^*, with radius ε_1, downwards assigning smaller function values to those points to prevent convergence to that previously computed solution [44]. As can be observed, the other maxima are left unchanged (see Fig. 4).

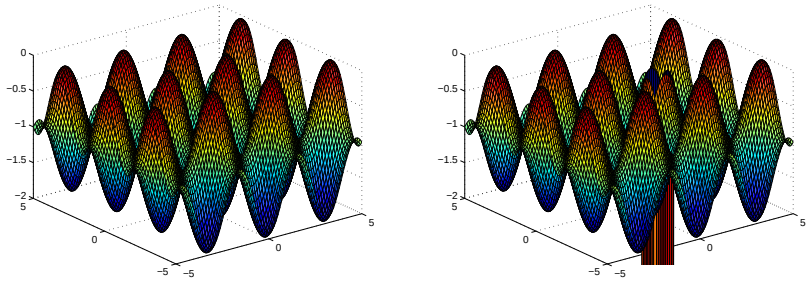

Fig. 4 Plot of $f(x)$ (left) and $f_2(x)$ (right) in Example 2.

Thus, the SSA method, at each iteration, solves a global programming problem using the ASA algorithm, where the objective function of the problem resulted from a local application of the function stretching technique that aims to eliminate the previously detected maximizer leaving the other maximizers unchanged. This process is repeated until no other solution is encountered. The mathematical formulation of the $j+1$-order problem in the sequence of problems is the following:

$$\max_{l \le x \le u} f^{j+1}(x) \equiv \begin{cases} f_2^j(x) \text{ if } x \in V_{\varepsilon_j}(x_j^*), \\ f^j(x) \text{ otherwise} \end{cases} \tag{7}$$

where x_j^* is the solution detected in the j-order problem, and the following notation is used: f_2^j is the stretched function obtained from f^j after transformations (5) and (6), for any j, where $f^1 = f$, and $f_2^1 = f_2$.

Algorithm 2 below presents, in summary, the strategy SSA for MPP (3). As previously stated the algorithm terminates when no more solutions are detected during a predefined number of consecutive iterations, K_{iter}, or a maximum number of function evaluations is reached, $nf_{\max}$. The conditions for the inner cycle (in Step 2.2)

aim at defining an adequate radius (ε_j) for the neighborhood of each solution computed in Step 2.1, in a way to adjust for each x_j^* the convenient neighborhood. In the final stage of the algorithm, a local search procedure is applied to each computed solution to improve accuracy.

Algorithm 2 SSA algorithm

1: **Given:** δ_0, ε_0, $\varepsilon_{\max}$. Set $f_{\max} = f^1(l)$, $j = 1$ and $p = 0$
2: **While** the stopping conditions are not met **do**

2.1 Compute $x_j^* = \arg\max_{l \le x \le u} f^j(x)$ using Algorithm 1
2.2 **While** $\left| f^j(x_j^*) - f_{\max} \right| \le \delta_0$ or $\Delta > \varepsilon_{\max}$ **do**
Set $p = p + 1$ and $\Delta = p\varepsilon_0$
Randomly generate $\widetilde{x}_i \in V_\Delta(x_j^*)$, $i = 1, \ldots, 2n$
Find $f_{\max} = \max_{i=1,\ldots,2n}\{f^j(\widetilde{x}_i)\}$
2.3 Update the optimal set X^* and set $\varepsilon_j = \Delta$
2.4 Set $j = j + 1$ and $p = 0$

3: Apply a local search procedure to the optimal set X^*

Example 3. Consider the classical optimization problem known as Branin problem [20].

$$\max\ f(x) \equiv -\left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 - 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) - 10,$$

where the feasible region is defined as $\mathscr{F} = \left\{x \in \mathbb{R}^2 : -5 \le x_1 \le 10 \wedge 0 \le x_2 \le 15\right\}$.

This problem has three global maximizers $x_1^* = (-\pi, 12.2750)^T$, $x_2^* = (\pi, 2.2750)^T$ and $x_3^* = (9.4248, 2.475)^T$ with a maximum value of -0.39789.

The SSA algorithm solves this problem in 0.45 seconds, needs 2442 function evaluations and detects the following maximizers $(-3.1416E+00, 1.2275E+01)$, $(9.4248E+00, 2.4750E+00)$ and $(3.1416E+00, 2.2750E+00)$, with global value $-3.9789E-01$. Since the SSA algorithm is a stochastic technique, the problem was solved thirty times. In this case all the solutions were identified in all runs. The results were obtained using a Inter Core 2 Duo, T8300, 2.4 GHz with 4 GB of RAM. The parameters of the algorithm are set as follows: $\delta_0 = 5.0$, $\varepsilon_0 = 0.1$, $\varepsilon_{\max} = 1.0$, $K_{\text{iter}} = 5$ and $nf_{\max} = 100\,000$.

2.2 Deterministic methods

Deterministic methods for global optimization are able to solve a problem with a required accuracy in a finite number of steps. Unlike stochastic methods, the outcome

of the algorithm does not depend on pseudo random variables. In general, they provide a theoretical guarantee of convergence to a global optimum. When compared with stochastic methods they may rely on structural information about the problem and in some cases they require some assumptions on the objective function, for example, they may require Lipschitz continuity of f over the feasible region [14, 20, 21, 31].

There are deterministic methods that combine the branch-and-bound method with successive refinement of convex relaxations of the initial problem [15], others use a non-differentiable technique based on the method of optimal set partitioning [27], and in [28] partitioning ideas are combined with some derivative information. An important subclass of methods for locating the solutions (maximizers and minimizers) of a continuous function inside bound constraints, like problem (3), consist of two phases: first, a partition of the feasible set is made and a set of finite points are generated and evaluated in order to detect good approximations to solution points; then, a local search method is applied in order to improve the accuracy of the approximations found in the first phase (e.g. [10, 11, 48, 51]).

DIRECT is a deterministic method that has been designed to find the global solution of bound constrained and non-smooth problems where no derivative information is needed [14, 25, 26]. DIRECT is an acronym for DIviding RECTangles and is designed to completely explore the search space, even after one or more local solution have been identified. The algorithm begins by scaling the domain into the unit hypercube and the objective function is evaluated at the center of the domain, where an upper bound is constructed. DIRECT computes the objective function at points that are the centers of hyperrectangles. At each iteration, new hyperrectangles are formed by dividing those that are more promising, in the sense that they potentially contain a required global solution, and the objective function is evaluated at the centers of those hyperrectangles. Based on those objective function values, the method is able to detect new promising hyperrectangles.

Another interesting subclass of deterministic methods for global optimization is based on the idea of branch and bound. Methods based on interval analysis [2, 19, 61] fall in this subclass. Interval analysis arises from the natural extension of real arithmetical operations to interval operations. Its use for global optimization was presented in 1992 [19]. Using interval operations, the interval algorithm splits successively the initial feasible region $[l,u]$ into small subintervals. The subintervals that do not contain the global solution are discarded and the others are further subdivided and analyzed. This process terminates when the width of the subintervals are below a predefined accuracy or no interval remains to be subdivided. Interval methods require high computational costs since the complexity rises exponentially with the dimension of the problem [19, 20].

The most known and used deterministic method is the branch-and-bound (BB) method. It has been mainly used in discrete optimization. The main idea in a BB method is the recursive decomposition of the original problem into smaller disjoint subproblems until the required solution is detected. In this context, smaller means either a strict smaller problem dimension or a strict smaller feasible region. The partition of the feasible region is the most used branching rule in continuous pro-

gramming. This decomposition should guarantee that the global solution is at least in one of the generated subproblems. The method compares the lower and upper bounds for fathoming each subregion. The subregion that contains the optimal solution is found by eliminating subregions that are proved not to contain the optimal solution.

BB-type methods are characterized by four natural rules: branching, selection, bounding and elimination. Branching is concerned with further refinement of the partition. The selection rule is also very important, greatly affects the performance of the algorithm and aims at deciding which subregion should be explored next.

The method starts with a set I^0 that contains the feasible region assumed to be a compact set. An algorithm should be provided to compute an upper bound value, f_U, such that $f_U \geq f(x)$ for all $x \in [l,u]$ that will be improved as subproblems are solved. At each iteration, the method has a list $\mathscr{L}$ of subsets I^k of I^0. An upper bound f_U^k of the maximum objective function value on I^k is computed for every subset in $\mathscr{L}$. A global lower bound f_L of the maximum function value over the feasible region is defined by the f value of the best feasible solution found.

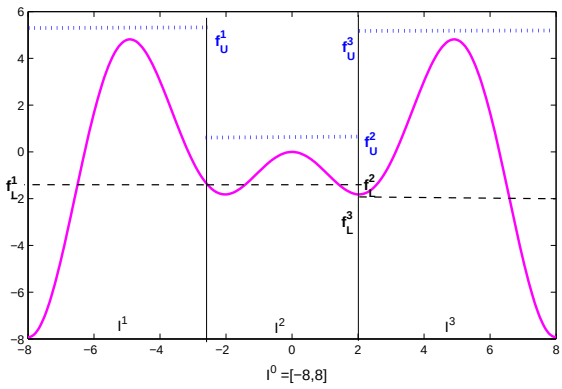

Fig. 5 Branching applied to the continuous Example 1.

Figure 5 illustrates a branching rule applied to the function in Example 1. The set $I^0 = [-8,8]$ was partitioned into I^1, I^2 and I^3. $f(x)$ is represented by a solid line. The lower bounds, f_L^k, are the higher function values at the boundaries of the subintervals and are represented by dashed lines. The upper bounds, f_U^k, represented in the figure by dotted lines, are computed using a simple procedure. In this case, all the subintervals should be explored and subdivided again by the branching rule, since no upper bound is lower than any lower bound.

A subregion I^k can be removed from the list $\mathscr{L}$ if:

i) it cannot contain any feasible solution;
ii) it cannot contain the optimal solution since $f_U^k < f_L$;
iii) there is no use in splitting I^k since the size of the set is smaller than a predefined tolerance δ.

A crucial parameter of the BB method is the positive δ-precision. This tolerance is used in the stopping criteria in a way that a solution within a δ-precision is obtained. The algorithm also stops when the list $\mathscr{L}$ is empty. When solving discrete problems, the parameter δ can be set to zero and the BB algorithm is finite. However, in continuous optimization, the bounding operation is required to be consistent, i.e., any infinitely decreasing sequence of successive refined partitions I^k on I^0 satisfies

$$\lim_{k\to\infty}(f_L^k - f_U^k) = 0 \tag{8}$$

where f_L^k and f_U^k are the lower and upper bounds, respectively, of the problem with feasible region I^k. This consistency condition implies that the required δ-precision solution is achieved after a finite number of steps and the BB algorithm is therefore finite.

In the multilocal programming context, to compute the solutions of (3), the BB method is combined with strategies that keep the solutions that are successively identified during the process. The method also avoids visiting those subproblems which are known not to contain a solution [20, 21]. The main step of the proposed multilocal BB method is to solve a sequence of subproblems described as

$$\max f(x) \text{ for } x \in I^{i,j} \text{ and } i = 1,\ldots,n_j \tag{9}$$

where $I^{i,j} = [l_1^{i,j}, u_1^{i,j}] \times \cdots \times [l_n^{i,j}, u_n^{i,j}]$, and the subsets $I^{i,j}$, for $i = 1,\ldots,n_j$, belong to a list, herein denoted by $\mathscr{L}^j$, that can have a local solution that satisfies condition (2). The method starts with the list $\mathscr{L}^0$, with the set $I^{1,0} = [l,u]$, as the first element and stops at iteration j when the list $\mathscr{L}^{j+1}$ is empty. The generic scheme of the multilocal BB algorithm can be formally described as shown in Algorithm 3. Furthermore, the algorithm will always converge due to the final check on the width of the subinterval $I^{i,j}$ (see the stopping conditions in Step 3 of the algorithm). A fixed value, $\delta > 0$, is provided in order to guarantee a δ-precision solution.

Algorithm 3 Multilocal BB algorithm

1: **Given:** $\delta_0 > 0$, $\delta > 0$
2: Consider f^0 the solution of problem (9), for $I^{1,0} = [l,u]$, set $j = 0$ and $n_0 = 1$
3: **While** $\mathscr{L}^{j+1} \neq \emptyset$ and $\max_i \left\{ \left| \left| u^{i,j} - l^{i,j} \right| \right| \right\} \geq \delta$ **do**

3.1 Split each set $I^{i,j}$ into intervals, for $i = 1,\ldots,n_j$; set $\mathscr{L}^{j+1} = \left\{ I^{1,j+1},\ldots,I^{n_{j+1},j+1} \right\}$
3.2 Solve problem (9), for all subsets in $\mathscr{L}^{j+1}$. Set $f^1,\ldots,f^{n_{j+1}}$ to the obtained maxima values
3.3 Set $f^0 = \max_i \left\{ f^i \right\}$ for $i = 0,\ldots,n_{j+1}$. Select the subsets $I^{i,j+1}$ that satisfy the condition:

$$\left| f^0 - f^i \right| < \delta_0$$

3.4 Reorganize the list $\mathscr{L}^{j+1}$; update n_{j+1}
3.5 Set $j = j+1$

To illustrate the practical behavior of the Algorithm 3, the problem presented in Example 3 is used. The multilocal BB algorithm solves this problem in 37.1 seconds, needs 9331 function evaluations and finds the following maximizers $(3.1416E+00, 2.2750E+00)$, $(-3.1416E+00, 1.2275E+01)$ and $(9.4248E+00, 2.4750E+00)$ with global value $-3.9789E-01$. As it was expected, the multilocal BB algorithm is computationally more demanding than the SSA algorithm.

2.3 Numerical experiments

This part of the section aims to report the results of applying the Algorithm 2 to solve bound constrained MPP. The Algorithm 3 was not used due to its high time consuming. First, an experiment with a varied dimensional problem is analyzed for five different values of n. Then a large dimensional problem is solved by the SSA algorithm. The problems were solved using a Inter Core 2 Duo, T8300, 2.4 GHz with 4 GB of RAM. The parameters in the algorithm are set as follows: $\delta_0 = 20.0$, $\varepsilon_0 = 0.1$, $\varepsilon_{\max} = 1.0$, $K_{\text{iter}} = 5$ and $nf_{\max} = 100\,000$.

2.3.1 Experiment with a varied dimensional problem

Example 4. Consider the classical optimization problem known as n-dimensional Test (n-dT) [12]:

$$\begin{aligned} \max f(x) \equiv & -\tfrac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i) + \varpi\sum_{i=1}^{n}(x_i - 2.90353)^2 \\ \text{s.t. } & -5 \le x_i \le 5, i = 1,\ldots,n \end{aligned}$$

for $\varpi = 0$ (classical problem) and $\varpi = 0.3$ (modified). This problem has 2^n local maxima in the set $[-5,5]^n$ and the global is located at $(-2.9035,\ldots,-2.9035)$. The 2-dT function for the classical problem and $n = 2$ is plotted in Fig. 6. The global maximizer is $(-2.9035,-2.9035)$ with a value of $f = 78.332$ and the local maxima are located at $(-2.9036, 2.7468)$ (with $f = 64.196$), $(2.7468,-2.9035)$ (with $f = 64.196$) and $(2.7468, 2.7468)$ (with $f = 50.059$).

Results regarding the classical problem in Example 4 for $n = 2,4,6,8,10$ are shown in Table 1. The table depicts a summary of the results obtained by SSA algorithm. The average value of the solutions found for the global maximum, in all the runs, f^*_{avg}, the average number of function evaluations (obtained in all 30 runs, when computing the global), $nf^{\text{eval}}_{\text{avg}}$, the average (over all runs) of the CPU time required to converge to all the solutions identified by the algorithm (in seconds), CPU(s), the best solution found for the global maximum during the 30 runs, f^*, and the average number of solutions identified by the algorithm, n_{sol}, are displayed. Table 2 reports the same results for the modified problem ($\varpi = 0.3$) in Example 4. The SSA algorithm was able to identify several maximizers during the process, in both tested

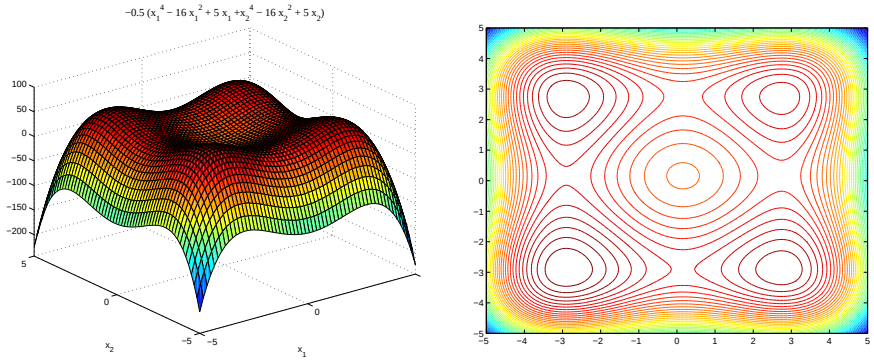

Fig. 6 Plot of the classical 2-dT problem

problems (classical and modified), although not all maximizers are detected in all runs. We may conclude that the efficiency of the algorithm is not greatly affected by the dimension of the problem.

Table 1 Results of the SSA algorithm for Example 4, considering $\varpi = 0$.

Problem	f^*_{avg}	$nf^{\text{eval}}_{\text{avg}}$	CPU(s)	f^*	n_{sol}
2-dT	7.8332E+01	1067	0.17	7.8332E+01	2
4-dT	1.5667E+02	3159	0.29	1.5667E+02	2
6-dT	2.3500E+02	10900	0.75	2.3500E+02	2
8-dT	3.1333E+02	36326	2.28	3.1333E+02	1
10-dT	3.9166E+02	58838	3.71	3.9166E+02	1

Table 2 Results of the SSA algorithm for Example 4, considering $\varpi = 0.3$.

Problem	f^*_{avg}	$nf^{\text{eval}}_{\text{avg}}$	CPU(s)	f^*	n_{sol}
2-dT	9.8911E+01	1386	0.34	9.8911E+01	2
4-dT	1.9782E+02	2796	0.25	1.9782E+02	2
6-dT	2.9673E+02	10110	0.69	2.9673E+02	1
8-dT	1.2471E+03	30641	1.95	1.2471E+03	1
10-dT	1.5588E+03	56604	3.58	1.5588E+03	1

2.3.2 Experiment with a large dimensional problem

Here we aim at analyzing the performance of the SSA algorithm when solving a large dimensional MPP.

Example 5. Consider the following optimization problem with a multimodal objective function [30]:

$$\max f(x) \equiv -\sum_{i=1}^{n} \sin(x_i) + \sin(\frac{2x_i}{3})$$
$$\text{s.t. } 3 \leq x_i \leq 13, i = 1, \ldots, n$$

which has an analytical global optimum of $1.216n$. Figure 7 contains the plot of $f(x)$ when $n = 2$. The global maximizer is located at $(5.3622, 5.3622)$. The other maximizers in $[3,13]^2$ are: $(10.454, 5.3622)$ (with $f = 1.4393$), $(5.3622, 10.454)$ (with $f = 1.4393$) and $(10.454, 10.454)$ (with $f = 0.4467$). The other optimum is a minimum with $f = -0.4467$ at $(8.3961, 8.3961)$. Table 3 contains the results obtained by the SSA algorithm for two different values of n: 50 and 100. Clearly, the SSA algorithm is able to solve large-dimensional problems, detecting some solutions, in a reasonable time. The number of function evaluations and the CPU time are smaller in the case of $n = 100$. We remark that these results were obtained with $nf_{\max} = 1\,000\,000$.

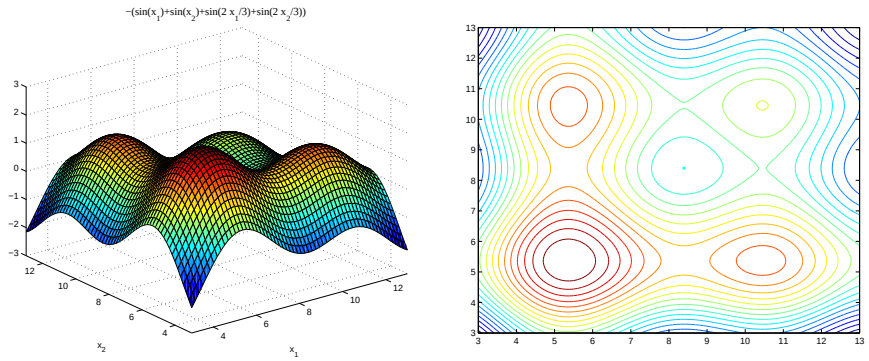

Fig. 7 Plot of $f(x)$ of Example 5 for $n = 2$.

Table 3 Results of the SSA algorithm for Example 5.

n	f^*_{avg}	$nf^{\text{eval}}_{\text{avg}}$	CPU(s)	f^*	n_{sol}
50	6.0799E+01	944761	287	6.0799E+01	4
100	1.2160E+02	383038	104	1.2160E+02	6

2.4 Synopsis

Approaches aiming at computing multiple solutions of bound constrained MPP are addressed. The first proposal is a stochastic method based on a function stretching technique and the simulated annealing algorithm. A deterministic method is also proposed. It relies on a branch-and-bound-type method that is able to keep the solutions found so far. The results reported with a varied dimensional problem show that the performance of the SSA algorithm is not greatly affected by problem's dimension. The computational cost of implementing the multilocal BB algorithm is much higher than that of the SSA algorithm. The ability of the SSA algorithm to tackle large dimensional problems was investigated using a classical example with various dimensions.

3 Constrained Multilocal Programming

In general, constrained optimization problems are more difficult to solve than unconstrained or bound constrained problems, specially when the feasible region is not convex and is very small when compared with the whole search space. There is a metric ρ given by the ratio between the feasible region and the search space that can be used to measure the difficulty of solving a problem. With a stochastic method, ρ can be estimated by the ratio between the number of feasible solutions and the total number of solutions randomly generated [29]. Feasible regions made of disjointed regions are also difficult to handle, in particular by gradient-based methods. Stochastic methods are in general well succeeded when solving this type of difficult problems. Different constrained search spaces have motivated the development of a variety of constraint-handling techniques. The three main classes of methods to handle constraints are:

- methods that use penalty functions;
- methods based on biasing feasible over infeasible solutions;
- methods that rely on multi-objective optimization concepts.

We refer the reader to [34, 54] and to the references therein. There are also other techniques that aim at repairing infeasible solutions. In [60], a method that uses derivative information from the constraint set to repair infeasible points is proposed in a hybrid particle swarm optimization context.

Penalty function-based methods are the most well-known class of methods to handle constraints in nonlinear optimization problems. These techniques transform the constrained problem into a sequence of unconstrained subproblems by penalizing the objective function f whenever constraints are violated. Then, the goal is to force constraint violation to zero – adding a positive penalization in minimization problems, or subtracting a positive penalization in maximization problems. The penalty method relies on a penalty function as the objective function of the problem which depends on f, on a penalty term and a (at least one) positive penalty parame-

ter. This is an iterative process where the solutions of the unconstrained subproblems are approximations to the solution of the constrained problem.

To solve the constrained MPP in the form presented in (1), some theory and practice of penalty methods is addressed in the remaining part of this section.

3.1 The penalty function method

A variety of sophisticated penalties exist in the class of penalty function methods [20, 36, 57]. They are developed to address efficiently the issue related with constraint-handling in problems with different structures and types of constraints. Additive penalties define a penalty function of the form

$$\phi(x;\mu) = f(x) - \mathscr{P}(g(x),\mu) \tag{10}$$

where $f(x)$ is the objective function in problem (1) and $\mathscr{P}$, known as the penalty term, depends on the constraint functions $g(x)$ and a positive penalty parameter μ. The penalty term should be zero when the point is feasible and then $\phi(x;\mu) = f(x)$, and is positive when the point is infeasible. The penalty term aims at penalizing constraint violation directing the search towards the feasible region and at the same time looking upwards for a point with the largest f. On the other hand, multiplicative penalties have the form

$$\phi(x;\mu) = f(x)\mathscr{P}_{\text{mult}}(g(x),\mu)$$

where $\mathscr{P}_{\text{mult}}(g(x),\mu)$ is a function that should take the value one when the point is feasible and smaller than one for infeasible points. There is no special rule to design a penalty function. Experiments show that penalties that depend on the distance from feasibility are better than those that rely on the number of violated constraints alone.

Different penalty terms have been devised including the death, static, dynamic, annealing and adaptive penalties. Death and adaptive penalties are appropriate for population-based stochastic algorithms. Death penalty does not require any penalty parameter although can be computationally expensive trying to find feasible points when the problem is highly constrained. Static penalties do not depend on the current iteration number and a constant value is set to all infeasible points. With a dynamic penalty, the penalty parameter increases with the iteration number and with the distance to feasibility. Most of the time, the dynamic penalty term also relies on other parameters that depend on the problem at hand, and it is not an easy task to determine the best values for those parameters. Well succeeded applications of dynamic penalties within particle swarm optimization algorithms appear in [30, 41]. Annealing penalties depend on a parameter known as temperature that approaches zero as iterations proceed. In methods based on adaptive penalties, the penalty parameters are updated every iteration according to information gathered from the whole population of points. Adaptive penalties are proposed in [5] in conjunction

with a genetic algorithm. A penalty adapting algorithm used with an ant colony optimization aiming at eliminating the need for trial-and-error penalty parameter determination is proposed in [1]. We refer to [9] for details concerning these penalties, advantages and drawbacks during implementation.

Another common classification of penalty functions in classical optimization is based on interior and exterior penalty functions [6, 7]. Exterior penalties are used more often than interior penalties since an exterior penalty function does not require an initial feasible point to start the iterative process. Furthermore, algorithms based on interior penalty functions are more complex since all generated points should be maintained inside the feasible region throughout the whole iterative process. A well-known interior penalty is the logarithmic barrier function and works only with inequality constraints.

Here, we are specially interested in exterior penalty functions of the additive type. Three different penalty functions are described and tested with a benchmark set of problems. Although setting the initial value for the penalty parameter as well as its updating scheme are usually critical in algorithm's performance, they are not yet well-defined issues. Nevertheless these issues are addressed since convergence to the solution is to be promoted and accelerated. Thus, details concerning the most appropriate strategies for updating the penalty and other related parameters are presented.

Our implementation of the penalty framework aims to penalize only the inequality constraints. Each subproblem of the sequence that is solved for a fixed value of the penalty μ is the bound constrained multilocal optimization problem

$$\begin{array}{l} \max\ \phi(x;\mu) \\ \quad \text{s.t. } l_i \leq x_i \leq u_i,\ i = 1,...,n \end{array} \tag{11}$$

To illustrate the effect on the penalty function ϕ as the penalty parameter μ increases, a one-dimensional example is used.

Example 6. Consider the problem

$$\max f(x) \equiv e^x - x^2 \ \text{ s.t. } \ x \leq 1 \text{ and } x \in [-5,5].$$

Figure 8 shows on the left plot the penalty function ϕ_1 that depends on the penalty term $\mathscr{P}(x,\mu) = \mu \max\{0, x-1\}$ and on the right plot the ϕ_2 that depends on the penalty term $\mathscr{P}(x,\mu) = \mu(\max\{0,x-1\})^2$, for the three values of $\mu = 1, 10, 100$. As it can be seen, in the feasible region $[-5,1]$, the penalty function coincides with $f(x)$, the function ϕ_2 is smoother at $x = 1$ (the solution of the problem) than ϕ_1, and the larger the μ the more difficult the problem is.

$\mathbf{L}_{1/2}$ penalty function. A variant of a dynamic nonstationary penalty function is herein used to solve constrained MPP [30, 41]. In these papers, particle swarm optimization algorithms are implemented in conjunction with the penalty technique. The penalty term of the herein simplified variant, denoted by $\mathrm{l}_{1/2}$ penalty function, is defined as

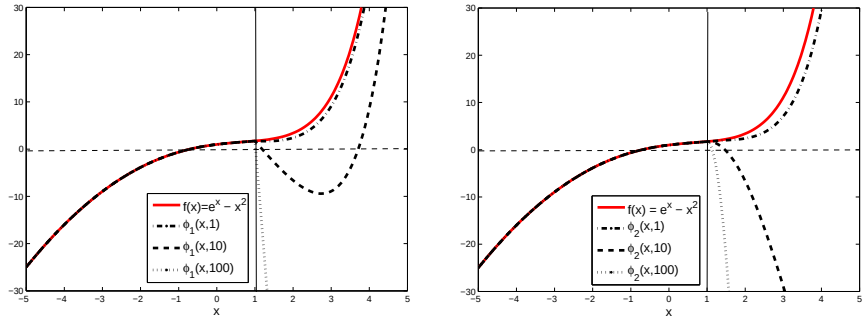

Fig. 8 Plot of $f(x)$ and ϕ_1 (on the left) and ϕ_2 (on the right) relative to Example 6.

$$\mathscr{P}_{1/2}(x,\mu) = \mu \sum_{j=1}^{m} \left(\max\left\{0, g_j(x)\right\}\right)^{\gamma(g_j(x))} \tag{12}$$

where the power of the constraint violation, $\gamma(.)$, may be a violation dependent constant. The simplest approach sets $\gamma(z) = 1$ if $z \leq 0.1$, and $\gamma(z) = 2$, otherwise. This is a nonsmooth function and derivative-free methods should be applied when solving problem (11). Unlike the suggestions in [30] and [41], the penalty parameter in (12) will not be changing dynamically with the iteration number. To define an appropriate updating scheme for μ one has to consider a safeguarded scheme to prevent the subproblems (11) from becoming ill-conditioned as the penalty parameter increases [7]. An upper bound $\mu_{\max}$ is then defined and the update is as follows:

$$\mu^{k+1} = \min\left\{\tau\mu^k, \mu_{\max}\right\}, \text{ for } \tau > 1 \text{ and } \mu_{\max} >> 1, \tag{13}$$

given an initial value $\mu^0 > 0$, where k represents the iteration counter. Thus, the sequence of solutions $\{x^*(\mu^k)\}$, from (11), will converge to the solution x^* of (1) and $\phi(x^*(\mu^k);\mu^k) \to f(x^*)$ as $k \to \infty$.

L_2-exponential penalty function. We now extend the use of a continuous l_2-exponential penalty function to the constrained multilocal optimization problem. This penalty function was previously incorporated into a reduction-type method for solving semi-infinite programming problems [43]. The penalty term depends on the positive penalty parameter μ and other two fixed positive parameters v_1, v_2:

$$\mathscr{P}_2^{\exp}(x, v_1, v_2, \mu) = \frac{v_1}{\mu}\left(e^{\mu\theta(x)} - 1\right) + \frac{v_2}{2}\left(e^{\mu\theta(x)} - 1\right)^2, \tag{14}$$

where $\theta(x) = \max_{j=1,\ldots,m}[g_j(x)]_+$ and the $[g_j(x)]_+$ represents $\max\{0, g_j(x)\}$. Clearly $\theta(x)$ is the infinity norm of the constraint violation. The tuning of the penalty parameter previously described in (13) also applies to this penalty function.

Hyperbolic penalty function. Another proposal uses the 2-parameter hyperbolic penalty function [56]. This is a continuously differentiable function that depends on two positive penalty parameters, in general different for each constraint, $\mu_{1,j}$ and $\mu_{2,j}$, $j = 1,\ldots,m$,

$$\mathscr{P}^{\text{hyp}}(x,\mu_1,\mu_2) = \sum_{j=1}^{m} \mu_{1,j} g_j(x) + \sqrt{\mu_{1,j}^2 [g_j(x)]^2 + \mu_{2,j}^2}. \tag{15}$$

This penalty is made to work as follows. In the initial phase of the process, μ_1 increases, causing a significant increase of the penalty at infeasible points, while a reduction in penalty is observed for points inside the feasible region. This way the search is directed to the feasible region since the goal is to minimize the penalty. From the moment that a feasible point is obtained, the penalty parameter μ_2 decreases. Thus, the parameters $\mu_{1,j}$ and $\mu_{2,j}$ are updated, for each $j = 1,\ldots,m$, as follows:

$$\begin{cases} \mu_{1,j}^{k+1} = \tau_1 \mu_{1,j}^k \text{ and } \mu_{2,j}^{k+1} = \mu_{2,j}^k, & \text{if } \max\{0, g_j(x^k)\} > 0 \\ \mu_{2,j}^{k+1} = \tau_2 \mu_{2,j}^k \text{ and } \mu_{1,j}^{k+1} = \mu_{1,j}^k, & \text{otherwise} \end{cases}$$

for each $j = 1,\ldots,m$, where $\tau_1 > 1$ and $\tau_2 < 1$.

Multilocal penalty algorithm. The multilocal penalty (MP) algorithm can be implemented using the stretched simulated annealing algorithm when solving subproblem (11), or the multilocal BB, both previously described in Subsections 2.1 and 2.2 respectively. Details of the main steps of the algorithm are shown in Algorithm 4. The algorithm is described for the simpler penalty function, see (12). Adjustments have to be made when the penalty functions (14) and (15) are used.

Algorithm 4 MP algorithm

1: **Given**: μ^0, $\mu_{\max}$, τ, δ_0, ε_0, $\varepsilon_{\max}$. Set $k = 0$
2: **While** the stopping conditions are not met **do**
3: Set $L^k = 0$ and $j = 0$
4: **While** inner stopping conditions are not met **do**

4.1 Set $p = 0$ and $j = j + 1$
4.2 Compute $x_j^*(\mu^k) = \arg\max_{l \le x \le u} \phi^j(x;\mu^k)$ using Algorithm 2 or Algorithm 3
4.3 **While** $\left|\phi^j\left(x_j^*(\mu^k),\mu^k\right) - \widetilde{\phi}_{\max}\right| \le \delta_0$ or $\Delta > \varepsilon_{\max}$ **do**
Set $p = p + 1$ and $\Delta = p\varepsilon_0$
Randomly generate $\widetilde{x}_i \in V_\Delta(x_j^*)$, $i = 1,\ldots,2n$
Find $\widetilde{\phi}_{\max} = \max_{i=1,\ldots,2n}\{\phi^j(\widetilde{x}_i,\mu^k)\}$
4.4 Set $L^k = L^k + 1$ and $\varepsilon_j = \Delta$

5: $\mu^{k+1} = \min\{\tau\mu^k, \mu_{\max}\}$
6: Set $X^* \leftarrow X^*(\mu^k)$ and $k = k + 1$

3.2 Numerical experiments

Here, we aim to compare the effectiveness of the SSA algorithm when coupled with a penalty function method to compute multiple solutions. The above listed penalty functions, $l_{1/2}$ penalty, l_2-exponential penalty and the hyperbolic penalty are tested.

Stopping conditions. The stopping conditions for the multilocal penalty algorithm are:

$$\left\| X^*(\mu^k) - X^*(\mu^{k-1}) \right\| \leq \varepsilon_x \text{ or } k > k_{\max} \tag{16}$$

and the inner iterative process (in Step 2 of Algorithm 4) terminates if L^k does not change for a specified number of iterations, K_{iter}, or a maximum number of function evaluations is reached, $nf_{\max}$.

Setting parameters. In this study the selected values for the parameters resulted from an exhaustive set of experiments. Here is the list: $\varepsilon_x = 10^{-3}$, $k_{\max} = 1000$ and the parameters for the $l_{1/2}$ penalty function are $\mu^0 = 10$, $\mu_{\max} = 10^3$ and $\tau = 10$. The parameters used in the l_2-exponential penalty function are $\nu_1 = 100$ and $\nu_2 = 100$. The parameters used in the Hyperbolic penalty function are $\mu^0_{1,j} = \mu^0_{2,j} = 10$ for $j = 1, \ldots, m$, $\tau_1 = \sqrt{10}$ and $\tau_2 = 0.1$. The parameters of the SSA algorithm are set as follows: $\delta_0 = 5.0$, $\varepsilon_0 = 0.1$, $\varepsilon_{\max} = 1.0$, $K_{\text{iter}} = 5$ and $nf_{\max} = 100\,000$. The problems were solved in a Inter Core 2 Duo, T8300, 2.4 GHz with 4 GB of RAM.

Experiments. For the first part of our comparative study, we use a well-known problem described in Example 7.

Example 7. Consider the camelback objective function

$$f(x) = -(4 - 2.1x_1^2 + x_1^4/3)x_1^2 - x_1x_2 + 4(1 - x_2^2)x_2^2$$

which has four local maxima and two minima in the set $-2 \leq x_i \leq 2, i = 1,2$. The two global maxima are located at (0.089482, -0.712656) and (-0.089482, 0.712656). Here, we define the constrained problem:

$$\begin{aligned} &\max\ f(x) \\ &\text{s.t. } g(x) \equiv x_1^2 + (x_2 - 1)^2 - 4 \leq 0, \\ &\quad -2 \leq x_i \leq 2, i = 1,2 \end{aligned} \tag{17}$$

and illustrate the behavior of the MPA when using SSA algorithm to solve the bound constrained subproblems. Figure 9 shows the 3D plot and contour lines of $f(x)$ as well as of $g(x) \leq 0$. This nonconvex problem has three maxima in the interior of the feasible region.

The problem in (17) was solved using the MP algorithm combined with the hyperbolic penalty function. The method identified two global solutions $x_1^* = (-8.9842E-02, 7.1266E-01)^T$ and $x_2^* = (8.9842E-02, -7.1266E-01)^T$ with the global value $1.0316E+00$. The local maximizer $x_3^* = (-1.7036E+00, 7.9608E-01)^T$ with

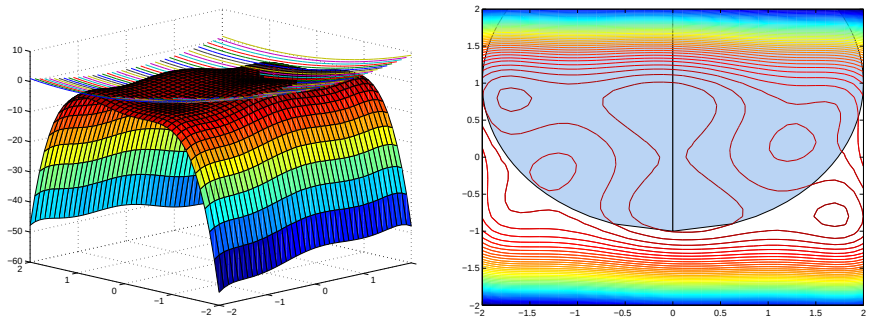

Fig. 9 Plot of $f(x)$ and $g(x) \leq 0$ in Example 7.

value $f(x_3^*) = 2.1546E-01$ was also detected. To solve this problem, the MP algorithm needed 2.14 seconds of CPU time and 10535 functions evaluations, both average number in 30 runs.

To further analyze the performance of the multilocal penalty algorithm when coupled with SSA, a set of six benchmark problems, described in full detail in [29], is used. In this study, small dimensional problems ($n \leq 10$ and $m \leq 13$) with a nonlinear objective function, simple bounds and inequality constraints were tested. They are known in the literature as g04, g06, g08, g09, g12 and g18. Details of the selected problems are displayed in Table 4, where 'Problem' refers to the problem number, 'type of $f(x)$' describes the type of objective function, 'f_{opt}-global' is the known global solution (all are minimization problems), n is the number of variables and m is the number of inequality constraints.

Table 4 Details of the constrained problems selected from [29].

Problem	type of $f(x)$	f_{opt}-global	n	m
g04	quadratic	$-3.0665E+04$	5	6
g06	cubic	$-6.9618E+03$	2	2
g08	general	$-9.5825E-02$	2	2
g09	general	$6.8063E+02$	7	4
g12	quadratic	$1.0000E+00$	3	1
g18	quadratic	$-8.6603E-01$	9	13

Table 5 contains the results obtained with the penalties $\mathscr{P}_{1/2}$, $\mathscr{P}_2^{\text{exp}}$ and $\mathscr{P}^{\text{hyp}}$, when combined with the SSA algorithm. The f^* is the best solution found for the global minimum during all the 30 runs, $nf_{\text{avg}}^{\text{eval}}$ indicates the average number of function evaluations required to obtain the global minimum (over the 30 runs) and n_{sol} represents the number of solutions identified by the algorithm.

Table 5 Results for the MP algorithm, combined with SSA.

	$\mathscr{P}_{1/2}$			$\mathscr{P}_2^{\exp}$			$\mathscr{P}^{\text{hyp}}$		
	f^*	$nf_{\text{avg}}^{\text{eval}}$	n_{sol}	f^*	$nf_{\text{avg}}^{\text{eval}}$	n_{sol}	f^*	$nf_{\text{avg}}^{\text{eval}}$	n_{sol}
g04	$-3.067+04$	156154	12	$-3.067+04$	62337	1	$-3.067+04$	18352	1
g06	$-6.962E+03$	27550	1	$-6.962E+03$	6472	1	$-6.962E+03$	15766	1
g08	$-9.583E-02$	79771	5	$-9.583E-02$	67753	5	$-9.583E-02$	8624	1
g09	$6.787E+02$	309719	1	$6.787E+02$	183806	1	$6.787E+02$	117638	1
g12	$1.000E+00$	202219	1	$1.000E+00$	302134	1	$1.000E+00$	313211	1
g18	$-8.660E-01$	945000	2	$-8.660E-01$	845375	4	$-8.660E-01$	339213	4

3.3 Synopsis

We have described some important issues related with the implementation of penalty function methods in classical optimization. A proposal focused on a penalty framework is shown when multiple solutions of constrained optimization problems are required. Three penalty functions have been presented and discussed. The numerical results obtained when the penalty function method is used to solve constrained MPP are reported. A comparison between the three penalty functions is included. The subproblems that emerge from the multilocal penalty strategy are bound constrained MPP and they may be solved by the two proposed strategies, either the stretched simulated annealing algorithm or the multilocal BB algorithm. However, the numerical experiments reported in this section use the MP algorithm which relies on the stretched simulating annealing, since this is by far the most efficient version. Last, we have shown that the penalty function method is effective in solving constrained MPP, in particular when some penalty functions are used.

4 Engineering Applications

In the last part of the chapter, a real-world application of multilocal programming in the engineering field is presented. Phase stability studies are multilocal programming problems frequently found in the chemical engineering area with special interest in process design and optimization. These studies, still a current subject for scientists and engineers, are specially difficult, since the feasible region is very small and not convex. In this section the mathematical formulation of the problem is initially given as well as a very brief summary of the strategies and optimization techniques used so far. Following, some numerical results are presented and discussed, and the main findings outlined.

4.1 Phase stability

Separation processes are fundamental and ubiquitous operations in the chemical based industries. However, to design and optimize such separation operations, thermodynamic equilibrium conditions must be known. A severe problem causing enormous difficulties in this regard is that the number and identity of phases present at equilibrium are generally not known [46], which makes phase stability analysis obligatory. At a fixed temperature, pressure and global composition the problem is, therefore, to evaluate if the system is globally stable regarding the separation in two or more liquid phases.

The phase stability criteria based on the Gibbs free energy of mixing, or derived properties, are multiple, but the minimization of the tangent plane distant function ($TPDF$), firstly proposed by Baker et al. [4], and first implemented by Michelsen [35], is usually applied, and accepted to be a reliable and potent methodology for stability studies. Considering the Gibbs free energy of mixing (ΔG) of a multicomponent mixture, at a given temperature (T) and pressure (P), to be described as $\Delta g(x) = \frac{\Delta G}{RT} = f(T,P,x)$, where x is the vector of n mole fraction compositions characterizing that mixture and R is the ideal gas constant. For an initial feed composition, z, at a fixed system pressure and temperature, the tangent plane equation (Δg_{tp}) at that point is:

$$\Delta g_{tp}(x) = \Delta g(z) + \sum_{i=1}^{n} \left(\frac{\partial \Delta g}{\partial x_i} \right) \Bigg|_{x=z} (x_i - z_i).$$

In this way the tangent plane distance function ($TPDF$) is calculated by:

$$TPDF(x) = \Delta g(x) - \Delta g_{tp}(x).$$

Among the several thermodynamic models possible to apply, NRTL model [47] is one of the most successful in the representation of equilibrium properties of multicomponent liquid mixtures, and is frequently found in commercial software for process simulation and design. Therefore, NRTL model is here applied for which:

$$\Delta g = \sum_{i=1}^{n} x_i ln(x_i) + \sum_{i=1}^{n} x_i \left(\frac{\sum_{j=1}^{n} \tau_{ji} G_{ji} x_j}{\sum_{l=1}^{n} G_{li} x_l} \right)$$

where τ_{ji} and G_{ji} are interaction parameters between components j and i, calculated by $G_{ji} = \exp(-\alpha_{ji}\tau_{ji})$, being α the non-randomness parameter. They are all readily available in the open literature.

To evaluate if a mixture of a given global composition shows phase instability the following nonlinear multilocal optimization problem must be solved:

$$\begin{aligned} &\min\ TPDF(x) \\ &\text{s.t.}\ \sum_{i=1}^{n}(x_i) - 1 = 0 \\ &\quad 0 \le x_i \le 1 \quad \text{and} \quad i = 1,\ldots,n. \end{aligned}$$

The necessary and sufficient condition for stability is that at the global minimum the $TPDF(x)$ function is nonnegative. Phase instability will be observed otherwise. In that event the following step is to find the number of phases in equilibrium as well as the composition of each phase.

Due to the mathematical complexity of the thermodynamic models, the minimization of the $TPDF$ and location of all the stationary points are demanding tasks, requiring robust numerical methods, since these functions are multivariable, nonconvex, and highly nonlinear [8]. Strictly speaking, to check phase stability only the global minimum is needed. However, the identification of all stationary points is very important because the local minima in $TPDF$ are good initial guesses for the equilibrium calculations [13, 49].

Floudas and Gounaris [16] have very recently reviewed different strategies and optimization techniques for phase stability and phase equilibrium calculations. Thus, only aspects of relevance for the optimization methods and examples explored in this section are briefly mentioned. In fact, the vast majority of the researchers state that many techniques are initialization dependent, and may fail by converging to trivial solutions or be trapped in local minima [8, 13, 16, 33, 49], features which are under attention in the numerical examples given in the following pages. Hence, the performance analysis of new numerical techniques is still of enormous importance concerning phase stability and equilibria studies.

Particularly, several variants of the simulated annealing method have been widely applied, and importantly studies have been performed concerning the so-called 'cooling schedule', by fixing the control parameters to the best values [13, 46, 58, 66]. Naturally, a compromise must be made between efficiency and reliability, analyzing the probability of obtaining the global minimum within a reasonable computational effort. On the other hand, a branch and bound algorithm has been used with several thermodynamic models [59, 65]. These authors claim that it can solve effectively the global stability problem, but only a few studies have been carried out.

Due to space limitations only two relevant examples are now presented using SSA.

Example 8. Consider the binary system water (1) + butyl glycol (2) at 5 ^{o}C. It might seem a simple example, but this is a canonical example, where multiple stationary points and local solutions can be found. Additionally, for some compositions, in [37] it was concluded that the stationary points found in [17], using the interval Newton method, are not true roots as shown by the simulated annealing method.

The NRTL parameters used in the calculations are given in Table 6, while Table 7 compiles the results obtained at four different global compositions z.

Confirming the results from [37], at the first two compositions only one stationary point was found, giving the indication that only one liquid phase will be formed. On the contrary, the other two compositions present a negative value of the $TPDF$ at

Table 6 NRTL parameters in Example 8 [17].

Components	i	j	τ_{ij}	τ_{ji}	$\alpha_{ij} = \alpha_{ji}$
water/butyl glycol	1	2	1.2005955	1.4859846	0.121345

Table 7 Numerical results for the binary system water + butyl glycol.

z	CPU(s)	f^*	x^*
(5.00E-02, 9.50E-01)	0.22	0.0000E+00	(5.00E-02, 9.50E-01)
(1.00E-01, 9.00E-01)	0.27	0.0000E+00	(1.00E-01, 9.00E-01)
(2.50E-01, 7.50E-01)	0.30	-9.2025E-02	(8.79E-01, 1.21E-01)
		0.0000E+00	(2.50E-01, 7.50E-01)
		8.4999E-05	(2.96E-01, 7.04E-01)
(5.00E-01, 5.00E-01)	0.20	-3.4091E-02	(1.43E-01, 8.57E-01)
		-2.7355E-02	(8.36E-01, 1.64E-01)
		0.0000E+00	(5.00E-01, 5.00E-01)

the global minimum, suggesting phase instability. At the global composition (0.25, 0.75) it must be noted the closeness of two stationary points, which can introduce difficulties when applying the stretched technique as well as the small magnitude of the function at the stationary point. The performance of the SSA can be assessed by verifying that all the 30 runs converge to the function value (f^*) at the stationary point (x^*). It also must be stressed that the average time is much uniform when comparing with the results in [37] and [17].

Example 9. Consider now the ternary system n-propanol (1) + n-butanol (2) + water (3) at 25 ^{o}C. The NRTL parameters needed are compiled in Table 8.

Table 8 NRTL parameters in Example 9 [66].

Components	i	j	τ_{ij}	τ_{ji}	$\alpha_{ij} = \alpha_{ji}$
propanol/butanol	1	2	-0.61259	0.71640	0.30
propanol/water	1	3	-0.07149	2.74250	0.30
butanol/water	2	3	0.90047	3.51307	0.48

This is also a reference system in the study of phase stability, presenting, like in the previous example, multiple stationary points. Table 9 presents a complete list of the results found for two global compositions. In both cases the $TPDF$ function is negative indicating phase splitting. It must again be stressed the closeness of some stationary points and the very small magnitude of the function. The average time although longer than in the previous example is still very uniform.

Table 9 Numerical results for the ternary system n-propanol + n-butanol + water.

z	CPU(s)	f^*	x^*
(1.20E-01, 8.00E-02, 8.00E-01)	2.42	-7.4818E-04	(5.97E-02, 2.82E-02, 9.12E-01)
		-3.0693E-06	(1.30E-01, 8.91E-02, 7.81E-01)
		0.0000E+00	(1.20E-01, 8.00E-02, 8.00E-01)
(1.30E-01, 7.00E-02, 8.00E-01)	2.34	-3.2762E-04	(7.38E-02, 3.03E-02, 8.96E-01)
		-8.6268E-07	(1.38E-01, 7.56E-02, 7.87E-01)
		0.0000E+00	(1.30E-01, 7.00E-02, 8.00E-01)

4.2 Synopsis

The phase stability of two mixtures was studied at different global compositions using the SSA algorithm. It proved to be very reliable and robust even in the cases where the stationary points are very close. Additionally, it was possible to find short CPU times for all the seven conditions investigated. The results found so far will soon be checked and extended to compositions near to the plait point and to systems containing three liquid phases, hardly even considered [18], or to quaternary systems with multiple stationary points.

References

1. Afshar, M.H.: Penalty adapting ant algorithm: application to pipe network optimization. Eng. Optim. 40, 969-987 (2008)
2. Alefeld, G., Mayer, G.: Interval analysis: theory and applications. J. Comput. Appl. Math. 121, 421-464 (2000)
3. Ali, M.M., Gabere, M.N.: A simulated annealing driven multi-start algorithm for bound constrained global optimization. J. Comput. Appl. Math. 233, 2661-2674 (2010)
4. Baker, L.E., Pierce, A.C., Luks, K.D.: Gibbs energy analysis of phase equilibria. Soc. Petrol. Eng. J. 22, 731-742 (1982)
5. Barbosa, H.J.C., Lemonge, A.C.C.: An adaptive penalty method for genetic algorithms in constrained optimization problems. In: Iba H. (ed.) Frontiers in Evolutionary Robotics, I-Tech Education Publ., Austria (2008)
6. Bertsekas, D.P.: Constrained Optimization and Lagrange Multiplier Methods. Academic Press, New York (1982)
7. Bertsekas, D.P.: Nonlinear Programming, 2nd edition. Athena Scientific, Belmont (1999)
8. Bonilla-Petriciolet, A., Vásquez-Román, R., Iglesias-Silva, G.A., Hall, K.R.: Performance of stochastic global optimization methods in the calculation of phase analyses for nonreactive and reactive mixtures. Ind. Eng. Chem. Res. 45, 4764-4772 (2006)
9. Coello, C.A.C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Technical Report (48 pages), CINVESTAV-IPN, Mexico (2002)
10. Coope, I.D., Watson, G.A.: A projected Lagrangian algorithm for semi-infinite programming. Math. Program. 32, 337-356 (1985)
11. Csendes, T., Pál, L., Sendín, J.O.H., Banga, J.R.: The GLOBAL optimization method revisited. Optim. Lett. 2, 445-454 (2008)

12. Fanelli, S.: A new algorithm for box-constrained global optimization. J. Optim. Theory Appl. 149, 175-196 (2011)
13. Ferrari, J.C., Nagatani, G., Corazza, F.C., Oliveira, J.V., Corazza, M.L.: Application of stochastic algorithms for parameter estimation in the liquid-liquid phase equilibrium modeling. Fluid Phase Equilib. 280, 110-119 (2009)
14. Finkel, D.E., Kelley, C.T.: Convergence analysis of the DIRECT algorithm. Optim. Online 14, 1-10 (2004)
15. Floudas, C.A.: Recent advances in global optimization for process synthesis, design and control: enclosure all solutions. Comput. Chem. Eng. 23, S963-S973 (1999)
16. Floudas, C.A., Gounaris, C.E.: A review of recent advances in global optimization. J. Glob. Optim. 45, 3-38 (2009)
17. Gecegormez, H., Demirel, Y.: Phase stability analysis using interval Newton method with NRTL model. Fluid Phase Equilib. 237, 48-58 (2005)
18. Guo, M., Wang, S., Repke, J.U., Wozny, G.: A simultaneous method for two- and three-liquid-phase stability determination, AIChE J. 50, 2571-2582 (2004)
19. Hansen, E.R., Walster, G.W.: Global Optimization Using Interval Analysis. 2nd edition, Marcel Dekker, Inc., New York (2004)
20. Hendrix, E.M.T., G.-Tóth, B.: Introduction to Nonlinear and Global Optimization. Springer, New York (2010)
21. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 3rd edition. Springer, Berlin (1996)
22. Ingber, L.: Very fast simulated re-annealing. Math. Comput. Model. 12, 967-973 (1989)
23. Ingber, L.: Simulated annealing: practice versus theory. Math. Comput. Model. 18, 29-57 (1993)
24. Ingber, L.: Adaptive simulated annealing (ASA): lessons learned. Control Cybern. 25, 33-54 (1996)
25. Jones, D.R., Perttunen, C.C., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. 79, 157-181 (1993)
26. Jones, D.R.: Direct global optimization algorithm. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, pp. 725-735. Springer (2009)
27. Kiseleva, E., Stepanchuk, T.: On the efficiency of a global non-differentiable optimization algorithm based on the method of optimal set partitioning. J. Glob. Optim. 25, 209-235 (2003)
28. León, T., Sanmatias, S., Vercher, E.: A multilocal optimization algorithm. TOP 6, 1-18 (1998)
29. Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.A.C., Deb, K.: Problem definitions and evaluation criteria for the CEC2006 special session on constrained real-parameter optimization. Technical Report (2006)
30. Liu, J.L., Lin, J.H.: Evolutionary computation of unconstrained and constrained problems using a novel momentum-type particle swarm optimization. Eng. Optim. 39, 287-305 (2007)
31. Liuzzi, G., Lucidi, S., Piccialli, V.: A partition-based global optimization algorithm. J. Glob. Optim. 48, 113-128 (2010)
32. McDonald, C.M., Floudas, C.A.: Global optimization for the phase stability problem. AIChE J. 41, 1798-1814 (1994)
33. McDonald, C.M., Floudas, C.A.: Global optimization for the phase and chemical equilibrium problem: application to the NRTL equation. Comput. Chem. Eng. 19, 1111-1139 (1995)
34. Michalewicz, Z.: A survey of constraint handling techniques in evolutionary computation methods. Proceedings of the 4th Annual Conference on Evolutionary Programming pp. 135-155 (1995)
35. Michelsen, M.L.: The isothermal flash problem. Part I. Stability. Fluid Phase Equilib. 9, 1-19 (1982)
36. Mieltinen, K., Mäkelä, M.M., Toivanen, J.: Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. J. Glob. Optim. 27, 427-446 (2003)
37. Nagatani, G., Ferrari, J., Cardozo Filho, L., Rossi, C.C.R.S., Guirardello, R., Oliveira, J.V., Corazza, M.L.: Phase stability analysis of liquid-liquid equilibrium with stochastic methods. Braz. J. Chem. Eng. 25, 571-583 (2008)

38. Parsopoulos, K.E., Plagianakos, V., Magoulas, G., Vrahatis, M.N.: Objective function stretching to alleviate convergence to local minima. Nonlinear Anal. 47, 3419-3424 (2001)
39. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through particle swarm optimization. Nat. Comput. 1, 235-306 (2002)
40. Parsopoulos, K.E., Vrahatis, M.N.: On the computation of all global minimizers through particle swarm optimization. IEEE Transaction on Evolutionary Computation 8, 211-224 (2004)
41. Petalas, Y.G., Parsopoulos, K.E., Vrahatis, M.N.: Memetic particle swarm optimization. Ann. Oper. Res. 156, 99-127 (2007)
42. Pereira, A.I.P.N., Fernandes, E.M.G.P.: On a reduction line search filter method for nonlinear semi-infinite programming problems. In: Sakalauskas, L., Weber, G.W., Zavadskas, E.K. (eds.) Euro Mini Conference Continuous Optimization and Knowledge-Based Technologies, pp. 174-179 (2008)
43. Pereira, A.I.P.N., Fernandes, E.M.G.P.: Numerical experiments with a continuous L_2-exponential merit function for semi-infinite programming. In: Simos, T.E., Psihoyios, G. (eds.) International Electronic Conference on Computer Science, AIP Vol. 1060(1), pp. 1354-1357, Springer-Verlag (2008)
44. Pereira, A.I.P.N., Fernandes, E.M.G.P.: A reduction method for semi-infinite programming by means of a global stochastic approach. Optim. 58, 713-726 (2009)
45. Pereira, A.I.P.N., Fernandes, E.M.G.P.: Constrained multi-global optimization using a penalty stretched simulated annealing framework. In: Simos, T.E., Psihoyios, G., Tsitouras, Ch. (eds.) Numerical Analysis and Applied Mathematics, AIP Vol. 1168, pp. 1354-1357, Springer-Verlag (2009)
46. Rangaiah, G.P.: Evaluation of genetic algorithms and simulated annealing for phase equilibrium and stability problems. Fluid Phase Equilib. 187-188, 83-109 (2001)
47. Renon, H., Prausnitz, J.M.: Local compositions in thermodynamic excess functions for liquid mixtures. AIChE J. 14, 135-144 (1968)
48. Sepulveda, A.E., Epstein, L.: The repulsion algorithm, a new multistart method for global optimization. Struct. Multidiscip. Optim. 11, 145-152 (1996)
49. Tessier, S.R., Brennecke, J.F., Stadtherr, M.A.: Reliable phase stability analysis for excess Gibbs energy models. Chem. Eng. Sci. 55, 1785-1796 (2000)
50. Tsoulos, L.G., Lagaris, I.E.: MinFinder: locating all the local minima of a function. Comput. Phys. Commun. 174, 166-179 (2006)
51. Tu, W., Mayne, R.W.: Studies of multi-start clustering for global optimization. Int. J. Numer. Methods Eng. 53, 2239-2252 (2002)
52. Voglis, C., Lagaris, I.E.: Towards "Ideal Multistart". A stochastic approach for locating the minima of a continuous function inside a bounded domain. Appl. Math. Comput. 213, 216-229 (2009)
53. Wang, Y.J.: Derivative-free simulated annealing and deflecting function technique for global optimization. J. Appl. Math. Comput. 1-2, 49-66 (2008)
54. Wang, Y., Cai, Z., Zhou, Y., Fan, Z.: Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. Struct. Multidiscip. Optim. 37, 395-413 (2008)
55. Wu, Z.Y., Bai, F.S., Lee, H.W.J., Yang, Y.J.: A filled function method for constrained global optimization. J. Glob. Optim. 39, 495-507 (2007)
56. Xavier, A.: Hyperbolic penalty: a new method for nonlinear programming with inequalities. Int. Trans. Oper. Res. 8, 659-671 (2001)
57. Yeniay, Ö.: Penalty function methods for constrained optimization with genetic algorithms. Math. Comput. Appl. 10, 45-56 (2005)
58. Yushan, Z., Zhihong, X.: A reliable method for liquid-liquid phase equilibrium calculation and global stability analysis. Chem. Eng. Commun. 176, 113-160 (1999)
59. Yushan, Z., Zhihong, X.: Calculation of liquid-liquid equilibrium based on the global stability analysis for ternary mixtures by using a novel branch and bound algorithm: application to UNIQUAC Equation. Ind. Eng. Chem. Res. 38, 3549-3556 (1999)
60. Zahara, E., Hu, C.H.: Solving constrained optimization problems with hybrid particle swarm optimization. Eng. Optim. 40, 1031-1049 (2008)

61. Zhang, X., Liu, S.: Interval algorithm for global numerical optimization. Eng. Optim. 40, 849-868 (2008)
62. Zhigljavsky, A., Zilinskas, A.: Stochastic Global Optimization. Optimization and Its Applications, Springer (2007)
63. Zhu, W.: A class of filled functions for box constrained continuous global optimization. Appl. Math. Comput. 169, 129-145 (2005)
64. Zhu, W., Ali, M.M.: Solving nonlinearly constrained global optimization problem via an auxiliary function method. J. Comput. Appl. Math. 230, 491-503 (2009)
65. Zhu, Y., Inoue, K.: Calculation of chemical and phase equilibrium based on stability analysis by QBB algorithm: application to NRTL equation. Chem. Eng. Sci. 56, 6915-6931 (2001)
66. Zhu, Y., Xu, Z.: A reliable prediction of the global phase stability for liquid-liquid equilibrium through the simulated annealing algorithm: application to NRTL and UNIQUAC equations. Fluid Phase Equilib. 154, 55-69 (1999)